Hybridizing Cartesian Genetic Programming and Harmony Search for Adaptive Feature Construction in Supervised Learning Problems

Andoni Elola[a], Javier Del Ser[a,b,c,*], Miren Nekane Bilbao[a], Cristina Perfecto[a], Enrique Alexandre[d], and Sancho Salcedo-Sanz[d]

[a] *University of the Basque Country UPV/EHU. Alameda Urquijo S/N, 48013 Bilbao, Spain*
[b] *TECNALIA. P. Tecnologico Bizkaia, Ed. 700, 48160 Derio, Spain*
[c] *Basque Center for Applied Mathematics (BCAM), 48009 Bilbao, Spain*
[d] *Universidad de Alcalá. 28805 Alcalá de Henares, Madrid, Spain*

Abstract

The advent of the so-called Big Data paradigm has motivated a flurry of research aimed at enhancing machine learning models by following very diverse approaches. In this context this work focuses on the automatic construction of features in supervised learning problems, which differs from the conventional selection of features in that new characteristics with enhanced predictive power are inferred from the original dataset. In particular this manuscript proposes a new iterative feature construction approach based on a self-learning meta-heuristic algorithm (Harmony Search) and a solution encoding strategy (correspondingly, Cartesian Genetic Programming) suited to represent combinations of features by means of constant-length solution vectors. The proposed feature construction algorithm, coined as Adaptive Cartesian Harmony Search (ACHS), incorporates modifications that allow exploiting the estimated predictive importance of intermediate solutions and, ultimately, attaining better convergence rate in its iterative learning procedure. The performance of the proposed ACHS scheme is assessed and compared to that rendered by the state of the art in a toy example and three practical use cases from the literature. The excellent performance figures

*Corresponding author: javier.delser@tecnalia.com (Prof. Dr. Javier Del Ser). TECNALIA. P. Tecnologico Bizkaia, Ed. 700, 48160 Derio, Spain. Tl: +34 946 430 50. Fax: +34 901 760 009. E-mail: javier.delser@tecnalia.com.

obtained in these problems shed light on the widespread applicability of the proposed scheme to supervised learning with legacy datasets composed by already refined characteristics.

Keywords: Feature construction; Supervised Learning; Cartesian Genetic Programming; Harmony Search.

1. Introduction

Predictive analytics are broadly conceived as the family of supervised machine learning models aimed at inferring unknown outcomes from a system based on a set of observed variables or features [1]. Albeit supervised learning models date back to several decades ago, predictive analytics have nowadays regained momentum by virtue of the *in crescendo* availability of data in most fields of knowledge. Hot topics such as Intelligent Systems [2] and Big Data [3, 4] evince the increasing relevance of predictive modeling among different disciplines and the subsequent need for enhancing and innovating through all its compounding processing steps [5]: 1) data preparation and cleansing, with different strategies to impute missing and/or illegal data depending on their alphabet; 2) novelty/outlier detection; 3) feature processing, where the original dataset is processed/transformed/filtered so as to describe the essential features of the data and reduce the complexity of the subsequent predictive model; 4) model selection, where diverse alternatives have so far been reported in the literature characterized by different controlling parameters, training algorithms, discriminative capability and generalization properties; 5) model tuning; and 6) model performance assessment when predicting a set of unseen examples. It is only by thoroughly elaborating on each of the above steps that a good predictive model is generated.

This manuscript gravitates on the third processing step as enumerated above: feature processing. The literature has been specially profitable in this regard, with *de facto* classifications depending on the selective or constructive nature of the feature processing approach at hand. On one hand, *feature selection* schemes essentially select a subset of the original features by following strategies (filter, wrapper or embedded methods). Interestingly for the scope of this manuscript meta-heuristically empowered feature selection schemes have lately come into scene in a diversity of scenarios [6]–[13] with particular emphasis in Energy applications [14, 15] and Bioinformatics [16, 17]. On the other hand, feature extraction/construction or dimensionality reduction algorithms transform the original dataset to a feature space of fewer dimensions, which can be done by resorting to elements from linear statistics [18] or newer findings in the field of non-linear manifold learning and low-dimensional embedding [19].

This research work focuses on this second category, specifically on the construction of features via wrapper methods. This class of methods are of paramount utility when dealing with legacy datasets, i.e. datasets whose compounding features result from raw information preprocessed through application-specific signal processing stages. In such situations there is no access to the original data from which such features were extracted, hence jeopardizing the adoption of embedded schemes with known potential in highly multidimensional datasets (e.g. deep learning). The scope is also placed on the readability of the constructed features, which not only is useful for assessing mathematical properties therefrom (e.g. trends, correlations), but also becomes a requirement for certain application scenarios where supervision by a higher-level entity and/or the preservation of privacy are crucial, such as the risk assessment in bank insurance, the diagnosis of diseases and the personalized prescription of medical treatments. From a technical perspective this sought explicitness for the constructed feature set can be provided by Evolutionary Programming [20], a branch of Evolutionary Computation that aims at iteratively refining *computer programs* based on a measure of their quality or fitness. In the context of mathematical programs, this term stands for a combination or function of different variables (features) based on an alphabet of operator functions (e.g. $\{+, -, \times, \div\}$). Such programs can be represented as tree structures, which can be in turn evolved via evolutionary crossover and mutation processes towards regions of progressively higher optimality as measured by the fitness function at hand. When put in the context of feature construction, each evolved program represents a combination of features (i.e. a newly constructed feature), whereas the fitness function is given by the performance of the wrapped predictive model when trained with the evolved feature set. Indeed this has been the technical approach followed by a number of contributions by the research community where the good performance of Evolutionary Programming has been evinced in diverse practical applications of predictive modeling (see [21]–[30] and the comprehensive survey in [31]).

The work presented in this paper takes a step further in the state of the art in the above field by proposing a novel wrapper approach based on the combination of Cartesian Genetic Programming [32] and Harmony

Search (hereafter denoted as HS, [33]). On the one hand, Cartesian Genetic Programming permits to encode (represent) programs by means of strings of integers, which numerically encode the operators that relate variables to each other, their connections to the set of input features and the resulting output features fed to the model. On the other hand, Harmony Search is a meta-heuristic solver that has been widely shown to outperform other bio-inspired optimization algorithms in many applications [34]. In this manuscript we propose to blend together these two techniques to yield a feature construction wrapper that in addition, exploits information about the predictive relevance of the produced feature set so as to enhance the convergence properties of the overall search process. The performance of the derived feature construction scheme is evaluated over four supervised learning problems – namely, the well-known `WINE` dataset, leaf-based plant classification (`LEAF`, [35]), classification of radar returns from the ionosphere (`IONOSPHERE`, [36]) and vehicle type recognition (`VTR`, [37]) – with results that dominate the best scores obtained to date. To the best of the authors' knowledge, this is the first contribution in the literature hybridizing Cartesian Genetic Programming with Harmony Search for feature construction in supervised learning.

The rest of the paper is structured as follows: Section 2 formally poses the construction of explicit features in supervised learning scenarios as a mathematical optimization problem. Next, Section 3 and subsections therein delves into the proposed algorithmic approach by outlining its overall working procedure and detailing each of its compounding modules. Experimental results over the four considered datasets are presented and discussed in Sections 4 and 5 and, finally, Section 6 ends the paper by drawing conclusions and sketching several lines of future research.

2. Feature Construction as an Optimization Problem

Mathematically speaking a supervised learning problem departs from a set of available data instances $\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^N$, with N denoting the number of instances or examples, x_n^d the d-th feature for example n and $D \doteq |\mathbf{x}_n|$ $\forall n \in \{1, \ldots, N\}$ the number of features or *dimensionality.* Since we deal with supervised learning, samples in $\mathbf{X}$ are associated to a value of the target variable to be predicted, which are all collected in the label vector $\mathbf{y} \doteq \{y_n\}_{n=1}^N$. The goal of a supervised learning algorithm is to infer the pattern relating $\mathbf{x}_n$ to its corresponding label y_n. This can be accomplished by a model $M_{\boldsymbol{\theta}} : \mathcal{X}^D \mapsto \mathcal{Y}$ that maps a given data instance or sample $\mathbf{x}$ to its estimated target variable y. The model $M_{\boldsymbol{\theta}}(\cdot)$ can be constructed (trained, learned) based on a set of training examples $\{\mathbf{X}^{tr}, \mathbf{y}^{tr}\} \subset \{\mathbf{X}, \mathbf{y}\}$ and the parameters $\boldsymbol{\theta}$ of the model at hand. The remaining data samples $\mathbf{X}^{test} \doteq \mathbf{X} - \mathbf{X}^{tr}$ and their supervised labels are left out for testing the predictive performance $\Psi(\mathbf{y}^{test}; \mathbf{y}^{pred})$ of the model when processing unseen data. The design is hence to maximize this performance measure. To this end, a common practice is to randomly partition the training set into K folds, retain a subset for validating the model and use the remaining $K-1$ subsets as training data. By repeating this process K times a cross-validated estimate of model prediction performance can be computed, which is utilized for tuning the parameter configuration $\boldsymbol{\theta}$ of the model.

In this manuscript the maximization of the average performance of the model is approached by constructing a new feature set $\mathbf{X}' \doteq \{\mathbf{x}'_n\}_{n=1}^N$ with $D' \doteq |\mathbf{x}'_n|$, such that a model $M' : \mathcal{X}^{D'} \mapsto \mathcal{Y}$ can be trained to yield a better performance score than the same model when fed with the original set of features $\mathbf{X}$. As mentioned in Section 1 the scope of this work is placed on the construction of readable features based on the original data $\mathbf{X}$ and a set of explicit mathematical operators $\mathcal{F} \doteq \{f_1, \ldots, f_P\}$. Each sample $\mathbf{x}_n \in \mathbf{X}$ is mapped to the space spanned by a set of constructed features $\{x^{d,\prime}\}_{d=1}^{D'}$, each expressed as an combination of the original feature set $\{x^d\}_{d=1}^{D}$ through a operator subset $\mathcal{F}^{d,\prime} \subseteq \mathcal{F}$. For instance, if $\mathcal{F} = \{+, -, \times, \div, \cos, \sin, \exp\}$ and $D = 7$, examples of constructed features with $D' = 3$ could be given by

$$x^{1,\prime} = \cos(x^1 + x^2) + x^3 \times \exp(x^5 - x^2) \quad (\mathcal{F}^{1,\prime} = \{+, -, \times, \cos, \exp\}), \tag{1}$$

$$x^{2,\prime} = \exp(x^3 - \sin(x^5)) \quad (\mathcal{F}^{2,\prime} = \{-, \cos, \sin\}), \tag{2}$$

$$x^{3,\prime} = \cos(x^2 \times (x^1 - x^7)/x^4) \quad (\mathcal{F}^{3,\prime} = \{-, \times, \div, \cos\}). \tag{3}$$

It should be clear that any given operator subset $\mathcal{F}^{1,\prime}$ can result in different expressions, as the original features involved and their relative order in the expression may vary without impacting on the operator subset at hand. This is the reason why such combinations (*programs*) are often represented as trees, with variables (original features) located at leaf nodes and operators at intermediate nodes. The transformed dataset $\mathbf{X}'$ is next used for training and tuning a model $M'_{\boldsymbol{\theta}}$ towards achieving a better performance score than that operating on $\mathbf{X}$ under the following criterion:

$$\underset{\{x^{d,\prime}\}_{d=1}^{D'}}{\text{Maximize}} \; \Psi_{\boldsymbol{\theta}}^{val} \doteq \mathbb{E}\left\{\Psi\left(\mathbf{Y}^{val}; M'_{\boldsymbol{\theta}}\left(\mathbf{X}^{val,\prime}\right)\right)\right\}, \tag{4a}$$

$$\text{subject to} \;\; x^{d,\prime} \in \mathcal{X}^{d,\prime} \; \forall d \in \{1,\ldots,D'\}, \tag{4b}$$

$$x^{d,\prime} = \Omega^d(\mathcal{F}^{d,\prime}, \{x^d\}_{d=1}^{D}), \; \forall d \in \{1,\ldots,D'\}, \tag{4c}$$

$$\mathcal{F}^{d,\prime} \in \mathcal{F}, \; \forall d \in \{1,\ldots,D'\}, \tag{4d}$$

where $\mathbb{E}\{\,\cdot\,\}$ denotes expectation over every possible validation subset $\mathbf{X}^{val,\prime} \subset \mathbf{X}'$, which can be approximated by the average score produced by the aforementioned K-fold cross-validation procedure. In words, the optimal features $\{x^{d,\prime}\}_{d=1}^{D'}$ will be given by those constructed upon $\{\mathcal{F}^{d,\prime}\}_{d=1}^{D'}$ and leading to an expected maximal cross-validated performance score when used as predictive inputs to a supervised learning model $M'_{\boldsymbol{\theta}}(\,\cdot\,)$. In the above set of constraints Expression (4c) implicitly defines $\Omega^d(\mathcal{F}, \{x^d\}_{d=1}^{D})$ as a particular ordered arrangement of the operators contained in $\mathcal{F}^{d,\prime}$ and the variables in $\{x^d\}_{d=1}^{D}$. The number of different operator subsets $\mathcal{F}^{d,\prime}$, original features finally involved in the constructed characteristic $x^{d,\prime}$ and the specific arrangement of these variables within the constructed feature gives rise to a search space whose large dimensionality requires computationally efficient solvers capable of learning and evolving programs towards regions of progressively increased optimality. This is indeed the main purpose of the proposed Adaptive Cartesian Harmony Search (ACHS) approach, which is next described.

3. Proposed Feature Construction Approach

In order to tackle the above problem in a computationally efficient fashion we propose a novel feature construction algorithm whose overall working procedure is illustrated in Figure 1 and algorithmically described in Algorithm 1. The proposed scheme blends together elements from wrapper and embedded methods for feature processing. On one hand, the setup relies on a predictive learning model $M_{\boldsymbol{\theta}}$ capable of internally estimating the relevance of each input variable when predicting the target variable at hand. This estimation can be accomplished as an inherent result of the training procedure of the model itself (as in e.g. tree models) or by incorporating side processes aimed at this end, such as the so-called Relief approach later explained in the manuscript. This estimated relevance is exploited at a wrapper optimization approach, which iteratively evolves a set of *programs* that represent explicit expressions of constructed variables. The dataset resulting from each potential set of transformed features proposed by the solver is evaluated by the predictive model, whose cross-validated score is adopted as the fitness function that drives the search procedure. At this point it should be remarked that the novelty of our work does not only resides in the specific implementation of the general scheme in Figure 1, but rather in the exploitation of the predictive relevance of the constructed features to enhance the convergence behavior of the search procedure.

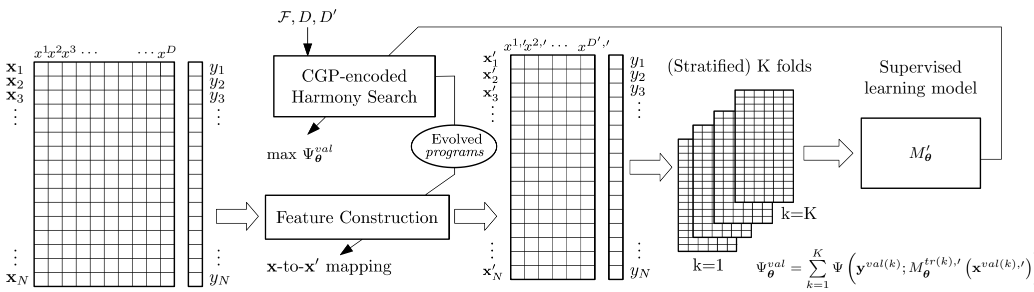

Figure 1: Overall diagram of the proposed feature construction approach.

We now elaborate on a specific implementation of the above feature construction framework that couples the widely reported good search capability of Harmony Search as a meta-heuristic solver with the constant-length program encoding strategy known as Cartesian Genetic Programming. The search procedure of the former will be utilized as the core heuristic of the meta-heuristic optimization wrapper, whose evolved solutions will be represented by the latter. This specific implementation of the proposed framework constitutes the ACHS scheme for constructing predictively optimal explicit feature sets.

3.1. Optimization Wrapper: Harmony Search

First proposed by Geem et al [33], HS is a human-experience-derivative-based meta-heuristic solver that has been shown to perform satisfactorily in several application domains [34]. In essence the operators that drive the HS search procedure work by evolving partial solutions in a similar fashion to conventional crossover and mutation operators in Genetic Algorithms. However, as will be shown later in the article it is in the inspiration found in the musical improvisation process such as note pitch and harmony memory and its impact on the definition of such operators where HS differs from other

Algorithm 1: Proposed feature construction algorithm (ACHS).

Data: Data instances $\{\mathbf{x}_n\}_{n=1}^N$, with $\mathbf{x}_n = \{x_n^d\}_{d=1}^D$, set of operators $\mathcal{F}$, number of constructed features D', number of iterations $\mathcal{T}$.

Result: Data instances $\{\mathbf{x}'_n\}_{n=1}^N$ with constructed features $\{x^{d,\prime}\}_{d=1}^{D'}$.

Initialize φ CGP-encoded individuals (harmonies) at random;
Set feature importances w_ϕ^d to 1 for $d \in \{1, \dots, D'\}$ and $\phi \in \{1, \dots, \varphi\}$;
for $t \leftarrow 1$ **to** $\mathcal{T}$ **do**
    Apply HS operators (Subsection 3.1) using the $w_{\boldsymbol{\theta}}^d$-dependent progression law in Expression (8) (Subsection 3.3);
    for $\phi \leftarrow 1$ **to** φ **do**
        Transform data instances $\{\mathbf{x}_n\}_{n=1}^N$ to $\{\mathbf{x}'_n\}_{n=1}^N$ based on the set of constructed features represented by the ϕ-th harmony newly improvised by the CGP-encoded HS solver;
        Compute a cross-validated score for underlying model $M_{\boldsymbol{\theta}}$;
        Extract averaged feature importances w_ϕ^d computed over the trained model corresponding to every fold;
    end
    Concatenate and sort the previous and newly improvised harmonies by their cross-validated score;
    Filter out the worst φ harmonies, including their feature importance vectors (Subsection 3.3);
end
The dataset $\{\mathbf{x}'_n\}_{n=1}^N$ composed by features $\{x^{d,\prime}\}_{d=1}^{D'}$ is given by the first CGP-encoded harmony within the memory of harmonies;

evolutionary algorithms [38, 39]. Due to its population-based nature HS relies on a set of candidates φ that are iteratively refined by means of intelligent combinations and mutations. The Harmony Memory (HM) consists of the candidates that sound better (under a fitness criterion) along time. The HM is updated whenever any of the φ improvised harmonies sounds better than any of the φ harmonies kept in the HM.

Before proceeding further with the fundamentals of this meta-heuristic algorithm, it should be noted that HS and in general other bio-inspired optimization methods have lately been involved in a hot debate within the research community. Some investigations have questioned the inherent novelty of metaphor-based optimization approaches in light of their resemblance to foundational methods from Evolutionary Computation [40]–[43], subsequently unleashing several rebuttals against the criticism shed over such nature-inspired heuristics [44]–[46]. Recently more constructive proposals have been made in the literature to standardize the functional definition of meta-heuristics [47, 48]. Nevertheless, the work presented in this paper is focused on the impartial use of HS as the heuristic engine to evolve CGP-encoded programs, i.e. without entering any debate on the originality of this meta-heuristic algorithm.

Going back to the algorithmic steps underlying the HS solver, its seminal version as contributed in [33] begins by initializing the set of φ stored harmonies in the HM. This is usually accomplished by sampling each *note* of every harmony (i.e. each entry of every solution) uniformly at random from its corresponding alphabet. Once this is done several improvisation operators are applied to each note of the HM for a given number of iterations[1] $\mathcal{T}$:

- The Harmony Memory Considering Rate (HMCR), driven by the probabilistic parameter $\xi \in [0, 1]$, which sets the probability that the new value for a certain note is drawn from the values of this same note in the other $\varphi - 1$ harmonies.

- The Pitch Adjustment Rate (PAR), controlled by the probabilistic parameter $\delta \in [0, 1]$ which sets the probability to execute subtle adjustments in the chosen harmony by improvising the probability to replace the actual value of the note at hand with any of its neighboring values within the alphabet on which it is defined. Many pitch adjustment approaches have been reported in the literature dealing with integer- and real-valued variables (see e.g. [49, 50] and references therein). Without loss of generality in what follows we will resort to the seminal definition of the pitch adjustment operator established in [33], although other pitch adjusting policies can be also considered.

- More recently, another operator has been proposed in the literature, Random Selection Rate (RSR, $\upsilon \in [0, 1]$), which establishes the probability that the new value for a given note will be drawn uniformly at random (i.e. without any neighborhood consideration) from its corresponding alphabet [38].

[1] In practice any other stop criterion can be imposed, but in any case it reduces to the set of operators being applied for a number of iterations given by the adopted criterion.

At this point it is important to remark that a good convergence behavior of the HS solver when applied to a given problem depends roughly from several algorithmic aspects. To begin with, a minimum-redundancy solution encoding strategy should be designed, capable of spanning the whole search space while maintaining at the same time the neighborhood relationship between locally close solutions in their encoded representations. Likewise the solver should meet a balance between explorative and exploitative search well-suited to the problem being tackled, which can be attained via a proper selection of the parameters guiding the HS search process. Regarding this latter aspect, there are different parameter tuning strategies, from the most naive approach (i.e. keeping the HS parameters $\{\xi, \delta, \upsilon\}$ fixed to a value) to more elaborated schemes, such as progressively varying the value of the parameters along iterations. In [51] a linear progression of HS parameters with the iteration index t was presented. More recently, the authors in [52] proposed a more flexible logarithmic progression that generalizes its linear counterpart and allows for tuning the convexity of the parameter progression in the range $t \in \{1, \ldots, \mathcal{T}\}$ by using a new design parameter $\zeta \in \mathbb{R}^+$. By denoting starting and ending values of the parameters with sub-indexes s and e respectively, such a progression is given by

$$\eta(t) = \eta_s \left[1 - \vartheta(\eta_s, \eta_e) \left(\frac{\log(t-1)}{\lambda(\eta_s, \eta_e, \zeta) \log(\mathcal{T}-1)} \right)^{\frac{1}{\zeta}} \right], \tag{5}$$

where $\eta \in \{\xi, \delta, \upsilon\}$, $\vartheta(\eta_s, \eta_e) \triangleq \text{sgn}(\eta_s - \eta_e)$,

$$\lambda(\eta_s, \eta_e, \zeta) \triangleq \left[\vartheta(\eta_s, \eta_e) \left(1 - \frac{\eta_e}{\eta_s} \right) \right]^{-\zeta}, \tag{6}$$

and $\text{sgn}(x) = 1$ if $x \geq 0$ (-1 otherwise). The fact that the above logarithmic progression is controlled by just one parameter allows tuning the convexity of the parameter progression along iterations. This provides a higher degree of design flexibility for the overall algorithm at an affordable computational penalty in terms of the time taken to refine such a new parameter.

3.2. Solution Encoding: Cartesian Genetic Programming

In what relates to solution encoding, it is important to recall that the particularly adopted strategy should represent explicit combinations of the input variables in a numerical fashion that keeps most of the expected properties for an encoding to be suited for HS (i.e. minimum representation, constant length, redundancy and vicinity relationship among values for a given note driven by the fitness at hand). These requirements are met by Cartesian Genetic Programming (CGP), a highly efficient and flexible form of Genetic Programming [32]. CGP represents computational structures as a string of integers and can easily encode computer programs, electronic circuits [53], neural networks [54, 55], mathematical equations and other computational structures [56, 57]. In this encoding integers are used to encode the function nodes in the graph, the connection between nodes (variables), the connections to inputs and the locations in the graph where outputs are taken from. In other words, the genotype of CGP is a fixed-length list of integers values, from which its phenotype (i.e. the program it represents) is inferred.

The types of computational node functions used in CGP are decided by the user and are listed in a look-up table indexed by integer values. Each node represents a function and is encoded by a number of genes. One of them is the address of the computational node function in the look-up table. The remaining node genes indicate where the node gets data from. These genes represent addresses, known as connection genes. They take their inputs in a feed-forward manner from either the output of nodes in a previous column or from a program input. The program data inputs are given by $\{x^1, \ldots, x^D\}$, where D is the number of program inputs. Each node output has also an address and this can be the input of the next node. The selection of the number of intermediate nodes and their arrangement in terms of number of hidden layers and amount of nodes per layer are flexible and can be tailored for the problem at hand.

Figure 2 exemplifies the process to build mathematical expressions. There are three type of nodes: input nodes (variables), output nodes (programs) and function nodes. As described above, the user defines the functions and the layout of the CGP. In Figure 2 there are two input nodes (x^1 and x^2), nine hidden nodes and two output nodes ($x^{1,\prime}$ and $x^{2,\prime}$). Each genotype is composed by three genes, the first one indicates the function (defined in the table) and the remaining genes define the input data. In some cases, the last number is marked with a special symbol ($\Diamond$ in Figure 2) to indicate that it is unused. Output nodes have only a single gene each, which indicates the node from which there is an incoming edge.

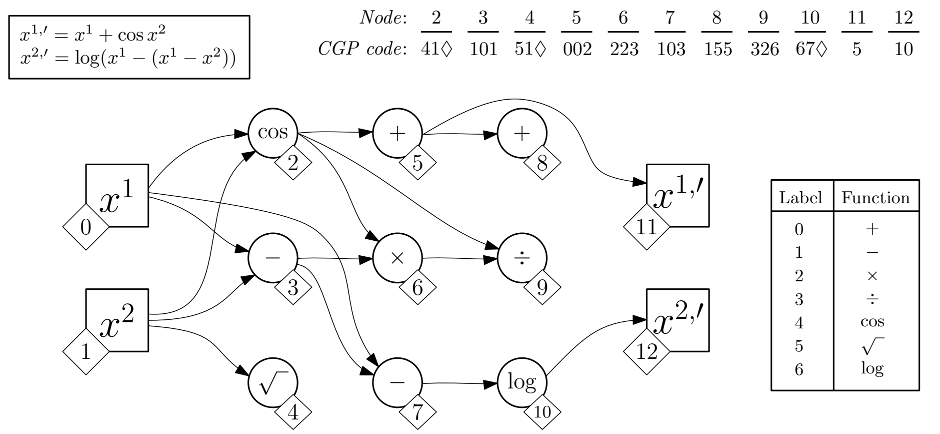

Figure 2: General form of CGP for representing mathematical expressions. Following the notation in Section 2, $\mathcal{F} = \{+, -, \times, \div, \cos, \sqrt{\ }, \log\}$, $D = 2$ and $D' = 2$.

3.3. Exploiting Predictive Relevance in the Feature Optimization Process

Besides hybridizing CGP and HS, another novel ingredient of the ACHS algorithm proposed in this paper is the exploitation of the predictive relevance of the iteratively constructed feature set in the search procedure of the heuristic wrapper. By considering an original feature set represented by the variable vector $\mathbf{x} = \{x^d\}_{d=1}^{D}$, the memory of program candidates HM is initialized by encoding each element using CGP, which creates a new constructed feature set $\mathbf{x}' = \{x^{d,\prime}\}_{d=1}^{D'}$ using the scheme exemplified in Figure 2. As such, each row of the HM (i.e. each candidate solution) is a newly constructed feature set built upon a combination of the original features $\mathbf{x}$ through a subset of the overall operator subset $\mathcal{F}$. In order to drive the search process of the HS solver, each of the rows in the HM is evaluated by using a supervised learning model $M(\cdot)$, whose cross-validated average prediction score (e.g. accuracy, R^2 or any other metric alike) measured over the given dataset is adopted as the fitness function of the HS algorithm.

Before starting the next generation, the predictive importance or relevance of each feature within $\mathbf{x}'$ is computed either from the trained model or by resorting to algorithmic approaches specifically designed to this end. In this regard some supervised learning models such as tree-based classifiers allow for numerically assessing the predictive power of each input feature by quantifying the information gain (or alternatively, the so-called Gini impurity) for all branches below such a feature, and optionally weighting the gain at each branch by the number of samples classified thereby. Universal esti-

Algorithm 2: Original Relief algorithm.

Data: Data instances $\{\mathbf{x}_n\}_{n=1}^{N}$, with $\mathbf{x}_n = \{x_n^d\}_{d=1}^{D}$.

Result: Feature relevances $\mathbf{w} = \{w^d\}_{d=1}^{D}$.

Initialize all weights in $\mathbf{w}$ to zero, i.e. $w^d = 0\ \forall d \in \{1, \ldots, D\}$;
for $z \leftarrow 1$ **to** Z **do**
  Randomly select an instance $\mathbf{x}_n$, with $n \in \{1, \ldots, N\}$;
  Find nearest hit $\mathbf{x}_n^{\mathcal{H}}$ and nearest miss $\mathbf{x}_n^{\mathcal{R}}$;
  for $d \leftarrow 1$ **to** D **do**
    $w^d = w^d - \delta^2(\mathbf{x}_n, \mathbf{x}_n^{\mathcal{H}}) + \delta^2(\mathbf{x}_n, \mathbf{x}_n^{\mathcal{R}})$;
  end
end

mators for inferring the predictive relevance in general supervised learning models have been also proposed in the literature. For instance, in [58, 59] an algorithm coined as Relief was shown to estimate attributes according to how well their values distinguish among the instances that are near to each other. Given a data instance $\mathbf{x}_n$, Relief searches for its two nearest neighbors: one for the same class (nearest hit, $\mathbf{x}_n^{\mathcal{H}}$) and the other from a different class (nearest miss, $\mathbf{x}_n^{\mathcal{R}}$). The original algorithm selects Z training instances, where Z is an user-defined parameter. By denoting the normalized difference between the values of the d-th attribute for two instances $\mathbf{x}_n$ and $\mathbf{x}_{n'}$ as $\delta(x_n^d, x_{n'}^d)$, the normalized predictive relevance $w^d \in [0, 1]$ for the d-th attribute is iteratively updated as

$$w^d = w^d - \delta^2(\mathbf{x}_n, \mathbf{x}_n^{\mathcal{H}}) + \delta^2(\mathbf{x}_n, \mathbf{x}_n^{\mathcal{R}}), \tag{7}$$

which is repeated Z times as summarized in Algorithm 2. The algorithm was later extended giving rise to ReliefF [60], a more robust version of its predecessor that can tolerate incomplete and noisy data and manage multiclass problems by finding one near miss for each different class and averaging their contribution for updating the importances $\mathbf{w}$.

When put in the context of our article, it should be obvious that each row $\phi \in \{1, \ldots, \varphi\}$ (harmony) of the population of harmonies iteratively evolved by the HS solver will correspond to a different set of newly proposed features with potentially higher predictive power. Therefore the normalized relevance of the d'-th feature in harmony ϕ will be given by $w_\phi^{d'} \in [0, 1]$. This predictive information is exploited by redefining the progression law of two of the operators of HS (namely, ξ and υ) so as to reflect the fact that when a newly improvised feature is found to be relevant for the supervised problem in question, the HS solver should not move far away from the *vicinity* of the program upon which it is defined. Based on this rationale the progression in Expression (5) is modified as

$$\eta_\phi^{d'}(w_\phi^{d'}) = \eta_s\left[1 - \left(\frac{\log(w_\phi^{d'}+1)}{\lambda(\eta_s,\eta_e,\zeta)\log(2)}\right)^{\frac{1}{\zeta}}\right], \tag{8}$$

where $\eta \in \{\xi, \upsilon\}$ and the arbitrary factor ζ depends on the generation. Without loss of generality in this work we assume a linearly increasing progression of ζ with the iteration index t as

$$\zeta(t) = \zeta_s + \frac{\zeta_e - \zeta_s}{\mathcal{T}-1}(t-1), \tag{9}$$

whose impact on $\eta_\phi^{d'}(w_\phi^{d'})$ is plotted in Figure 3 for $\eta_s = 0.1$, $\eta_e = 0.75$ and different values of t. Note that η is different for each element of the HM depending on its relative importance and the iteration index. It should be also remarked that the proposed scheme is flexible enough to accommodate any arbitrary form of dependence between ζ and t.

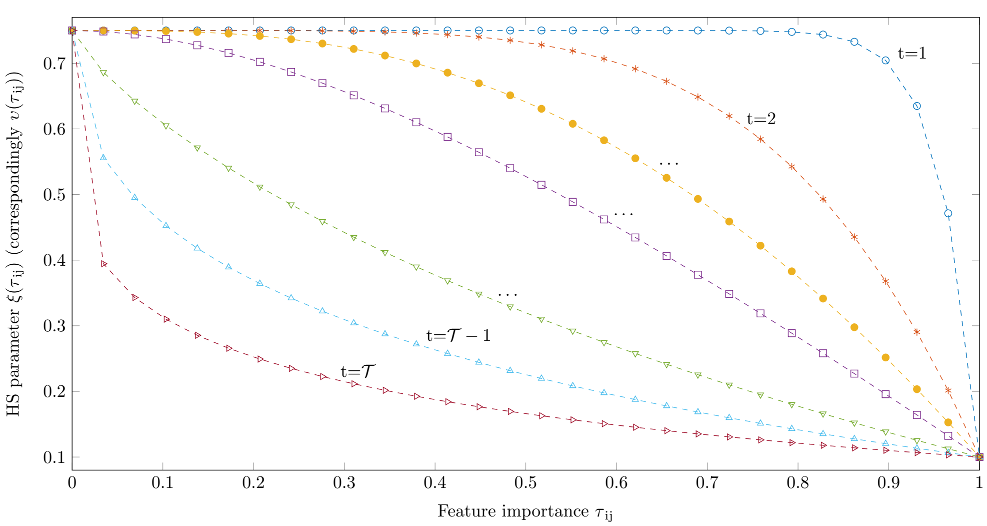

Figure 3: Logarithmic progression of the HS parameter $\eta_\phi^{d'}(w_\phi^{d'})$ versus the feature importance $w_\phi^{d'}$ for several values of ζ assuming a linear progression of this latter parameter with the generation index t.

4. Cases of Study and Learning Models

As has been already mentioned in the introduction the performance of the proposed ACHS approach has been experimentally assessed over four different datasets:

- `WINE` dataset: this is a relatively small dataset consisting of $N = 178$ samples and $D = 13$ original features [61]. The data correspond to results of a chemical analysis of wines grown in the same region but delivered from three different cultivars. The aim is to classify among 3 different classes of wine. This first well-known dataset serves as a toy example to elucidate the expected computational benefits of constructing new feature sets with ACHS.

- `LEAF` dataset: this is a publicly available repository for the classification of plants based on morphological features extracted from digitally processed leaf images [35]. Specifically, samples in this datasets correspond to specimens of different plant species, whose features are built upon automatically segmented images of 720×960 pixels, which have been further inspected to ensure comparability between different plant classes. This process results in $N = 171$ samples of $D = 15$ features corresponding to 15 different plant species [62]. Results obtained with this dataset aim at verifying whether feature importances do make a difference in terms of convergence when fed back to the solver and utilized to tailor the values of the HS parameters.

- `IONOSPHERE` dataset: these data correspond to signals recorded by a phased array of 16 high-frequency antennas. The aim is to find evidence of free electrons in the ionosphere, in such a way that "good" radar returns are those signals that are evidence of the presence of some structure in the ionosphere, whereas "bad" radar returns are those signals that did pass through the ionosphere as a result of the lack of structure in this atmospheric shell. The dataset is comprised by a total of $D = 34$ features and $N = 351$ instances. More details on the dataset are provided in [36]. This third dataset will be used to evaluate to which extent the progression of ζ with t impacts on the convergence of the algorithm through the resulting concavity of the curves relating the feature relevance to the value of the HS parameter.

- `VTR` dataset: in this case the objective is to classify among different classes of vehicles using recorded engine sounds. The original database for the experimental work consists of $N = 574$ vehicle sounds corresponding to 311 cars, 88 motorbikes and 175 trucks. The sound of each vehicle has been recorded in a single file with a sampling frequency of 11025 kHz and 16 bits per sample. All sound samples have been recorded in flat two-way road sections with a speed limit of 50 kilometers per hour, what prevents them from gear shifting and acceleration. Only a segment of 1 second length has been considered for classification, centered on the instant where the vehicle is passing exactly in the front of the capturing sensor array. All sound files have been divided into frames of 512 samples each, from which $D = 66$ features have been extracted. All features are described in more detail in [37]. This case study is included to show that the constructed characteristics are universally good across different classifiers, i.e. the produced feature set possesses higher predictive power when input to classifiers that are different than the one utilized for their generation.

In this paper we used several supervised learning models to classify data and measure the importance of each input feature. One of such techniques is Random Forest (RF), one of the most utilized classification and regression algorithms by virtue of its ability to classify large datasets with excellent accuracy and small chance to overfit. RF splits the data into a number of subsets and creates a tree for each of them (weak learners), from which the predicted output of the RF is taken by aggregating the individual predictions of all its compounding trees. RF also provides an embedded method for quantifying the values of the feature importances: the decrease in the classification margin is computed if the variables are permuted among weak learners. Results are averaged over the entire ensemble and divided by the standard deviation [63].

The second algorithm utilized in the experiments is the 1-Nearest Neighbor (1-NN) algorithm: it simply classifies a new example $\mathbf{x}_n$ by finding the training example $(\mathbf{x}_{n'}, y_{n'})$ that is closest to $\mathbf{x}_n$ according to a distance measure. Unless otherwise stated, all tested models assume the conventional Euclidean distance. Another well-known classification method included in the benchmark is the well-known Support Vector Machine, which essentially finds a gap in the feature space that is as wide as possible between data samples corresponding to different categories. Such a gap can be set to be linear or instead, can be shaped arbitrarily by using the so-called *kernel trick*, which indirectly maps samples to high-dimensional feature spaces where the points can be separated linearly. For multi-class classification, one-against-all or one-against-one strategies can be used [64]. The last classification method considered in this paper is the Extreme Learning Machine (ELM, [65]). This is a novel and fast learning method based on the structure of multi-layer perceptrons. The most significant characteristic of the ELM training procedure lies on the fact that is carried out just by randomly setting the weights connecting the inputs to the nodes of the hidden layer, and then obtaining the pseudo-inverse of the matrix linking the hidden layer to the outputs of the model.

For the experimental setup, the starting and ending values of the HS parameters were fixed: $\xi_e = 0.1$, $\xi_s = 0.75$, $\upsilon_e = 0.05$ and $\upsilon_s = 0.15$. The CGP tree for feature construction was also fixed. The starting and ending values of the arbitrary factor ζ, the size of the HM, the number of generations $\mathcal{T}$ and other parameters of the model – $\boldsymbol{\theta}$ in Expression (4a) – were optimized for each problem via off-line simulations over a value grid, with the maximization of the average cross-validated accuracy score of $M_{\boldsymbol{\theta}}$ when fed with the produced features by HS as the criterion to select one configuration or another[2]. Weights and biases of the 50 hidden neurons of the ELM model utilized for the `VTR` dataset have been drawn uniformly at random from the ranges $[-1, 1]$ and $[0, 1]$, respectively. The number of tree learners in the RF models has been set to 100 in all cases. The remaining parameters of all models in the benchmarks have been tuned using a procedure similar to the above exhaustive grid search with cross-validation. Features of all datasets have been normalized via Z-scores.

5. Results and Discussion

5.1. Results obtained with the `WINE` Dataset

We trained two different ACHS-based supervised learning models to obtain $D' = 3$ features with enhanced predictive power when classifying the samples within this dataset. For the first one, we used RF to measure feature importance and to classify the data (namely ACHS_1 model). The

[2]Further details on the utilized value grid, the final configuration used in the experiments and the source code can be provided upon demand to the corresponding author.

second option was to use 1-NN for classification purposes and ReliefF to compute feature importance (ACHS_2). Both ACHS_1 and ACHS_2 were compared with the results obtained with Principal Component Analysis (PCA) and the results obtained selecting the 3 most important variables according to the importance estimated by RF and ReliefF. To obtain the optimum feature expressions, we set apart 25% of the dataset for test and we used 10-fold stratified cross-validation over the remaining 75% of the dataset. We computed all results using $\mathcal{T} = 50$ iterations, changing the folds at each iteration to avoid overfitting. Both cross-validated accuracy scores – represented as *median (Interquartile Range, IQR)* – and those obtained for the test set are reported in order to check whether the cross-validated score of each algorithm is representative of the score obtained for the test set.

Table 1 summarizes the results obtained by each algorithm for a given train/test split. The worst results were obtained by those classifiers based on ReliefF-based feature selection, with cross-validated accuracy scores ranging from 91.56% (90.51%-92.61%) using RF to 90.69% (89.86%-91.59%) using 1-NN. By using the three most important features selected by the RF algorithm, the figures of merit when RF and 1-NN are used as predictive model increase to 96.16% (95.52%-96.32%) and 95.80% (95.11%-96.45%), respectively. The best results were achieved by using the features obtained by the ACHS_2 in the training process and using 1-NN classifier, with an accuracy of 98.40% (97.80%-98.60%). The accuracy scores obtained over the test set are within the statistical margin covered by the cross-validated metrics.

The train/test split, feature construction, model building and score reporting were next further performed for 10 repetitions in order to evaluate the influence of the stochastic nature of the search heuristic, the random selection of the train/test split and the training algorithm (in the case of RF) on the predictive performance of the proposed scheme. Test scores obtained by each algorithm in the benchmark over such repetitions were compared to those of the best performing scheme (ACHS_2 + 1-NN, as measured by its median test accuracy computed over repetitions) by using the Mann-Whitney U test to measure the equality of medians. A p-value below 0.05 was obtained for all cases, hence performance gaps can be deemed statistically significant (different median) with respect to the test scores obtained by ACHS_2 with 1-NN. The produced features (*programs*) corresponding to the repetition scoring the highest test accuracy are the following:

$$x^{1,\prime} = x^1 + x^{10}, \quad x^{2,\prime} = x^{13} - x^9, \quad x^{3,\prime} = \log{(x^7 + x^{11})}. \tag{10}$$

Scheme	Feature processing method	Cross-validated accuracy (%)	Test accuracy (%)
ReliefF + RF	Selection	91.56 (90.51-92.61)	93.33
PCA + 1-NN	Construction	94.40 (93.90-95.49)	93.33
ReliefF + 1-NN	Selection	90.69 (89.86-91.59)	95.56
RF + RF	Selection	96.16 (95.52-96.32)	95.56
RF + 1-NN	Selection	95.80 (95.11-96.45)	95.56
PCA + RF	Construction	95.48 (94.94-96.10)	95.56
$ACHS_1$ + RF	Construction	97.41 (96.33-97.90)	97.78
$ACHS_1$ + 1-NN	Construction	97.46 (96.60-97.76)	97.78
$ACHS_2$ + RF	Construction	96.80 (95.54-97.07)	97.78
$ACHS_2$ + 1-NN	Construction	98.40 (97.80-98.60)	100.00

Table 1: Results obtained with the `WINE` dataset, different feature processing criteria and classification methods. Cross-validated accuracy scores are represented as *median (IQR)*.

5.2. Results obtained with the `LEAF` Dataset

In this second dataset the aim of the experiments discussed in what follows is twofold. On one hand features produced by the proposed ACHS approach will be compared, in terms of predictive performance, to those reported in [62]. On the other hand we will assess the impact of incorporating information about the predictive importance of the iteratively produced features in the operators of the HS solver. A RF model will be utilized for both classification and estimation of feature importances, which can be done efficiently as the latter information is produced as a byproduct of the training procedure of the model.

Regarding the first goal pursued in this second dataset, using $\mathcal{T} = 80$ and using 10 train/test splits (with proportions set to 75%/25%) the sets of newly improvised features produced by ACHS for such splits produce a mean test accuracy score of 92.54%, with IQR equal to (91.81%-93.66%). These statistics are notably superior to the 88.82% average accuracy score reported in [66] when using a RF model with the original set of features. For a fair comparison we have verified that this latter scheme scores a mean test score of 87.30% (86.62%-89.15%) when computed over the same train/test splits used in our study. This gap becomes even more relevant when jointly analyzed with the slight dimensionality reduction of the ACHS approach

($D = 15$ original features versus $D' = 13$ constructed characteristics).

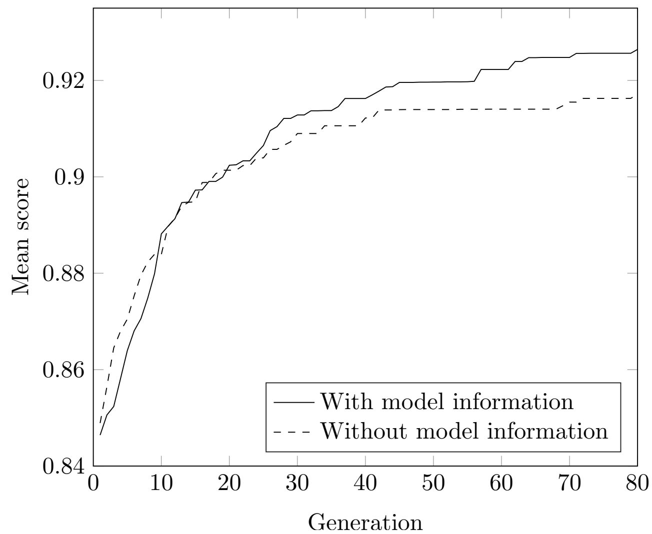

Figure 4: Mean accuracy score of the ACHS approach – averaged over 10 Monte Carlo realizations – with and without model information in the definition of the operators controlling the HS-based optimization wrapper. While differences among them are not relevant in the first iterations, a clear performance gap arises as the search becomes more exploitative.

The discussion with this dataset follows by Figure 4, where convergence plots (average cross-validation score over generations) of ACHS are depicted for the cases when the inner HS solver exploits the predictive relevance estimated by the learning model and when this information is not utilized anyhow. Cross-validation scores have been averaged over 10 repetitions of the HS algorithm for a given train/test split. As inferred from the curves therein both alternatives behave quite similarly in the early generations of the HS wrapper; however, as the search process becomes more exploitative, a clear gap widens between both schemes supporting our postulated benefit of incorporating the predictive relevance in the definition and progression of the HS parameters.

5.3. Results obtained with the `IONOSPHERE` *Dataset*

In this third dataset RF was used to quantify the feature importances and to classify the data based on the selection of the $D' = 5$ features with highest predictive potential. Although it is not the main purpose of this case study, for a train/test split of 75%/25% we obtained a *median (IQR)* cross-validation accuracy of 95.12% (94.22%-95.89%) by using our ACHS scheme, 10 folds and $\mathcal{T} = 50$ iterations. A RF model with the original feature set scored 93.10% (92.44%-93.70%), whereas PCA+RF showed an accuracy of 91.23% (90.86%-91.54%). Differences were found to be significant by using the Mann-Whitney U test over these cross-validated accuracy score sets.

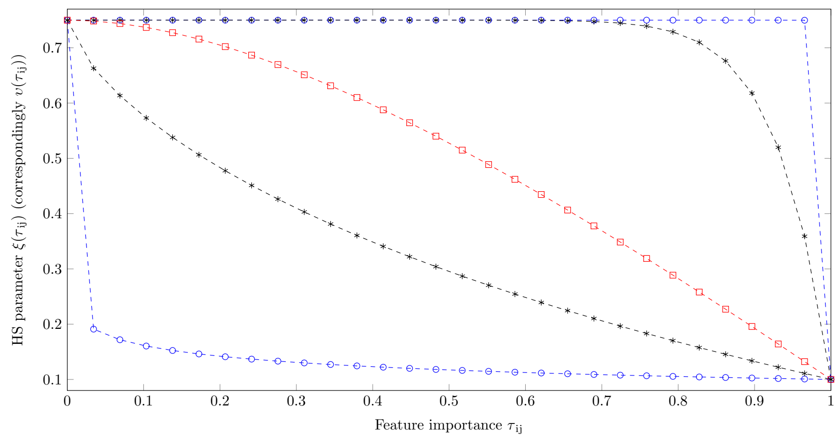

Figure 5: Progression curves of an HS parameter (either $\xi(\tau_{ij})$ or $\upsilon(\tau_{ij})$) from 0.9 (for the first generation of the HS search process) to 0.1 (for the last generation) as a function of the feature importance τ_{ij} for three different values of ζ_s and ζ_e.

Following the intended purpose of this dataset we analyzed the convergence and cross-validated scores of the ACHS approach with different values of ζ_s and ζ_e, which yields a different concave/convex shape for the curve relating the feature importance τ_{ij} to the value of the HS parameter through Expression (9). Figure 5 depicts different curves for the used $\{\zeta_s, \zeta_e\}$ values, with the same color used for the starting and ending curves of each of the 3 cases. Red curves correspond to the special case when $\zeta_e = \zeta_s$. We applied ACHS to the aforementioned train/test split of the `IONOSPHERE` dataset for 10 repetitions, and computed the median cross-validated accuracy for every generation of the HS solver. Figure 6 shows the obtained plots for every $\{\zeta_s, \zeta_e\}$ case and $\mathcal{T} \in \{50, 100\}$ generations. The case where $\zeta_e = \zeta_s$ (red) renders the worst results in all cases, which demonstrates the performance gain that the concavity/convexity tuning by ζ_s and ζ_e through Expressions (8) and (9) can provide. Indeed the other cases (blue and black) converge to better median results, especially when the number of generations of the HS solver is lower.

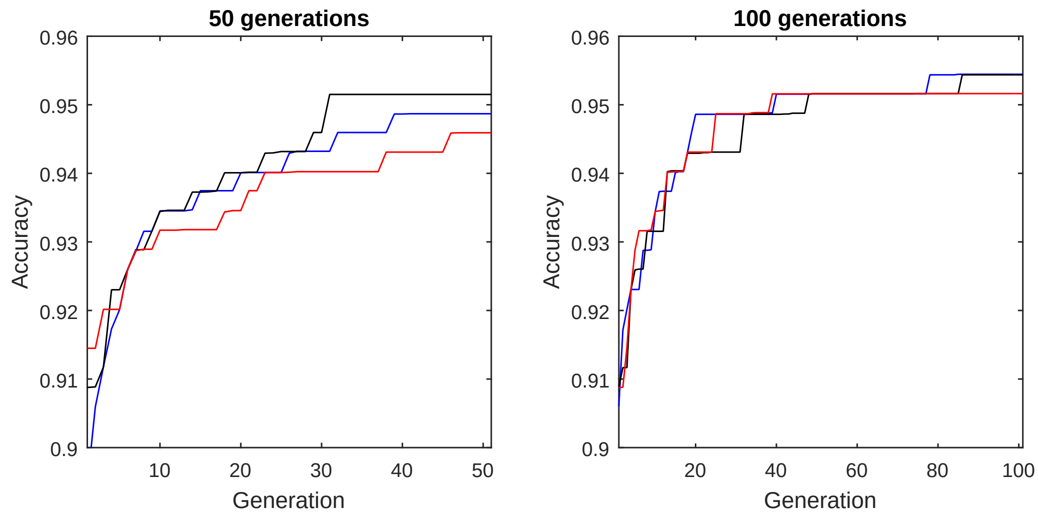

Figure 6: Convergence plots of the median cross-validated accuracy score obtained by ACHS+RF, averaged over 10 repetitions of the HS solver over a 75%/25% train/test split, using $\mathcal{T} = 50$ (left) and $\mathcal{T} = 100$ (right) generations and the three different $\{\zeta_s, \zeta_e\}$ cases depicted in Figure 5.

5.4. Results obtained with the `VTR` *Dataset*

The dataset was divided into a 60%/40% train/test split, from which 5-fold cross-validation will be used for producing a set of $D' = 17$ new features (in the order of the set used in the landmark reference of this dataset [37]). The authors in [37] proposed to hybridize Genetic Algorithms for feature selection and ELM as the predictive model, hence comparing the performance of this hybrid scheme to that provided by different alternative classifiers. In this paper the same experiment has been repeated in order to compare the obtained results and the universality of both two algorithms. To this end we used ReliefF to measure the feature importances and ELM as the classification method. The ELM has 3 output neurons, from which the maximum value was considered to be the class predicted by the ELM. A sigmoid activation function $g(x)$ has been also selected for all compared counterparts. We next trained a RF classifier, a SVM with different kernels and a 1-NN model using the constructed features produced by the ACHS approach with ELM as its core learner. For the SVM, a one-vs-all strategy was considered. The 1-NN classifier is configured to use the Mahalanobis distance, as it scored best in an off-line 5-fold cross-validation benchmark among different options for the distance metric.

Classifier	Kernel	Test accuracy
RF	—	87.89%
SVM	$k(\mathbf{x}, \mathbf{x_i^T}) = \mathbf{x} \cdot \mathbf{x}_i^T$ (linear)	88.42%
SVM	$k(\mathbf{x}, \mathbf{x}_i^T) = \exp(-\gamma \left\|\mathbf{x} - \mathbf{x_i^T}\right\|^2)$ (RBF)	94.21%
SVM	$k(\mathbf{x}, \mathbf{x}_i^T) = (\mathbf{x} \cdot \mathbf{x}_i^T + 1)^3$ (polinomial)	97.37%
1-NN	—	97.89%
ELM	—	98.33%
SVM	$k(\mathbf{x}, \mathbf{x}_i^T) = (\mathbf{x} \cdot \mathbf{x}_i^T + 1)^2$ (polinomial)	98.95%

Table 2: Test accuracy of different classifiers using the features extracted previously by ACHS with ELM as the core classifier.

Table 2 summarizes the accuracy scores obtained by each algorithm over the test data left aside before any feature construction process. The best performance was obtained using ELM and SVM with order two polynomial kernel. In particular the mean accuracy averaged over the output of 100 independent ELM models[3] over a given train-test split was 98.33%, which outperforms the 93.74% accuracy previously obtained in [37]. Furthermore, the accuracy score reported in this reference was computed based on a single training of the ELM model. Interestingly, other classifiers such as 1-NN or SVM with other kernel functions also produced good performance scores. In this case RF yielded worse accuracy figures, possibly due to the fact that ACHS was not trained using RF as the core classifier. On the other hand, the feature selection method proposed in [37] scored the best performance using a multi-class ELM classifier (MC-ELM), with the average probability of correct classification reported to amount up to 98% over the test set. The MC-ELM was composed by three different binary ELM-based classifiers arranged in a one-versus-all voting strategy: one ELM per class trained to distinguish the samples of each class from the samples of the remaining classes. The classification of an unknown sound file was done according to the maximum output among ELMs. For the rest of classifiers used in [37] RF showed a test accuracy of 86.7%, and 1-NN and the SVM classifiers did not reach the 85% test accuracy threshold.

There indeed lies the benefit of the proposed algorithm: as shown in

[3]The purpose of averaging the accuracy over this pool of independent models is to assess the impact of the stochasticity of the ELM training procedure on the obtained results.

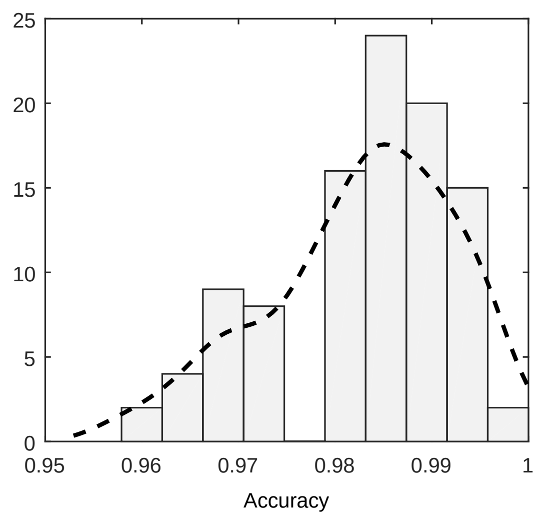

Figure 7: ELM performance histogram over 100 ELM models (see Footnote 3).

Table 2, in this dataset ACHS is proven to construct features that are *universally* good across different classifiers, disregarding whether they are the same model than that used for learning the constructed feature set. Besides, our simple ELM showed better performance than their MC-ELM. Figure 7 depicts the histogram of the accuracy performance attained by 100 independently trained ELM models over the test split at hand. Relevantly is to highlight that there are model instances where the accuracy reaches 1 (i.e. 100%), whereas the major density of this histogram is concentrated above 98%. Nevertheless, results were calculated for a specific train-test split not necessarily equal to the one used in [37]. All in all, the better results in Table 2 show the outperforming behavior of the ACHS approach and its universality when resorting to different supervised models in this fourth dataset.

5.5. Discussion

In order to get an uniform overview on the performance of the proposed ACHS scheme, we here discuss a final set of simulations performed with algorithms based exclusively on the RF model and train/test data splits fixed for every dataset. In this way we isolate the contribution of ACHS to the performance of the overall model and avoid any bias due to the heterogeneity of the models around which discussions in previous sections have been held. Specifically we explore three different schemes: RF using the original feature set, RF using a reduced feature subset selected by their predictive importance (RF+RF), and the proposed ACHS with RF as its core learner for both feature extraction and prediction. Performance of each scheme is given in terms of the mean accuracy score measured over 10 execution of the algorithm over 10 different realizations, namely, 10 distinct train/test splits of the dataset at hand. This averaging process not only minimizes the effects of the selection of train/test splits in databases of relatively small size as the ones chosen in this benchmark, but also permits to account for the randomness of the HS operators.

Dataset	**Baseline** (Literature)	**RF** (original)	**RF+RF** (selection)	**ACHS+RF** (proposed)
`WINE`	—	97.11% ☒	96.59% ☐	97.89%
`LEAF`	88.82% [66]	82.92% ☐	78.32% ☐	93.19%
`IONOSPHERE`	92.50% [63]	93.12% ☐	87.76% ☐	94.20%
`VTR`	86.77% [37]	85.74% ☐	88.44% ☐	97.77%

Table 3: Summary of mean test accuracy scores (computed over 10 realizations) achieved for the datasets and algorithms in this benchmark. Train/test splits within each realization are kept fixed for every dataset so that features and scores for all algorithms in the benchmark are based on the same data subsets. The number of selected (RF+RF) and constructed (ACHS+RF) features is also set equal for a fair comparison.

Results are summarized in Table 3. The statistical significance of the performance gaps between each algorithm and the proposed ACHS+RF – by the Mann-Whitney U test with 0.05 as the confidence threshold – being denoted as ☐ (statistically meaningful) or ☒ (statistically meaningless). Also included are in the table scores of RF-based models reported in the literature for `LEAF`, `IONOSPHERE` and `VTR`. Except for `LEAF` due to the use of an augmented set of features in [66], it can be noticed that the results of RF with the original set of features and the baseline scores are similar to each other, which confirms that our discussed figures are aligned with those in the literature. In all datasets the proposed ACHS+RF is shown to be superior, with statistical significance in the majority of cases.

Before finishing this discussion, it should be noted that the computational complexity of the proposed feature construction algorithm is higher than that of feature selection schemes. This fact limits its applicability to static learning scenarios where no time constraints are imposed for constructing the predictive model and/or with stationary data.

6. Conclusions

This manuscript has delved into a novel feature construction framework for supervised learning problems. The proposed scheme, coined as ACHS, blends together 1) a heuristic wrapper that relies on Cartesian Genetic Programming and the Harmony Search solver; and 2) the predictive relevance of the constructed features produced by the model wrapped by the former. The solution encoding convention provided by Cartesian Genetic Programming is shown to conveniently match the constant-length encoding requirements of the naïve HS algorithm. Furthermore, the incorporation of the predictive importance of the constructed features to the HS search procedure expedites the convergence of the overall wrapper method. The performance of the proposed ACHS approach has been assessed over four datasets with different yet related purposes: to shed light on its potential over the thoroughly studied `WINE` dataset, and to prove that it outperforms the state of the art related to two other recently contributed datasets for supervised learning, namely, `LEAF`, `IONOSPHERE` and `VTR`. The accuracy scores attained by the proposed ACHS have been shown to be promising, with statistically meaningful performance gains that motivate its widespread application to practical supervised learning problems.

Acknowledgements

This work has been funded in part by the Basque Government under the ELKARTEK program (BID3A project, grant ref. KK-2015/0000080). The authors would also like to thank the anonymous referees for their constructive comments and recommendations.

Bibliography

[1] E. Siegel, Predictive Analytics: The Power to Predict who will Click, Buy, Lie, or Die, John Wiley & Sons, 2013.

[2] M. Negnevitsky, Artificial Intelligence: a Guide to Intelligent Systems, Pearson Education, 2005.

[3] S. Lohr, The age of big data, New York Times 11 (2012).

[4] F. Provost, T. Fawcett, Data science and its relationship to big data and data-driven decision making, Big Data 1 (2013) 51–59.

[5] J. Han, M. Kamber, J. Pei, Data Mining: Concepts and Techniques, Elsevier, 2011.

[6] B. Xue, M. Zhang, W. N. Browne, X. Yao, A survey on evolutionary computation approaches to feature selection, IEEE Transactions on Evolutionary Computation 20 (2016) 606–626.

[7] J. Yang, V. Honavar, Feature subset selection using a genetic algorithm, in: Feature extraction, construction and selection, Springer, 1998, pp. 117–136.

[8] C.-L. Huang, J.-F. Dun, A distributed pso–svm hybrid system with feature selection and parameter optimization, Applied Soft Computing 8 (2008) 1381–1391.

[9] S.-W. Lin, Z.-J. Lee, S.-C. Chen, T.-Y. Tseng, Parameter determination of support vector machine and feature selection using simulated annealing approach, Applied Soft Computing 8 (2008) 1505–1512.

[10] C.-M. Wang, Y.-F. Huang, Evolutionary-based feature selection approaches with new criteria for data mining: A case study of credit approval data, Expert Systems with Applications 36 (2009) 5900–5908.

[11] K. Drozdz, H. Kwasnicka, Feature set reduction by evolutionary selection and construction, in: KES International Symposium on Agent and Multi-Agent Systems: Technologies and Applications, Springer, 2010, pp. 140–149.

[12] B. Xue, M. Zhang, W. N. Browne, Particle swarm optimisation for feature selection in classification: Novel initialisation and updating mechanisms, Applied Soft Computing 18 (2014) 261–276.

[13] B. Xue, M. Zhang, W. N. Browne, A comprehensive comparison on evolutionary feature selection approaches to classification, International Journal of Computational Intelligence and Applications 14 (2015) 1550008.

[14] S. Salcedo-Sanz, J. A. Portilla-Figueras, J. Munoz-Bulnes, J. del Ser, M. N. Bilbao, A novel harmony search algorithm for one-year-ahead energy demand estimation using macroeconomic variables, in: International Joint Conference SOCO14-CISIS14-ICEUTE14, Springer, 2014, pp. 251–258.

[15] S. Salcedo-Sanz, A. Pastor-Sánchez, J. Del Ser, L. Prieto, Z. Geem, A coral reefs optimization algorithm with harmony search operators for accurate wind speed prediction, Renewable Energy 75 (2015) 93–101.

[16] T. Jirapech-Umpai, S. Aitken, Feature selection and classification for microarray data analysis: Evolutionary methods for identifying predictive genes, BMC bioinformatics 6 (2005) 1.

[17] M. Banerjee, S. Mitra, H. Banka, Evolutionary rough feature selection in gene expression data, IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 37 (2007) 622–632.

[18] J. P. Cunningham, Z. Ghahramani, Linear dimensionality reduction: Survey, insights, and generalizations, Journal of Machine Learning Research 16 (2015) 2859–2900.

[19] J. A. Lee, M. Verleysen, Nonlinear Dimensionality Reduction, Springer Science & Business Media, 2007.

[20] X. Yao, Y. Liu, G. Lin, Evolutionary programming made faster, IEEE Transactions on Evolutionary Computation 3 (1999) 82–102.

[21] K. Krawiec, Genetic programming-based construction of features for machine learning and knowledge discovery tasks, Genetic Programming and Evolvable Machines 3 (2002) 329–343.

[22] H. Guo, L. B. Jack, A. K. Nandi, Feature generation using genetic programming with application to fault classification, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 35 (2005) 89–99.

[23] M. G. Smith, L. Bull, Genetic programming with a genetic algorithm for feature construction and selection, Genetic Programming and Evolvable Machines 6 (2005) 265–281.

[24] K. Neshatian, M. Zhang, M. Johnston, Feature construction and dimension reduction using genetic programming, in: AI 2007: Advances in Artificial Intelligence, Springer, 2007, pp. 160–170.

[25] L. Shao, L. Liu, X. Li, Feature learning for image classification via multiobjective genetic programming, IEEE Transactions on Neural Networks and Learning Systems, 25 (2014) 1359–1371.

[26] K. Neshatian, M. Zhang, P. Andreae, A filter approach to multiple feature construction for symbolic learning classifiers using genetic programming, IEEE Transactions on Evolutionary Computation 16 (2012) 645–661.

[27] M. W. Aslam, Z. Zhu, A. K. Nandi, Feature generation using genetic programming with comparative partner selection for diabetes classification, Expert Systems with Applications 40 (2013) 5402–5412.

[28] D. Y. Harvey, M. D. Todd, Automated feature design for numeric sequence classification by genetic programming, IEEE Transactions on Evolutionary Computation 19 (2015) 474–489.

[29] A. Cano, S. Ventura, K. J. Cios, Multi-objective genetic programming for feature extraction and data visualization, Soft Computing (2015) 1–21.

[30] B. Tran, B. Xue, M. Zhang, Genetic programming for feature construction and selection in classification on high-dimensional data, Memetic Computing 8 (2016) 3–15.

[31] P. G. Espejo, S. Ventura, F. Herrera, A survey on the application of genetic programming to classification, IEEE Transactions on Systems, Man, and Cybernetics, Part C 40 (2010) 121–144.

[32] J. F. Miller, P. Thomson, Cartesian genetic programming, in: Genetic Programming, Springer, 2000, pp. 121–132.

[33] Z. W. Geem, J. H. Kim, G. Loganathan, A new heuristic optimization algorithm: harmony search, Simulation 76 (2001) 60–68.

[34] D. Manjarres, I. Landa-Torres, S. Gil-Lopez, J. Del Ser, M. N. Bilbao, S. Salcedo-Sanz, Z. W. Geem, A survey on applications of the harmony

search algorithm, Engineering Applications of Artificial Intelligence 26 (2013) 1818–1831.

[35] E. J. Pauwels, P. M. de Zeeuw, E. B. Ranguelova, Computer-assisted tree taxonomy by automated image recognition, Engineering Applications of Artificial Intelligence 22 (2009) 26–31.

[36] V. G. Sigillito, S. P. Wing, L. V. Hutton, K. B. Baker, Classification of radar returns from the ionosphere using neural networks, Johns Hopkins APL Technical Digest 10 (1989) 262–266.

[37] E. Alexandre, L. Cuadra, S. Salcedo-Sanz, A. Pastor-Sánchez, C. Casanova-Mateo, Hybridizing extreme learning machines and genetic algorithms to select acoustic features in vehicle classification applications, Neurocomputing 152 (2015) 58–68.

[38] Z. W. Geem, Novel derivative of harmony search algorithm for discrete design variables, Applied Mathematics and Computation 199 (2008) 223–230.

[39] Z. W. Geem, K.-B. Sim, Parameter-setting-free harmony search algorithm, Applied Mathematics and Computation 217 (2010) 3881–3889.

[40] K. Sörensen, Metaheuristics – the metaphor exposed, International Transactions in Operational Research 22 (2015) 3–18.

[41] D. Weyland, A critical analysis of the harmony search algorithm – how not to solve sudoku, Operations Research Perspectives 2 (2015) 97–105.

[42] M. Padberg, Harmony search algorithms for binary optimization problems, in: Operations Research Proceedings, Springer, 2012, pp. 343–348.

[43] D. Weyland, A rigorous analysis of the harmony search algorithm: How the research community can be misled by a "novel" methodology, Modeling, Analysis, and Applications in Metaheuristic Computing: Advancements and Trends (2012) 72.

[44] Z. W. Geem, Research commentary: Survival of the fittest algorithm or the novelest algorithm?, International Journal of Applied Metaheuristic Computing (IJAMC) 1 (2010) 75–79.

[45] M. Saka, O. Hasançebi, Z. Geem, Metaheuristics in structural optimization and discussions on harmony search algorithm, Swarm and Evolutionary Computation 28 (2016) 88–97.

[46] J. H. Kim, Harmony search algorithm: A unique music-inspired algorithm, in: 12th International Conference on Hydroinformatics, HIC 2016, 2016.

[47] J. Swan et al., A research agenda for metaheuristic standardization, in: Proceedings of the XI Metaheuristics International Conference, 2015.

[48] S. Adriaensen, A. Nowé, Towards a white box approach to automated algorithm design, in: International Joint Conferences on Artificial Intelligence (IJCAI), 2016, pp. 554–560.

[49] M. El-Abd, An improved global-best harmony search algorithm, Applied Mathematics and Computation 222 (2013) 94–106.

[50] Z. W. Geem, State-of-the-art in the structure of harmony search algorithm, in: Recent Advances In Harmony Search Algorithm, Springer, 2010, pp. 1–10.

[51] M. Mahdavi, M. Fesanghary, E. Damangir, An improved harmony search algorithm for solving optimization problems, Applied Mathematics and Computation 188 (2007) 1567–1579.

[52] J. Del Ser, M. Matinmikko, S. Gil-López, M. Mustonen, Centralized and distributed spectrum channel assignment in cognitive wireless networks: a harmony search approach, Applied Soft Computing 12 (2012) 921–930.

[53] S. Zhan, J. F. Miller, A. M. Tyrrell, A developmental gene regulation network for constructing electronic circuits, in: Evolvable Systems: From Biology to Hardware, Springer, 2008, pp. 177–188.

[54] M. M. Khan, G. M. Khan, J. F. Miller, Efficient representation of recurrent neural networks for markovian/non-markovian non-linear control problems, in: International Conference on Intelligent Systems Design and Applications (ISDA), IEEE, 2010, pp. 615–620.

[55] M. M. Khan, A. M. Ahmad, G. M. Khan, J. F. Miller, Fast learning neural networks using cartesian genetic programming, Neurocomputing 121 (2013) 274–289.

[56] J. F. Miller, Cartesian genetic programming, Springer, 2011.

[57] S. Luke, Essentials of metaheuristics. lulu, 2009, Available for free at http://cs. gmu. edu/sean/book/metaheuristics/. There is no corresponding record for this reference (2011).

[58] K. Kira, L. A. Rendell, The feature selection problem: Traditional methods and a new algorithm, in: AAAI, volume 2, 1992, pp. 129–134.

[59] K. Kira, L. A. Rendell, A practical approach to feature selection, in: Proceedings of the ninth international workshop on Machine learning, 1992, pp. 249–256.

[60] I. Kononenko, E. Šimec, M. Robnik-Šikonja, Overcoming the myopia of inductive learning algorithms with relieff, Applied Intelligence 7 (1997) 39–55.

[61] M. Lichman, UCI machine learning repository, 2013. URL: `http://archive.ics.uci.edu/ml`.

[62] P. F. Silva, A. R. Marçal, R. M. A. da Silva, Evaluation of features for leaf discrimination, in: Image Analysis and Recognition, Springer, 2013, pp. 197–204.

[63] L. Breiman, Random forests, Machine learning 45 (2001) 5–32.

[64] C.-W. Hsu, C.-J. Lin, A comparison of methods for multiclass support vector machines, IEEE Transactions on Neural Networks 13 (2002) 415–425.

[65] G.-B. Huang, D. H. Wang, Y. Lan, Extreme learning machines: a survey, International Journal of Machine Learning and Cybernetics 2 (2011) 107–122.

[66] E. Elhariri, N. El-Bendary, A. E. Hassanien, Plant classification system based on leaf features, in: Computer Engineering & Systems (ICCES), 2014 9th International Conference on, IEEE, 2014, pp. 271–276.