Computational Intelligence and Neuroscience

Biomedical Classification Problems Automatically Solved by Computational Intelligence Methods

Luis Carlos Padierna,[1] Carlos Villaseñor-Mora,[1]

[1] Departamento de Ingenierías Química, Electrónica y Biomédica, División de Ciencias e Ingenierías, Universidad de Guanajuato, León, 37150, México

Correspondence should be addressed to Luis Carlos Padierna; lc.padierna@ugto.mx

Abstract

Biomedical classification problems are of great interest for both medical practitioners and computer scientists. Due to the harmful consequences of a wrong decision in this ambit, computational methods must be carefully designed to provide a reliable tool for helping physicians to obtain accurate predictions on unseen cases. Computational Intelligence (CI) provides robust models to perform optimization, classification and regression tasks. These models have been previously designed, mainly based on the expertise of computer scientists, to solve a vast number of biomedical problems. As the number of both CI algorithms and biomedical problems continues to grow, selecting the right method to solve a given problem becomes more challenging. To deal with this complexity, a systematic methodology for selecting a suitable model for a given classification problem is required. In this work, we review the more promising classification and optimization algorithms, and reformulate them into a synergic framework to automatically design and optimize pattern classifiers. Our proposal, including state-of-the-art evolutionary algorithms and support vector machines, is tested on a variety of biomedical problems. Experimental results on benchmark datasets allow us to conclude that the automatically designed classifiers reach higher or equal performance than those designed by computer specialists.

Introduction

Biomedical engineering is a concept that bridges medicine and technology, and includes fields of specialization such as: bioimaging, bioinstrumentation, biomolecular analysis, biomechanics and biomaterials [1]. Research conducted in these fields frequently faces classification problems, for instance when deciding: if a gene can be a candidate of risk to develop Autism [2], if a patient presents a Parkinson disease based on the information obtained by acoustic biomarkers [3], if a microscopic image of breast tumor tissue provides evidence that it is benign or malignant [4] or if thermal patterns can help on the diagnosis of diabetic food [5].

Computational Intelligence have provided methods to solve biomedical classification problems for years. However, the task of selecting an appropriate set of algorithms for a given problem has been mainly delegated to an expert. Among classification methods, Support Vector Machines (SVMs) are one of the most successful approach due to its generalization capability by conducting structural risk minimization, absence of local minima by solving a quadratic programming problem and representation based on few parameters [6] [7] [8].

The performance of SVMs is highly dependent on two of its components: kernel function and hyper-parameters [9] [10]. With respect to the kernel functions, different scenarios have emerged through the years to ensure that the best kernel function is used for a classification task. These scenarios include kernel generation from primary operations [11] [12] [13], and multiple kernel combination from existing kernels [14] [15] [16] [17] [18] [19]. Out of these, the latter has proved to be more convenient, since it combines the flexibility of kernel generation with the effectiveness of pre-designed kernels.

With respect to the hyper-parameter tuning, several optimization methods have been applied, from the simple Grid Search and Random Search [20], to the more sophisticated Evolutionary Strategies [21], Genetic Algorithms [22], Bio-inspired Metaheuristics [23] [24] [25], and Estimation of Distribution Algorithms (EDAs) [26], among others. Recently, it has been shown that EDAs perform SVM hyper-parameter tuning more efficiently than other methods when solving biomedical classification tasks [27]. Some studies have described strategies for simultaneous kernel selection (or kernel generation) and hyper-parameter optimization [12] [18]; however, it can be argued that they lack convergence efficiency, explore a limited search space and are prone to overfitting.

In this work, a novel method is formulated to solve the issues of previous studies by combining the advantages of evolutionary programming with the guided exploration of estimation of distribution algorithms. Our method, hereafter referred as Smart Evolution of Ensemble Kernel for Support Vector Machines (SEEKS), consists of a Genetic Programming (GP) mechanism [28] able to build new multiple kernels based on different kernel families (or to select the best single kernel) and an EDA [29] adapted to the GP mechanism and aimed to build probability models based on estimates of the hyper-parameter distributions. Thus, our method performs hyper-parameter tuning of kernels as these are being evolved without adding any significant overhead.

Through robust experimentation it is shown that SEEKS automatically achieves simultaneous kernel design and hyper-parameter tuning for SVM classifiers as successfully as previous methods designed by specialists, with significantly higher computational efficiency and improved parameter convergence. Next section provides the background theory to clearly understand the tasks of SVM-kernel design and hyper-parameter tuning. In Section III, our SEEKS method is justified and described. Section IV presents the experimental methodology followed to test our proposal. The results obtained in terms of performance of the generated SVMs are discussed in Section V along with the analysis of the convergence and efficiency of SEEKS. Finally, conclusions and directions for future work are offered in Sect. VI.

Materials and Methods

II A. Kernel Functions for Support Vector Machines

Given a set of m training data points $\{\mathbf{x}_i, y_i\}_{i=1}^{m}$, where $\mathbf{x}_i \in R^d$ is the i-th input vector and $y_i \in \{+1, -1\}$ its corresponding class label; a SVM classifier in dual form can be formulated, introducing the Lagrange multipliers $\boldsymbol{\alpha}$ and following the Karush-Kuhn-Tucker conditions, as [30]:

$$Max\ L(\boldsymbol{\alpha}) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} y_i y_j \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{z}_j)$$
$$s.t. 0 \leq \alpha_i \leq \mathrm{C},\ \forall i = 1, \ldots, m\ \ \textstyle\sum_{i=1}^{m} \alpha_i y_i = 0 \tag{1}$$

where C is called a penalty factor, $\mathbf{x}_i, \mathbf{z}_j \in R^d$ are the i-th and j-th input vector, respectively; and the function $K(\mathbf{x}, \mathbf{z})$ defined on $R^d \times R^d$ is called a kernel if there exists a map ϕ from the space R^d to the Hilbert space, $\phi: R^d \rightarrow H$ such that $K(\mathbf{x}, \mathbf{z}) = \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle$ [10]. The kernel function can take different forms, as those in TABLE 1. Hyper-parameters are those parameters of a kernel plus the parameters of a specific SVM formulation (e.g. the penalty factor C). Choosing among different kernels is equivalent to choosing among different SVM models [9]. A deterministic way to select the most appropriate kernel for a given classification problem is still unknown. Moreover, kernel selection is becoming more challenging because the number of valid kernels continues to grow as new kernel families are proposed. These families include: wavelet kernels [31], non-parametric kernels [32] and orthogonal kernels [33] [34] [35] [36]. Kernels in TABLE 1 have proved to be valid kernels since they satisfy the necessary and sufficient conditions established in the Mercer's theorem [37]. Briefly stated, this theorem affirms that the series $\sum_{i=1}^{\infty} \lambda_i \phi_i(\mathbf{x}) \phi_i(\mathbf{z})$, in terms of eigenfunctions $\phi_i \in L_2(X \subseteq R^d)$ and positive associated eigenvalues λ_i, converges absolutely and uniformly to $K(\mathbf{x}, \mathbf{z})$ when the latter is symmetric and positive semidefinite. To be positive semidefinite a kernel function must comply with $\iint K(\mathbf{x}, \mathbf{z}) g(\mathbf{x}) g(\mathbf{z}) d\mathbf{x} d\mathbf{z} \geq 0$.

TABLE 1. Classic, wavelet, non-parametric and orthogonal kernels

Kernel (code)	Expression	Eq.
Linear (L)	$K_{Lin}(\mathbf{x}, \mathbf{z}) = \mathbf{x}^T \mathbf{z}$	(2)
Polynomial (P)	$K_{Pol}(\mathbf{x}, \mathbf{z}) = (\mathrm{a}\mathbf{x}^T \mathbf{z} + \mathrm{b})^{\mathrm{n}}$	(3)
Radial (R)	$K_{RBF}(\mathbf{x}, \mathbf{z}) = exp(-\gamma \lVert \mathbf{x} - \mathbf{z} \rVert^2)$	(4)
Wavelet (W)	$K_{Wav}(\mathbf{x}, \mathbf{z}) = \prod_{j=1}^{d} \left(h \times e^{\frac{-\lVert (x_j - z_\mathrm{j}) \rVert^2}{2a^2}} \right)$ with $h = cos\left(1.75 \times \left(\mathbf{x}_j - \mathbf{z}_j\right)/a\right)$	(5)
Non-param. (K_{11})	$K_{11}(\mathbf{x}, \mathbf{z}) = 1 + \sum_{i=1}^{3} (-\lVert \mathbf{x} - \mathbf{z} \rVert / \tau)^i / i$	(6)
Non-param. (K_{13})	$K_{13}(\mathbf{x}, \mathbf{z}) = 1 - sin(\pi \lVert \mathbf{x} - \mathbf{z} \rVert / 2\, \tau)$	(7)
Legendre (E)	$K_{Leg}(\mathbf{x}, \mathbf{z}) = \prod_{j=1}^{d} \sum_{i=0}^{n} P_i(\mathrm{x}_j) P_i(z_j)$	(8)
s-Hermite (H)	$K_{Her}(\mathbf{x}, \mathbf{z}) = \prod_{j=1}^{d} \sum_{i=0}^{n} He_i(x_j) He_i(z_j)(2^{-2n})$	(9)
Gegenbauer (G) cf. [36] for details	$K_{Geg}(\mathbf{x}, \mathbf{z}) = \prod_{j=1}^{d} \sum_{i=0}^{n} C_i^{\alpha}(x_j) C_i^{\alpha}(z_j) w_{\alpha}(x_j, z_j) u(C_i^{\alpha})^2$ With $w_{\alpha}(x_j, z_j) = \left((1 - x^2)(1 - z^2)\right)^{\alpha - 0.5} + \epsilon$ $u(C_i^{\alpha}) = \left(\sqrt{n+1} \times \lvert C_i^{\alpha}(1) \rvert\right)^{-1}$	(10)

A current research trend consists in combining two or more kernels to increase the accuracy rate and generalization capability of SVMs. A combination of Mercer kernels is also valid under the closures in TABLE 2 (for a formal proof of these closures, cf. [38]). This approach has been followed in several relevant works [14] [18] [19] [32] [34] [35] [36], and has introduced the concept of a Multiple Kernel, denoted: $K_\eta(\mathbf{x},\mathbf{z}) = f_\eta\left(\left\{K_i\left(\mathbf{x}^i,\mathbf{z}^i\right)\right\}_{i=1}^{P} \vee \eta\right)$ where $\mathbf{x}^i, \mathbf{z}^i \in R^{d_i}$; the kernel functions, $\{K_i\colon R^{d_i} \times R^{d_i} \to R\}_{i=1}^{P}$ take P feature representations (not necessarily different) of data instances; and the combination function $f_\eta\colon R^p \to R$ can be linear or nonlinear. The parameter η indicates that a certain set of predefined kernels is used (i.e., the kernels and their parameters are known before training) [39].

TABLE 2. Closures allowing the valid combination of kernels

$K(x,z)$	Description
$K(x,z) = K_1(x,z) + K_2(x,z)$	Closure under the sum.
$K(x,z) = aK_1(x,z)$	Multiplication by scalar,$a \in R^+$.
$K(x,z) = K_1(x,z)K_2(x,z)$	Closure under product.
$K(x,z) = f(x)f(z)$	$f(\cdot)$ is a real-valued function on X.
$K(x,z) = K_3\big(\phi(x),\phi(z)\big)$	Kernel composition.

II B. Metaheuristic Algorithms

Metaheuristics refers to a family of approximate optimization techniques that provide acceptable solutions in a reasonable time for solving complex problems [40]. There exist several types of metaheuristics; however, only Estimation of Distribution and Genetic Programming algorithms will be considered in the implementation of SEEKS since these are the more promising methods found so far in literature related to kernel evolution [12] [18] [19] and hyper-parameter tuning of SVMs [23] [21] [20] [24] [25]. The selection of an EDA among other metaheuristics is based on two reasons: first, its iterative mechanism naturally adapts to the GP algorithm; and second, it was proved to be the optimal hyper-parameter tuner of SVM classifiers when solving biomedical problems [27]. EDAs explore a solution space by iteratively building and sampling explicit probabilistic models of candidate solutions that guide the search [29]. The generic procedure is shown in LISTING 1.

LISTING 1. General Estimation of Distribution Algorithm

```
Generate initial population of $N$ solutions: $S^{(0)} = \{\mathbf{s}_i\}_{i=1}^{N}$ at iteration $t = 0$ uniformly
while (stopping criteria are not met)
    Compute objective function of each solution $g\left(S^{(t)}\right)$
    Select a subset of the best solutions, $S_M^{(t)}$ from $S^{(t)}$
    Build a probabilistic model $(11)$ based on $S_M^{(t)}$
    Sample $\boldsymbol{P}^{(t)}$ to generate new solutions $S'^{(t)}$
    Substitute / Incorporate $S'^{(t)}$ into $S^{(t)}$
    Update $t = t + 1$
end while
Output the best solution found, $\mathbf{s}^*$, as the result
```

Taking a specific probabilistic model, an EDA takes a certain name. Two of these particular cases of probabilistic models are considered in this work: the Univariate Marginal Distribution Algorithm (UMDA) [41] and the Boltzmann Univariate Marginal Distribution Algorithm (BUMDA) [42]. The UMDA builds a Gaussian model $\boldsymbol{P}^{(t)}$ from a set of solutions, with parameters $S = \{\mathbf{s}_i\}_{i=1}^{M}$ given by:

$$\mu_\kappa = \frac{1}{M}\sum_{i=1}^{M} s_{i,\kappa} \quad \text{and} \quad \sigma_k = \left(\frac{1}{M-1}\sum_{i=1}^{M}\left(s_{i,\kappa} - \mu_\kappa\right)^2\right)^{1/2} \qquad (11)$$

where M is the number of solutions considered to compute these parameters and $s_{i,\kappa}$ represents the κ-th component of solution $\mathbf{s}_i$, which is a D-dimensional vector in $\Re^D$.

The BUMDA modifies UMDA by employing a model based on the Boltzmann distribution and incorporating a truncation selection method to accelerate convergence as well as to free the user from having to set the number of samples that are selected for parameter estimation. The parameters of the Gaussian distribution used by the BUMDA are:

$$\mu_\kappa = \frac{1}{\tilde{g}} \sum_{i=1}^{M} s_{i,\kappa}\, g(\mathbf{s}_i) \quad \sigma_\kappa = \left(\frac{1}{\tilde{g}+1} \sum_{i=1}^{M} \left(s_{i,\kappa} - \mu_\kappa\right)^2 g(\mathbf{s}_i) \right)^{\frac{1}{2}} \quad (12)$$

where M is adjusted by the automatic truncation method, $g(\mathbf{s}_i)$ is the objective function value of the i-th solution, and $\tilde{g}$ represents the sum of the objective function values over the M selected solutions.

The GP algorithm arose as an extension of the conventional genetic algorithm for discovering computer programs using the expressiveness of symbolic representation. The search space for GP is the space of all possible expressions that can be recursively created by compositions of the available functions and variables for a given problem [28]. In the case of the kernel evolution problem, functions can be any operator in TABLE 2 and variables any kernel as those in TABLE 1. The major difference of GP with respect to other evolutionary algorithms is that in GP the solutions are stored in variable-sized structures, commonly in parse trees where operations are internal nodes and variables are the leaves [43]. These tree-based structures are used to combine kernels in our proposal. The pseudocode of the GP method is shown in LISTING 2.

LISTING 2. General Genetic Programming Algorithm

```
Generate initial population of solutions: $S^{(0)} = \{\mathbf{s}_i\}_{i=1}^{N}$ at iteration $t = 0$ uniformly
while (stopping criteria are not met)
    Compute objective function of each solution $g(S^{(t)})$
    Select $m$ individuals for reproduction $S_r^{(t)} \subset S^{(t)}$
    Apply variation operators to $S_r^{(t)}$ and keep the offspring $S_o^{(t)}$
    Compute objective function of each new candidate solution $g(S_o^{(t)})$
    Integrate candidates $S_o^{(t)}$ to new population according to replacement operators
    Update $t = t + 1$
end while
Output the best solution found, $\mathbf{s}^*$, as the final result.
```

III. SMART EVOLUTION OF KERNELS FOR SVMs

Previous works have constructed single and multiple kernels by means of Evolutionary Algorithms [12] [13] [14] [18] [19]. The first attempt to evolve kernels appears to be [14], where genetic algorithms were employed to explore possible kernel combinations. Subsequent studies [12] [18] [19] utilized GP with tree data structures to encode multiple kernels. Those works using GP adopted two different approaches: the first one focuses on combining vector operators to discover new kernel functions; this strategy introduces several problems (from numerical errors to poor results), mainly because it is prone to generate invalid kernels. The alternative approach combines predefined kernels under some of the closures in TABLE 2. Although this latter strategy has shown to be more consistent and it is adopted as part of our proposed method, it presents three major limitations:

(i) The hyper-parameters of evolved kernels were assigned unsystematically, leaving the burden of this task to the guided search of the GP algorithm. This presents two problems: the search space is extended dramatically, and the GP method is overloaded, since it should control the convergence of both hyper-parameters and kernel shape.

(ii) Evolved kernels obtained from the GP mechanism are prone to overfitting, since the search process is guided just by the accuracy index (without considering if datasets are unbalanced or the amount of data required to build the decision function).

(iii) The terminal set, and thus the basis of the GP search space, was limited to combinations of classic kernels (Linear, Sigmoid, Polynomial and RBF).

Our current proposal removes these limitations of previous works through the following improvements:

(i) The GP search space is reduced and its control on the kernel shape convergence is increased by delegating the hyper-parameter tuning task to a synergic EDA mechanism.

(ii) One advantage of SVMs over another classifiers is the property to estimate its generalization capability as function of the proportion of support vectors (PSV). Some performance indexes that consider both accuracy and PSV have been successfully used in [27] [44].

(iii) Recently developed kernels have shown advantages when combined with classical kernels. In [34] [35] [45] mixtures of classical, wavelet and orthogonal polynomial kernels were shown to reach better performance than classic kernels. As a natural next step, SEEKS enhances this expertise-based way of combining kernels by adapting a systematic tool for automatically exploring combinations of kernels from different families. To date, no single work has been found that reports neither the hyper-parameter tuning nor the evolution of kernels based on different kernel families. Thus, another contribution is the potentially first analysis of these kernels in an evolutionary methodology.

III A. The SEEKS Algorithm

SEEKS is formulated to take advantage of the GP and EDAs mechanisms so that, on each iteration, kernel evolution and hyper-parameter tuning are performed simultaneous and independently. The algorithm is overviewed in LISTING 3 and the sequence of steps is detailed below.

1. In the first step, N kernel combinations are codified as binary trees, randomly populated from the lists of kernels and operators described in TABLE 1 and TABLE 2, respectively. Hyper-parameters vectors are all of cardinality $\kappa = 6$, since $\mathbf{h} = (C, \gamma, a, b, n, \alpha)^T$. Figure 1 illustrates two possible initial kernel trees with corresponding hyper-parameter vectors.

2. In step two, each pair of kernel tree $\mathbf{k}$ and hyper-parameter vector $\mathbf{h}$ is evaluated as a SVM classifier on the same k-fold cross validation for a given dataset. The objective function $g(\mathbf{k}, \mathbf{h})$ is the k-fold classification average performance. Then, the kernel population is sorted by this average.

3. Common stopping criteria include: a max number of iterations reached, a tolerance between the best solution found so far and the best possible objective function value, a small deviation on the population performance, etc. If criteria are not satisfied, then, at iteration t perform the following steps:

4. The indexes of the best M kernel trees are used to update $\boldsymbol{P}^{(t)}$ for all continuous parameters. In the case of the discrete parameter n a multinomial model is estimated based on the distribution of the M best degrees. After this, kernel trees are again selected from the population $K^{(t)}$ by a method such as tournament, reward-based or binary selection, etc.

5. Crossover and one-point mutation are used as variation operators. To avoid uncontrolled growth a bloat-control method, such as Tarpeian or others [46] is recommended.

6. Sample the value of the required hyper-parameters for the new individuals from the current probability model $\boldsymbol{P}^{(t)}$ and train the SVMs corresponding to the new individuals. Again, following the same k-fold cross-validation scheme using the partition obtained in step 2.

7. Update the population $K^{(t)}$ by replacing the worst-performing individuals with the top-performing new individuals from $K_o^{(t)}$. Increment the iteration counter and go to step 2.

LISTING 3. Pseudocode of the SEEKS Algorithm

```
Initialize a population of kernel trees $K^{(0)} = \{\mathbf{k}_i\}_{i=1}^{N}$ and a set of hyper-parameter vectors $H^{(0)} = \{\mathbf{h}_i\}_{i=1}^{N}$ following a uniform distribution. Set iteration $t = 0$
Compute objective function $g(K^{(0)}, H^{(0)})$ for each kernel with its corresponding parameters on the same validation data.
while (stopping criteria are not met)
    Select the best $M$ parameter vectors to update a probability model $\boldsymbol{P}^{(t)}$ by equation (11) or (12). Also select kernel trees for reproduction $K_r^{(t)} \subset K^{(t)}$ with a crossover method.
    Apply variation operators to $K_r^{(t)}$ and keep the offspring $K_o^{(t)}$
    Sample new hyper-parameter vectors from $\boldsymbol{P}^{(t)}$ and compute $g(K_o^{(t)}, H^{(t)})$ of each new kernel tree.
    Integrate candidates $K_o^{(t)}$ to a new population according to replacement operators and update $t = t + 1$
end while
Output the best solution found, $(\mathbf{k}^*, \mathbf{h}^*)$, as the result.
```

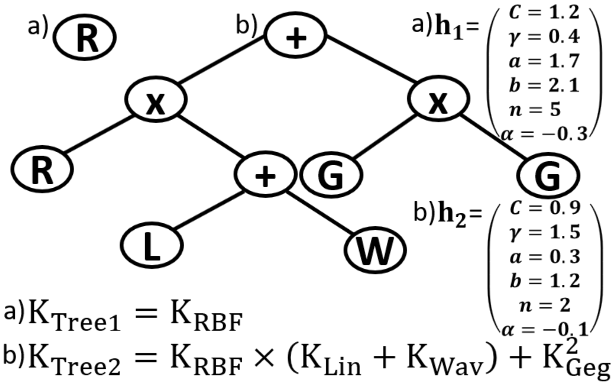
a)
R
b)
+
x
x
R
+
G
G
L
W
a)$\mathbf{h}_1$= $C = 1.2$, $\gamma = 0.4$, $a = 1.7$, $b = 2.1$, $n = 5$, $\alpha = -0.3$
b)$\mathbf{h}_2$= $C = 0.9$, $\gamma = 1.5$, $a = 0.3$, $b = 1.2$, $n = 2$, $\alpha = -0.1$
a)$\mathrm{K}_{\mathrm{Tree1}} = \mathrm{K}_{\mathrm{RBF}}$
b)$\mathrm{K}_{\mathrm{Tree2}} = \mathrm{K}_{\mathrm{RBF}} \times (\mathrm{K}_{\mathrm{Lin}} + \mathrm{K}_{\mathrm{Wav}}) + \mathrm{K}_{\mathrm{Geg}}^{2}$

Figure 1. Kernel tree decoding. a) Kernel selection is possible when the root node is a kernel, irrelevant hyper-parameters are ignored. b) Kernel construction combining four basic kernels, hyper-parameters are shared.

IV. EXPERIMENTAL DESIGN

IV A. Biomedical Problems

To test SEEKS method fifteen benchmark biomedical classification problems were selected from related works. Datasets (and short labels) corresponding to these problems are: breast cancer prediction (*breast*), chronic kidney disease prediction (*chronic*), vertebral column orthopaedic normality (*column_2C*), wart treatment results by using cryotherapy (*cryotherapy*), type 2 diabetes diagnosis (*diabetes*), identification of altered sperm concentration (*fertility*), survival prediction after surgery of breast cancer (*haberman*), determination of heart disease (*heart*), wart treatment results by using immunotherapy (*immuno*), liver disorders caused by alcohol (*liver*), discrimination of benign and malignant mammographic masses (*mammo*), Parkinson prediction based on voice measurements (*parkinsons*), post-operative life expectancy in lung cancer patients after thoracic surgery (*thoracic*), prediction on the donation of blood (*transfusion*) and prognostic on breast cancer (*wpbc*).

All datasets are publicly available at the UCI Machine Learning Repository [47]. Their characteristics and results of previous studies are summarized in TABLE 3. The best reported rates in classification accuracy by using the RBF kernel or Multiple kernels are also included as a basereference for comparison. All datasets were scaled to the range $[-1,1]$ to prevent influence of attributes with dominating values and to preserve the conditions required by orthogonal kernels.

TABLE 3. Summary of Biomedical Classification Problems

	Dataset short label	Total cases (positive - negative)	Fts[1]	Best Accuracy[2] reported with RBF or *Multiple Kernels*	References
1	breast	683 (239-444)	10	*98.03* | 97.31 | 97.18	[18] [48] [27]
2	chronic	400 (150-250)	24	99.60	[27]
3	column_2C	310 (100-210)	7	87.00 | 86.02	[20] [27]
4	cryotherapy	90 (43-47)	6	91.00	[49]
5	diabetes	768 (268-500)	8	*81.25* | 77.73 | 76.83	[18] [48] [50]
6	fertility	100 (12-88)	9	88.00 | 89.19 | 88.04	[20] [27] [8]
7	haberman	306 (81-225)	3	73.55 | 75.77 | 75.91	[48] [51] [35]
8	heart	270 (120-150)	13	*86.98* | 83.70 | 84.67	[18] [50] [27]
9	immuno	90 (19-71)	7	88.00 | 85.46	[49] [52]
10	liver	345 (145-200)	7	72.45 | 74.20 | 74.78	[50] [53] [8]
11	mammo	961(445-516)	5	86.44	[51]
12	parkinsons	195 (48-147)	22	95.30 | 95.98 | 98.88	[24] [27] [54]
13	thoracic	470 (70-400)	17	85.30 | 85.15	[20] [27]
14	transfusion	748 (178-570)	4	75.00 | 80.53	[20] [51]
15	wpbc	194 (46-148)	33	80.09 | 81.22	[50] [51]

[1] Fts is the number of attributes (features) that model a case.

[2] Variations on accuracy are due to the SVM solver algorithm applied and validation scheme followed on each study (dataset pre-processing and partition, hyper-parameter tuning strategy, etc.). For instance, results of multiple kernels in [18] used a unique 80-20 data partition for learning-validation and testing data, respectively.

IV B. Kernel Evolution Settings

For training of SVMs with tree kernels, the LIBSVM [55] solver was selected to achieve a direct comparison against previous works. Training a standard SVM has an algorithmic complexity between $O(N^2)$ and $O(N^3)$, with N the number of input vectors [56]. Furthermore, the associated Gram matrix of size $N \times N$ must be allocated. Thus, the computational cost of evolving SVMs is high for each evaluation of the fitness function (classification accuracy obtained by 5-fold cross validation). Taking into account this cost and the fact that only small improvements of candidate solutions have been observed after the 15th generation [18], in our experiments the number of generations was set to 15 as stopping criterion.

The performance index that guides the search of SEEKS is the rate in classification accuracy. However, high classification rates may hide overfitting to the training data. One way to observe this case is through an estimate of the generalization capability of a SVM, such as the proportion of support vectors (PSV = SV/N, where SV is the number of essential support vectors and N is the number of instances in the training dataset) that defines the SVM hyperplane [9]. Therefore, producing a good solution with the minimum PSV is a desirable goal when evolving kernels. For the sake of a direct comparison against previous works: the PSV is not used to guide the evolution of SVMs, but it is adopted as a complementary performance measure; parameters including tree depth, mutation and crossover rates, as well as methods for initialization, selection, variation and replacement were set to the values used in [18]. The function set was defined so that results could be directly compared with those reported in [34] and [35]. The terminal set includes the identifiers of kernels presented in TABLE 1. This configuration is provided in TABLE 4.

IV C. Parameter Settings and Experiment Objectives

The standard C-SVM considers few hyper-parameters, namely: the penalty factor C, and the kernel parameters, such as the decaying parameter γ, the degree n, and the dilation factor a for the RBF, orthogonal and Wavelet kernels, respectively. The work of Sun et al. [50] is the only one that we have identified that addresses the hyper-parameter tuning of orthogonal kernels (by means of Grid Search). No studies have been found about the influence of the Wavelet kernel parameters, so the values of the dilation parameter a are taken from the original work [31]. The non-parametric and Linear kernels do not possess parameters. From the analysis to these works, the hyper-parameter setting used to perform tuning is summarized in

TABLE 5.

TABLE 4. Configuration of SEEKS for kernel evolution

Parameter	Koch et al. 2012	Diosan et al 2012	Souza et al 2017	**SEEKS**
Population size	20	50	200	**100**
Max Tree depth	---	10	100[1]	**2-5**
Generations	2500[2]	50	100	**20**
Mutation rate	---[3]	30%	30%	**30%**
Crossover rate	---	80%	90%	**90%**
Function set (FS)	$+, -, \times$, %, exp, norm	$\times, +$, exp	$\times, +, exp$ log, tanh	$\times, +$
Terminal set (TS)	x, z c_1, c_2, c_3, c_4	P, R, S, c_1, c_2	$(x^T z)$, $\|\|x-z\|\|^2$ others	**L, P, R, W, $\boldsymbol{K_{11}}$,$\boldsymbol{K_{13}}$, E, G, H**
Initialization	Grow	Grow	GE	**Grow**
Selection method	Tourn-8	Tourn-2	Tourn-2	**Tourn-2**

[1] As GE do not use a tree structure, a max depth tree is not required. Instead, the max size of an expression is indicated.
[2] Instead of generations, Koch et al used 2500 evaluations on 20 runs.
[3] Values marked with "---" were not reported by the authors of that experiment.

TABLE 5. Configuration of SEEKS for Hyper-parameter tuning

Parameter	Experimental Setting
Previous Works	
RBF kernel decaying γ [18]	$\gamma_{qt} = q \cdot 10^t$, $q = \{1,2,\dots,9\}$, $t = \{-5, -4, \dots, -1\}$
Polynomial order n [18]	$n \in \{1,2,\dots,15\}$
Regularization C, Kernel decaying γ [50]	$C \in \{2^{-1}, 1, 2^1, \dots 2^5\}$ $\gamma \in \{2^{-6}, 2^{-5}, \dots, 1\}$
Polynomial order n Kernel decaying γ, [50]	$n \in \{2,3,\dots,6\}$ $\gamma \in \{0.1, 0.25, 0.5, 1, 1.5, 2, 2.5, 3\}$
SEEKS	
Regularization C	$C \in (0, 32]$
RBF kernel decaying γ	$\gamma \in (0, 4]$
Polynomial order n	$n \in \{1, 2, \dots, 6\}$
Wavelet dilation factor and classic polynomial scale a	$a \in (0, 2]$
Classic polynomial offset b	$b \in [0, 5]$
Gegenbauer α	$\alpha \in (-0.5, 1.5]$
UMDA-Sample size M	25% of the population

Two experiments were performed with the experimental setting described in TABLES 2-5. The objective of the first experiment is to analyse the effect of incorporating an EDA to the GP mechanism, which is the main idea of the SEEK algorithm. Both EDAs detailed in Sect II, UMDA and BUMDA, were implemented so that three different kernel evolution algorithms were compared. These algorithms are referred as GP, GP-U and GP-B as short names for the standard Genetic Programming with totally random hyper-parameter setting, GP with UMDA and GP with BUMDA, respectively.

The second experiment is aimed to observe if there exist improvements on classification performance when varying the terminal set of kernels being evolved. Three different terminal sets were employed: the set of classic kernels ($clas = \{L, R, P\}$), the set of modern kernels ($mod = \{K\_11, K\_13, H, E, W, G\}$), and all kernels ($all = \{L, R, P, K\_11, K\_13, H, E, W, G\}$). Both experiments were carried out on the same framework, following the methodology described in LISTING 4. Also, they were performed on an Intel® Core™ i9-9980XE CPU (18 cores) running at 3.0 GHz and with 16 GB of RAM. The algorithms were implemented in the Java programming language using multithreading, where each thread execute an independent call of the SEEKS algorithm. Experimental results were analysed with the Python programming language and are reported on the following section.

LISTING 4. Experimental Methodology

```
INPUT: User-defined parameters for kernel evolution, hyper-parameter ranges, number of trials, performance metrics, etc.
OUTPUT: Performance Indexes (PI: Accuracy, PSV, G-mean, etc.)
FOR EACH experimental trial $(i)$ DO:                                                   $i = 1, …, 5$
    $(K_i^{(0)}, H_i^{(0)}) \leftarrow$ Generate new initial population of kernel trees and hyper-parameters vectors      Same initial population for all algorithms
    FOR EACH $D \in Datasets$ DO:
        Partition $D$ into $F = \{Train_j, Test_j\}_{j=1}^{5}$                          5-fold cross valid.
        FOR EACH algorithm-terminal set $(A, T)$ pair RUN IN PARALLEL:              $A = \{GP, GPU, GPB\}$  $T = \{clas, mod, all\}$
            $(\boldsymbol{K}_i^*, \boldsymbol{H}_i^*) \leftarrow SEEKS_A(K_i^{(t)}, H_i^{(t)}, F, T)$        Listing 3
            $PI_{i,A,T} \leftarrow save\_file_{A,T}(\boldsymbol{K}_i^*, \boldsymbol{H}_i^*)$              nine files per dataset
```

Results and Discussion

Experimental results of all the algorithms evaluated are summarized in TABLE 6. With respect to the first experiment, from TABLE 6 one can observe that, in almost all cases, both versions of the SEEKS algorithm (GP-U and GP-B) reached higher accuracies than the GP with random hyper-parameter search. This is an important finding, because it means that the effect of an EDA coupled to the GP mechanism helps to improve the performance on classification rate of evolved SVMs. However, the information provided by the average and standard deviation on accuracy is not enough to verify if there exists any gain in efficiency; and it is insufficient to understand why the tree evolutionary methods differ.

TABLE 6. Average accuracies and standard deviations reached by the nine pairs (algorithm, terminal set). Statistics were computed from around 2500 kernels evaluated during the last five generations of each algorithm.

	algorithm	GP_clas		GP-U_clas		GP-B_clas		GP_mod		GP-U_mod		GP-B_mod		GP_all		GP-U_all		GP-B_all	
	dataset	**mean**	**std**	**mean**	**std**	**mean**	**std**	**mean**	**std**	**mean**	**std**	**mean**	**std**	**mean**	**std**	**mean**	**std**	**mean**	**std**
1	breast	97.45	0.2	**97.6**	0.2	97.51	0.2	97.51	0.2	97.59	0.2	**97.62**	0.2	97.49	0.1	97.52	0.1	**97.55**	0.1
2	chronic	**99.64**	0.4	99.46	0.3	99.63	0.4	**99.95**	0.1	99.88	0.1	**99.95**	0.1	99.87	0.2	**99.92**	0.1	99.88	0.1
3	column_2C	85.91	0.4	86.17	0.4	**86.2**	0.4	86.39	0.4	86.47	0.3	**86.92**	0.4	86.44	0.5	86.59	0.6	**86.72**	0.6
4	cryotherapy	92.55	1.6	**93.48**	1.8	93.46	1.0	95.04	1.5	95.56	1.2	**95.81**	1.3	95.19	1.5	97.24	1.2	**97.36**	0.7
5	diabetes	77.6	0.4	77.69	0.4	**77.78**	0.6	77.9	0.5	78.11	0.4	**78.12**	0.5	77.86	0.6	78.01	0.6	**78.05**	0.6
6	fertility	88.82	0.4	**88.87**	0.4	88.86	0.4	89.3	1.1	**90.14**	1.8	89.9	1.0	89.15	0.3	**89.52**	0.5	89.36	0.5
7	haberman	75.49	0.7	**75.83**	0.9	75.72	0.9	75.79	0.7	75.76	0.7	**76.38**	0.7	75.71	0.7	75.49	0.6	**76.05**	0.8
8	heart	84.23	0.7	84.25	0.7	**84.36**	0.5	85.27	0.7	85.59	0.5	**85.91**	0.6	85.04	0.6	85.43	0.8	**85.67**	0.7
9	immuno	82.1	1.3	**82.39**	1.2	**82.39**	1.1	87.08	1.2	**87.8**	1.2	87.48	1.2	85.13	0.9	86.13	0.8	**86.3**	0.5
10	liver	74.18	0.6	**74.67**	1.0	74.56	0.7	74.17	0.7	74.61	1.1	**75.09**	0.8	74.32	0.9	74.51	1.0	**75.17**	1.1
11	mammo	82.97	0.5	**83.07**	0.6	83	0.6	82.56	0.4	**83.02**	0.5	**83.02**	0.4	82.67	0.3	83.1	0.5	**83.33**	0.6
12	parkinsons	96.02	0.4	96.17	0.3	**96.23**	0.5	96.25	0.3	96.37	0.4	**96.52**	0.4	96.79	0.5	**96.89**	0.5	96.79	0.5
13	thoracic	85.16	0.2	85.15	0.1	**85.17**	0.2	85.25	0.3	85.33	0.3	**85.41**	0.4	85.33	0.5	**85.34**	0.4	85.33	0.4
14	transfusion	79.37	0.3	79.48	0.3	**79.58**	0.3	79.07	0.4	79.33	0.4	**79.43**	0.5	79.55	0.5	**79.78**	0.6	79.77	0.6
15	wpbc	82.31	0.7	82.46	0.7	**82.66**	0.7	81.54	1.2	81.66	1.0	**82.19**	1.5	81.71	0.7	82.11	0.9	**82.3**	1.0
	Per terminal set	1		7.5		7.5		0.5		2.5		12		0		5		10	
	Averall	0		0		1		0.5		2		6.5		0		2		3	
	Final Rank	6		6		4		5		3		**1**		6		3		**2**	

In order to visualize how the kernel evolution process was performed, a density estimation [57] based on the distribution of all tree kernels (without duplicates) evaluated during the last five generations of the SEEKS algorithm and the baseline GP, is presented for each dataset in Figure 2. The terminal set with all kernels was used for the three algorithms. As can be observed from the figure, most densities for the GP algorithm (red curves) present a large area to the left side, thus indicating that most of the evaluated kernels reached lower performance than those generated by GP-U (blue curves) or GP-B (green curves). An interesting case is that of the mammographic dataset, which obtained similar average accuracy on the nine algorithms tested. For this case it could be hard to determine which algorithm is preferable, but after revealing the notable skewness of GP method, it is possible to justify that the SEEKS algorithm is a better option.

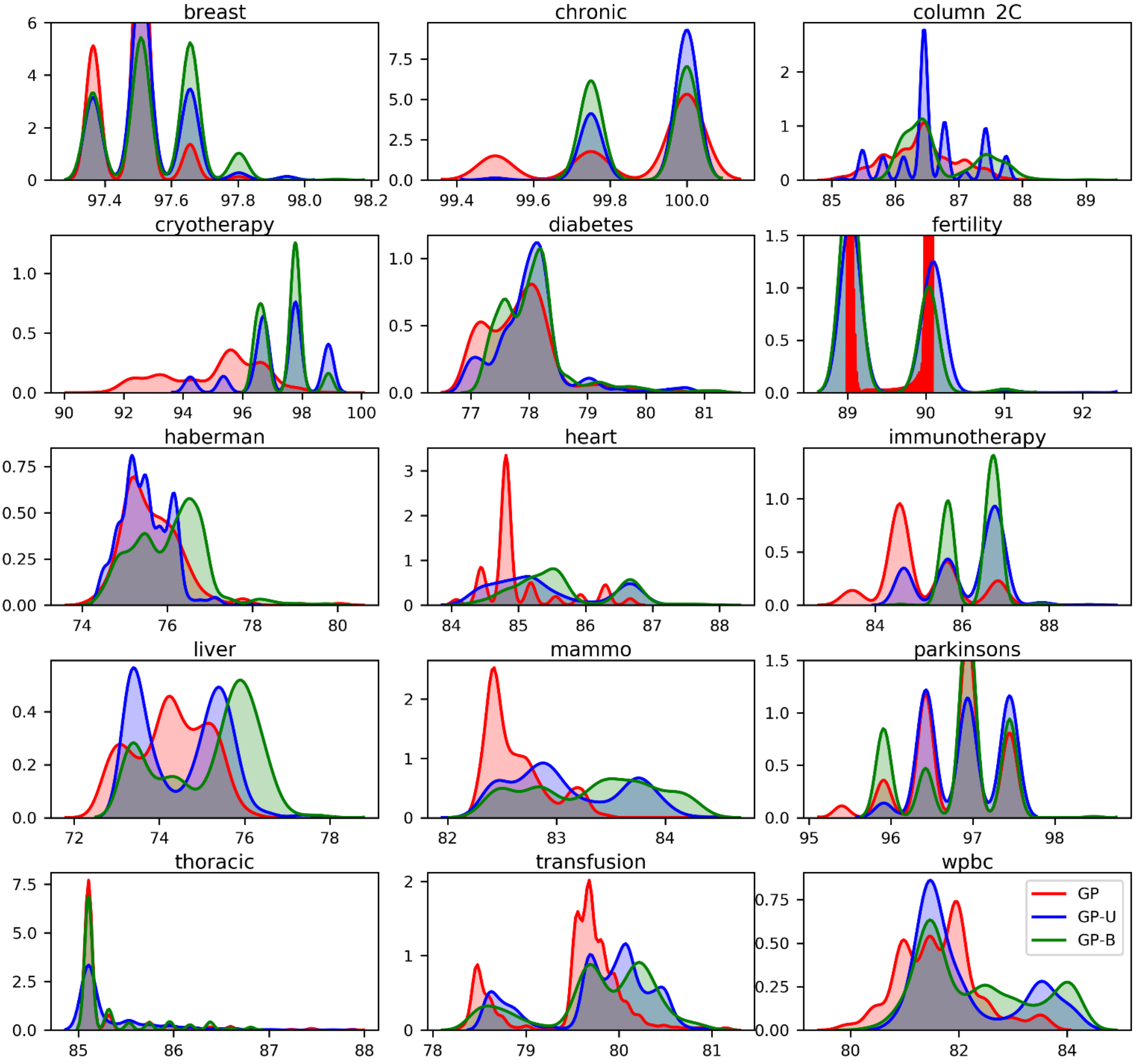

Figure 2. Density estimations based on the distribution of all SVMs with kernel trees evaluated during the last 5 generations (around 2500 SVMs minus duplicates for each dataset). Densities illustrate convergence to the best performance on accuracy (x-axis). Higher peaks to the right indicate better kernels were found.

The information synthetized in Figure 2 not only is useful to compare SEEKS against GP. It also helps to estimate how hard or easy is to solve a given datasets. For instance, datasets like breast, chronic, cryotherapy and parkinsons seems to be easily solvable, while diabetes, haberman, liver and thoracic are not.

With respect to the second experiment TABLE 6 shows that independently of the evolutionary mechanism employed, algorithms taking the modern or full set of kernels reach better performance than using the classic set. Among the nine algorithms, those formed by GP-B with modern or full set of kernels present the best results. In order to ease the comparison of terminal sets Figure 3 illustrates the max, average, and min performance on all datasets. From this figure it can be observed for which datasets the modern (blue dots) or full (green dots) set of kernels reach better or equal performance than the classic (solid red line) set of kernels.

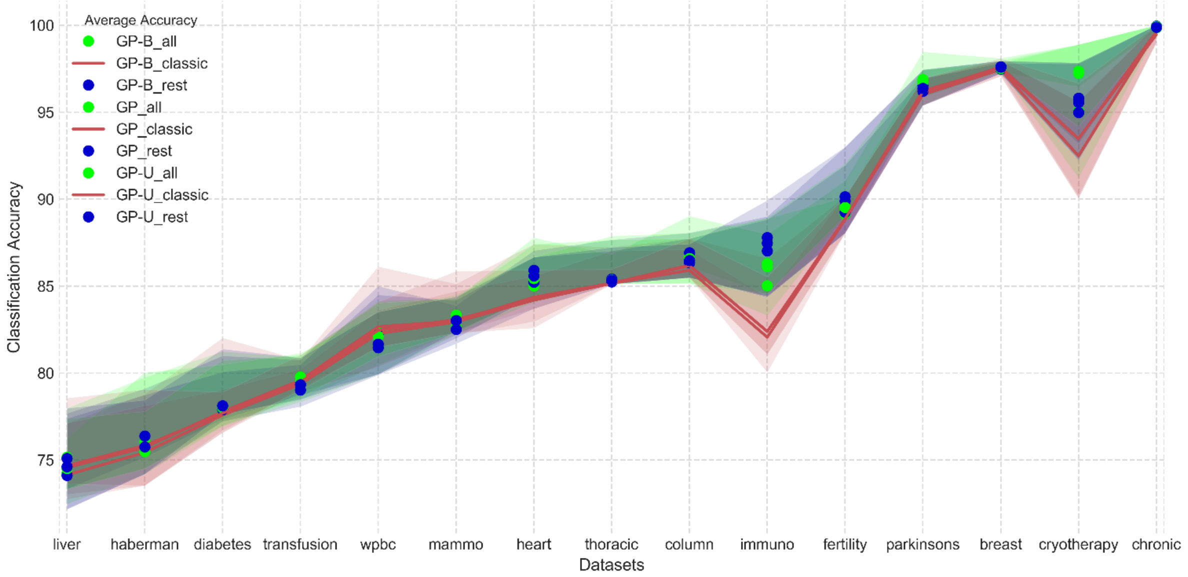

Figure 3. Performance of all algorithms sorted by accuracy and grouped by terminal set. The red solid line represents the GP with random hyper-parameter search. Shadowed areas are limited with the max and min values. The same data that in Figure 2 and Table 6 was used.

Conclusions

The main conclusion of the present study is that the proposed SEEKS method automatically designed and optimized an effective SVM classifier for each of the considered classification problems. Effectiveness relies on the fact that, for all the problems, SEEKS designed an SVM with equal or higher performance than the SVMs with an optimized single kernel or a multiple kernel produced by a standard evolutionary algorithm. The SEEKS required fewer iterations than the control method to produce SVM classifiers with said performance, showing that it is a more efficient method.

Furthermore, SEEKS mechanism solves the two problems presented when hyper-parameter tuning is delegated to the standard GP method: the search space was reduced and the control in converge was increased.

On the other hand, the introduction of new kernel families showed to be very helpful, since many of the classification problems required an orthogonal, a non-parametric or a wavelet kernel to be solved with high performance.

The SEEKS method succeeded in simultaneously performing hyper-parameter tuning, kernel selection and kernel design. Kernel selection was performed when evolving trees containing a single kernel, and kernel design when any combination of two or more kernels was tested. Hyper parameter tuning occurred at each generation, but it also served as an intensification mechanism when the SEEKS identified a single kernel that later dominated the population. This case typically occurred at late generations.

Two major directions to future improvements of this work can be stated. First, the implementation of SEEKS reported can be regarded as a proof-of-concept version of the algorithm. Many modified versions can be formulated that may lead to improved characteristics. The other possible improvement consists in using a combined index (Acc-PSV) or to reformulate the SVM evolution as a multi-objective optimization problem in order to reduce the PSV while maintaining the optimal accuracy within the evolution strategy. We are currently addressing these two aspects and are eager to report our new findings. Finally, since the SEEKS strategy is independent of the problem domain, applications to domains other than classification (such as regression or density estimation) can be easily conducted in future work

Data Availability

All pre-processed datasets, files with the performance indexes and generated kernel trees, complementary figures and codes are available at: https://github.com/padiernacarlos/SEEKS. For copyright reasons, the source code will be available once this paper is accepted for publication.

Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

Funding Statement

This work was financially supported by the following bodies: Universidad de Guanajuato under the program *Convocatoria Institucional de Investigación Científica* with the research project [**CIIC-232/2019**]; Secretaría de Innovación, Ciencia y Educación Superior under the program *Empuje Científico y Tecnológico Modalidad Apoyo a Investigadores Jóvenes* with the research project [**IJ-19-36**] and CONACYT under the program *Ciencia Básica* with the research project [**CB-2016-288605**].

Acknowledgments

The authors want to acknowledge the support provided by the División de Ciencias e Ingenierías, Universidad de Guanajuato; during the research and preparation of the manuscript.

Supplementary Materials

All supplementary materials can be found at https://github.com/padiernacarlos/SEEKS.

References

[1] W. M. Saltzman, Biomedical Engineering: Bridging Medicine and Technology, Cambridge: Cambridge University Press, 2009.

[2] S. Cogil and L. Wang, “Support vector machine model of developmental brain gene expression data for priorization of Autism risk gene candidates,” *Bioinformatics*, vol. 32, no. 23, pp. 3611-3618, 2016.

[3] L. Naranjo, C. J. Pérez, J. Martín and Y. Campos-Roca, “A two-stage variable selection and classification approach for Parkinson's disease detection by using voice recording replications,” *Computer Methods and Programs in Biomedicine*, vol. 142, pp. 147-156, 2017.

[4] F. A. Spanhol, L. S. Oliveira, C. Petitjean and H. Laurent, “A Dataset for Breast Cancer Histopathological Image Classification,” *IEEE Transactions on Biomedical Engineering,* vol. 63, no. 7, pp. 1455-1462, 2016.

[5] M. Adam, E. Y. Ng, S. L. Oh, M. L. Heng, Y. Hagiware, J. H. Tan, J. W. Tong and U. R. Acharya, “Automated characterization of diabetic foot using nonlinear features extracted from thermograms,” *Infrared Physics & Technology,* vol. 89, pp. 325-337, 2018.

[6] I. Yoo, P. Alafaireet, M. Marinov, K. Pena-Hernandez, R. Gopidi, J.-F. Chang and L. Hua, “Data Mining in Healthcare and Biomedicine: A Survey of the Literature,” *Journal of Medical Systems,* vol. 36, pp. 2431-2448, 2012.

[7] S. Maldonado and J. López, “Alternative second-order cone programming formulations for support vector classification,” *Information Sciences,* vol. 268, pp. 328-341, 2014.

[8] Y. Xu, Z. Yang and X. Pan, “A Novel Twin Support-Vector Machine With Pinball Loss,” *IEEE Transactions on Neural Networks and Learning Systems,* vol. 28, no. 2, pp. 359-370, 2017.

[9] V. Vapnik, Statistical Learning Theory, New York: John Wiley and Sons, 1998.

[10] N. Deng, Y. Tian and C. Zhang, Support Vector Machines, Boca Ratón: CRC Press, 2013.

[11] C. M. S. M. S. M. T. Gagné, “Genetic Programming for Kernel-based Learning with Co-evolving Subsets Selection.,” in *Proceedings of Parallel Problem Solving,* vol. 4193, T. Runarsson, H. Beyer, E. Burke, J. Merelo-Guervós, L. Whitley and X. Yao, Eds., Reykjavik, Reykjavik, Springer Verlag, 4193 (4193), 2006, pp. 1008-1017.

[12] P. Koch, B. Bischl, O. Flasch, T. Bartz-Beielstein, C. Weihs and W. Konen, "Tuning and Evolution of Least-Squares Support Vector Machines.," *Evolutionary Intelligence,* pp. 1-30, 2011.

[13] A. Sousa, A. Lorena and M. Basgalupp, “GEEK: Grammatical Evolution for Automatically Evolving Kernel Functions,” *Trustcom/BigDataSE/ICESS,* pp. 941-948, 2017.

[14] T. Howley and M. Madden, “The genetic kernel support vector machine: Description and evaluation,” *Artificial Intelligence Review,* pp. 379-395, 2005.

[15] K. Sullivan and S. Luke, “Evolving kernels for support vector machine classification,” in *Proceedings of the 9th annual Conference on Genetic and Evolutionary Computation,* London, 2007.

[16] A. Majid, A. Khan and A. M. Mirza, “Combination of support vector machines using genetic programming,” *International Journal of Hybrid Intelligent Systems,* vol. 3, no. 2, pp. 109-125, 2006.

[17] A. Gijsberts, G. Metta and L. Rothkrantz, “Evolutionary optimization of least-squares support vector machines,” in *Data Mining,* New York, Springer, 2010, pp. 277-297.

[18] L. Dioşan, A. Rogozan and J. Pecuchet, “Improving classification performance of support vector machine by genetically optimising kernel shape and hyper-parameters,” *Applied Intelligence,* vol. 36 , no. 2, pp. 280-294, 2012.

[19] B. Zamani, A. Akbari and B. Nasersharif, “Evolutionary combination of kernels for nonlinear feature transformation,” *Information Sciences,* pp. 95-107, 2014.

[20] R. Mantovani, A. Rossi, J. Vanschoren and B. d.-C. A. Bischl, “Effectiveness of Random Search in SVM hyper-parameter tuning,” in *International Joint Conference on Neural Networks (IJCNN),* 2015.

[21] T. Phienthrakul and B. Kijsirikul, “Evolutionary strategies for hyperparameters of support vector machines based on multi-scale radial basis function kernels,” *Soft Computing,* vol. 14, pp. 681-699, 2010.

[22] M. Zhao, C. Fu, L. Ji, K. Tang and M. Zhou, “Feature selection and parameter optimization for support vector machines: A new approach based on genetic algorithm with feature chromosomes,” *Expert Systems with Applications,* vol. 38, no. 5, pp. 5197-5204, 2011.

[23] S.-W. Lin, K.-C. Ying, S.-C. Chen and Z.-J. Lee, “Particle swarm optimization for parameter determination and feature selection of support vector machines,” *Expert Systems with Applications,* vol. 35, no. 4, pp. 1817-1824, 2008.

[24] L. Shen, H. Chen, Yu, W. Kang, B. Zhang, H. Li, Y. Bo and D. Liu, “Evolving support vector machines using fruit fly optimization for medical data classification,” *Knowledge-Based Systems,* vol. 96, no. 15, pp. 61-75, March 2016.

[25] A. Tharwat, A. E. Hassanien and B. E. Elnaghi, “A BA-based algorithm for parameter optimization of Support Vector Machine,” *Pattern Recognition Letters,* October 2016.

[26] L. C. Padierna, C. Martin, A. Rojas, H. Puga, R. Baltazar and F. Héctor, “Hyper-Parameter Tuning for Support Vector Machines by Estimation of Distribution Algorithms,” in *Nature-Inspired Design of*

Hybrid Intelligent Systems, Vols. Studies in Computational Intelligence, 667, P. Melin, O. Castillo and J. Kacprzyk, Eds., Springer, 2017, pp. 787-800.

[27] A. Rojas-Domínguez, L. C. Padierna, J. M. Carpio, H. J. Puga and H. Fraire, "Optimal Hyper-parameter Tuning of SVM Classifiers with Application to Medical Diagnosis," *IEEE Access,* vol. 6, pp. 7164-7176, 2017.

[28] J. R. Koza, Genetic programming: on the programming of computers by means of natural selection, Massachusetts: MIT press, 1992.

[29] M. Hauschild and M. Pelikan, "An introduction and survey of estimation of distribution algorithms," *Swam and Evolutionary Computation,* vol. 1, pp. 111-128, 2011.

[30] J. Shawe-Taylor and N. Cristianini, Kernel Methods for Pattern Analysis, New York: Cambridge University Press, 2004.

[31] L. Zhang, W. Zhou and L. Jiao, "Wavelet Support Vector Machine," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics),* vol. 34, no. 1, pp. 34-39, 2004.

[32] A. D. Essam and T. Hamza, "New empirical nonparametric kernels for support vector machines classification," *Applied Soft Computing,* no. 13, pp. 1759-1765, 2013.

[33] Z. Pan, H. Chen and X. You, "Support vector machine with orthogonal Legendre kernel," in *International Conference on Wavelet Analysis and Pattern Recognition*, Xian, 2012.

[34] S. Ozer, C. Chen and H. Cirpan, "A set of new Chebyshev kernel functions for support vector machine pattern classification," *Pattern Recognition,* vol. 44, no. 7, pp. 1435-1447, 2011.

[35] V. H. Moghaddam and J. Hamidzadeh, "New Hermite orthogonal polynomial kernel and combined kernels in Support Vector Machine classifier," *Pattern Recognition*, vol. 60, pp. 921-935, 2016.

[36] L. C. Padierna, M. Carpio, A. Rojas-Domínguez, H. Puga and H. Fraire, "A novel formulation of orthogonal polynomial kernel functions for SVM classifiers: The Gegenbauer family," *Pattern Recognition,* vol. 84, pp. 211-225, 2018.

[37] J. Mercer, "Functions of Positive and Negative Type, and their Connection with the Theory of Integral Equations," *Philosophical transactions of the royal society of London. Series A,* pp. 415-446, 1909.

[38] N. Christianini and J. Shawe-Taylor, An Introduction to SVM and other Kernel Based Methods., Cambridge, U.K.: Cambridge University Press, 2000.

[39] M. Gönen and E. Alpaydin, "Multiple Kernel Learning Algorithms," *Journal of Machine Learning Research,* pp. 2211-2268, 2011.

[40] E. Talbi, Metaheuristics: from design to implementation, New Jersey: John Wiley., 2009.

[41] H. Mühlenbein, "The equation for response to selection and its use for prediction. Evol. Comput.," *Evolutionary Computation,* vol. 5, no. 3, pp. 303-346, 1997.

[42] S. I. Valdez, A. Hernández and S. Botello, "A Boltzmann based estimation of distribution algorithm," *Information Sciences,* vol. 236, pp. 126-137, 2013.

[43] D. Ashlock, Evolutionary Computation for Modeling and Optimization, New York: Springer, 2006.

[44] Y. Zhang and P. Zhang, "Machine training and parameter settings with social emotional optimization algorithm for support vector machine," *Pattern Recognition Letters,* pp. 36-42, 2015.

[45] M. Tian and W. Wang, "Some sets of orthogonal polynomial kernel functions," *Applied Soft Computing,* vol. 61, pp. 742-756, 2017.

[46] S. Luke and L. Panait, "A Comparison of Bloat Control Methods for Genetic Programming," *Evolutionary Computation,* vol. 14, no. 3, pp. 309-344, 2006.

[47] D. Dua and C. Graff, *UCI Machine Learning Repository [http://archive.ics.uci.edu/ml],* Irvine, CA: University of California, School of Information and Computer Science, 2019.

[48] A. López, X. Li and W. Yu, "Support Vector Machine Classification for Large Datasets Using Decision Tree and Fisher Linear Discriminant," *Future Generation Computer Systems (36) 57-65,* vol. 36, pp. 57-65, 2014.

[49] A. Cüvitoglu and Z. Isik, "Evaluation Machine-Learning Approaches for Classification of Cryotherapy and Immunotherapy Datasets," *International Journal of Machine Learning and Computing,* vol. 8, no. 4, pp. 331-335, 2018.

[50] L. Sun, K.-A. Toh and Z. Lin, "A center sliding Bayesian binary classifier adopting orthogonal polynomials," *Pattern Recognition,* vol. 48, no. 6, pp. 2013-2028, 2015.

[51] H. l. Chen, B. Yang, S. j. Wang, G. Wang, D. y. Liu, H. z. Li and W. b. Liu, “Towards an optimal suppport vector machine classifier using a parallel particle swarm optimization strategy,” *Applied Mathematics and Computation,* vol. 239, pp. 180-197, 2014.

[52] Y. F. Hernández-Julio, M. J. Prieto-Guevara, W. Nieto-Bernal, I. Meriño-Fuentes and A. Guerrero-Avendaño, “Framework for the Development of Data-Driven Mamdani-Type Fuzzy Clinical Decision Support Systems,” *Diagnostics,* vol. 9, no. 2, p. 52, 2019.

[53] J. Zhao, Z. Yang and X. Yitian, “Nonparallel least square support vector machine for classification,” *Applied Intelligence,* pp. 1-10, 2016.

[54] M. Li, X. Lu, X. Wang, S. Lu and N. Zhong, “Biomedical classification application and parameters optimization of mixed kernel SVM based on the information entropy particle swarm optimization,” *Computer Assited Surgery,* vol. 21, no. 1, pp. 132-141, 2016.

[55] C.-C. Chang and L. Chih-Jen, “LIBSVM: A library for support vector machines,” *ACM Transactions on Intelligent Systems and Technology,* 2011.

[56] I. Tsang, J. Kwok and P.-M. Cheung, “Core Vector Machines: Fast SVM Training on Very Large Data Sets,” *Journal of Machine Learning Research,* pp. 363-392, 2005.

[57] B. Silverman, Density Estimation for Statistics and Data Analysis, Boca Raton: Routledge, 2018.