





In-depth analysis of SVM kernel learning and its components

Ibai Roman¹ Roberto Santana¹
Alexander Mendiburu¹
Jose A. Lozano^{1,2}

✉ ibai.roman@ehu.eus

¹Intelligent Systems Group,
University of the Basque Country, UPV/EHU
Paseo Manuel de Lardizabal 1, 20018 Donostia, Spain
²Basque Center for Applied Mathematics, BCAM
Alameda de Mazarredo 14, 48009 Bilbao, Spain

Abstract

The performance of support vector machines in non-linearly-separable classification problems strongly relies on the kernel function. Towards an automatic machine learning approach for this technique, many research outputs have been produced dealing with the challenge of automatic learning of good-performing kernels for support vector machines. However, these works have been carried out without a thorough analysis of the set of components that influence the behavior of support vector machines and their interaction with the kernel. These components are related in an intricate way and it is difficult to provide a comprehensible analysis of their joint effect. In this paper we try to fill this gap introducing the necessary steps in order to understand these interactions and provide clues for the research community to know where to place the emphasis. First of all, we identify all the factors that affect the final performance of support vector machines in relation to the elicitation of kernels. Next, we analyze the factors independently or in pairs and study the influence each component has on the final classification performance, providing recommendations and insights into the kernel setting for support vector machines.

Keywords: SVM, Kernel Learning, Genetic Programming and Automatic Machine Learning

1 Introduction

Support Vector Machines (SVMs) [52] have been, for a long time, the reference paradigm in supervised classification and regression. Although the field is nowadays overwhelmed by the application of deep learning approaches, SVMs are still one of the best alternatives when the requirements of deep neural networks are not met. When applied to binary classification problems, SVMs separate samples from the two different classes by means of a hyperplane that maximizes the gap to the nearest samples in order to ensure a proper generalization. SVMs can even handle non-linearly-separable problems by means of a kernel function [4], and when this kernel meets Mercer’s condition [33], the optimal hyperplane can be found.

Although SVMs are a suitable tool to solve classification problems, they involve several components that should be adjusted in order to obtain a good performance. Among these, the choice of the kernel heavily influences the performance of SVMs, and there is no rule of thumb to select it. While some standard kernels proposed in the literature are straightforwardly used in several applications, tailored kernels produce much better results as each problem has specific characteristics [42, 43]. In order to achieve an automated machine learning approach, several works in the literature pose the kernel selection as a search problem in the space of kernels with no human intervention [25, 12, 28].

However, there are still some open questions that have not been answered in the current kernel search literature regarding the search method, the space of kernels where this search is carried out, and the interaction with other components of SVMs. To start with, a search space which contains all (and only) the Mercer kernels has not been described yet. Instead, previously proposed methods pose some sort of limitations in the search space of kernels, whether only considering a subset of all the Mercer kernels or also including some which are non-Mercer. Several challenges have to be dealt with in relation to this topic: How does the selected search space of kernels influence the results of SVMs? Which are the regions of the search space according to the characteristics of the kernels on which the search efforts should focus?

Once the space of possible kernels has been defined, the next relevant question is the selection of a strategy to carry out the search. Most of the works in the literature have proposed various heuristic algorithms to solve this search problem, Genetic Programming (GP) being one of the most used methods [25, 12, 28]. However, there is a lack of knowledge about many aspects related to the specific characteristics of the kernel function optimization problem, and in particular, about how these characteristics relate to the way the GP search for optimal solutions is accomplished. Furthermore, there is no clear understanding of the relative performance of GP compared to other simpler search strategies, since other optimization methods have rarely been applied to this problem. Relevant questions in this area are: What is the relevance of the search method with respect to the characteristics of the chosen search space? Which characteristics should a search algorithm have in order to efficiently explore the kernel space?

Apart from the choice of the kernel, there are other components of SVMs that interact in a complex manner, which hinders the identification of the essential elements that are necessary to obtain a good performance in the classification task. However, in most of the previous works little attention has been paid to the rationale behind the choice of those components of SVMs and how these choices influence the dynamics and results of the kernel search.

The learning of the kernel itself is often divided into the kernel structure search and the tuning of its hyperparameters. This tuning process is one of the steps involved in the SVM learning, whose essential role is usually overlooked in the literature. The key questions are: What is the relevance of finding the right hyperparameters? Which is the best method for finding them? How much computational effort needs to be used to optimize the hyperparameters?

Beyond the setting of the kernel and its hyperparameters, the commonly used flexible variant of SVMs has its own parameter (C), whose role is to deal with the overfitting of the model. Although, the choice of C strongly influences the effectiveness of the final classifier, it is not clear in the literature what the interactions of this parameter are with the rest of the components. For instance, what is the contribution of the C parameter to the performance of SVMs with a particular choice of the kernel? Which is the interplay between the kernel hyperparameter setting and the C parameter setting?

Finally, for an automated kernel search, we not only need to assess the quality of the solution on the training data but also to implicitly capture how it will generalize to new data. If a wrong evaluation measure is chosen, then, an apparently good solution (in terms of the measure) may overfit the data and produce poor results at the prediction stage. The choice of the objective function used to evaluate the quality of the kernel also has an impact on the roughness of the kernel search space, and therefore on the performance of the search methods. Most of the previous works in the literature have used the classifier accuracy as the metric of choice. However, are there better metrics to guide the search for optimal kernels?

Trying to shed some light on these issues, in this paper we analyze the components involved in the structural learning of kernels. We start by considering each component independently, and then, we proceed by addressing the way they interact to influence the behavior of SVMs. In the study of these components and their interactions, we introduce some guidelines to improve the performance of SVMs.

The remainder of the paper is structured as follows: In the next section, a background on SVM classification is provided, including the presentation of standard kernel functions and an overview of the SVM method. Next, we present our research regarding the different components that take part in the kernel search. In Section 4, we analyze the questions regarding the kernel space, and in Section 5, we address the influence of the search method. The interactions between the hyperparameter tuning and C parameter setting are studied in Section 6, and Section 7 focuses on the metrics used to measure the performance of the SVMs. Finally, in Section 8, the conclusions and future work are presented.

2 Support Vector Machine Classification

Support Vector Machines (SVMs) were introduced by Vapnik in 1963 [51] as non-probabilistic linear classifiers to solve binary classification problems. Later, probabilistic variants [40] of SVMs and extensions to multi-class problems [8] were proposed.

In a supervised binary classification scenario, linear classifiers, such as SVMs, classify these samples by means of a hyperplane. Nevertheless, there are many ways to position this hyperplane. SVMs are characterized by the use of a hyperplane that maximizes the separation, or margin, between classes.

In the most trivial case, where the samples are linearly-separable, the hard-margin formulation can be used. Then, the margin from the plane to the solutions of each class is maximized to achieve a better generalization. Given some data $D = \{\mathbf{x}_i, y_i\}_{i=1}^n$ ($n \in \mathbb{N}$), where $y_i \in \{-1, +1\}$ indicates the class $\mathbf{x}_i \in \mathbb{R}^d$ ($d \in \mathbb{N}$) belongs to, the maximal separating hyperplane can be found by solving the following optimization problem:

$$\begin{aligned} & \min \left(\frac{1}{2} \mathbf{w} \mathbf{w}^T \right) \\ & \text{subject to} \\ & y_i (\mathbf{w} \mathbf{x}_i^T + b) \geq 1, \forall i \in \{1, 2, \dots, n\} \end{aligned} \tag{1}$$

where \mathbf{w} is the normal vector of the hyperplane, n refers to the number of samples in the dataset and b corresponds to a special parameter in SVMs, often called bias.

The label assigned to each new sample \mathbf{x}_* is determined by the following function:

$$y_* = \text{sgn}(\mathbf{w} \mathbf{x}_*^T + b) \tag{2}$$

where $\text{sgn}(a)$ returns $+1$ if a is positive, and -1 otherwise.

On the contrary, the soft-margin formulation allows linear SVMs to be used with non-linearly-separable data by introducing the hinge loss function ($L(\mathbf{w}, b) = \max(0, 1 - y_i(\mathbf{w} \mathbf{x}_i^T + b))$) with the error variable ζ_i :

$$\begin{aligned} & \min \left(\frac{1}{2} \mathbf{w} \mathbf{w}^T + C \sum_{i=1}^n \zeta_i \right) \\ & \text{subject to} \\ & y_i (\mathbf{w} \mathbf{x}_i^T + b) \geq 1 - \zeta_i \text{ and} \\ & \zeta_i \geq 0, \forall i \in \{1, 2, \dots, n\} \end{aligned} \tag{3}$$

where C is the regularization parameter. If its value is large, having a small hinge loss will be more important than having large margins. Therefore, SVMs will reduce the margin of the hyperplane in order to classify correctly as many training points as possible. On the other hand, if the value of C is small, increasing margins will be more important than reducing the hinge loss. Thus, SVMs

will assume some classification errors to have large margins. In the extreme case, when C is tiny, SVMs will behave similarly to the hard-margin case.

Equation (3) can be simplified by solving its Lagrangian dual:

$$\begin{aligned} & \max \left(\sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{x}_i \mathbf{x}_j^T \right) \\ & \text{subject to} \\ & \sum_{i=1}^n \alpha_i y_i = 0 \text{ and} \\ & C \geq \alpha_i \geq 0, \forall i \in \{1, 2, \dots, n\} \end{aligned} \tag{4}$$

In this dual formulation, α_i can be found by means of quadratic programming methods [27]. Then, \mathbf{w} and b can be calculated as follows:

$$\begin{aligned} \mathbf{w} &= \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i \\ b &= y_B - \mathbf{w} \mathbf{x}_B^T \end{aligned} \tag{5}$$

where \mathbf{x}_B and y_B are the values for a sample on the boundary of the margin.

2.1 Kernel trick

When data is not linearly-separable in the original space, there might be some feature space \mathcal{V} where a hyperplane can classify the data. In [4] a mapping of the data to a higher dimensional space was proposed, called the *kernel trick*.

A kernel function k can be defined as an inner product in some Hilbert space $k(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle_{\mathcal{V}}$, given a feature map $\phi : \mathbb{R}^d \rightarrow \mathcal{V}$. Thus, replacing the dot product operations in Equation (4) with the kernel function k is equivalent to mapping the data to the feature space \mathcal{V} and computing the SVM in such space, which allows non-linearly-separable classification problems to be solved. In addition, the quadratic programming problem required to find the optimal hyperplane is convex as long as the kernel function meets Mercer's condition [5]. To hold this condition, the kernel $k : \mathcal{S} \times \mathcal{S} \rightarrow \mathbb{R}$ on the set \mathcal{S} must satisfy:

$$\int_{\mathcal{S}} \int_{\mathcal{S}} g(x) k(x, x') g(x') dx dx' \geq 0 \tag{6}$$

for any square integrable function $g(x)$.

If k satisfies Equation (6), then the matrix M , where $m_{ij} = k(x_i, x_j)$, $\forall x_1, \dots, x_n \in \mathcal{S}$ and $\forall n \in \mathbb{N}$, is (i) symmetric, i.e., $M = M^T$, and (ii) a Positive semi-definite (PSD) matrix. A matrix is PSD if $\mathbf{u} M \mathbf{u}^T \geq 0$ for all real vectors $\mathbf{u} \in \mathbb{R}^n$, which is equivalent to saying that all its eigenvalues are non-negative.

2.2 Standard kernel functions

The kernel functions can be divided into two main families: stationary and non-stationary kernels [20].

A stationary kernel is translation invariant. Among the stationary kernels, we focus on isotropic kernels, as they are the most used kernel functions in the literature. Such kernels can be defined by the following equation:

$$k(\mathbf{x}, \mathbf{x}') = \widehat{k}(r)$$

$$r = \left\| \frac{\mathbf{x}}{\theta_l} - \frac{\mathbf{x}'}{\theta_l} \right\| \quad (7)$$

where \widehat{k} is a function that guarantees that the kernel satisfies Mercer's condition and θ_l is the lengthscale hyperparameter. The lengthscale hyperparameter can be also a vector that expresses the relevance of each dimension d , as suggested in Automatic Relevance Determination (ARD) approaches [32, 36].

On the contrary, in non-stationary kernels, the output of the kernel may vary with translation transformations of the input space. Within this family, the most common ones are those that depend on the dot product of the input vectors, which are usually referred to as dot-product kernels:

$$k(\mathbf{x}, \mathbf{x}') = \widehat{k}(s)$$

$$s = \left(\frac{\mathbf{x} - \theta_s \mathbf{1}}{\theta_l} \right) \left(\frac{\mathbf{x}' - \theta_s \mathbf{1}}{\theta_l} \right)^T \quad (8)$$

where θ_l is again the lengthscale hyperparameter, θ_s is the shift hyperparameter and $\mathbf{1}$ is a vector of ones.

Table 1 shows eleven standard kernels used in different applications of SVMs [25, 28, 12]. The Radial Basis Function (RBF) kernel, also known as the Squared Exponential kernel, is one of the most popular choices, and it is described as k_{RBF} in the table. This kernel is known to capture the smoothness property of the data.

2.3 Kernel and parameter setting overview

As explained in the previous section, the application of SVMs requires optimizing the weights (w) and bias (b), as well as setting the C parameter. Apart from these general parameters of SVMs, in kernel learning approaches, the kernel structure and its hyperparameters (Θ) must also be searched for. All these components are depicted in Figure 1.

In order to find the best kernel structure for a certain problem, all the components must be properly set. In this work we discuss the different kernel search methods and analyze the interplay between the SVM components.

Kernel function expressions	
Constant	$k_{CON}(\mathbf{x}, \mathbf{x}') = \theta_0$
White Noise	$k_{WN}(\mathbf{x}, \mathbf{x}') = \theta_0 \delta(\mathbf{x}, \mathbf{x}')$
Exponential	$k_E(r) = \theta_0^2 \exp(-r)$
γ exponential	$k_{E\gamma}(r) = \theta_0^2 \exp(-r^\gamma)$
RBF	$k_{RBF}(r) = \theta_0^2 \exp(-\frac{1}{2}r^2)$
Matern12	$k_{M12}(r) = \theta_0^2 \exp(-r)$
Matern32	$k_{M32}(r) = \theta_0^2 (1 + \sqrt{3}r) \exp(-\sqrt{3}r)$
Matern52	$k_{M52}(r) = \theta_0^2 (1 + \sqrt{5}r + \frac{5}{3}r^2) \exp(-\sqrt{5}r)$
Rat. Quadratic	$k_{RQ}(r) = \theta_0^2 (1 + \frac{1}{2\alpha}r^2)^{-\alpha}$
Periodic	$k_{PER}(r) = \theta_0^2 \exp\left(-\frac{2\sin^2(\pi r)}{\theta_p^2}\right)$
Linear	$k_{LIN}(s) = s$

Table 1: Standard kernel functions. θ_0 and θ_p are the kernel hyperparameters, called amplitude and period respectively. r is described in Equation (7), while s is presented in Equation (8).

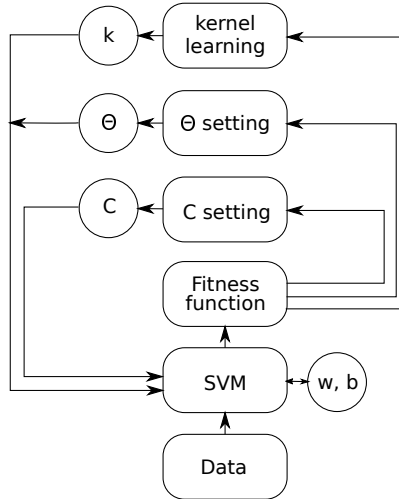


Figure 1: Kernel search diagram for SVMs. The elements that take part in the kernel search are shown in rectangles, while the associated parameters are displayed in circles.

3 Literature Review

An important choice in the SVM setting shown in Section 2.3 is the kernel function that will be used. In the early stages of SVM research, the standard kernel functions introduced in Table 1 were applied [35]. Most of the studies used expert knowledge to select the most suitable kernel among the standard ones,

although some authors also proposed cross-validation techniques [26] or decision trees [1] to carry out this selection. Once the kernel structure was selected, the hyperparameters were optimized to find the most appropriate ones depending on the characteristics of the problem [31], often using grid search techniques with k -fold cross-validation. Alternatively, more advanced techniques to optimize the lengthscale hyperparameter of the RBF kernel [30] or the C parameter [6] were proposed, avoiding the need of evaluating the SVM. Recently, the hyperparameter optimization problem has been analyzed from a multi-objective perspective, proposing evolutionary algorithms to obtain the Pareto optimal selection of the hyperparameters for Gaussian and polynomial kernel functions [38].

Then, methodological advances allowed the search for more complex kernel structures beyond the standard kernels. First, problem-specific kernel functions were manually developed. For example, in [54], an improved kernel function was proposed, based on the Gaussian, polynomial and sigmoid kernels functions. Later, automatic methods were proposed to compose, with no human intervention, ad-hoc kernels for particular problems. Although other techniques, such as deep-kernel-based architectures [7], have been proposed, Genetic Programming (GP) [29] has been one of the most used approaches for learning new kernels for SVMs. [25, 12, 28].

4 A key player: the kernel

Kernel design is usually done by combining basic modules, where a search procedure looks for the best performing combination. The two aspects which must be taken into account in this process are: (1) a grammar that includes the modules to be used and the rules for combining them, and (2) a search algorithm that defines the way the search is conducted. In this section we focus on the first aspect (modules and rules), and Section 5 is devoted to discussing some of the search methods that have been used to find the best kernel in such spaces.

As previously mentioned, the search space can be described by a grammar which specifies the building blocks of the kernels and the rules for combining them. Ideally, it is desirable to obtain a Mercer kernel, so that convergence to the optimal hyperplane is guaranteed. Nonetheless, defining a grammar which satisfies that all the kernels are Mercer kernels is not an easy task and, in fact, there is no proposal in the literature in this sense. The different grammars proposed in the literature can be classified depending on how they deal with the Mercer condition: kernel composition approaches and methods based on basic mathematical expressions.

In kernel composition approaches, such as [12] and [48], the grammar is usually composed of a set of kernels such as those shown in Table 1 together with some Mercer-condition-preserving composition rules (operations). These operations guarantee that, if the source kernel satisfies the Mercer's condition, the resulting kernels will also satisfy it [15, 14]. Although all the solutions created by means of kernel composition have convergence guarantees in SVMs, not all Mercer kernels can be explored through this method. Moreover, the

search may end up with extremely complex structures that can be too cost-intensive to evaluate depending on the application domain [48].

On the other hand, the second type of approaches are based on using basic mathematical expressions as building blocks. An example is shown in Figure 2, where a RBF kernel is represented as a mathematical expression tree. Note that these grammars may also contain the hyperparameters of the kernel, which allow the search to explore more flexible kernels, which are able to adapt to different scenarios by means of modifying their hyperparameters.

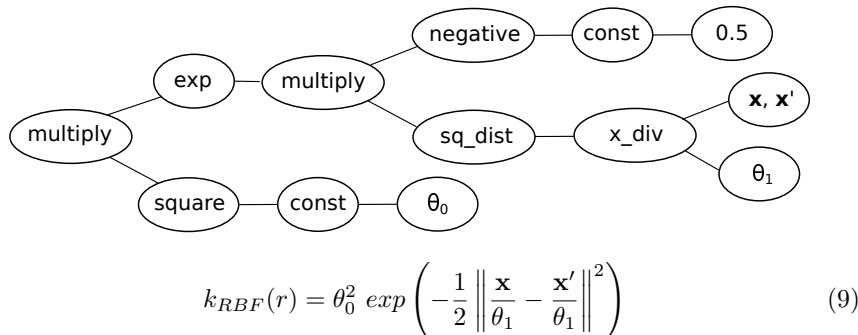


Figure 2: RBF kernel represented as a basic mathematical expression tree.

These proposals are more flexible and potentially better because, in addition to allowing the design of the kernels that can be constructed by means of kernel composition grammars, they allow a richer and wider set of kernels to be explored, built from scratch, without any previous bias.

On the contrary, in approaches based on the use of basic mathematical expressions, the dimensionality of the search space is clearly higher than in kernel composition approaches, which could make it more difficult to find good-performing kernels.

Another undesirable downside of being able to compose more compact and flexible kernels is that these methods may generate non-Mercer kernels during the search. The simplest method to deal with this problem is simply not to guarantee that kernels meet Mercer’s condition [3, 24, 19, 49]. As a result, the optimization algorithm used to find the optimal hyperplane for SVMs may not converge. On the contrary, the authors of [28], [10] and [25] checked Mercer’s condition for every kernel and those that do not meet this condition are penalized or discarded. In [10] and [25], a method for penalizing (giving the worst possible fitness value) the non-Mercer kernels at evaluation time is proposed. Besides, in [28], if during the random generation a kernel does not meet this condition, it is discarded and the process is repeated.

4.1 Relevance of periodic elements

Depending on the classification problem that is being solved by means of SVMs, some elements of the grammar might be crucial. The absence of certain elements may limit the result. However, most of the kernel learning approaches do not single out as relevant issues the grammar nor the choice of the elements that compose it.

In order to illustrate the importance of including the appropriate elements in the grammar, we compare the classic RBF kernel to the Periodic kernel (PER) in a subset of well-known problems. The RBF kernel is known for capturing the smoothness property of the data, as elements close to each other in terms of Euclidean distance have a high kernel value and it smoothly decreases as the distance increases. The elements needed to compose the RBF kernel are present in most of the grammars reported in the kernel learning literature, regardless of the grammar type choice, whether they use kernel composition or are based on basic mathematical expressions. On the other hand, the Periodic kernel is based on the RBF kernel but adds a spectral transformation to the space [23] in order to model periodic patterns in the data. Periodic elements, such as the spectral transformation, have been overlooked in the kernel search literature, and only some approaches [3] include them in their grammars.

If the Periodic kernel obtains a better result than the RBF, it may indicate that some periodic patterns are present in the data. If so, having the spectral transformation in the grammar is essential to achieve good results.

Although some of the kernel search approaches have been used to solve particular problems, most works in the literature test their proposals in the UCI classification datasets [13], shown in Table 2, in order to compare their results with those reported in previous works.

Classification problem	Samples	Variables	Classes
pima	768	8	2
ionosphere	351	34	2
heart statlog	270	13	2
glass2	163	9	2
liver disorder	345	6	2
breast cancer Wisconsin	569	30	2

Table 2: Characteristics of the UCI problems studied in this work.

In these UCI datasets, the RBF kernel has a reasonably good performance, close to the state-of-the-art kernels created by composition [12]. It might indicate that modeling the smoothness property of the data is enough to achieve good classification results [16].

To widen the scope of the analysis, we searched for other types of classification problems, where new kernel properties, apart from smoothness, could be necessary to obtain more accurate results. In a preliminary experiment, we used the Penn Machine Learning Benchmarks (PMLB) [37] to find datasets where

RBF does not perform so well, possibly indicating that other kernel properties are needed.

After evaluating the SVMs with the RBF kernel in all the PMLB databases, we selected the 8 problems where the lowest accuracy was obtained. The characteristics of these datasets are shown in Table 3. In the non-binary PMLB problems, the one-vs-one approach was used.

Classification problem	Samples	Variables	Classes
calendarDOW	399	31	5
contraceptive	1473	9	3
GAMETES Epistasis 0.1H	1600	19	2
GAMETES Epistasis 0.4H	1600	19	2
GAMETES Heterogeneity 50	1600	19	2
GAMETES Heterogeneity 75	1600	19	2
parity5+5	1124	10	2
Hill Valley with noise	1212	100	2

Table 3: Characteristics of the PMLB problems for which the RBF kernel obtained the worst accuracies.

In order to compare the results of the RBF kernel to the Periodic kernel, we ran a second experiment with the Periodic and RBF kernels in the previously shown UCI and PMLB databases.

The dataset was partitioned twice. A random fold of 20% of the data is selected as the test set, and a 4-fold cross-validation was used to set C and the hyperparameters for each kernel. The SVMs were fitted in each fold of the training set for each combination of C (2^{-5} to 2^{15} , at powers of 2^2 as in [47]) and the hyperparameters (2^{-5} to 2^4 , at powers of 2), with a limit of 1000 evaluations. Then, the combination with the best average accuracy was selected to be evaluated in the test set.

In Table 4 the results of this experimentation are shown. Although in the UCI datasets the results of both kernels are similar, important performance gains can be obtained in the *GAMETES* PMLB datasets when using the periodic kernel instead of the RBF.

In spite of being very similar kernels, there are remarkable performance differences between the RBF and Periodic kernels depending on the database. The limited capacity of the RBF kernel to model the *GAMETES* databases restricts the classification performance of the SVMs. This suggests that including the elements of the Periodic kernel in the kernel search grammar might be crucial, and highlights the importance of a careful selection of the elements that compose the grammar.

4.2 Proposed grammar

In order to investigate the importance of the selected search space, and taking into account the grammars proposed in the literature, we designed a grammar

	Classification problem	RBF	PER
UCI	pima	0.772	0.758
	ionosphere	0.939	0.944
	heart statlog	0.826	0.837
	glass2	0.809	0.773
	liver disorder	0.743	0.726
	breast cancer Wisconsin	0.972	0.974
PMLB	calendarDOW	0.621	0.624
	contraceptive	0.547	0.548
	GAMETES Epistasis 0.1H	0.562	0.668
	GAMETES Epistasis 0.4H	0.709	0.797
	GAMETES Heterogeneity 50	0.660	0.714
	GAMETES Heterogeneity 75	0.669	0.701
	parity5+5	0.931	0.905
	Hill Valley with noise	0.815	0.801

Table 4: Mean accuracies in the test set for the RBF kernel and the Periodic kernel in UCI and PMLB classification problems. The numbers in bold indicate the best result for each problem. UCI databases are shown in the top 6 rows of the table, while PMLB problems are at the bottom of the table.

based on basic mathematical expressions by means of which all the kernels of Table 1 can be composed. The production rules of this grammar are shown in Table 5.

The *scalar* non-terminal is the start symbol of the grammar. It also includes the $+$, \times , and $^{\wedge}$ arithmetic operators, with their usual meanings (addition, product and power, respectively). Note that we only allow hyperparameters as the exponent in the power operator. The same interpretation is given to the unary operators. The power to the minus one is also added as an unary operator in order to allow division operations. Then, the input vectors are converted into scalars by means of the square distance and dot product non-terminals, as described in Section 2.2. Similarly, constant and noise non-terminals, whose values depend on the input hyperparameter, are included. The subtraction and the division of an input vector by a hyperparameter are also incorporated. In addition, the grammar also contains the spectral transformation, in order to allow periodic kernels, as in [23]. Finally, $(\mathbf{x}, \mathbf{x}')$ (the input vectors of the kernel) and θ (the hyperparameters) are the terminals of this grammar.

4.2.1 Random kernel generation

In order to randomly generate kernel expressions, we propose a strongly-typed grow method based on the work presented in [29]. This approach creates kernels from scratch, without any knowledge of previously proposed kernels. This is achieved by a recursive process where, at each step, a random terminal or a random operator is added.

<i>kernel</i> :	<i>scalar</i>	start symbol
<i>scalar</i> :		
	$scalar^{hp}$	power
	$scalar + scalar$	add
	$scalar \times scalar$	multiply
	$scalar^{-1}$	div
	e^{scalar}	exp
	\sqrt{scalar}	sqrt
	$scalar^2$	square
	$- scalar$	negative
	$\tanh(scalar)$	tanh
	$\ \mathbf{invec} - \mathbf{invec}'\ ^2$	sq_distance
	$\mathbf{invec} \cdot \mathbf{invec}'^T$	dot_product
	hp	constant
	$hp \times \delta(\mathbf{x}, \mathbf{x}')$	noise
	;	
$(\mathbf{invec}, \mathbf{invec}')$:		
	$([\sin(\mathbf{invec}) \quad \cos(\mathbf{invec})],$	
	$[\sin(\mathbf{invec}') \quad \cos(\mathbf{invec}')])$	spectral
	$(\frac{\mathbf{invec}}{hp}, \frac{\mathbf{invec}'}{hp})$	x_div
	$(\mathbf{invec} - \mathbf{1}hp, \mathbf{invec}' - \mathbf{1}hp)$	x_rest
	$(\mathbf{x}, \mathbf{x}')$	input
	;	
hp :		
	0.5 1 2 3 5	
	θ_0 θ_1 ... θ_t	
	;	

Table 5: Proposed grammar for SVMs. t indicates the number of different hyperparameters allowed in the grammar (in this work, it is set to $t = 20$).

When generating random solutions, some of the solutions may be too complex in terms of the number of elements in the expression, and others too simple or trivial. Thus, we propose a method to control the depth of the generated expressions by setting a minimum (d_{min}) and a maximum depth (d_{max}). As can be seen in Table 5, some of the non-terminals have the same symbol in both sides of the production rule. These non-terminals guarantee that, once selected, the iterative procedure can continue growing this branch, i.e., they are recursive.

As can be seen in Algorithm 1, during the creation process, we select a uniformly random production rule depending on the current symbol. If the minimum depth has not been reached, only recursive non-terminals are used. Then, until the maximum depth is reached, any non-terminal can be selected. Finally, when the maximum depth is reached, only the terminals and the non-

recursive non-terminals are used, limiting the depth of the expression.

Algorithm 1 Random Generation of expression trees

```

1: procedure TYPEDGROW( $d_{min}, d_{max}, type$ )
2:    $termexprs = \text{GETTERMINALEXPRES}(type)$ 
3:    $nonreexprs = \text{GETNONRECURSIVEEXPRES}(type)$ 
4:    $reexprs = \text{GETRECURSIVEEXPRES}(type)$ 
5:    $candexprs = \emptyset$ 
6:   if  $d_{min} \leq 2$  then
7:      $candexprs = candexprs \cup nonreexprs \cup termexprs$ 
8:   end if
9:   if  $2 \leq d_{max}$  then
10:     $candexprs = candexprs \cup reexprs$ 
11:  end if
12:  if  $candexprs$  is  $\emptyset$  then ▷ No candidate expressions
13:     $candexprs = terms \cup notnests \cup nests$ 
14:  end if
15:   $expr = \text{RANDOMCHOICE}(candexprs)$ 
16:   $inputtypes = \text{GETINPUTTYPES}(expr)$ 
17:  if  $inputtypes$  is  $\emptyset$  then ▷  $expr$  is terminal
18:    return  $expr$ 
19:  end if
20:  for  $inputtype$  in  $inputtypes$  do
21:     $subexpr = \text{TYPEDGROW}(d_{min} - 1, d_{max} - 1, inputtype)$ 
22:     $expr = \text{APPEND}(expr, subexpr)$ 
23:  end for
24:  return  $expr$ 
25: end procedure

```

4.2.2 Dealing with non-Mercer kernels

In order to mitigate the evaluation of non-Mercer kernels, we followed the approach used by the authors of [28]: check the positive definiteness of the matrix generated by a kernel for some random data and attempt the generation of the kernel again if this matrix is not PSD.

As mentioned in Section 2.1, any matrix generated by a Mercer kernel has to be symmetric and also PSD. To identify non-Mercer kernels, we generate w random uniformly distributed datasets $X = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)$ (where $\mathbf{x}_i \in \mathbb{R}^d$, $i \in \{1, \dots, n\}$ and $n \in \mathbb{N}$) and check the M matrix produced by the kernel for each dataset. If any M matrix matches the following cases, the generation process of the kernel is repeated:

- $M \neq M^T$: As previously mentioned, the matrix given by a Mercer kernel should be symmetric.

- Any m_{ii} is negative: It has been proved [53] that, if any of the elements in the main diagonal are negative, the matrix is not PSD.
- Any of the eigenvalues of M is negative: Similarly, all the eigenvalues of the matrix should be non-negative.

With this method, and using the grammar shown in Table 5, 9 trials were required on average in the experiments conducted in this work to create a kernel that generates a PSD matrix. Note that meeting these conditions is not sufficient for a kernel to be Mercer. Although some non-Mercer kernels may pass the PSD check, it was not a problem during the experimentation, as the SVM optimization did not converge to the global optimum only with 3.16% of the kernels that passed the PSD check.

4.3 Increasing grammar experiment

Following the experiment introduced in Section 4.1, we conducted a more in-depth experiment to observe the influence of the grammar in the search of the best kernel for the SVM classification. Particularly, we wanted to measure the effect that the addition of certain elements to the grammar has in randomly generated kernels and in the performance of the SVMs that uses these kernels.

In order to carry out this experiment we obtain a series of grammars that are able to create kernels that can generate PSD matrices and each grammar in the sequence comprises all the elements from the previous one. We start with the minimum possible grammar, including the input vector and the dot product operator from which only a simplified version of the Linear kernel can be created. To create the next grammar, we add one random element to the first one, and test whether a random kernel generated with this new grammar, which contains this new element, passes the PSD check. $d_{min} = 5$ and $d_{max} = 15$ were set to control the depth of the generated expressions. If the randomly generated kernel fails the test, we try adding another element to the grammar. If after testing all the elements, none of the new grammars is able to generate a random kernel that passes the PSD check, we try to add pairs of elements to the grammar. On the other hand, if the new grammar is able to generate random kernels that pass the PSD check, this new grammar is added to the sequence, and the process is repeated adding a new element to it. We repeat this experiment obtaining 10 series of 27 grammars for each UCI and PMLB database. Then, for each grammar in the sequence, we randomly generate 18 kernels and evaluate them. The evaluation of these kernels is carried out following the same setting used in Section 4.1.

Figure 3 illustrates one of these experiments in the *GAMETES Epistasis 0.4H* database. It can be seen that, when including certain elements in the grammar, the accuracy of SVMs increases. For example, when the *spectral* non-terminal is included, in combination with the exponential, 60% accuracy can be achieved. Moreover, in the 16th iteration, the inclusion of the *multiplication* non-terminal improves the accuracy up to 74%. In the following iterations,

the *sq_distance* non-terminal slightly increases the performance when selected. Overall, the performance of the best kernel of each iteration (created by means of a richer grammar) shows an increasing trend.

To obtain a general view of the experiment, in Figure 4 the average accuracy for each iteration and database is shown. In some problems, such as *pima*, *heart statlog* and *breast cancer Wisconsin* the results of the first iteration, i.e., the simplified Linear kernel, can not be improved with the addition of new elements. This is consistent with the results of Table 4 where the rest of the standard kernels barely outperform the results of the Linear kernel. However, for the rest of the problems, there is a clear increase in accuracy when new elements are added to the grammar. This is especially visible in the *glass2*, *parity5+5* and *GAMETES* databases.

In summary, the increasing grammar experiment shows that including a wide set of elements in the grammar is beneficial for any kernel structure search attempt. Thus, our first recommendation would be to include as many elements described in the literature as possible. Particularly, we have observed in the experiment of Section 4.1 that the spectral element is very important for some particular problems. A possible drawback derived from a very rich grammar would be a wider search space, which makes it more difficult for a search algorithm to find a good performing combination (kernel). However, as it is discussed in Section 5, even a basic random search algorithm with a limited budget is able to provide competitive results.

5 Kernel structure search

The main component in the SVM kernel search methods is the optimization of the structure of the kernel itself. GP has been one of the most widely used approaches when addressing these optimization tasks. GP is an evolutionary algorithm designed to search in a predefined space of computer programs, i.e., kernel functions in our case. By means of this technique, kernels have been obtained that produce better results in terms of accuracy than standard kernels [24, 28].

However, there is a lack of knowledge on many aspects related to the specific characteristics of the kernel function optimization problem, and in particular about how these characteristics relate to the way the GP search for optimal solutions is accomplished. Furthermore, there is not a clear understanding of the relative performance of GP to other simpler search strategies since other optimization methods have only rarely been applied to the kernel function optimization problem.

5.1 Kernel structure search experiment

Once the importance of the grammar and the search space has been introduced, we then focus on the kernel structure search step. For this purpose, we designed an experiment to compare the performance of GP with other kernel structure

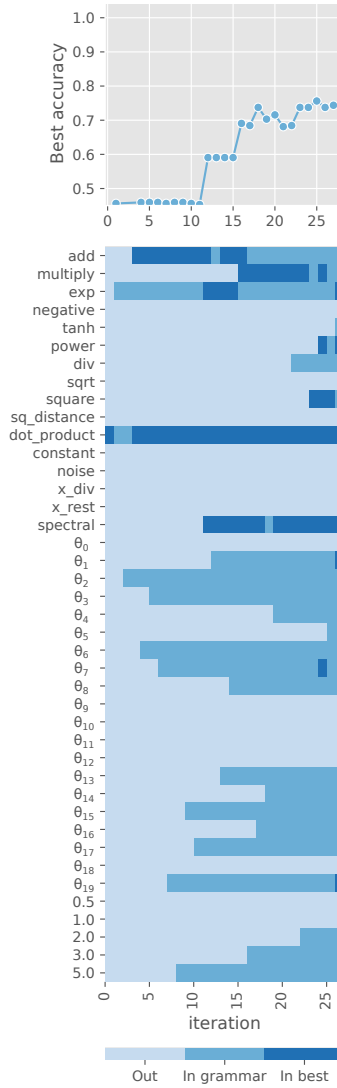
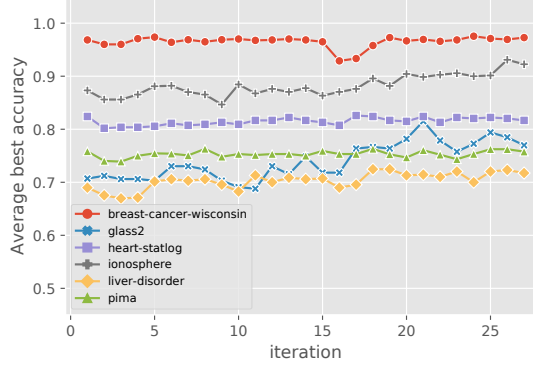
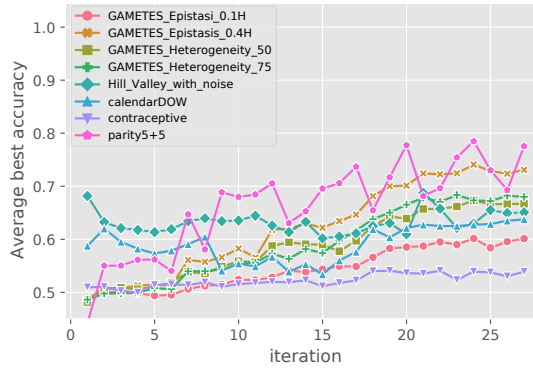


Figure 3: Increasing grammar experiment in the *GAMETES Epistasis 0.4H* dataset. The figure at the bottom shows the elements present in the grammar at each iteration. The lightest blue color indicates that an element is out of the grammar, while the darker blue color expresses that this element is included. If the best kernel of that iteration contains a certain element the darkest blue color is shown. At the top, the accuracy on the test set of the best random kernel (selected according to the training set) is plotted.



(a) UCI



(b) PMLB

Figure 4: Average accuracy on the test set of the best kernel (selected according to the training set) at each iteration.

search approaches, including also the standard kernels shown in Table 1, for the UCI and PMLB databases.

The GP method studied in this experiment is based on the mathematical expression grammar introduced in Section 4.2. As shown in Algorithm 2, in this approach, an initial population of N random kernels is generated with a minimum (d_{min}) and a maximum depth (d_{max}), and evaluated. After selecting the S best individuals, the algorithm randomly chooses between a mutation or a crossover operator (with probability p_m and p_{cx} respectively, where $p_{cx} = 1 - p_m$) to generate an offspring population of N new individuals. After evaluating all the individuals in this offspring population, the previously selected individuals are added to generate the next population that consists of $N + S$ individuals. This procedure is repeated for G generations, until the last population is evaluated and the best individual found during the whole process is returned.

Algorithm 2 GP algorithm for SVM kernel learning

```
1: procedure GP( $N, G, S, p_m, p_{cx}, d_{min}, d_{max}$ )
2:    $pop = \text{GENRANDPOP}(N, d_{min}, d_{max})$ 
3:    $\text{EVALUATE}(pop)$ 
4:    $all = pop$ 
5:    $i = 0$ 
6:   while  $i < G - 1$  do
7:      $sel = \text{SELECT}(pop, S)$ 
8:      $offspring = \text{VARIATE}(sel, N, p_m, p_{cx})$ 
9:      $\text{EVALUATE}(offspring)$ 
10:     $all = all \cup offspring$ 
11:     $pop = sel \cup offspring$ 
12:     $i = i + 1$ 
13:  end while
14:   $best = \text{SELECT}(all, 1)$ 
15:  return  $best$ 
16: end procedure
```

In order to assess the contribution that each component of the proposed GP algorithm makes to the kernel search, we introduce three algorithms to be used as a baseline in the experiments.

First, we describe a random search algorithm that generates N kernels following the method described in Section 4.2.1. Next, the best solution is chosen according to the cross validated accuracy in the training set.

Secondly, in order to measure the gain produced by the crossover operator in the GP setting, we propose a hill-climbing algorithm, which does not make use of this operator. This procedure generates an initial kernel, from which a second kernel is created by applying a random mutation. Then, the best kernel in terms of accuracy is selected. This procedure is repeated for N evaluations.

Finally, we also introduce a GP variant, without the spectral element in the grammar (sGP), in order to develop the results of the experiment shown in Section 4.1.

All these algorithms were coded in Python, based on the EA software *DEAP*¹ [17], and made publicly available in the Python Package Index². The code used to carry out the SVM classification with evolved kernels has been also published³.

In all these approaches, before the evaluation of every kernel, the hyperparameters and C were optimized as in the experiment in Section 4.1. In the kernel structure search approaches, the hyperparameters were set according to the grammar shown in Section 4.2 and optimized in a grid of 2^{-5} to 2^4 , at powers of 2, with a limit of 1000 evaluations. In order to find the hyperparameters

¹<https://deap.readthedocs.io>

²<https://pypi.org/project/evocov/>

³<https://pypi.org/project/ksvmllib/>

	Classification problem	LIN	M32	M52	RBF	PER
UCI	pima	0.779	0.781	0.782	0.783	0.785
	ionosphere	0.889	0.956	0.957	0.956	0.958
	heart-statlog	0.859	0.859	0.859	0.860	0.869
	glass2	0.715	0.809	0.813	0.813	0.855
	liver-disorder	0.697	0.737	0.742	0.743	0.747
	breast-cancer-Wisconsin	0.981	0.984	0.984	0.983	0.984
PMLB	calendarDOW	0.592	0.622	0.625	0.627	0.651
	contraceptive	0.516	0.564	0.565	0.565	0.572
	GAMETES_Epistasi_0.1H	0.501	0.576	0.578	0.582	0.684
	GAMETES_Epistasis_0.4H	0.503	0.678	0.686	0.690	0.794
	GAMETES_Heterogeneity_50	0.502	0.641	0.647	0.650	0.716
	GAMETES_Heterogeneity_75	0.517	0.659	0.664	0.665	0.736
	parity5+5	0.497	0.555	0.563	0.632	0.722
	Hill_Valley_with_noise	0.866	0.833	0.828	0.820	0.835

Table 6: Results of the Kernel search experiment in the training set for the standard kernels. The mean accuracy achieved by each kernel is shown. The numbers in bold indicate the best result for each problem while taking into account the results of the structure search methods shown in Table 7.

of the standard kernels (Their descriptions can be found in the Table 1), we used a more exhaustive grid search for a fair comparison: 2^{-5} to 2^4 , at powers of $2^{0.1}$, with a limit of 486000 evaluations. In the GP approach, in each of the $G = 27$ generations, $N = 18$ kernels were created and the best $S = 4$ kernels were chosen as seeds for new individuals. The mutation and crossover probabilities were set to $p_m = 0.4$ and $p_{cx} = 0.6$ respectively. Similarly, $N = 486$ evaluations were carried out in the random search and hill-climbing methods. Each configuration was repeated 10 times.

As can be seen in tables 6 and 7, the GP approach improves the training set results of the standard kernels in all the UCI and PMLB databases, except in the *GAMETES* problems, where it is not able to achieve the same accuracy as the Periodic kernel. Notable performance gains were achieved in the *glass2*, *parity5+5* and *Hill Valley with noise* problems by using the GP method. The hill climbing and the random search methods obtain results similar to the best standard kernel in most of the databases. Comparing GP to the random search and the hill climbing methods, the best average accuracies are achieved by GP.

The results obtained in the training set by the different models, are used to determine the best performing model in the test set. However, the performance of the models changes when applied to the test set, probably due to the overfitting effect. Thus, in spite of obtaining good accuracy values in the training set, the GP approach is not able to maintain those results in the test set. Tables 8 and 9 show that this issue is clearly visible in most databases where slight differences were observed in the training set between the GP and the best standard kernels. However, in other datasets, such as *glass2*, *parity5+5* and *Hill*

	Classification problem	Random	HC	sGP	GP
UCI	pima	0.788	0.788	0.793	0.793
	ionosphere	0.955	0.958	0.962	0.962
	heart-statlog	0.876	0.878	0.878	0.878
	glass2	0.860	0.862	0.836	0.909
	liver-disorder	0.745	0.750	0.756	0.763
	breast-cancer-Wisconsin	0.984	0.985	0.986	0.986
PMLB	calendarDOW	0.651	0.660	0.642	0.665
	contraceptive	0.569	0.570	0.569	0.573
	GAMETES_Epistasi_0.1H	0.679	0.680	0.596	0.681
	GAMETES_Epistasis_0.4H	0.791	0.793	0.700	0.793
	GAMETES_Heterogeneity_50	0.704	0.706	0.667	0.709
	GAMETES_Heterogeneity_75	0.725	0.732	0.673	0.733
	parity5+5	0.957	0.938	0.990	0.999
	Hill_Valley_with_noise	0.796	0.817	0.886	0.905

Table 7: Results of the Kernel search experiment in the training set for the kernel structure search methods. The mean accuracy achieved by each kernel search method is shown. The numbers in bold indicate the best result for each problem. sGP indicates the spectral-less GP approach.

	Classification problem	LIN	M32	M52	RBF	PER
UCI	pima	0.759	0.766	0.773	0.766	0.765
	ionosphere	0.875	0.942	0.945	0.942	0.944
	heart-statlog	0.830	0.820	0.824	0.820	0.815
	glass2	0.710	0.818	0.800	0.794	0.800
	liver-disorder	0.684	0.722	0.732	0.735	0.713
	breast-cancer-Wisconsin	0.973	0.969	0.971	0.970	0.971
PMLB	calendarDOW	0.577	0.620	0.621	0.623	0.619
	contraceptive	0.521	0.555	0.553	0.550	0.548
	GAMETES_Epistasi_0.1H	0.467	0.561	0.562	0.559	0.675
	GAMETES_Epistasis_0.4H	0.489	0.697	0.708	0.717	0.797
	GAMETES_Heterogeneity_50	0.482	0.648	0.651	0.653	0.721
	GAMETES_Heterogeneity_75	0.488	0.665	0.670	0.668	0.720
	parity5+5	0.474	0.542	0.718	0.892	0.865
	Hill_Valley_with_noise	0.819	0.849	0.841	0.825	0.837

Table 8: Results of the Kernel search experiment in the test set for the standard kernels. The mean accuracy achieved by each kernel is shown. The numbers in bold indicate the best result for each problem while taking into account the results of the structure search methods shown in Table 9.

Valley with noise, the GP kernel learning method is clearly a better choice in the training set, and these results are also visible in the test set.

GP achieves better average accuracy values in the test set than the other

	Classification problem	Random	HC	sGP	GP
UCI	pima	0.762	0.763	0.756	0.759
	ionosphere	0.945	0.945	0.944	0.946
	heart-statlog	0.806	0.811	0.815	0.802
	glass2	0.773	0.803	0.776	0.836
	liver-disorder	0.712	0.719	0.714	0.719
	breast-cancer-Wisconsin	0.975	0.969	0.972	0.973
PMLB	calendarDOW	0.626	0.619	0.609	0.616
	contraceptive	0.547	0.547	0.553	0.552
	GAMETES_Epistasi_0.1H	0.676	0.675	0.560	0.675
	GAMETES_Epistasi_0.4H	0.796	0.796	0.722	0.796
	GAMETES_Heterogeneity_50	0.719	0.719	0.665	0.712
	GAMETES_Heterogeneity_75	0.714	0.713	0.672	0.708
	parity5+5	0.985	0.965	0.998	1.000
	Hill_Valley_with_noise	0.786	0.818	0.865	0.910

Table 9: Results of the Kernel search experiment in the test set for the kernel structure search methods. The mean accuracy achieved by each kernel search method is shown. The numbers in bold indicate the best result for each problem. sGP indicates the spectral-less GP approach.

kernel structure search methods, in the *GAMETES Epistasis 0.4H* and *contraceptive* databases. In some other problems, such as *pima*, *heart statlog*, *liver disorder* and *GAMETES Epistasis 0.4H* problems, the exploitation oriented mutation operator of the hill climbing algorithm generates the best kernels for the test set. On the other hand, the exploration oriented behavior of the random search seems to be less prone to overfit, achieving the best results in the *breast cancer Wisconsin*, *calendarDOW* and two *GAMETES Heterogeneity* problems.

Besides, the GP approach with the spectral element in the grammar shows a better performance than the spectral-less variant in the training set in all the problems, especially in the *glass2*, *calendarDOW*, *Hill Valley with noise* and *GAMETES* problems. In tables 8 and 9, it can be seen that these differences are also notable in the test set. These problems probably include some periodic patterns that can be better modeled when the spectral element is present.

We conducted a statistical test to assess the existence of significant differences among the methods in the test set. For each database, we applied Friedman’s test [18] and we found significant differences ($\alpha = 0.05$) in the *ionosphere*, *glass2*, *contraceptive*, *parity5+5* and all *GAMETES* databases (p-values can be seen in Figure 5). Then, for each configuration, we applied a post-hoc test based on Friedman’s test as in [9], and adjusted the results with Shaffer’s correction [46]. The results are shown in Figure 5, and in Table 10, where a summary of the statistical tests is presented.

Overall, the GP method is the best performing approach, obtaining significantly better results than the Linear and Matern kernels in some problems. On the contrary, there are not many statistical differences between the struc-

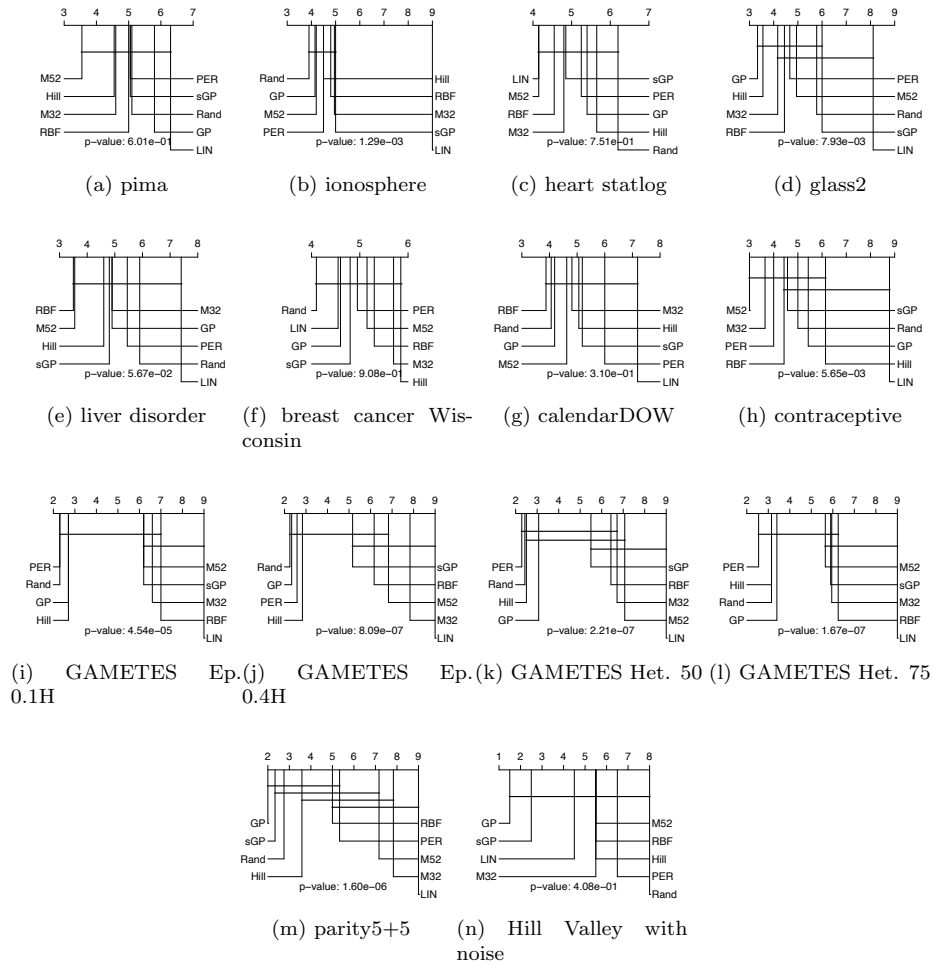


Figure 5: Critical difference diagrams in UCI and PMLB datasets. Search methods are ordered following their rankings. The methods with no significant differences among them are matched with a straight line.

	LIN	RBF	M32	M52	sGP	PER	Hill	Rand	GP	Worse
LIN	0	1	2	2	2	6	7	6	7	33
RBF	0	0	0	0	0	0	0	0	0	0
M32	0	0	0	0	1	1	1	2	2	7
M52	0	0	0	0	0	1	0	1	1	3
sGP	0	0	0	0	0	0	0	0	0	0
PER	0	0	0	0	0	0	0	0	0	0
Hill	0	0	0	0	0	0	0	0	0	0
Rand	0	0	0	0	0	0	0	0	0	0
GP	0	0	0	0	0	0	0	0	0	0
Better	0	1	2	2	3	8	8	9	10	

Table 10: Summary table of the statistical testing. The number of databases where the method in the column is significantly better than the method in the row is shown.

ture search methods. Among the standard kernels, the periodic kernel shows significantly better results than the linear kernel in 6 databases, and improves the performance of Matern32 in *GAMETES Epistasis 0.4H*, and Matern52 in *GAMETES Heterogeneity 50*.

Although the GP approach improves the results of the standard kernels in the training set, and it has a better exploration-exploitation balance than random search and hill climbing, these results cannot be transferred to the test set, probably due to overfitting issues. In the test set, there are no significant differences between the GP approach and the simpler kernel structure search methods. It is also important to notice that a single change in the grammar can produce a greater impact in the results than the search method itself, as in the *GAMETES* problems, where the average differences between the results of the GP with and without the spectral element are higher than the gap between the hill climbing and the standard GP approach.

As a final note, we can question the importance of the kernel search method compared to the importance of selecting an appropriate search space. According to the experiments, the GP method shows the best results. Nevertheless, the absence of statistical differences with the random search suggests that the efforts of the practitioners should focus on the design of an adequate search space rather than on the design of the best possible search algorithm. It is also worth mentioning the small differences we found between the training and test results. Not having a measure of complexity of the models in the kernel learning approaches has probably generated models that are too dependent on the training set.

6 Hyperparameter and C optimization

We have shown that in SVMs there are several variables to optimize apart from the kernel structure, such as the kernel hyperparameters and the C parameter. Hyperparameters, being part of the kernel, change the transformed space, while C parameter balances the trade-off between increasing the margin and assuming greater hinge loss. In this section, we review the literature about the C parameter and hyperparameter setting and investigate the interplay of these variables for several kernels.

6.1 Hyperparameter setting

During the kernel learning process, kernel hyperparameters must be carefully set. A change in the hyperparameters can be as relevant as a change in the structure of the kernel. These hyperparameters clearly influence the results of the kernel function, and therefore, the performance of SVMs.

In the initial kernel learning approaches, the hyperparameters were not even included in the grammar [24, 25, 10, 49]. In other methods, random constants were incorporated to the grammar, which can be interpreted as hyperparameters that are learned together with the structure [48, 39, 21, 2, 19, 47]. Alternatively, hyperparameters can be also learned apart from the structure in a secondary optimization procedure. The most common hyperparameter optimization method is grid search [22, 12, 28, 34, 11], although more complex methods have been also tried, such as particle swarm optimization [44]. As can be seen, choosing the right hyperparameters for the kernel remains an open question.

6.2 C parameter setting

The value of C also influences the evaluation of the quality of the kernels generated during the learning process. The simplest approach to fairly compare the kernels is to set a constant value for the C parameter ($C = c$) [10, 22]. However, this approach also creates a bias in the kernel selection process to that constant value of C .

The opposite approach is to run an exhaustive search in a reduced set of values [28]. Here, a kernel-performance-maximizing C is selected for each of the visited kernels, increasing the computational cost of the search. We can classify these approaches depending on the method used to deal with the optimization of C , along with the search of a kernel and its hyperparameters: (i) the approaches that use a nested search procedure, where we optimize C for each kernel structure and hyperparameters visited in the search (ii) the methods that optimize C together with the kernel hyperparameters [28] and (iii) the algorithms that optimize C together with the kernel as a parameter of the kernel itself [47].

Finally, the C parameter can be selected based on the characteristics of the evaluation of the kernel in the data as shown in [6]. For each kernel, a good value of C is approximated while reducing the evaluation cost similar to the fixed case.

Analogous to the hyperparameter tuning problem, the C parameter setting poses many questions when searching for the most suitable kernel. There is a clear interplay between these parameters, and also a trade-off between quality and computational cost.

6.3 Interplay between optimization procedures

By means of the following experiment, we would like to investigate the interaction between the kernel hyperparameters and the C parameter. Particularly, we analyze the performance and the overfitting of SVMs with different kernel hyperparameters and C parameter values for several kernel structures in the mentioned UCI and PMLB datasets.

We have selected some of the standard kernels shown in Table 1. For each kernel, three hyperparameter configurations are tested: a set of default hyperparameters ($\theta_i = 1$), a random set of hyperparameters, and an optimized set of hyperparameters according to a grid search (2^{-5} to 2^4 , at powers of 2, with a limit of 1000 evaluations) that maximizes the accuracy in the training set for the C value proposed in [6]. Furthermore, for each kernel and hyperparameter configuration, 20 values (2^{-5} to 2^{15} , at powers of 2^2) for C are tried apart from the C value proposed in [6].

In Figures 6 and 7, the results of the experiment are shown. For each database, the accuracy in the test set is represented by a heatmap. In the X axis the different values of C are shown, while in the Y axis the different kernels and their hyperparameters can be seen.

The best results are achieved with the optimized hyperparameters. This can be clearly seen in the *parity5+5* database, where the Periodic, RBF and Matern52 kernels obtain their best results when optimizing their hyperparameters. Regarding the influence of C , note that lower values of C show a lower performance in almost every configuration. Although there are some databases, such as *breast cancer Wisconsin* or *contraceptive*, where the C parameter has very little influence, in the rest of the problems certain C values are required to achieve the best possible performance. Also, it is worth mentioning that configurations with optimized hyperparameters show a more consistent performance for different values of C . Finally, it can be seen that the C value suggested in [6] shows good accuracy values overall.

On the whole, there is a clear influence of the kernel structure and hyperparameters in the results, but also the C parameter can drastically change the quality of the prediction. In order to set the values of the hyperparameters and the C parameter, a grid search is highly recommended due to their strong interactions. Searching the hyperparameters on a grid and using the data based approach shown in [6] to set the C parameter could be a good approximation in the cases where the exhaustive search is not computationally affordable.

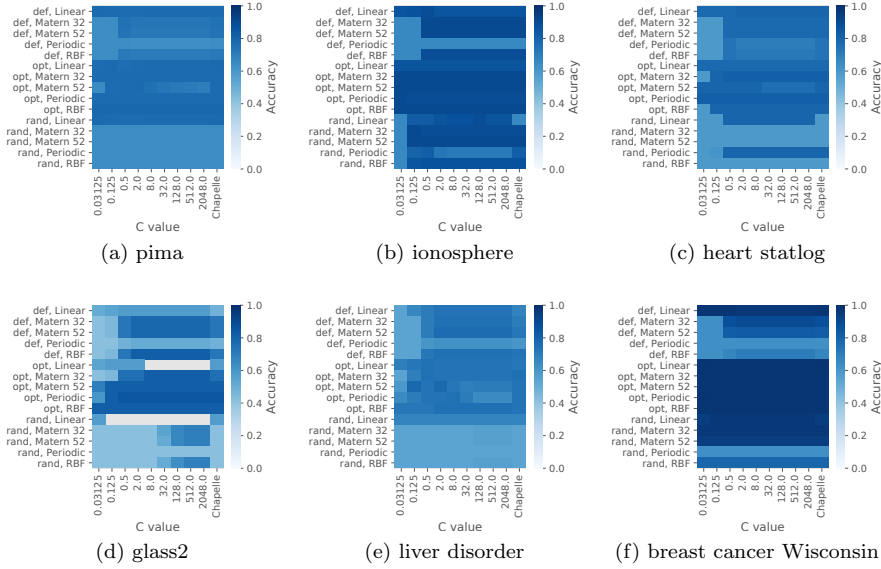


Figure 6: Accuracy in the test set for different values of C per kernel and hyperparameter optimization methods for UCI datasets. *Chapelle* indicates the C value suggested in [6]. *def* indicates the default set of hyperparameters, while *opt* and *rand* refer to the optimized and random hyperparameters respectively. The light gray areas indicate the C , kernel and hyperparameter combinations for which the optimal hyperplane of the SVM could not be computed.

7 Metrics

Another important aspect of the kernel learning is the metric used to evaluate its performance. The selected metric should be informative about the performance of the kernels in the training set, but also needs to provide some clues as to the generalization ability of the kernel. Furthermore, it has a direct influence on the search method, as the roughness of the search landscape heavily depends on this choice.

In the SVM kernel search literature, almost every proposal uses accuracy [19, 24, 47, 34, 2, 48, 11, 44, 12, 49, 22] or classification error related metrics [28, 25, 21] to measure the goodness of the kernel. As these metrics are discrete, some of the approaches include tiebreakers to deal with the same results when comparing similar kernels [25, 49]. There is very little knowledge about the performance of other metrics. For example, in [50], the authors tried to estimate the goodness of the matrix generated by the kernel measuring the intra/extra class similarity ratio instead of evaluating the SVMs.

The accuracy may seem the best choice according to the literature, but it

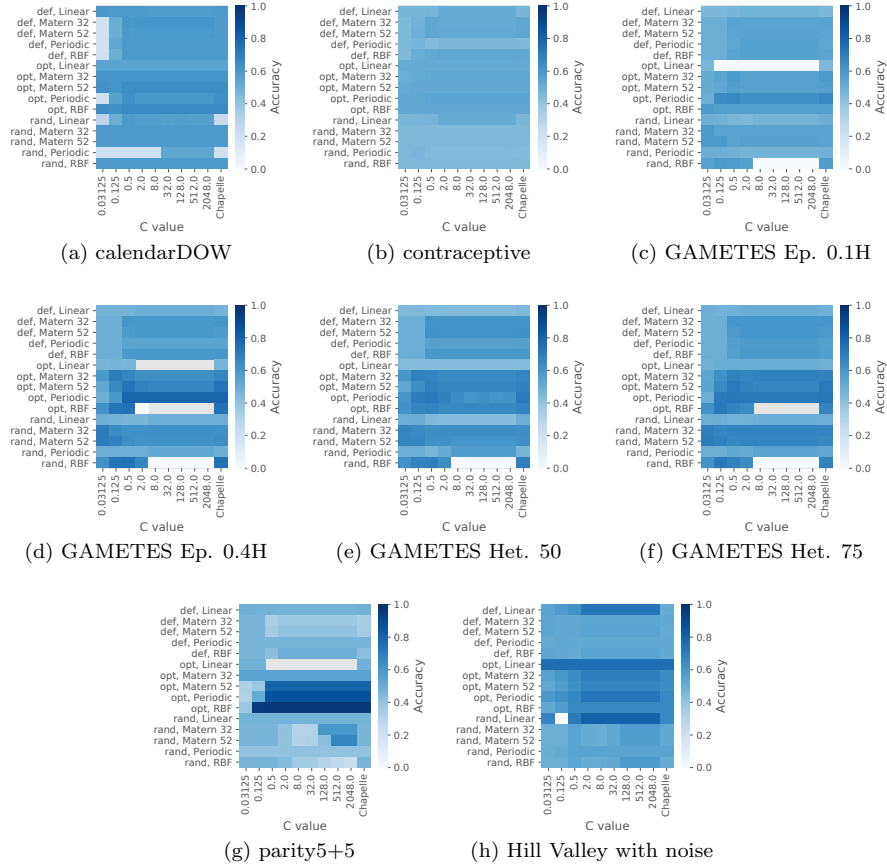


Figure 7: Accuracy in the test set for different values of C per kernel and hyperparameter optimization methods for PMLB datasets. *Chapelle* indicates the C value suggested in [6]. *def* indicates the default set of hyperparameters, while *opt* and *rand* refer to the optimized and random hyperparameters respectively. The light gray areas indicate the C , kernel and hyperparameter combinations for which the optimal hyperplane of the SVM could not be computed.

also has some drawbacks. As previously mentioned, this measure requires some methods to deal with tie results. It produces a search landscape where we can not directly obtain the gradient of this metric, disallowing the usage of many search methods that exploit this feature to optimize the C parameter, the kernel or its hyperparameters. On the other hand, the accuracy does not include any information about the generalization ability of the model and requires some k-fold cross-validation to mitigate overfitting, which is very computationally

demanding.

7.1 Likelihood based metric for the SVM kernel selection

In order to provide a more robust measure to guide the kernel selection, we propose a Bayesian Information Criterion (BIC) [45] based measure for SVMs, SVMBIC. This measure uses Platt scaling [40] to obtain probabilistic predictions of the SVM model and includes a complexity penalization according to the number of hyperparameters. It can be described as follows:

$$SVMBIC(k_i) = -2 \sum_{i=0}^n \log p(y_i | \mathbf{x}_i, k_i, \boldsymbol{\theta}_{i,best}) + q \log n \quad (10)$$

where q is the number of hyperparameters of the kernel and n is the number of data points in X . $\boldsymbol{\theta}_{i,best}$ is the best hyperparameter set for the kernel structure k_i .

Being a continuous measure, SVMBIC provides a smoother landscape than the accuracy, as small changes in the SVM parameters produce little variations in their values. Therefore, gradient based approaches can be used to optimize the hyperparameters or C . Besides, it includes an explicit complexity penalization.

The following experimental scenario was designed as a test for this new measure against the commonly used accuracy metric. By using GP as described in Section 5, we search for new kernels in the UCI datasets. In order to take advantage of properties of the BIC measure, we have optimized the hyperparameters based on a multi-start variation of Powell’s conjugate direction method [41] for every kernel, while for the experimental setting with the accuracy measure, we perform a grid search to find the best parameters. In both cases the same amount of evaluations was allowed (1000). The C parameter was set following a grid search as in the experiment described in Section 4.1.

The results of the experiment are summarized in Table 11. As expected, in all the problems the kernel structures optimized with accuracy as metric achieve better results in the training set than those learned by means of the SVMBIC metric, particularly in the *GAMETES Heterogeneity 50* problem. However, when compared in the test set, the performance gap between these two methods is undoubtedly lower for most of the problems. In fact, for *heart statlog* and *contraceptive* problems, the SVMBIC method outperforms the results obtained by using accuracy as the metric.

Moreover, in Table 11, the average number of hyperparameters of the best kernels obtained in each search procedure is shown. As can be seen, the SVMBIC approach clearly penalizes the number of hyperparameters, showing a lower average number of hyperparameters per kernel than the accuracy guided approach.

According to the results of our experimentation, SVMBIC can be used to obtain simpler kernels than those obtained with accuracy, close to them in terms of performance, even outperforming the latter in some of the runs. The complexity penalization of the solutions, together with the continuous nature

	Classification problem	# of HPs		Training set		Test set	
		Ac.	BIC	Ac.	BIC	Ac.	BIC
UCI	pima	4.2	0.1	0.793	0.770	0.759	0.758
	ionosphere	4.8	0.4	0.962	0.939	0.946	0.942
	heart statlog	3.4	0.0	0.878	0.847	0.802	0.820
	glass2	4.8	0.7	0.909	0.835	0.836	0.803
	liver disorder	5.4	0.3	0.763	0.727	0.719	0.712
	breast-cancer-wisconsin	4.6	0.0	0.986	0.978	0.973	0.973
PMLB	calendarDOW	5.7	0.7	0.665	0.592	0.616	0.594
	contraceptive	4.5	0.8	0.573	0.552	0.552	0.553
	GAMETES Epistasis 0.1H	4.8	1.6	0.681	0.674	0.675	0.670
	GAMETES Epistasis 0.4H	4.1	1.9	0.793	0.734	0.796	0.739
	GAMETES Heterogeneity 50	4.4	1.1	0.709	0.576	0.712	0.576
	GAMETES Heterogeneity 75	5.3	1.0	0.733	0.681	0.708	0.675
	parity5+5	3.8	1.7	0.999	0.987	1.000	1.000
	Hill_Valley_w._noise	4.0	1.8	0.905	0.863	0.910	0.884

Table 11: SVMBIC (BIC) measure compared to accuracy (Ac.). Average number of hyperparameters of the best kernels are shown on the left. The mean accuracy achieved by each kernel search metric (in the training and test sets) is shown in the right most columns. The numbers in bold indicate the best result for each problem.

of the SVMBIC measure, can contribute to improving the performance of a GP-like search.

8 Conclusions

Kernel functions are a key element of SVMs, as its performance strongly depends on them. Although the previous works in automated kernel search for SVMs have focused on the search algorithm itself, there are other components of the method that influence and condition the performance of SVMs that have not received the same attention. In this work, we have identified those components and analyzed the interactions between them, with the aim of obtaining a more general view about the kernel search for SVMs, making the following contributions:

- Identification of the components that influence the performance of SVMs: Apart from the weights of the hyperplane and the bias of the SVM, the kernel structure, its hyperparameters and the C parameter are also important for the efficiency of the method. The practitioner should consider all these components as a whole, instead of focusing on just one of them.
- The intrinsic limitation of using a reduced set of databases to evaluate kernel search strategies has been exposed: We have identified a number of

datasets where the behavior of standard kernels is far from being optimal. We highlight the need to extend the benchmark of datasets and increase the variety of characteristics that the datasets exhibit.

- Analysis of the kernel space: The kernel space where the search is carried out is a key element for the automated kernel search. We have proposed a basic mathematical expression based grammar, and proved the influence of including different elements in the performance of SVMs. All in all, it is worth expanding the search space by adding new elements to the grammar.
- Insights about the kernel structure search have been provided: Several methods have been proposed to learn the structure of the kernel for SVMs. We have compared the performance of the GP to other simpler search methods, obtaining marginal gains over them. Including the right elements in the grammar can be more important than the search method itself when trying to find the best kernel structure.
- Study the interplay of the hyperparameter tuning and C parameter setting: We have shown that the value of the hyperparameters and the value of C can drastically change the behavior of SVMs. We have also provided guidance on setting those parameters during the kernel search.
- A novel metric for the SVM kernel search has been proposed: Although the accuracy has been the standard measure in the SVM classification problems, it also presents some challenges. We propose a metric based on the BIC measure, which can overcome these problems.

Further research on grammar definition is suggested, as performance gains have been identified when additional elements are included. The addition of new grammatical elements could allow the application of SVM to new fields. We also propose to continue the work done in the study of the interplay of the hyperparameters and C , since the kernel search strongly depends on the setting of these parameters. On the other hand, we have seen that SVMBIC, in contrast to the traditional accuracy metric, is a continuous metric that introduces an explicit complexity penalization. New methods for finding kernels and hyperparameters can be designed so that SVMBIC metric can take advantage of these properties.

Acknowledgements

This work has been supported by the Spanish Ministry of Science and Innovation (project PID2019-104966GB-I00), and the Basque Government (IT1244-19 and ELKARTEK programs). Jose A. Lozano is also supported by BERC 2018-2021 (Basque government) and BCAM Severo Ochoa accreditation SEV-2017-0718 (Spanish Ministry of Science and Innovation).

Conflict of interest

The authors declare that they have no conflict of interest.

References

- [1] Ali S, Smith-Miles KA (2006) A meta-learning approach to automatic kernel selection for support vector machines. *Neurocomputing* 70(1):173–186, DOI 10.1016/j.neucom.2006.03.004, URL <http://www.sciencedirect.com/science/article/pii/S0925231206001056>
- [2] Alizadeh M, Ebadzadeh MM (2011) Kernel evolution for support vector classification. In: 2011 IEEE Workshop on Evolving and Adaptive Intelligent Systems (EAIS), pp 93–99, DOI 10.1109/EAIS.2011.5945924
- [3] Bing W, Wen-qiong Z, Ling C, Jia-hong L (2010) A GP-based kernel construction and optimization method for RVM. In: 2010 The 2nd International Conference on Computer and Automation Engineering (ICCAE), vol 4, pp 419–423, DOI 10.1109/ICCAE.2010.5451646
- [4] Boser BE, Guyon IM, Vapnik VN (1992) A Training Algorithm for Optimal Margin Classifiers. In: *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, ACM, New York, NY, USA, COLT '92, pp 144–152, DOI 10.1145/130385.130401, URL <http://doi.acm.org/10.1145/130385.130401>, event-place: Pittsburgh, Pennsylvania, USA
- [5] Burges CJ, Crisp DJ (2000) Uniqueness of the SVM solution. In: *Advances in Neural Information Processing Systems*, pp 223–229
- [6] Chapelle O (2002) *Support Vector Machines: Induction Principle, Adaptive Tuning and Prior Knowledge*. PhD thesis, LIP6
- [7] Cho Y, Saul LK (2009) Kernel Methods for Deep Learning. In: Bengio Y, Schuurmans D, Lafferty JD, Williams CKI, Culotta A (eds) *Advances in Neural Information Processing Systems 22*, Curran Associates, Inc., pp 342–350, URL <http://papers.nips.cc/paper/3628-kernel-methods-for-deep-learning.pdf>
- [8] Crammer K, Singer Y (2001) On the Algorithmic Implementation of Multiclass Kernel-based Vector Machines. *Journal of Machine Learning Research* 2(Dec):265–292, URL <http://www.jmlr.org/papers/v2/crammer01a.html>
- [9] Demšar J (2006) Statistical Comparisons of Classifiers over Multiple Data Sets. *Journal of Machine Learning Research* 7:1–30, URL <http://dl.acm.org/citation.cfm?id=1248547.1248548>

- [10] Dioşan L, Rogozan A, Pecuchet JP (2007) Evolving kernel functions for SVMs by genetic programming. In: Sixth International Conference on Machine Learning and Applications (ICMLA 2007), pp 19–24, DOI 10.1109/ICMLA.2007.70
- [11] Dioşan L, Rogozan A, Pecuchet JP (2008) Optimising Multiple Kernels for SVM by Genetic Programming. In: Evolutionary Computation in Combinatorial Optimization, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, pp 230–241, DOI 10.1007/978-3-540-78604-7_20, URL https://link.springer.com/chapter/10.1007/978-3-540-78604-7_20
- [12] Dioşan L, Rogozan A, Pecuchet JP (2012) Improving classification performance of Support Vector Machine by genetically optimising kernel shape and hyper-parameters. Applied Intelligence 36(2):280–294, DOI 10.1007/s10489-010-0260-1, URL <https://link.springer.com/article/10.1007/s10489-010-0260-1>
- [13] Dua D, Graff C (2017) UCI Machine Learning Repository. University of California, Irvine, School of Information and Computer Sciences, URL <http://archive.ics.uci.edu/ml>
- [14] Durrande N, Ginsbourger D, Roustant O (2012) Additive covariance kernels for high-dimensional Gaussian process modeling. Annales de la Faculté de Sciences de Toulouse Tome 21(numéro 3):p. 481–499, URL <https://hal.archives-ouvertes.fr/hal-00644934>
- [15] Duvenaud D (2014) Automatic model construction with Gaussian processes. Thesis, University of Cambridge, URL <http://www.repository.cam.ac.uk/handle/1810/247281>
- [16] Duvenaud D, Lloyd J, Grosse R, Tenenbaum J, Zoubin G (2013) Structure Discovery in Nonparametric Regression through Compositional Kernel Search. In: Proceedings of The 30th International Conference on Machine Learning, pp 1166–1174, URL <http://jmlr.org/proceedings/papers/v28/duvenaud13.html>
- [17] Fortin FA, Rainville FMD, Gardner MA, Parizeau M, Gagné C (2012) DEAP: Evolutionary Algorithms Made Easy. Journal of Machine Learning Research 13(Jul):2171–2175, URL <http://www.jmlr.org/papers/v13/fortin12a.html>
- [18] Friedman M (1937) The use of ranks to avoid the assumption of normality implicit in the analysis of variance. Journal of the American Statistical Association 32(200):675–701
- [19] Gagné C, Schoenauer M, Sebag M, Tomassini M (2006) Genetic Programming for Kernel-Based Learning with Co-evolving Subsets Selection. In: Parallel Problem Solving from Nature - PPSN IX, Lecture Notes

- in Computer Science, Springer, Berlin, Heidelberg, pp 1008–1017, DOI 10.1007/11844297_102, URL https://link.springer.com/chapter/10.1007/11844297_102
- [20] Genton MG (2002) Classes of Kernels for Machine Learning: A Statistics Perspective. *Journal of Machine Learning Research* 2:299–312, URL <http://dl.acm.org/citation.cfm?id=944790.944815>
- [21] Gijsberts A, Metta G, Rothkrantz L (2010) Evolutionary Optimization of Least-Squares Support Vector Machines. In: *Data Mining, Annals of Information Systems*, Springer, Boston, MA, pp 277–297, DOI 10.1007/978-1-4419-1280-0_12, URL https://link.springer.com/chapter/10.1007/978-1-4419-1280-0_12
- [22] Girdea M, Ciortuz L (2007) A Hybrid Genetic Programming and Boosting Technique for Learning Kernel Functions from Training Data. In: *Ninth International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC 2007)*, pp 395–402, DOI 10.1109/SYNASC.2007.71
- [23] HajiGhassemi N, Deisenroth M (2014) Analytic Long-Term Forecasting with Periodic Gaussian Processes. In: *Proceedings of Machine Learning Research*, pp 303–311, URL <http://proceedings.mlr.press/v33/hajighassemi14.html>
- [24] Howley T, Madden MG (2005) The Genetic Kernel Support Vector Machine: Description and Evaluation. *Artificial Intelligence Review* 24(3-4):379–395, DOI 10.1007/s10462-005-9009-3, URL <https://link.springer.com/article/10.1007/s10462-005-9009-3>
- [25] Howley T, Madden MG (2006) An Evolutionary Approach to Automatic Kernel Construction. In: *Artificial Neural Networks – ICANN 2006*, Springer, Berlin, Heidelberg, *Lecture Notes in Computer Science*, pp 417–426, DOI 10.1007/11840930_43, URL https://link.springer.com/chapter/10.1007/11840930_43
- [26] Hussain M, Wajid SK, Elzaart A, Berbar M (2011) A Comparison of SVM Kernel Functions for Breast Cancer Detection. In: *Imaging and Visualization 2011 Eighth International Conference Computer Graphics*, pp 145–150, DOI 10.1109/CGIV.2011.31
- [27] Joachims T (1998) Making large-scale SVM learning practical. *Tech. Rep. 1998,28*, Technical Report, URL <https://www.econstor.eu/handle/10419/77178>
- [28] Koch P, Bischl B, Flasch O, Bartz-Beielstein T, Weihs C, Konen W (2012) Tuning and evolution of support vector kernels. *Evolutionary Intelligence* 5(3):153–170, DOI 10.1007/s12065-012-0073-8, URL <https://link.springer.com/article/10.1007/s12065-012-0073-8>

- [29] Koza JR (1992) Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press
- [30] Li CH, Lin CT, Kuo BC, Chu HS (2010) An automatic method for selecting the parameter of the RBF kernel function to support vector machines. In: 2010 IEEE International Geoscience and Remote Sensing Symposium, pp 836–839, DOI 10.1109/IGARSS.2010.5649251, iSSN: 2153-7003
- [31] Li JB, Chu SC, Pan JS (2013) Kernel Learning Algorithms for Face Recognition. Springer Science & Business Media, google-Books-ID: Rky6BAAAQBAJ
- [32] MacKay DJC (1996) Bayesian Methods for Backpropagation Networks. In: Models of Neural Networks III, Physics of Neural Networks, Springer, New York, NY, pp 211–254, DOI 10.1007/978-1-4612-0723-8_6, URL https://link.springer.com/chapter/10.1007/978-1-4612-0723-8_6
- [33] Mercer J (1909) XVI. Functions of positive and negative type, and their connection the theory of integral equations. Philosophical Transactions of the Royal Society of London Series A, Containing Papers of a Mathematical or Physical Character 209(441-458):415–446, DOI 10.1098/rsta.1909.0016, URL <https://royalsocietypublishing.org/doi/10.1098/rsta.1909.0016>
- [34] Mezher MA, Abbod MF (2014) Genetic folding for solving multiclass SVM problems. Applied Intelligence 41(2):464–472, DOI 10.1007/s10489-014-0533-1, URL <https://link.springer.com/article/10.1007/s10489-014-0533-1>
- [35] Mohandes MA, Halawani TO, Rehman S, Hussain AA (2004) Support vector machines for wind speed prediction. Renewable Energy 29(6):939–947, DOI 10.1016/j.renene.2003.11.009, URL <http://www.sciencedirect.com/science/article/pii/S0960148103003860>
- [36] Neal RM (1996) Bayesian Learning for Neural Networks. Lecture Notes in Statistics, Springer-Verlag, New York, URL <https://www.springer.com/gp/book/9780387947242>
- [37] Olson RS, La Cava W, Orzechowski P, Urbanowicz RJ, Moore JH (2017) PMLB: a large benchmark suite for machine learning evaluation and comparison. BioData Mining 10(1):36, DOI 10.1186/s13040-017-0154-4, URL <https://doi.org/10.1186/s13040-017-0154-4>
- [38] Pei Y (2019) Automatic Decision Making for Parameters in Kernel Method. In: 2019 IEEE Symposium Series on Computational Intelligence (SSCI), pp 3207–3214, DOI 10.1109/SSCI44817.2019.9002691
- [39] Phienthrakul T, Kijisirikul B (2007) GPES: An algorithm for evolving hybrid kernel functions of Support Vector Machines. In: 2007 IEEE Congress

- on Evolutionary Computation, pp 2636–2643, DOI 10.1109/CEC.2007.4424803
- [40] Platt J (1999) Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in Large-Margin Classifiers* 10(3):61–74
- [41] Powell MJD (1964) An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The Computer Journal* 7(2):155–162, DOI 10.1093/comjnl/7.2.155, URL <http://comjnl.oxfordjournals.org/content/7/2/155>
- [42] Pree H, Herwig B, Gruber T, Sick B, David K, Lukowicz P (2014) On general purpose time series similarity measures and their use as kernel functions in support vector machines. *Information Sciences* 281:478–495, DOI 10.1016/j.ins.2014.05.025, URL <http://www.sciencedirect.com/science/article/pii/S0020025514005714>
- [43] Reitmaier T, Sick B (2015) The responsibility weighted Mahalanobis kernel for semi-supervised training of support vector machines for classification. *Information Sciences* 323:179–198, DOI 10.1016/j.ins.2015.06.027, URL <http://www.sciencedirect.com/science/article/pii/S0020025515004594>
- [44] Schuh MA, Angryk RA, Sheppard J (2012) Evolving Kernel Functions with Particle Swarms and Genetic Programming. In: Youngblood GM, McCarthy PM (eds) *Proceedings of the Twenty-Fifth International Florida Artificial Intelligence Research Society Conference, 2012*, AAAI Press, Marco Island, Florida, pp 80–85, URL <http://www.aaai.org/ocs/index.php/FLAIRS/FLAIRS12/paper/view/4479/4770.pdf>
- [45] Schwarz G (1978) Estimating the Dimension of a Model. *The Annals of Statistics* 6(2):461–464, DOI 10.1214/aos/1176344136, URL <https://projecteuclid.org/euclid.aos/1176344136>
- [46] Shaffer JP (2012) Modified Sequentially Rejective Multiple Test Procedures. *Journal of the American Statistical Association* URL <https://amstat.tandfonline.com/doi/abs/10.1080/01621459.1986.10478341>
- [47] Sousa ADM, Lorena AC, Basgalupp MP (2017) GEEK: Grammatical Evolution for Automatically Evolving Kernel Functions. In: *2017 IEEE Trustcom/BigDataSE/ICSS*, pp 941–948, DOI 10.1109/Trustcom/BigDataSE/ICSS.2017.334
- [48] Sullivan KM, Luke S (2007) Evolving Kernels for Support Vector Machine Classification. In: *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, ACM, New York, NY, USA, GECCO '07*, pp 1702–1707, DOI 10.1145/1276958.1277292, URL <http://doi.acm.org/10.1145/1276958.1277292>

- [49] Thadani K, Ashutosh, Jayaraman VK, Sundararajan V (2006) Evolutionary Selection of Kernels in Support Vector Machines. In: 2006 International Conference on Advanced Computing and Communications, pp 19–24, DOI 10.1109/ADCOM.2006.4289849
- [50] Valerio R, Vilalta R (2014) Kernel selection in support vector machines using gram-matrix properties. In: Proceedings of the 27th International Conference on Advances in Neural Information Processing Systems. Workshop on Modern Nonparametrics: Automating the Learning Pipeline, NIPS, vol 14, pp 2–4
- [51] Vapnik V (1963) Pattern recognition using generalized portrait method. *Automation and remote control* 24:774–780
- [52] Vapnik VN (1995) *The Nature of Statistical Learning Theory*. Springer-Verlag, Berlin, Heidelberg
- [53] Zhang F (2011) Positive Semidefinite Matrices. In: *Matrix Theory*, Universitext, Springer, New York, NY, pp 199–252, DOI 10.1007/978-1-4614-1099-7_7, URL https://link.springer.com/chapter/10.1007/978-1-4614-1099-7_7
- [54] Zhao L, Gai M, Jia Y (2018) Classification of Multiple Power Quality Disturbances Based on PSO-SVM of Hybrid Kernel Function. *Journal of Information Hiding and Multimedia Signal Processing* 10(1):138–146