UNIVERSITÀ DEGLI STUDI DI TORINO

DIPARTIMENTO DI SCIENZE VETERINARIE

Dottororato di Ricerca in

SCIENZE VETERINARIE PER LA SALUTE ANIMALE
E LA SICUREZZA ALIMENTARE

ENHANCING MACHINE LEARNING APPROACHES FOR BIODATA MINING IN VETERINARY SCIENCES

Tesi presentata da: Irene Azzali

Tutor: Mario Giacobini

Coordinatore del dottorato: Maria Teresa Capucchio

ANNI ACCADEMICI 2017/2018-2018/2019-2019/2020

Contents

Abstract

Vector-borne diseases (VBDs) are illnesses caused by parasites, bacteria or viruses that are transmitted by vectors, which include fleas, ticks and mosquitoes [57]. These diseases considerably impact the economic and public health, thus the development of effective prevention measures is essential. Besides the treatment and control of the disease itself, an approach to monitor VBDs is by means of vector control. Developing effective vector control measures allows for an immediate interruption of disease transmission and helps in disease eradication [28]. Vector abundance modelling is one of the approaches to address vector control. Modelling vectors means estimating the local abundance of vectors as a function of the features deemed informative on their dynamics. It is known that weather influences survival and reproduction rates of vectors, in turn influencing the rates of development, survival and reproduction of the pathogens they harbour. The prediction of vector occurrence and the understanding of vector-habitat link are therefore crucial to the early warning of pathogen circulation and to guide vector control strategies.

Modelling techniques share the principle to learn from data, however, we recognize two main approaches differing for their main purpose. Mathematical models, refers to all the mathematical formulas used to describe how variations in the response variable are generated by relationship with explanatory variables. The main objective of mathematical modelling is the discovery and the interpretation of the relationship among the included variables. Machine learning modelling is, instead, a set of techniques that learn the best model underlying data in order to make predictions on unseen data. The focus this time is in the accuracy of the forecast, penalizing the readability of variables interactions. The use of precise mathematical modelling techniques, statistical modelling, in the field of vector abundance modelling has been dominating respect to machine learning approaches. Statistical modelling, in fact, provides understanding of the mechanisms that influence vector abundance and thus suggests vector control strategies. Data involved to fulfil vector abundance prediction, however, are usually heterogeneous, involving different formats and types. The research is, therefore, moving towards machine learning (ML) models that can better catch the complex interplay between environment, climate and vectors. Compared with statistical methods, ML does not reside on assumptions (e.g. linearity of data structure). ML methods can handle complex data and implicit interactions among input factors, further inquiring on possible non-linear impacts of the covariates which are generally

difficult for statistical methods to recognize.

In an application perspective, models have their greatest utility when they can be used predictively and not simply as a means of exploring putative relationships among variables in the data. This consideration still leads towards further exploration of ML methods in VBDs abundance prediction. However the main drawback of ML methods is the lack of readability and interpretability. Many of ML techniques, in fact, are black-box methods, therefore the variables interaction responsible of the abundances remains unknown, providing less information for vector control. This drawback is overcame by a ML techniques called Genetic Programming (GP). GP automatically discovers solutions to problems by applying search principles analogous to those of natural evolution [60]. In theory, GP can cover any kind of problem whose candidate solutions can be measured and compared in terms of how well they solve the problem. One of GP most successful application is the discovery of the function describing features interaction in data, or in other words, modelling. Respect to classical ML methods, GP has the advantage to obtain readable models for the user, which allows for interpretation when the model formula is not too complex. Moreover, GP performs a natural selection of all the variables investigated to approximate the target, thus avoiding prior features selection. To the best of our knowledge GP has never been explored in the field of vector abundance prediction, although it has all the characteristic to be a promising technique.

In this work we focus on the most studied disease vectors: mosquitoes. Mosquito is in fact one of the deadliest animal in the world due to its great ability to carry and spread diseases that can cause death, such as Zika virus, West Nile virus, Chikungunya virus, dengue, and malaria [57]. More than half of the world population, moreover, live in areas where mosquitoes are present. The threat accompanying this vector determined, therefore, intensive studies on the topic of mosquito-borne diseases which provided us with many data on mosquito dynamics.

The overall goal of this work is to investigate the innovative use of GP in the field of vector abundance prediction. In particular we start from the available data of *Culex pipens* counts collected in the context of the surveillance programme promoted by IPLA [3] in Piedmont region. The dataset was already used in [11] to address mosquito abundance prediction by means of statistical modelling, thus we have previous results to compare GP performance. Moreover, to include GP in the framework of ML for eco-epidemiological problems, we benchmark GP performance against other ML approaches. We expect GP to improve at least the predictive accuracy of statistical modelling which limits the interaction between predictors. GP structure has even the advantage of being easy to modify in order to satisfy problem requirements. This feature is of great importance since we can enhance classical GP approach to properly deal with the different data format that we may face in vector abundance dataset. Time series are, in fact, frequently present in these dataset. The seasonal dynamics of vector population is likely to be associated with the fluctuation of climatic and

weather variables over time, thus time series data. At the moment, classical methods in statistical and ML modelling are mostly unable to handle time series as ordered sequence of values. The problem we are dealing with, in fact, is not the classical time series regression of predicting the future occurrence based on past ones. GP easy structure, instead, can be extended to work with raw time series data. Mathematical vectors are the suitable and immediate representation of time series and we can adapt GP to treat vectors as variables. The development of an innovative approach of GP in the context of epidemiological modelling will even be the springboard of a new ML technique that deals with vectors. By means of this work we want to make the veterinary community aware of GP technique as a modelling approach that can sum up three important goals of ecological modelling: readability of the model, ability to catch any functional forms describing the data and ability to adapt to data without distorting their nature.

Part I

Introduction and Motivations

Chapter 1

Introduction to the ecological problem of vector borne diseases

Vector borne infectious diseases (VBDs) are illnesses that result from an infection transmitted to humans by blood-feeding arthropods, such as mosquitoes, ticks, and fleas. Malaria, dengue fever, yellow fever and plague are examples of such illnesses. The WHO estimates that one-sixth of the illnesses and disabilities suffered worldwide is owing to vector-borne diseases, with more than half of the world's population currently at risk [57]. VBDs constitute an important cause of death, health inequity, brake on socio-economic development, and strain on health services. Continued progress in controlling these diseases is therefore an important contribution to global health, development and security, particularly since there is no vaccine for most of them.

Vector surveillance control is the primary mean to prevent VBDs, but, nonetheless, VBDs are still a threat worldwide. The development of vector resistance to insecticides, changes in public health programs, climatic change, increased mobility of humans, and urban growth are factors that contribute to the difficulty in controlling and eliminating vector borne diseases. Since controlled epidemiological experiments are usually not possible, modelling has played an important role in gaining a better understanding on how to mitigate the burden of VBDs. This role of modelling has been strongly confirmed by the European Centre for Disease Prevention and Control (ECDC) [24] that has recognized the spatial estimates of vector counts a crucial advice to assess VBDs risk.

Modelling means estimating a desired response through a mathematical representation of the dynamics underling it. Modelling vector abundance allows the prediction of occurrence which informs a number of epidemiological concepts. These include the early warning of pathogen introduction, the potential for pathogen transmission, its establishment and persistence, the consequent spread of the pathogen to new areas and the control of pathogen spread. Of more importance is, however, the insight that modelling gives

into the interactions of environmental and climatic factors driving the abundances. A prior knowledge of these dynamics allows for more effective early strategies of mitigation and control.

This thesis originates from a virus considered endemic in Piedmont region: West Nile Virus (WNV). An intensified and continuous WNV spread across northern Italy has been observed since 2008, which caused more than one hundred reported human infections until 2016 [21]. In this context, modelling implication has allowed the identification of informative patterns about potential disease outbreaks.

1.1 *Culex spp.* mosquitoes

The West Nile virus is transmitted predominantly by *Culex spp.* mosquitoes [16]. The habitat conditions of these mosquitoes strongly impact the population abundance. *Culex* common breeding site is stagnant water in a variety of natural and man-made containers, including tree holes, ditches and catch basins. They cannot develop in running water, and water that is present less than a week. In addition, mosquitoes are cold-blooded creatures, therefore they are not able to regulate their body heat and their temperature is essentially the same as their environment. Temperature and mosquito activityare thus related with the insects flourishing in moist, relatively warm environments [25]. Alterations to these traits can lead to substantial variations in vectorial capacity of mosquitoes that harbor and transmit pathogens. Many mosquito-borne diseases, in fact, exhibit substantial seasonality, due to strong links between environmental variables and vector life-cycles.

Culex pipiens is a *Culex spp.* complex native to Europe. Due to its spread and activity, *Culex pipiens* is considered a pest in urban environments. Since *Cx. pipiens* is broadly distributed and bites a wide range of hosts, the species enters into contact with a wide range of pathogens. In particular, *Cx. pipiens* appears to be a major vector of WNV in Europe [56].

1.2 West Nile Virus in Piedmont Region, Italy

WNV circulation was firstly detected in Italy in the late summer of 1998, when 14 horses located in Tuscany were confirmed for WNV infection by laboratory analyses [38]. After this outbreak, a national veterinary surveillance plan was implemented in 2001 under the coordination of the Italian Ministry of Health (MoH) and of the National Reference Centre for Exotic Diseases of Animals (Centro Studi Malattie Esotiche; CESME), with the aim to early detect new incursions of WNV. The largest outbreak, however, dates to 2008 when a WNV epidemic affected the regions in the Northeast of Italy surrounding the delta of the Po river. In this occasion, the first human case of WNV neuro-invasive infection in Italy was observed [23]. Following these events, the surveillance system was updated with the aim to detect as early as possible the WNV circulation [7].

Piedmont region was classified as a high risk area and therefore it was subject to the active surveillance promoted by the national plan. Crossed by the Po river from West to East and bounded to the East by the Ticino river, this region offers the suitable habitat to mosquitoes due to the extensive rice fields that dominate the northeast landscape. This awareness has resulted in the regional mosquito population surveillance programme that the "Istituto per le Piante da Legno e l'Ambiente" (IPLA) has been promoting and coordinating since 1997. Mosquitoes are weekly collected, from May to September (20 collections per year), from 36 CO2-baited traps randomly located in Casale Monferrato. This particular eastern area of Piedmont region offers, in fact, a favourable environment for the proliferation of mosquitoes. After the collections, all adult mosquitoes are sexed, counted and identified using proper keys. These data have been used to implement different treatments strategies in order to reduce mosquito abundance in the study area. More targeted vector control strategies, as well as the need to support decision making process, however, demand for the use of modelling techniques. In light of this consideration, Bisanzio et al. in [11] used the data of mosquito collections performed in the context of IPLA surveillance to evaluate the effects of environmental determinants on the spatial distribution of *Culex spp.* through rigorous statistical modelling. The predicted vector distribution maps allowed the identification of areas with high potential risk of WNV introduction and amplification in eastern Piedmont region.

1.3 Modelling mosquito abundance

Since the goals for decisions and practices in the field of epidemiological problems are the prevention and the control of a disease, modelling becomes a fundamental support. Applying models to mosquito dynamics allows for the identification of the variables that influence the behaviour of the vector based on past information, but also allows for the prediction of possible future scenarios. Building a model to predict mosquito counts involve as a primary step the selection of the modelling technique. While mathematical models are used in the epidemiological context to describe the mechanism of infection in a system [52, 27, 37, 26], statistical models are the suitable approach to describe the mechanism of the abundance. Statistical models formalize the relationship between variables in the form of a mathematical equation. The equation is based on the additivity of deemed informative factors plus random errors that account for uncertainty, whose effect on the response variable is based on coefficients estimation. These models are popular in the field of vector modelling, because they well deal with count data and, most important, they can be interpreted.

1.3.1 Statistical modelling of mosquito abundance

Statistical models fall into four main categories: multiple linear regression techniques, generalized linear models, mixed effect models and time series approaches. Related to the precise vector we focus on, mosquitoes,

we present the state of art concerning statistical modelling techniques, highlighting the reason beyond the choice of a particular category.

In [42], for example, the authors predicted potential West Nile virus (WNV) mosquito vector in Seattle region by means of multiple linear regression. More applications of multiple linear regression are found in [51] where this modelling technique was implied to predict adult female of *Aedes aegypti* based on meteorological variables.

Many researchers, however, found traditional linear regression too limiting due to the strict kind of association imposed and rather prefer the more flexible generalized linear models. In [30, 12, 15] generalized linear models were built to unveil the environmental drivers of mosquito abundances of some mosquito species.

Mosquito abundance data are always the results of scheduled collections of mosquitoes from fixed traps in the area of interest. In this perspective, mixed effect modelling approaches account for potential dependent structure in these data due to the repeated observations at a single trap and due to the similar abundances at nearby traps. In [62, 11] a generalized linear mixed model was used to predict mosquito abundance in northwest Italy, while in [31] the effects of environmental and weather factors on mosquito abundances in Ontario were investigated by means of linear mixed effect model.

Time series approaches, instead, are mainly used to forecast mosquito population based on meteorological factors. An empirical model to predict WNV mosquito vector abundance in Erie County was developed in [64] using time series analysis techniques and involving climatic variables. Poisson regression models were rather explored in [54, 73] to describe and predict mosquito counts.

In this work we just focus on multiple linear regression and generalized linear mixed models (GLMM). Multiple linear regression is selected because it is the basic modelling approach, while GLMM is selected because it was the first technique enrolled to tackle the mosquito problem in our area of interest [11]. In order to introduce the reader to the considered models, here below we provide a description of both the techniques.

From now on we use with interchangeability the words target, output and response to denote the dependent variable to be inferred and the word predictors, explanatory variables and features to denote the independent variables deemed informative about the dependent variable.

1.3.2 Multiple linear regression

Multiple linear regression (MLR or LR) is a statistical approach to model the relationship between the expected value of a continuous response y and a set of explanatory variables $x_i,\ i = 1, \ldots, n$ through a linear function [34], thus $E(y) = \beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n$. The parameters of the function are estimated from

data. The most common approach to fit parameters is least square (LS) method. The best parameters, according to LS technique, are the ones that minimizes the the sum of squared residuals (a residual being: the difference between a collected value, and the predicted value provided by a model). Given m observations of the response, the general model form can be rewritten as:

$$y_j = \beta_0 + \beta_1 x_{1_j} + \cdots + \beta_n x_{n_j} + \epsilon_j, \quad j = 1, \ldots, m,$$

where $\beta_i,\ i = 0, \ldots, n$ are the parameters and $\epsilon_j,\ j = 1, \ldots, m$ are the error terms. The residual errors ϵ_j represent the deviations of the observed values y_j from their means $E(y)$ and are assumed to be normally distributed.

1.3.3 Generalized linear mixed models

A GLMM is a statistical model that combines the characteristics of Generalized Linear Models (GLM) and mixed models [77]. Therefore, the linear function is assumed to link explanatory variables and a transformation of the expected value of the response, and both fixed and random effects join the predictors to introduce population-average effects and subject-specific effects. The general linear mixed model can be written therefore as:

$$g(E(Y)) = \beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n + b_1 u_1 + \cdots + b_p u_p,$$

where g is the link function, $\beta_0, \ldots, \beta_n$ are the fixed effects parameters, and $b_1, \ldots, b_p$ are the random effects parameter of $u_1, \ldots, u_p$ random effects. Maximizing the log-likelihood function with respect to $\beta_0, \ldots, \beta_n$ and $b_1, \ldots, b_p$ allows the fitting of the model.

1.3.4 Limits of statistical modelling

Every modelling technique has it pros and cons that guide the user in the selection of the optimal method for the problem under analysis. As previously shown, when dealing with mosquito counts statistical models are one of the first choices due to their immediate application and interpretation. We claim, however, the existence of some limitations that could be overcome by different modelling approaches.

Statistical models have fixed structures. The shape of the combination of variables is decided by experts knowledge or information gained on previous data. Once chosen, the collected data are used to determine the parameters, i.e. the coefficients of these combinations. Since the models have a human build formulation, high order interactions among variables that are not specified can not be detected. The factors deemed as informative about the abundances are heterogeneous, potentially including environmental time series as well as constant characteristics of the study area, therefore complex relationships are more likely.

We have mentioned that statistical models allow both the discovery of links and variables dynamics as well as the prediction of the response, thus the counts. The main focus of the method, however, is in the understanding of the relationship. The best relationship is inferred directly on the available collected data, in the sense that the fixed structure adjust its shape through parameters to describe at its best the data. Surely, a portion of the data can be cut off the adjusting phase to test the performance of the model on unseen data. The goodness of the model, however, is evaluated by means of the significance and robustness of the parameters rather than by means of performance on unseen values. Once discovered the dynamics, control strategies may wish to foresee the abundances to know in advance high risk areas, and this need requires the use of other kind of techniques focused on the accuracy of prediction rather than on interpretation.

The next section present to the reader the field of machine learning techniques. This new popular area provides modelling approaches able to overcome the main limitations of statistical modelling.

Chapter 2

Introduction to Machine Learning approaches

The reader of this dissertation, once arrived at this page, has probably understood that the classical approaches to figure out mosquito dynamics consist in forcing pre-defined structure to fit these dynamics. These approaches are actually statistical modelling approaches. The inclusion of new data and of new variables to reflect every possible scenario makes the dynamics to be inferred highly complex. Moreover, only prior knowledge or repeated tentative lead the choice of the fixed structures. The researcher with expertise in statistical models, therefore, would undoubtedly face the question: "How can I discover the best models underlying my data?". This is the time of machine learning introduction. Machine learning shifts the question from "How can I" to "How can computer".

Machine learning (ML) is a collection of computational methods that give the computer the ability to *learn* the solution to complex problems when we are not able to draw one. In this context, the problem is the prediction of mosquito dynamics, thus through ML we will make the computer learn the best model to describe and predict these dynamics. There are three broad types of ML, known as supervised Learning, unsupervised Learning, and Reinforcement Learning, but we only go deep into the understanding of *supervised learning*, which includes the problem we are dealing with.

2.1 Supervised learning

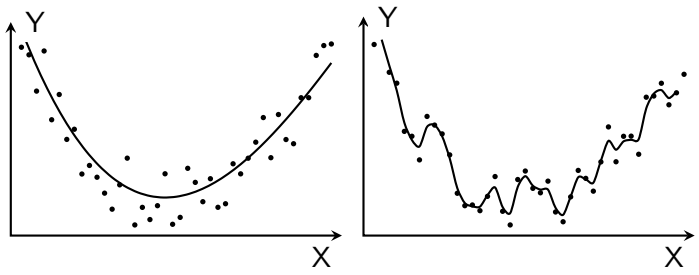
Supervised learning is a type of ML in which the system is presented with labelled data, thus data of the problem with the corresponding output. The goal of ML methods is to make the computer learn the mapping function joining data to label so well that, when presented with new data, the computer infers the correct label. To clarify, we consider X a set of input variables, such as some informative factors about mosquito

abundance. Letter Y, instead, represents the mosquito counts. Given to the computer data of X and the corresponding Y, through supervised machine learning the computer learns the mapping function f so that $f(X) = Y$. Once learnt, the computer is able to provide Y on new and unknown values of X. It is called supervised learning because the process of an algorithm learning from a dataset can be thought of as a teacher supervising the learning process. We know the correct answers, the algorithm iteratively makes predictions on the training data and is corrected by the teacher. Learning stops when the algorithm achieves an acceptable level of performance. Supervised learning is divided into two phase.

Phase 1 is the so called *training phase* in which the ML algorithm is fed by the labelled examples, thus the X and corresponding Y. Suppose we want to build the system to predict mosquito counts. We provide in this phase to the ML algorithm examples of environmental and climatic variables and their associated mosquito count. The algorithm may notice that the counts of rainy days are lower rather than the ones of sunny days. By looking at the examples, over and over, the algorithm recognizes this pattern. Therefore, the algorithm may find out that the map f is $2.8 \cdot \text{rainfall}$, thus $\text{mosquito} = 2.8 \cdot \text{rainfall}$ where rainfall is an entry of X and mosquito is actually Y.

Phase 2 is instead the so called *test phase* in which we provide to the algorithm unseen data and it has to return the label as learned is phase 1. This step consists in the real application of the system that, once trained during phase 1, is finally able to predict on unknown data. Referring to the example problem, once the algorithm has learnt f, if we give it new values of the input variables, thus new value of X, the system returns the corresponding mosquito count.

An important feature of the training phase is the acquisition of *generalization* capability. The system learning is based on training data, so it seems reasonable that the more training data the system has access to, the better it gets at guessing the map. This consideration is not true. The system in fact may become too specific on the training data, learning a mapping function that fits perfectly the training data, but its not capturing the general behaviour of the data. This phenomenon is called *overfitting*. Recalling our problem, imagine that we provide to the algorithm lots of data regarding mosquito collections in a sunny area. The variable concerning rainfall, therefore, may seem quite uninformative since it is mainly constant aver the whole dataset. The algorithm, then, may learn a function perfect for these data that do not take into account the rainfalls. When facing with new data regarding another area, the algorithm may fail to return the correct number of mosquitoes because it has not learnt with generalization. Figure 2.1 shows in the simplest case of one input variable a map f learnt with overfitting and a map f learnt with generalization. The curve in (b) captures closely the training points, but it is likely that a new point will be to closer to the curve in (a), rather than to the curve in (b).



(a) Generalization (b) Overfitting

Figure 2.1: Training properties

Overfitting is not visible during the test phase unless we select a portion of the whole labelled dataset as the test set, and use it just in phase 2. The labels will be then compared to the predictions made by the system to infer the accuracy. In case the accuracy of predictions in the test set is much lower than the accuracy of prediction in the training set, i.e. the other portion of the dataset, then overfitting is detected.

The problem of overfitting is now visible, but it would be better to avoid it. The simplest solution is the so-called early stopping, involving the partition of the whole dataset in three portions. The training and test set as mentioned before and the *validation* set. This last set is not used during the training, but involved in the training to measure how generally the algorithm is learning at every looking at training data. Figure 2.2 shows the mechanism. At every iteration, i.e. at every looking at training data, the map learnt by the algorithm is tested on the validation set. As soon as the predictions error on the validation stops to decrease, the training stops and test phase starts.

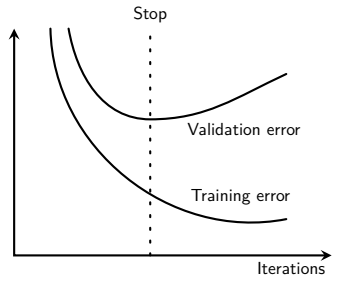


Figure 2.2: Training and validation set dynamics

The problem involved so far to clarify the explanations is the main epidemiological problem of the whole dissertation: modelling mosquito abundance. The variables in X are all the factors deemed informative about the counts, while Y is the abundance of mosquitoes. The mapping function f is the model, thus the combination of variables that determines the target Y. The class of problems with the goal of finding the link functions that can predict the value of the dependent attribute from the attribute variables is called *regression*. Modelling mosquito abundance belongs to this class. Regression problems are tasks usually

solved in the context of ML by supervised learning.

We have described until now how supervised learning manages regression problems by training and test, but how in practise the map f is discovered step by step during the training is still unknown. This process is what defines the different ML techniques of supervised learning. We recognize neural networks, decision trees and random forest, and regression trees. For the purpose of this work, we describe multilayer perceptron that is a neural network, and random forests. Moreover we opt to consider two advanced ML techniques that deserve to be explored: extreme gradient boosting and long short-term memory network. The first one is known to be the most accurate ML methods for prediction [4], while the second belongs to the family of recurrent neural network which addresses predictions involving time series. Since dataset concerning vector abundance typically consists of time series, the use of a proper technique to treat these data is mandatory.

2.1.1 Multilayer perceptron

Multilayer perceptron (MLP) is a type of artificial neural network that is patterned in combination of neurons called layers [66]. MLP consists of a set of source nodes that constitute the input layer, one or more hidden layers of computational nodes, and an output layer of computational nodes. The multiple layers are meant to capture more complex relationships among input variables, therefore we preferred MLP over simple perceptron in the context of mosquito abundance modelling. The MLP network is fed with input data in the input layer and the signal propagates in a forward way, layer by layer, towards the output layer that returns the predicted target. Each unit (neuron) posses a unique set of weights corresponding to all the connection that start from the unit. Weights are the parameters of the network to be trained in order to approximate the function that links the input to the output. MLP is fitted by the values of weights that minimizes a selected loss function.

2.1.2 Random forest

A regression tree is a ML technique that predict values of responses by learning decision rules derived from features [40]. To construct a regression tree, from top to bottom a test is applied to one of the features. Depending on the outcome of the test, we go to either the left or the right sub-branch of the tree. Eventually we come to a leaf node, where we make a prediction. This prediction averages all the training observations which reach that leaf. The tests, which are actually the rules defining the tree, determine a partition of the feature space. The partition such that the overall sums of squares residuals is minimized corresponds to the fittest tree. More robust than single regression tree are random forests. A random forest (RF) is an ensemble of regression trees that estimates the target by averaging individual tree predictions [39]. In this work, we use an RF that follows the Breiman bagging idea of trees ensamble. Instead of training each

tree on the same observations, each tree of the forest is trained over a fixed lower number of observations, randomly selected with replacement from the whole dataset of observations. This technique regularize the outputs, thus improving the generalization ability of the whole forest.

2.1.3 Extreme gradient boosting

Extreme gradient boosting (XGBoost) is an implementation of gradient boosted (GB) regression trees, with a difference in modelling details that generally allows XGBoost to obtain better performance [74]. Boosting is a technique where new models (regression trees) are added to correct the errors made by existing models. Specifically, gradient boosting is an approach where new models are created to fit the residuals of prior models and then added together to make the final model. The word "gradient" refers to the use of the gradient descent technique applied to minimize the loss when adding new models.

2.1.4 Recurrent Neural Network and Long Short-Term Memory

Recurrent neural network is a type of artificial neural network designed to handle sequence of data [33]. It memorizes information about the sequence of inputs and use it for accurate predictions by means of loops in the network that pass prior information forward. Basic RNN suffers from short-term memory: the more steps of a sequence to process, the more trouble to retain information about the previous steps. To solve this problem long short-term memory (LSTM) were developed [67]. LSTM learns long-term dependencies using gates that are capable of understanding what information to store and what to remove. These mechanisms are responsible for keeping track of the dependencies between the elements in the input sequence. We choose LSTM rather than basic RNN so that, in case of time series variables, the network can predict using information from the beginning of the series.

2.2 Machine learning in mosquito modelling: pros and cons

In the context of modelling mosquito abundance, ML adoption with neural networks [78, 65], random forests [12, 68, 65] and regression trees [20, 22] revealed a high predictive power, beating eventually the classical statistical methods [78, 68]. These techniques are gaining popularity among epidemiologists because they tackle problems for which classical statistical methods are not well-suited. There are, however, language and technical barriers that can make it difficult for epidemiologists to read and assess machine learning studies. We provide an overview of the advantages of ML as well as of the limitations that some ML approaches are trying to overcome. As already stated, ML is able to detect all patterns and trends beyond data that are not apparent to human. This property allows to draw insights from complex data, where the human formulation of statistical models is not sufficient. However, to conduct a proper training phase, the

amount of data has to be massive and of high quality. Again related to the training phase, the volume of data to be manage and the actual training process are time costly, but once this phase has been carried out, the model is available and ready to predict on any new data. Respect to statistical modelling, moreover, ML does not rely on any assumption about data.

Despite its great advantages, we have not discussed so far about the model returned by ML. The reason beyond this fact is that most of ML algorithms are black box method. This epithet means that it is not available to the user the model learnt by the algorithm. We can not therefore interpret the modelling result and infer the meaningful dynamics and patterns responsible of the target. In such a field, this lack is quite a huge disadvantage since the interpretation of the model is a fundamental characteristic. The choice of the ML model to deal with an epidemiological problem, therefore, must be carefully addressed.

The next Chapter is dedicated to the presentation of a particular ML method capable of combining the learning abilities of ML with the interpretability feature of statistical modelling. This technique is called Genetic Programming.

Chapter 3

Genetic Programming

3.1 Introduction to Evolutionary Algorithms

The machine learning approaches described in Chapter 2 mimic human knowledge acquisition to gain information about data. There is, however, another source of learning, the one that has been developing all species worldwide from ages ago: darwinian evolution. Evolution is the natural improvement of individuals or populations by means of several phenomena such as competition, struggle for survival and genetic inheritance.

Evolutionary algorithms (EA) are techniques that use the mechanisms of natural evolution to search the best solution to a problem. EAs should be defined as a nature-inspired /computational intelligence technique, which is part of the broader artificial intelligence methods. The application of EAs on supervised regression problem, however, include these techniques in the context of ML. An EA contains four overall steps: initialization, selection, genetic operators, and termination. These steps correspond, roughly, to a particular facet of natural selection. Figure 3.1 schematize the whole process of finding a solution with EA.

An EA starts with the *initialization* of a population of solution to the problem. The population contains an arbitrary number of possible solutions to the problem called *individuals*. Once a population is created, members of the population must be evaluated according to a *fitness function*. A fitness function is a function that takes into consideration the characteristics of an individual, and outputs a numerical representation of how viable of a solution it is. After *selecting* the top individuals according to the fitness function, these ones are used to create the next generation in the algorithm. Using the characteristics of the selected parents, new children are created that as mixture of the parents' qualities. The mixing processes are known as *crossover* and *mutation*. The children have their fitness evaluated and compete for survival with the current population to return the new generation. As in natural evolution, generation by generation the fitness of the population rises. Eventually, the algorithm must end. There are two cases in which this usually occurs:

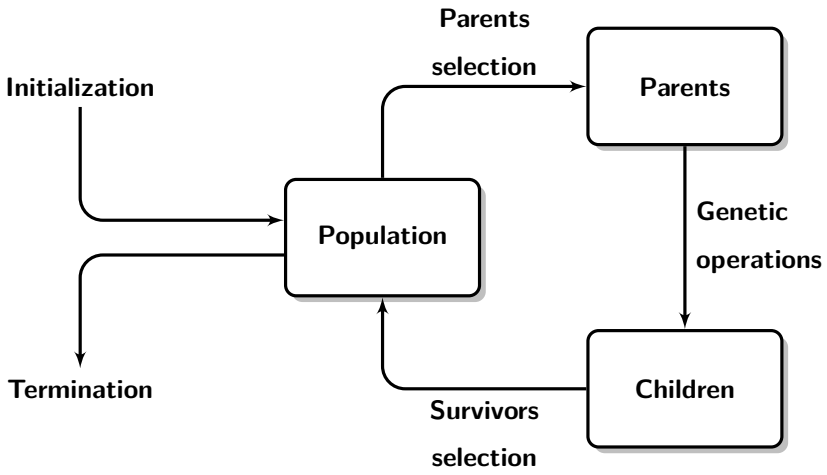


Figure 3.1: Flowchart of evolutionary algorithms.

either the algorithm has reached some maximum runtime, or the algorithm has reached some threshold of performance. At this point a final solution is selected and returned.

There are different types of EA which differ for the representation of the solution. According to the modelling problem of mosquito abundance, a suitable approach of EA is tree-based Genetic Programming (GP). We dedicate the next section to GP full presentation, giving emphasis to the features that make it an appropriate tool for epidemiological problem.

3.2 Genetic Programming: the Technique

Introduced in 1992 [60], Genetic Programming is a kind of EA that induces computer programs by evolutionary means [76]. According to the definition, therefore, the evolving solutions to the problem at hand are computer programs. Computer programs are collections of instructions that can be executed by a computer to perform a specific task. To give an example, $x + 3 \cdot (x + y + xy)$ is a computer program that can be evaluated, provided a value for x and y. In this perspective, a model of mosquito abundance, viewed as a combination of variables, is a computer program. A GP algorithm, thus, is capable of evolving models of mosquito abundance that over time better suits the data.

The basic process of GP to return the model of data follow the flow of Figure 3.2.

1. Randomly create an *initial population* of models using the available ingredients.
2. *Execute* each model and ascribe its *fitness*.
3. *Select* models from the population to participate in genetic operations.
4. Creates new individual by applying *genetic operations*.
5. *Repeat* from point 2 *until* an acceptable solution is found or some other stopping condition is met.
6. Return the *best model* so far.

Figure 3.2: Executional steps of basic GP.

Models in GP are expressed as syntax trees, built using the available ingredients provided by the user which are variables and constants, called *terminals*, and arithmetic operations, called *functions*. The set of all these ingredients is called *primitive set*. To give an example, let us assume that want to find out the relationship among $\{x_1, x_2, x_3\}$ responsible of y. The terminal set is $\{x_1, x_2, x_3\}$ plus some constants, and the functions set may be $\{+, \times, \max\}$, resulting in the primitive set $\{x_1, x_2, x_3\, 1, 0.4, +, *, \max\}$. A possible model for y can be therefore $\max(x_1 + x_2, x_1 + 0.4 * x_3)$, represented in tree structure as in Figure 3.3:

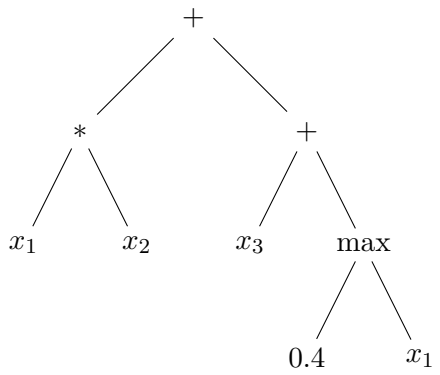


Figure 3.3: GP tree structure.

We now go into a deeper understanding of the main tools of GP architecture, previously mentioned in Figure 3.2. For a complete explanation, however, refer to [61].

Initial population. The starting point of GP is the random creation of a population of proposed models. The most frequent approaches to perform the initialization are the full method, the grow method and the *ramped half-and-half* method (RHH), a combination of the previous ones. All these techniques rely on a maximum tree depth defined by the user. In the full method nodes are taken randomly from the functions

set until the maximum depth is reached; in the grow method, instead, nodes are selected from the whole primitive set until the depth limit is reached. The RHH is the easiest way of combining the two previous methods: half of the population is created with the full method, the other half is created with the grow method.

***Execute* and ascribe a *fitness*.** The models execution consists in the evaluation of the trees expression substituting the values of the variables according to the dataset involved. The execution allows the calculation of a score, called *fitness*, for each model, which represents how well the model solve the problem at hand. Considering the example introduced before, the fitness of a model can be the mean squared errors between the value of y predicted by the model, and the value of y collected in the dataset. In this precise situation, the lower the fitness, the better the model.

Select. The models that do well, thus the ones with the better fitness, are probabilistically selected to breed and produce new models. This rule agrees with the Darwinian theory of reproduction: better individuals (models) are more likely to have children than inferior individuals. The most used sampling method is the *lexicographic parsimony pressure* introduced in [70]. This method chooses a random number of individuals from the population and the best of them is selected. The selection is based on the idea of placing fitness, then size in lexicographic order; that is, preferring smaller individuals only when fitness is identical. The process is repeated many times to provide a pool of best individuals.

Genetic Operations. The offsprings of the selected parents are created by two operators that simulate the Darwinian crossover and mutation. Given two selected parents, the *subtree crossover* selects a crossover node in each parent tree. Then, it creates the offspring by replacing the subtree rooted at the crossover node in a copy of the first parent with a copy of the subtree rooted at the crossover node in the second parent, as illustrated in Figure 3.4.

The *subtree mutation*, instead, randomly selects a mutation point in a tree and substitutes the subtree rooted there with a randomly generated subtree, as shown in Figure 3.5.

Repeat until. The population of offsprings join the current population and survival criteria select the new generation. The survival of the individuals is a two-phase process. First, all the individuals (parents and offsprings) are ordered by priority of survival that depends on the *elitism* level chosen. Each individual in the ordered list is granted survival, depending on the allowed population size. In order to combine the Darwinian evolutionary inspiration with the optimization need, the suitable elitism level is *keepbest*. The best individual from both parents and children is kept in the new population, so it receives the highest priority of survival,

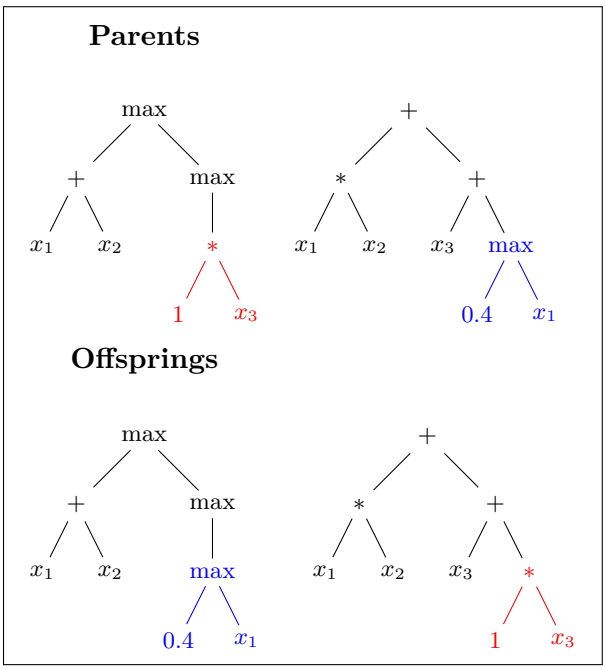


Figure 3.4: GP crossover. The swapping subtrees are marked in blue and red.

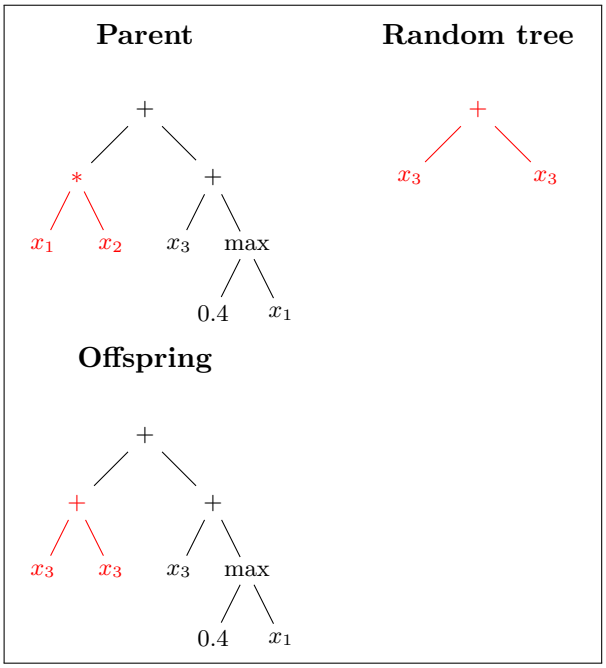


Figure 3.5: GP mutation. The mutating subtree and the random subtree are marked in red.

independently of being a parent or a child. The remaining individuals are ordered children first and then the parents.

The iterative process of measuring fitness and performing the genetic operations is repeated over many

generations until termination criteria are satisfied. These criteria can be fitness threshold or maximum number of generations.

Best model. The predictive model returned by GP algorithm is the model of the last evolved population with the best fitness.

A GP user has to be aware that every GP run returns a different model, because the algorithm starts from a randomly generated population subject to randomly genetic operations. It is mandatory therefore to perform several runs of GP and define a way to return just one final model among all the best ones.

3.2.1 Genetic Programming for mosquito abundance

There are few works that exploit GP in ecology, specifically marine ecology, and their results look very promising. In [71] GP is used to identify which environmental variables determine zooplankton abundance, while in [17] a similar approach is used to detect the drivers of planktonic population. To the best of our knowledge, GP has never been involved in problem concerning vector dynamics. Nonetheless we claim its implication in the epidemiological problem of predicting mosquito abundance due to its great properties that can sum up all the advantages of the modelling technique explored so far.

GP shares the same capability of ML methods of detecting complex interactions without requiring prior knowledge about data structure. Besides these features, GP has the great advantage of returning readable and even interpretable models since every tree structure can be explicated. Surely some tree structure are huge and a biological interpretation can be hard, but sometimes they reveal novel and unexpected interaction between variables. GP, moreover, is able to perform an implicit features selection. The survival of the fittest models during evolution, in fact, determines the survival of the variables they use which are surely the most meaningful ones since they are involved in the best predictive models. In addition, GP structure is highly modular and parametrized, and these properties allows the building and test of new methods from the initialization to the survival.

These characteristics of GP are of great importance in order to better align the modelling to the decision making process, thus further investigation of GP is highly recommended for vector abundance prediction.

Chapter 4

Thesis Goals and Motivations

The overall goal of this thesis is to explore the innovative use of GP in the field of vector abundance prediction. We start from the available data of *Culex pipens* counts collected in Piedmont region in the context of the surveillance programme promoted by IPLA [3]. The dataset was already used in [11] to address mosquito abundance prediction by means of statistical modelling, thus we have previous results to compare GP performance. Moreover, to include GP in the framework of ML for epidemiological problems, we benchmark GP performance against ML state of art.

In addition, we want to exploit GP easy structure to properly deal with the different data format that we may face. Time series are, in fact, frequently present in dataset concerning vector abundance, even in the specific dataset we are considering. The seasonal dynamics of vector population is likely to be associated with the fluctuation of climatic and weather variables over time, thus time series data. At the moment, classical methods in statistical and ML modelling are mostly unable to handle time series. In order to not collapse all the time series into a static features, classical methods, as shown in [11], split the series as different observations. This choice surely keep all the values of the series, but destroy the order information. The sequential order allows the identification of peaks and patterns and the lack of this knowledge penalize the accuracy of the modelling techniques. GP could potentially be extended to work with ordered sequences. This new approach will totally rely on raw data without prior transformation, and we claim it will bring great benefit to the accuracy since it will take informations directly from not distorted data. Moreover, this new approach, risen from problems in the field of epidemiology, will provide to the general modelling community a new method to properly treat any kind of ordered sequence (time series as well as space points).

The selection of GP as a modelling technique to be explored is based on several aspects. Primarily we reiterate that the monitoring of mosquito abundance can not be conducted without focusing even on the modelling methods. Investing enough effort to explore and enhance modelling techniques must, therefore, be of fundamental interest.

In particular, GP keeps the good abilities of ML in the context of complex problem without dismissing the interpretability of the model. These properties of GP has already been explored with great results in the field of ecological modelling, thus further exploration in the analogous field of epidemiological modelling are likely to reveal and confirm GP advantages.

We are aware that epidemiologists are unwilling to investigate ML methods due to the difficulty in understanding how the learning take place and where to focus to enhance it. GP provides a good bridge to let epidemiologist approach ML techniques. The learning algorithm and GP structure are easy and less obscure, thus it is even easier to think where to act to enhance results. To give an example, during a GP run it is possible to observe a slow evolution with a fitness of the population mainly constant. It is not difficult to understand that to speed up the process we can introduce genetic operators that produce more extreme modifications of individuals.

GP choice is further supported by the absence of a model selection to describe data. In statistical modelling it is fundamental to know the shapes of the different models and the meaning of the parameters to make a proper choice. In GP, instead, the only selection is in the predictive variables and in the mathematical operators to link the variables.

The dissertation is organized as follows. After the introduction in Chapters 1, 2, 3 to the problem and the methodologies, in Chapter 5 we investigate the application of GP on the mosquito problem mentioned before. The results revealed two main issues related to the time series involved in the dataset, that led us to the development of Vectorial GP(VE_GP). In Chapter 6 we presented the new approach of VE_GP as a GP technique suitable for time series prediction, potentially applied in a wider range of problems rather than only vector abundance. We provided a whole explanation of the technique and preliminary results of the application of VE_GP on benchmark problems, simulating the real problems we may face. Chapter 7 finally presents VE_GP application on mosquito problem. We compared the results against classical GP and we focused on what the new approach let us discover. We moreover put interest into the exploration of VE_GP as a concrete device to build up vector control strategies. We in fact interpreted the provided predictive model in terms of variables involved, variables associations and informative window of time identified. Since VE_GP is not only a tool to predict vector abundances, in Chapter 8 we tested its application to a problem belonging to a different area regarding the prediction of physiological time series. The dataset structure of this problem is analogous to the one of vector abundance prediction, thus this exploration let us strengthen VE_GP potential in real scenarios. Chapter 9 of the dissertation is dedicated to some attempts we made to enhance VE_GP predictive ability that actually made us realize the real goal of GP enhancement. Chapter 10, infact, presents a new approach of VE_GP that pave the way to the concept of assumption-free in ML. Finally IV makes the overall conclusion of the thesis.

Part II

Genetic Programming Application

Chapter 5

Genetic Programming on Mosquito Prediction

In this chapter, we report the investigation of genetic programming (GP) use in predicting mosquito abundance. To highlight the pros and cons of GP we compare the technique with machine learning (ML) and statistical methods already applied in the ecological field, in particular we considered a generalized linear mixed model (GLMM), a random forest (RF), an extreme gradient boosting (XGBoost) and a multilayer perceptron (MLP). The results revealed the high predictive accuracy of GP and the great capability of generalization. However, a critical issue emerged: none of these methods treats properly time series.

5.1 The dataset involved

We started the exploration of GP on mosquito abundance prediction using the *Cx. pipiens* counts from 2002 to 2006 collected in the context of IPLA surveillance plan. The choice of the years and of the mosquito species depends only on the quality of data. We considered the same predictive variables of [11], selected as the most informative about the abundance of mosquitoes. The variable *TWEEK* is the average Land Surface Temperature (LST) from 8 to 15 days prior to the trapping day derived from the Moderate Resolution Imaging Spectro radiometer (MODIS) satellite (National Aeronautics and Space Administration, NASA [75]). *RAIN* represents the cumulative rainfall from 10 to 17 days before trapping, registered by the nearest weather station to the trap [1]. Both these variables capture the effect of climatic conditions on mosquito population abundance during a precise window of time. Vegetation changes deemed influential on mosquito dynamics are caught by the Normalized Difference Vegetation Index (*NDVI*), derived again from MODIS [75] as 16 days average. While these variables change on each day of collection, some others are constant and inform about the environment surrounding the trap. *DISTU*, *DISTR*, and *DISTW* estimate the distance of sampling

locations to the nearest urban center, rice field and woodland respectively. The area covered by rice fields is registered in the variable *RICEA*. The elevation of the trap location (*ELEV*) is included among predictors to spot the impact of altitude. The last variable involved is *SIN*, a sinusoidal curve with a phase of one year. It is an artificial informer of the seasonality of mosquitoes, reflecting the shape of mosquito abundance across the year. Its value, thus, has a peak in the first week of August where experts know there is a peak in mosquito abundance.

The entire dataset used is thus composed of 3600 observations, each one corresponding to a collection of mosquitoes on a precise day and trap. Every observation contains the values of the nine predictors (*TWEEK*, *RAIN*, *NDVI*,*DISTU*, *DISTR*, *DISTW*, *RICEA*, *ELEV* and *SIN*) and the value of the dependent variable, or target, which is the number of mosquitoes collected.

To perform the experiments, we split the dataset into a learning and test set, following the natural order of the years. Collections from 2002 to 2005 are used as the learning set to tune and train the algorithms, while collections of 2006 form the test set, therefore they represent unseen data, used to test the generalization performance of the predictive models. This approach is naturally determined by the real problem at hand: developing predictive models for mosquito abundance. Thus, we used data from the past (2002-2005) to train models and we assessed their generalization ability by evaluating them on future predictions (2006).

5.2 The techniques involved

We used different tools and software to run the chosen algorithms, namely MATLAB R2018A, GPLab [72] and R. Nonetheless we do not provide a description of all the parameters, especially for the ones kept at the default value of the respective implementations. Some parameters that needed to be manually inserted were tuned according to the following strategy: we proposed a value for the parameter and we estimated the performance of the algorithm by running it 60 times on 60 different splits of the learning set; in each run the algorithm was trained on 75% of the learning set and validated on the remaining 25%; the median result over the 60 runs on the 25% measures the performance of the algorithm using that particular value of the parameter. In the end, we used the configuration that allowed us to obtain the best performance. In the following description, we specify which software was used and which parameters were tuned in each case. All the details about the techniques are found in Chapter 1.

5.2.1 Genetic programming

To perform GP experiments the preliminary parameter to be declared is the primitive set. In our case, the terminal set T consists of the predictors described in Section 5.1, plus random constants r between 0 and 1 generated in runtime when building trees. Therefore $T = \{TWEEK, RAIN, NDVI, DISTU, DISTR, DISTW,$

$RICEA, ELEV, SIN, r\}$. The functions set F includes the usual binary addition, subtraction and multiplication operators, plus a protected version of the division, known as `kozadivide`, that returns 1 when the denominator is equal to zero. Thus the set is $F = \{\texttt{plus}, \texttt{minus}, \texttt{times}, \texttt{kozadivide}\}$. Fitness is measured as the Root Mean Squared Error (RMSE) between the predicted values and the real mosquito abundances.

The computational tool used for GP experiments is GPLab [72], a public domain GP system implemented in MATLAB. The parameter setting considered is reported in the upper part of Table 5.1, and it corresponds to the default values provided by GPLab.

5.2.2 Generalized linear mixed model

Since the first modelling technique to approach our problem was the GLMM proposed in [11] we included this model in the comparison. The authors defined as random effects *RNDtrap*, which represented the spatial difference between traps (the subjects) and *SRNDtrap*, which represented the effect of the spatial location of each trap. The model expression provided is:

$$y = I + \beta_1 * RAIN + \beta_2 * TWEEK + \beta_3 * SIN + \beta_4 * ELEV + \beta_5 * DISTU + RNDtrap$$

where I is the intercept representing the fixed effect and y is the abundance of mosquitoes. All the analyses and training were performed using the R software package [2].

5.2.3 Random forest

In this work, we use an RF implemented in R [5], which requires the manual input of the number of trees participating in the forests. Therefore, according to the strategy described above, we tuned this parameter, investigating values from 100 to 700. The selected value was 700, as reported in Table 5.1.

5.2.4 Extreme gradient boosting

The implementation we chose for XGBoost is the one contained in R [6], which requires the number of rounds (iterations) to be specified by the user. According to the tuning technique described above, we investigated the best number of trees to configure, called the number of rounds, from 1 up to 20. This value and other main parameters of XGBoost in R are listed in Table 5.1.

5.2.5 Multilayer perceptron

We adopted multilayer perceptron implementation included in the Matlab Neural Network toolbox [55]. Following the strategy previously described, we tuned the number of hidden neurons considering a network with just one hidden layer. We explored all the values between 1 and the number of input variables as suggested by a frequently used rule of thumb [8]. All the main parameters used are described in Table 5.1.

Table 5.1: Parameters used in the experiments.

GP Parameters	
population size	500
max number of generations	100
initialization	Ramped Half and Half [61]
selection method	Lexicographic parsimony pressure [70]
elitism	best individual kept
crossover rate	0.9
mutation rate	0.1
max tree depth	17
RF Parameters	
number of trees	700
XGBoost Parameters	
η learning rate	0.3
max tree depth	6
number of rounds	7
MLP Parameters	
learning algorithm	Levemberg-Marquardt backpropagation [36]
hidden neurons	1
μ increase factor	0.1
μ decrease factor	10
epochs	1000

5.3 Experiments and results

5.3.1 Experiments setup

Our objective was to study GP's ability in predicting mosquito abundance during 2006, based on historical data collected from 2002 to 2005, and to show its competitiveness by comparing it with other methods. After a tuning phase of some parameters, RF, XGBoost, MLP and GLMM were trained on the learning set and then evaluated on the unseen collections of 2006. Since GP is a population based and stochastic technique, the training phase is conducted differently. We run the algorithm 60 times; in each time the population was trained on a random sample of 75% instances of the learning dataset. In each run, the individual of the final population that better performed in the remaining 25% of the learning dataset (called validation set) was selected as the best predictive model found by GP. The final 60 best models composed the sample population of GP models. Each of the GP models was then evaluated on the test set.

We measured the accuracy of prediction by comparing the RMSE between predicted and real collections on the test data. Statistical significance of the null hypothesis of no difference in performance between GP and each of the other method was based on one sample Wilcoxon signed rank test at $\alpha = 0.0125$, after Bonferroni correction. For further comparison, we measured the overfitting as the difference between the test and learning set RMSE.

5.3.2 Results

Table 5.2 summarizes the RMSE returned by each techniques on the test set. Regarding GP, we report the median RMSE over the 60 models, choosing the median instead of the mean as a more robust descriptor of outliers, which are likely to be found in stochastic methods. Table 5.3 presents the p-values of the Wilcoxon tests comparing GP with the other methods.

Table 5.2: Statistics about the RMSE of the different techniques on the test set.

GP	**RF**	**XGBoost**	**MLP**	**GLMM**
83.8 (median)	83.0	87.9	83.7	85.5

Table 5.3: Statistical significance of the difference in performance between GP and the other methods.

GP vs RF	GP vs XGBoost	GP vs MLP	GP vs GLMM
$p = 3.1 \cdot 10^{-7}$	$p = 1.7 \cdot 10^{-11}$	$p = 0.2$	$p = 1.2 \cdot 10^{-10}$

According to the statistical tests, GP performance differs from all the other methods except MLP. The boxplot in Figure 5.1 shows that GP is only outperformed by RF.

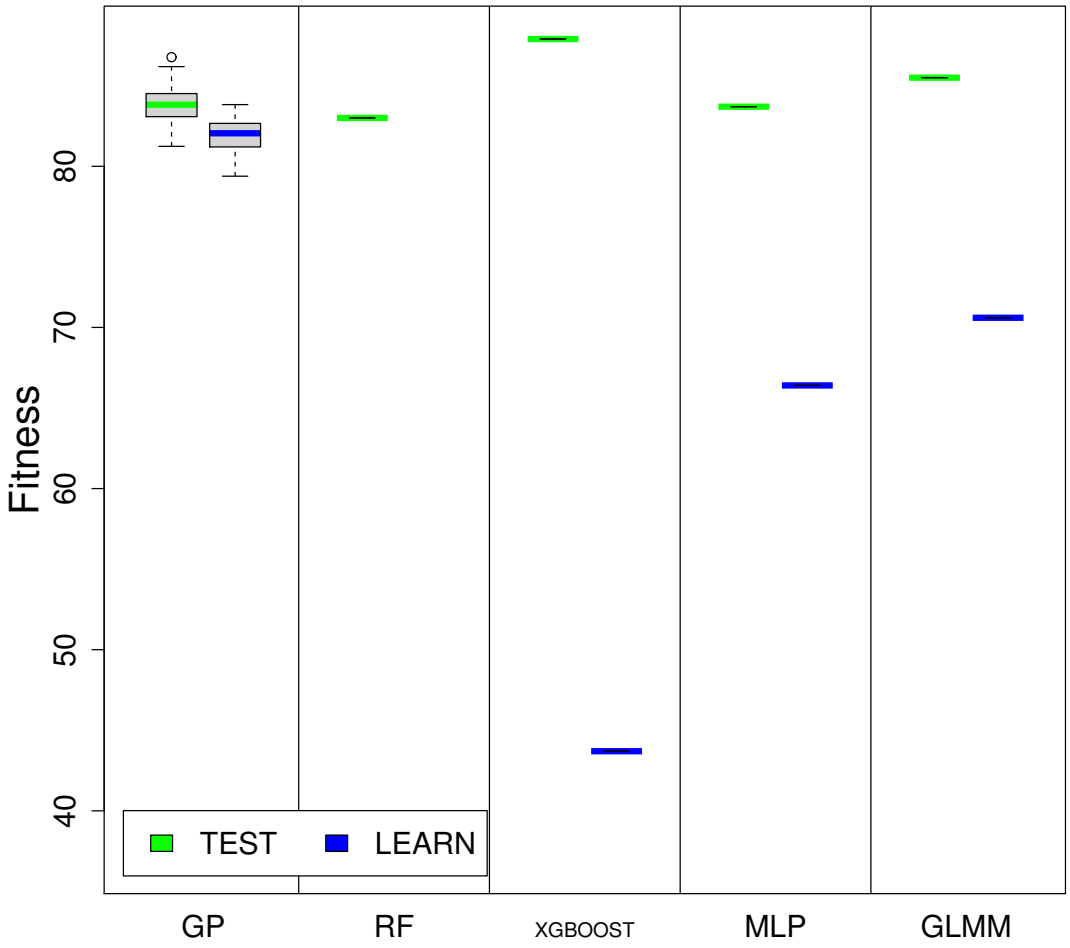


Figure 5.1: RMSE on both the learning and test set for the different algorithms. Test set results are plotted in grey, while learning set results are plotted in black.

The results on GP performance immediately suggest that the relationship among variables is more complex than the one previously designed with a GLMM in [11]. However, at first glance, GP does not seem to be the technique that returns the best result. Nonetheless, the quality of predictions should also take into account the quality of learning subject to overfitting. As mentioned above, we decided to quantify overfitting by calculating the difference between the learning and test median RMSE. For RF and MLP, this measure returns 46.1 and 17.4 respectively. This result indicates severe overfitting for RF. Contrarily, the difference between the learning and test RMSE for GP is only 1.8. We hypothesize that both RF and MLP, even though the latter being in a more reduced form, are not learning the existing relationship between the variables. The boxplot of Figure 5.1 shows that the same phenomenon appears even more substantially in XGBoost, which is generally considered the top machine learning method nowadays. A possible reason for

this fact, strengthened by the RF results, is that regression trees are not suitable for the problem at hand.

To corroborate the appropriateness of using GP, compared to the other studied techniques for the problem at hand, we calculated the percentage of GP models in the sample with lower RMSE compared to MLP and RF on the test set. Respectively 22% and 43% of GP models outperformed RF and MLP.

The comparison of GP against other models, however, should be treated as a way to frame GP performance in the context of ML modelling. The goal is not to prove GP as the outperforming technique.

5.3.3 The model for mosquito dynamics

Another competitive advantage of GP compared to other machine learning methods, including the ones studied here, is the possibility of reading and interpreting the model. Despite the fact that decision trees are easy to understand representations in logical terms, the process of averaging the results coming from multiple decision trees in an RF muddies the logic. Since model interpretability is a key feature of ecological modelling, GP shows its benefit in the field again.

We selected as the predictive model for mosquito abundance the best model on its validation set among the 60 best models. The resulting RMSE on the test set is equal to 83.4. The complex equation is still readable and interpretable, thus we report in Equation (5.1) its expression. The symbols used are the ones classically associated with the primitive functions reported in Table 5.1; when no symbol is found, a multiplication is occurring.

$$\begin{aligned}\#\text{Mosquitoes} =& \frac{DISTW \cdot SIN^3}{DISTU} + \frac{(TWEEK - DISTR + DISTW)SIN^7}{DISTR} \\ &+ \frac{DISTW \cdot RICEA \cdot SIN^5}{TWEEK} + 2SIN^5 + \frac{DISTW \cdot SIN^5}{DISTU} \\ &- DISTR \cdot SIN^3 + (TWEEK - DISTR)SIN^8 \\ &+ \frac{2SIN^3(2SIN + 1)}{DISTU} + \frac{DISTU \cdot SIN^3}{DISTR} + 2DISTW \cdot SIN^3 \\ &- \frac{DISTW \cdot SIN^2}{DISTR} - DISTR \cdot SIN + DISTW\end{aligned} \tag{5.1}$$

Since there were quite a lot of occurrences of `kozadivision` in the expression, we checked whether their result was frequently the constant 1 used to protect the division when the denominator is zero. Luckily, on the test set, the protected version of the division was not used too much, therefore the divisions involved are mainly true division.

The general expression reported in Equation (5.1) may be hard to interpret, but the analysis of the variables discarded and the general effect of the ones selected may provide meaningful information. To investigate the role of each variable in the prediction we firstly considered, as a subjective measurement, the

number of times each variable appears in the model. Table 5.4 shows these frequencies of occurrence of the single variables in the model.

Table 5.4: Frequency of each variable in the best model.

Variable	Frequency
TWEEK	3
RAIN	0
NDVI	0
DISTU	4
DISTR	7
DISTW	7
RICEA	1
ELEV	0
SIN	13

The implicit feature selection embedded in GP reveals that *RAIN*, *NDVI* and *ELEV* are not informative about mosquito abundance. Interestingly, the most frequent variable is *SIN*. We performed an analogous investigation also in the GLMM model. The standardized coefficients, in fact, give a measure of the change in the target (in standard deviations) for every standard deviation change in the predictors. Since the higher standardized coefficient is the one of *SIN* ($\beta_{SIN} = 0.02, \beta_{DISTU} = -0.003, \beta_{ELEV} = -0.007, \beta_{TWEEK} = -0.005, \beta_{RAIN} = 0.003$), we can state that also when using the GLMM this variable is considered as the most informative one. A similar analysis could not be carried out for the other techniques, since they were mainly black box methods.

5.4 What the results suggest

An issue that emerged from the results presented so far deserves further attention. The variable *SIN* is an artificial predictor included in the dataset to suggest the period of collection and therefore how high the abundance of mosquitoes is expected to be. Both GP and GLMM models strongly rely on this variable, which undoubtedly provides lots of information concerning mosquito abundance. However, this predictor is the result of prior knowledge about the problem and we would rather prefer the algorithms to directly infer from data which combination of environmental and climatic variables determines the fluctuations of mosquito abundance. The main problem not allowing this detection is the improper treatment of these environmental and climatic variables.

LST, NDVI, and the rainfall are time series predictors whose flow is related to seasons, thus they can be strong predictors of mosquito dynamics. Nonetheless, they are not recognized as such by GP (and even GLMM) probably because they are not treated as time series. Time series were managed as independent training cases by all the machine learning techniques studied in this paper, including GP. This element may cause a loss of information, which may deteriorate the ability to predict mosquito abundance.

These arguments inspired us to develop a new approach of GP with the main goal of solving time series issues and therefore becoming a powerful tool in the field of vector dynamics prediction. This new technique of GP is called vectorial genetic programming (VE_GP). As the adjective "vectorial" suggests, the technique is based on the introduction of vectors as new terminal representation suitable for time series. In Chapter 6 we will explain in details the new approach of VE_GP, focusing the attention on how VE_GP exploits the new terminal representation to extract knowledge from time series.

Published Original Research Article Azzali I., Vanneschi L., Mosca A., Bertolotti L., Giacobini M., Towards the use of genetic programming in the ecological modelling of mosquito population dynamics. *Genet Program Evolvable Mach*, 21, 629–642 (2020).

Chapter 6

Vectorial Genetic Programming

Among the several existing typologies of problems to which Genetic Programming (GP) [61] can be applied, symbolic regression is undoubtedly one of the most popular. The objective of symbolic regression is to find a function that describes the relationship between inputs and corresponding outputs, developing a model that can be used to make predictions on new inputs. Of great importance, belonging to the family of symbolic regression, is *panel data forecasting*. A panel dataset is a collection of observations for multiple subjects at different equal-spaced time intervals [13]. Therefore, if the independent variables among the M observations measured are $X_1^i, \ldots, X_N^i$ where $i = 1, \ldots, M$ and $X_K^i, \ldots, X_N^i$ change in time ($X_{j_t}^i$ with $i = 1, \ldots, M$, $j = K, \ldots, N$ and $t = 1, \ldots, T$ denoting time series variables) and Y is a dependent variable (Y_t^i with $t = 1, \ldots, T$ denoting a target time series variable), we can express the dataset as

$$\{X_1^i, \ldots, X_{K-1}^i, X_{K_1}^i, \ldots, X_{K_T}^i, \ldots, X_{N_1}^i, \ldots, X_{N_T}^i, Y_1^i, \ldots, Y_T^i\}$$

where $i = 1, \ldots, M$ refers to the subject being observed. The interest of panel data regression lies in predicting dependent variables which are hard to measure. To clarify, let us consider the example panel reported in Table 6.1. In this example, trap location characteristics are collected for traps and days in order to predict mosquito abundance. The standard GP approach can be easily applied to panel data regression, as described in Chapter 5. However, there can be a potential disadvantage. Data instances (fitness cases) are treated independently. Therefore the algorithm is not able to recognize that two (as in lines 1 and 2 in Table 6.1), or more, observations belong to the trap. This situation may result in a loss of knowledge regarding the time series, that may instead have been useful to effectively model the target.

The idea to overcome the problem is to design a novel GP system, that we call *Vectorial GP* (VE_GP), able to exploit the source of information provided by the additional dimension of time of panel datasets. To make the algorithm consider the whole time-series, we aggregate related data instances referring to the same entity in a vectorial representation, so that variables that change in time become *vectorial variables*.

Table 6.1: Example standard panel dataset.

Trap Id	Distance_city	Elevation	Rain	*Mosquitoes*
1	12	121	15.2	0
1	12	121	0	150
2	5.8	56	2.3	120
2	5.8	56	2.3	93
2	5.8	56	0	102
3	3.4	90	4.5	75

Therefore, the panel dataset represented in Table 6.1 is transformed into the representation reported in Table 6.2. In this configuration, a GP tree can be composed either by scalar terminals (to represent features

Table 6.2: Example vectorial panel dataset.

Trap Id	Distance_city	Elevation	Rain	*Mosquitoes*
1	12	121	[15.2,0]	[0,150]
2	5.8	56	[2.3,2.3,0]	[120,93,102]
3	3.4	90	4.5	75

such as Distance_City, Elevation and Trap ID) or by vectorial terminals (for instance to represent Raind and Mosquitoes). Moreover, the technique we propose adds new functions to the primitive set, defined with the purpose of describing the behaviour of temporal variables. From now on, we use the term *time series* to indicate an observation of a vectorial variable, also called *time series variable*. Therefore, a time series is a sequence of recorded values, belonging to one entity and represented as a vector.

6.1 Previous works about time series in GP

Working with time series in GP has always been a challenging problem due to the inherent difficulty of handling this type of data. Common strategies include feature extractions to reduce the series into scalar features [29] or element by element treatment, where each entry of the series is an independent terminal [32]. However, some previous works explored the idea of keeping the native data type of time series, the vector. Holladay et al. [41] introduced a vector-based GP to predict the feature vector of fixed length signals. In this

paper, vectors were possible inputs rather than scalars, and the primitive set included domain dependent functions that act on both of them. A more recent work of Bartashevich et al. [59] allows vectors as primitives and include vectorial functions such as the cross and dot product to build GP individuals. Other approaches were proposed to preserve the ordered essence of time series such as Vera et al. [19]. In this latter work the authors investigated the serial processing of data where the time sequence is presented to the algorithm in series so that the elements of a sequence are processed in the same order as they are recorded.

Starting from the idea of [41] we move further to provide a more exhaustive vector-based GP. Our VE_GP is less problem-specific and aims at providing an algorithm able to deal with any naturally ordered variable of any length. Moreover, in VE_GP we included new structures that advance the search space of the considered problems. VE_GP elevates the capabilities of previously proposed vector-based approaches and provides a more sophisticated technique.

6.2 Vectorial GP

VE_GP is built on top of the GPLab toolbox [72]. GPLab includes most of the traditional features usually found in many GP systems. We chose it because its highly modular structure makes it a particularly versatile, generalist and easily extendable tool, highly suited for testing new elements and techniques. Moreover, it is written in MATLAB, which provides a particularly appropriate environment to manage vectors. In the following paragraphs we describe the primitive set and other particular elements of VE_GP.

Functions of arity one

Since VE_GP is specific for time series variables, we integrated the set of classical arity one functions with aggregate functions. Standard aggregate functions collapse the whole time series variable into a single value of more significant meaning. They can be included in the primitive set when we deal with time series prediction based on past time series variables. We even face problems where the time series target flows simultaneously to the time series predictors, which means that the time instants, corresponding to the entries of a vector, are the same for both the target and the predictors. Therefore, specially meant for this latter problem, we added cumulative aggregate functions. These operators applied on a time series return a vector whose entries are the aggregate values of only previous time values. These versions of the aggregate function, the standard and the cumulative ones, allow GP to foresee any kind of time series, from the ones that take place during the recording of data to the future ones. All the arity one functions can be also easily applied to scalars, considering them as a vector of 1×1 dimension. The primitives of arity one used by VE_GP are described in Table 6.3.

Table 6.3: Description of functions of arity one. The columns represent the primitive function (first column), MATLAB name (second column) and the outcome of the function (third column) applied on a given vector v (in this example, v=[1, 2.5, 4.3, 0.7]).

Primitive function (pf)	**MATLAB name**	**pf(v)**
Mean	`V_mean`	`V_mean`$([1, 2.5, 4.3, 0.7]) = 2.1$
Max, Min	`V_max`, `V_min`	`V_max`$([1, 2.5, 4.3, 0.7]) = 4.3$ `V_min`$([1, 2.5, 4.3, 0.7]) = 0.7$
Sum	`V_sum`	`V_sum`$([1, 2.5, 4.3, 0.7]) = 8.5$
Mode	`V_mode`	`V_mode`$([1, 2.5, 4.3, 0.7]) = 0.7$
Length	`V_length`	`V_length`$([1, 2.5, 4.3, 0.7]) = 4$
2-Norm	`V_2norm`	`V_2norm`$([1, 2.5, 4.3, 0.7]) = 5.1$
Cumulative mean	`C_mean`	`C_mean`$([1, 2.5, 4.3, 0.7]) = [1, 1.8, 2.6, 2.1]$
Cumulative sum	`C_sum`	`C_sum`$([1, 2.5, 4.3, 0.7]) = [1, 3.5, 7.8, 8.5]$
Cumulative max, min	`C_max`, `C_min`	`C_max`$([1, 2.5, 4.3, 0.7]) = [1, 2.5, 4.3, 4.3]$ `C_min`$([1, 2.5, 4.3, 0.7]) = [1, 1, 1, 0.7]$
Exp, Log, Cos, Sin, Abs, Square, Cube, Sqrt	`V_exp`, `V_log`*, `V_cos`, `V_sin`, `V_abs`, `V_2`, `V_3`, `V_sqrt`* *(protected version as [61])	`V_exp`$([1, 2.5, 4.3, 0.7]) = [2.7, 12.2, 73.7, 2.0]$ `V_log`$([1, 2.5, 4.3, 0.7]) = [0, 0.9, 1.5, -0.4]$ `V_cos`$([1, 2.5, 4.3, 0.7]) = [0.5, -0.8, -0.4, 0.8]$ `V_sin`$([1, 2.5, 4.3, 0.7]) = [0.8, 0.6, -0.9, 0.6]$ `V_abs`$([1, 2.5, 4.3, 0.7]) = [1, 2.5, 4.3, 0.7]$ `V_2`$([1, 2.5, 4.3, 0.7]) = [1, 6.3, 18.5, 0.5]$ `V_3`$([1, 2.5, 4.3, 0.7]) = [1, 15.6, 79.5, 0.3]$ `V_sqrt`$([1, 2.5, 4.3, 0.7]) = [1, 1.6, 2.1, 0.8]$

Functions of arity two

Concerning arity two we included new functions inspired by classical vector operations. These functions can manage vectors of different lengths completing the shortest one with the null-element of the function

involved. In the case of a scalar and a vector as inputs we provided a specific evaluation in order not to consider scalars as 1×1 vectors. The primitives of arity two used by VE_GP are described in Table 6.4. The functions `VSUMW`, `V_W`, `VprW` and `VdivW` are called standard arity two functions.

Table 6.4: Description of functions of arity two. The columns represent the primitive function (first column), MATLAB name (second column), the outcome of the function (third column) applied on a given scalar s and vector v1 and the outcome of the function (fourth column) applied on two vectors v1 and v2 (in this example, s=0.2, v1=[1, 2.5, 4.3], v2=[0.1, 3.2, 4, 1.1]).

Primitive function (pf)	**MATLAB name**	**pf(s,v1)**	**pf(v1,v2)**
Element-wise sum	`VSUMW`	`VSUMW`$(0.2, [1, 2.5, 4.3]) = [1.2, 2.7, 4.5]$	`VSUMW`$([1, 2.5, 4.3], [0.1, 3.2, 4, 1.1]) = [1.1, 5.7, 8.3, 1.1]$
Element-wise	`V_W`	`V_W`$(0.2, [1, 2.5, 4.3]) = [-0.8, -2.3, -4.1]$	`V_W`$([1, 2.5, 4.3], [0.1, 3.2, 4, 1.1]) = [0.9, -0.7, 0.3 - 1.1]$
Element-wise product	`VprW`	`VprW`$(0.2, [1, 2.5, 4.3]) = [0.2, 0.5, 0.9]$	`VprW`$([1, 2.5, 4.3], [0.1, 3.2, 4, 1.1]) = [0.1, 8, 17.2, 1.1]$
Scalar	`VscalprW`	`VscalprW`$(0.2, [1, 2.5, 4.3]) = 1.6$	`VscalprW`$([1, 2.5, 4.3], [0.1, 3.2, 4, 1.1]) = 26.4$
Element-wise division	`VdivW`* *(protected version as [61])	`VdivW`$(0.2, [1, 2.5, 4.3]) = [0.2, 0.08, 0.05]$	`VdivW`$([1, 2.5, 4.3], [0.1, 3.2, 4, 1.1]) = [10, 0.8, 1.1, 0.9]$
Scalar Division	`VscaldivW`* *(protected version as [61])	`VscaldivW`$(0.2, [1, 2.5, 4.3]) = 0.3$	`VscaldivW`$([1, 2.5, 4.3], [0.1, 3.2, 4, 1.1]) = 12.8$

Parametric aggregate functions

We introduced parametric aggregate functions that apply the referring aggregate function only to the values belonging to the time window described by parameters. Regarding standard aggregate functions, the parameters p and q define respectively the initial and final position of the range to be considered. Therefore the standard aggregate function is applied to the input values of position $p, \ldots, q-1, q$. To have an admissible range $p < q$. Concerning cumulative aggregate functions, we remind that the output is a vector. The $i-$ith entry of the output depends on the values belonging to the window described by parameters p and q. In this case, p defines how far to look back from the i position determining the initial value of the range, while q defines how many values to consider. Thus, the $i-$ith entry of the output is the aggregate function applied on input values of position $i-(p-1), \ldots, i-(p-1)+q-1$. To have admissible range in this case $p > q$. The primitives of the parametric aggregate function used by VE_GP are described in Table 6.5. It is noteworthy

that many of the new functions are not replicable by the standard GP.

Table 6.5: Description of parametric aggregate functions. The columns represent the primitive function (first column), MATLAB name (second column), the outcome of the function (third column) applied on a given vector. For standard functions p is the initial position while q is the final position of the range, instead for cumulative functions p is the number of backward steps to be made in order to determine the initial position of the range while q is the amplitude of the range (in this example, v=[1, 2.5, 4.3, 0.7, 1.6], (p,q)=(2,3) for standard functions, (p,q)=(3,2) for cumulative functions).

Primitive function (pf)	**MATLAB name**	**pf$_{p,q}$(v)**
Mean in [p,q]	V_mean$_{p,q}$	V_mean$_{2,3}$([1, 2.5, 4.3, 0.7, 1.6]) = 3.4
Max in [p,q], Min in [p,q]	V_max$_{p,q}$, V_min$_{p,q}$	V_max$_{2,3}$([1, 2.5, 4.3, 0.7, 1.6]) = 4.3 V_min$_{2,3}$([1, 2.5, 4.3, 0.7, 1.6]) = 2.5
Sum in [p,q]	V_sum$_{p,q}$	V_sum$_{2,3}$([1, 2.5, 4.3, 0.7, 1.6]) = 6.8
Cumulative mean in [p,q]	V_Cmean$_{p,q}$	V_Cmean$_{3,2}$([1, 2.5, 4.3, 0.7, 1.6]) = [0, 1, 1.8, 3.4, 2.5]
Cumulative max in [p,q], Cumulative min in [p,q]	V_Cmax$_{p,q}$, V_Cmin$_{p,q}$	V_Cmax$_{3,2}$([1, 2.5, 4.3, 0.7, 1.6]) = [0, 1, 2.5, 4.3, 4.3] V_Cmin$_{3,2}$([1, 2.5, 4.3, 0.7, 1.6]) = [0, 1, 1, 2.5, 0.7]
Cumulative sum in [p,q]	V_Csum$_{p,q}$	V_Csum$_{3,2}$([1, 2.5, 4.3, 0.7, 1.6]) = [0, 0, 3.5, 6.8, 5]

Initialization

Given the new representation in VE_GP, several challenges arise. First, a big number of scalar inputs can cause a poor initial representation of new functions and terminals, as such barely used during the evolutionary process. Second, it is possible to obtain final solutions whose output is a scalar and not a vector because many of the integrated functions collapse a vector into a scalar. We designed a different initialization

strategy which releases unique and innovative characteristics of VE_GP during the evolution. The strategy is resumed in the procedure described in Figure 6.1. The motivation behind the second rule is to ensure a representative amount of trees in the initial population whose output is not a scalar. Similarly, the third rule ensures trees where the new functions are meaningfully used. In our opinion, the initialization strategy that we propose does not introduce significant bias in the evolutionary process, contrarily, it aims to aid VE_GP to free its full potential during the evolution.

Create an empty population P (the initial population) of size N.

1. Generate $n1$ trees in P using Ramped Half-and-Half initialization algorithm (RHH) [61];
2. Generate $n2$ trees in P using RHH such that each tree always generates an output which is a vector:
   (a) randomly generate a tree t by means of RHH;
   (b) randomly select a standard primitive function pf of arity two;
   (c) randomly select a vector-terminal v;
   (d) create the following tree, using post-fix notation, *(pf t v)*;
3. Generate $n3$ trees in P using RHH where aggregate primitive functions are forced to receive a vector-terminal as an input.

Figure 6.1: Proposed initialization strategy.

We furthermore initialized the values of the parameters for the aggregate functions. Since a time series variable can have a different length among fitness cases, we randomly set p and q between 1 and the maximum time series length for all the parametric aggregate functions.

Genetic operators

VE_GP includes a new type of mutation. Besides the classical one there is the mutation of aggregate function parameters. This operator allows parameters to evolve so that the most informative window where to apply the relative aggregate function is found. Firstly, the algorithm searches the tree for parametric aggregate functions and randomly selects one. Secondly, a random parameter is chosen and mutated according to the procedure reported in Table 6.6. Every time a genetic operator requires a new tree, parameters are set according to the initialization default values.

Table 6.6: Parameters mutation.

Standard aggregate functions parameters p,q	**Cumulative aggregate function parameters p,q**
• Random selection of p or q; • If p randomly change it from 1 to q; • If q randomly change it from p to the maximum time series length.	• Random selection of p or q; • If p randomly change it from 1 to the maximum time series length; • If q randomly change it from 1 to p.

6.3 Experiments to validate VE_GP approach

6.3.1 Benchmark problems

We tested the proposed VE_GP against a standard GP system (GP) on four benchmark problems. To investigate the competitiveness of VE_GP we chose a first problem where the target does not involve the new primitive functions in order to see if VE_GP is penalized by having unnecessary functions and structures. Three more problems include some of the new functions in the target, and they are meant to show the performances of both algorithms considering that GP can just try to approximate the new functions at its best.

Korns5 This benchmark problem is inspired by Korns problem number five for symbolic regression [35]. We chose to involve four variables of random numbers between -50 and 50 as the input, named $X1, X2, X3, X4$ respectively. Differently from the true Korns problem number five, the latter variable $X4$ for our experiment is a vector of length 10. The target expression is:

$$K5 = \texttt{VSUMW}(3.0, \texttt{VprW}(2.13, \texttt{V\_log}(X4))).$$

The dataset for VE_GP consists of 1000 instances, while for GP it consists of 10000 instances because we vertically untied the variable $X4$ and the target $K5$ to have the classical panel data representation.

Benchmark1 This benchmark problem is a new one that makes use of the aggregate functions implemented to produce the target. The vectorial dataset consists of 100 instances, where each instance is composed by four features: a random number between -10 and 10, another random number between -10 and 10, a vector of random numbers between 10 and 40 of length 10, and a vector of random numbers between -5 and 5 of

length 10. Naming the variables in order as $X1, X2, X3$ and $X4$ the target reads as follows:

$$B1 = \texttt{V\_W}\Big(\texttt{VSUMW}(X4, \texttt{V\_mean}(X3)), \texttt{V\_W}(\texttt{VscalprW}(X3, X4), X2)\Big).$$

Conversely, for the scalar dataset, we vertically untied the vectorial features so that it consists of 1000 instances.

Benchmark2 This benchmark problem involves the cumulative functions described in the previous section. The vectorial dataset is the same used for the previous benchmark B1, while the target is now

$$B2 = \texttt{VprW}\Big(\texttt{VdivW}(\texttt{VSUMW}(X3, X1), \texttt{V\_Cmean}(X4)), X2\Big).$$

Again for the scalar dataset we vertically untied the vectorial variables.

Benchmark3 This benchmark problem includes parametric aggregate functions. Therefore the evolution of parameters is integrated in VE_GP. The variables involved are five: $X1$ is a vector of length 20 of random numbers between 10 and 30, $X2$ is a random number between 50 and 60, $X3$ is a random number between 5 and 10, $X4$ is a random number between -2 and 2, and $X5$ is a random number between 0 and 1. The target is:

$$B3 = \texttt{VSUMW}(\texttt{VprW}(\texttt{V\_Cmin}_{3,3}(X1), \texttt{VdivW}(X2, X3)), X4).$$

The dataset for VE_GP consists of 100 instances, while for GP it consists of 1000 instances because, as for the other problems, we vertically untied the vectorial variables.

6.3.2 Parameters and statistical test

The experimental parameters used in all the problems are provided in Tables 6.7 and 6.8. They were essentially the same for both GP and VE_GP to facilitate the comparison between the techniques. We should remark that the choice of new terminal functions between cumulative or standard version depends on the chosen recording time of the time series involved. Fitness is calculated as the Root Mean Square Error (RMSE) between the output and the target. Since the output of trees built by VE_GP is supposed to be a vector, for this latter algorithm we calculated the RMSE vertically disbanding both output and target; in this way the measures of fitness are ensured to be comparable between the two techniques. Moreover, when a VE_GP tree wrongly produces scalars as an output, each scalar is replicated until the length of the corresponding target to make it a vector. We decided to penalize these trees by multiplying their fitness for a huge constant (100).

We tested both techniques on a total of 50 runs, each of which considers a different training and test data partition; from now on the term test set stands for unseen data. In particular, at the beginning of a

VE_GP run, 70% of the instances are randomly selected as the training set, while the remaining ones form the test set. We kept the same division for GP with correspondence of the observations, so that time series are not split. We performed a set of tests to analyse the statistical significance of the results. At first, the Kolmogorov-Smirnov test showed that final fitness data are not normally distributed and hence we opted for a rank-based statistic. Test decision for the null hypothesis of no difference in performance between GP and VE_GP were calculated with the Wilcoxon Rank Sum Test on the final test fitness. We opted for it because it is a non parametric test and considers the median which is more robust than the mean to outliers. In order to quantify the assuming difference in performance between the two approaches, we even used a Vargha Delaney A-test which is an index of effect size [10].

Table 6.7: Standard GP parameters.

	K5	**B1**	**B2**	**B3**
Runs	50	50	50	50
Population	500	500	500	500
Generations	100	100	100	100
Training-Testing division	70%-30% of vectorial instances	70%-30% of vectorial instances	70%-30% of vectorial instances	70%-30% of vectorial instances
Genetic operators	Crossover, probability 0.9-Mutation, probability 0.1	Crossover, probability 0.9-Mutation, probability 0.1	Crossover, probability 0.9-Mutation, probability 0.1	Crossover, probability 0.9-Mutation, probability 0.1
Initialization	Ramped Half-and-Half [61], max depth 6	Ramped Half-and-Half [61], max depth 6	Ramped Half-and-Half [61], max depth 6	Ramped Half-and-Half [61], max depth 6
Functions set	plus, minus, times, protected div as [61]	plus, minus, times, protected div as [61]	plus, minus, times, protected div as [61]	plus, minus, times, protected div as [61]
Terminals set	Input variables, random numbers	Input variables, random numbers	Input variables, random numbers	Input variables, random numbers
Selection for reproduction	Lexicographic Parsimony Pressure [70], tournament size=10	Lexicographic Parsimony Pressure [70], tournament size=10	Lexicographic Parsimony Pressure [70], tournament size=10	Lexicographic Parsimony Pressure [70], tournament size=10
Elitism	Replication probability 0.1, best individual is kept	Replication probability 0.1, best individual is kept	Replication probability 0.1, best individual is kept	Replication probability 0.1, best individual is kept
Maximum depth	17	17	17	17

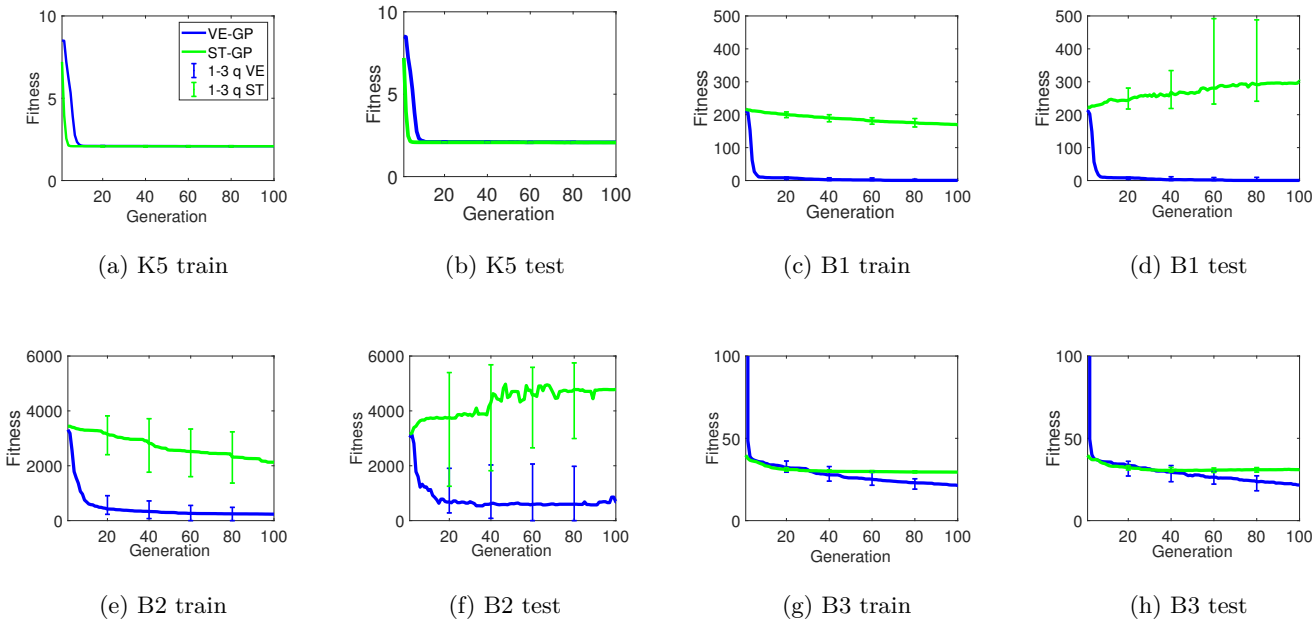
Table 6.8: Vectorial GP parameters.

	K5	**B1**	**B2**	**B3**
Runs	50	50	50	50
Population	500	500	500	500
Generations	100	100	100	100
Training-Testing division	70%-30% of instances	70%-30% of instances	70%-30% of instances	70%-30% of instances
Genetic operators	Crossover, probability 0.9-Mutation, probability 0.1	Crossover, probability 0.9-Mutation, probability 0.1	Crossover, probability 0.9-Mutation, probability 0.1	Crossover, probability 0.5-Mutation, probability 0.1-Mutation of parameters, probability 0.4
Initialization	30% Ramped Half-and-Half [61], 70% Ramped Half-and-Half with forced initialization	30% Ramped Half-and-Half [61], 70% Ramped Half-and-Half with forced initialization	30% Ramped Half-and-Half [61], 70%Ramped Half-and-Half with forced initialization	30% Ramped Half-and-Half [61], 70%Ramped Half-and-Half with forced initialization
Functions set	`VSUMW, V_W, VprW, VdivW, V_mean, V_max, V_min,V_sum`	`VSUMW, V_W, VprW, VdivW, V_mean, V_max, V_min, V_sum VscalprW`	`VSUMW, V_W, VprW, VdivW, V_Cmean, V_Cmax, V_Cmin, V_Csum`	`VSUMW, V_W, VprW, VdivW, V_Cmeanpq, V_Cmaxpq, V_Cminpq, V_Csumpq`
Terminals set	Input variables, random numbers	Input variables, random numbers	Input variables, random numbers	Input variables, random numbers
Selection for reproduction	Lexicographic Parsimony Pressure [70], tournament size=10	Lexicographic Parsimony Pressure [70], tournament size=10	Lexicographic Parsimony Pressure [70], tournament size=10	Lexicographic Parsimony Pressure [70], tournament size=10
Elitism	Replication probability 0.1, best individual is kept	Replication probability 0.1, best individual is kept	Replication probability 0.1, best individual is kept	Replication probability 0.1, best individual is kept
Maximum depth	17	17	17	17

6.4 Results

In this section, we analyse the performance achieved by the two algorithms on the four problems. The evolution fitness plot (Figure 6.2) shows the best fitness in each generation for the training and the test set, median of 50 runs. Besides evolution plots, there are boxplots based on the test fitness at the end of evolution. The statistical test comparing final test fitness between both techniques can be found in Table 6.9. In this table, p is the p-value of the Wilcoxon test with a 5% level of significance. The term A represents the value of the Vargha Delaney A-test. The test returns a number between 0 and 1, representing the probability

that a randomly selected observation from the first sample is bigger than a randomly selected observation from the second sample. In our specific case, the first sample is formed by the best fitness found by GP while the second sample is composed of the best fitness found by VE_GP. It is important to remember that fitness is measured via RMSE, therefore, the lower it is, the better performance it means. Vargha and Delaney in [10] provided a suggested threshold for interpreting the size of the difference: 0.5 means no difference at all, up to 0.56 indicates a small difference, up to 0.64 indicates medium and anything over 0.71 is large. The same intervals apply below 0.5.



(a) K5 train (b) K5 test (c) B1 train (d) B1 test

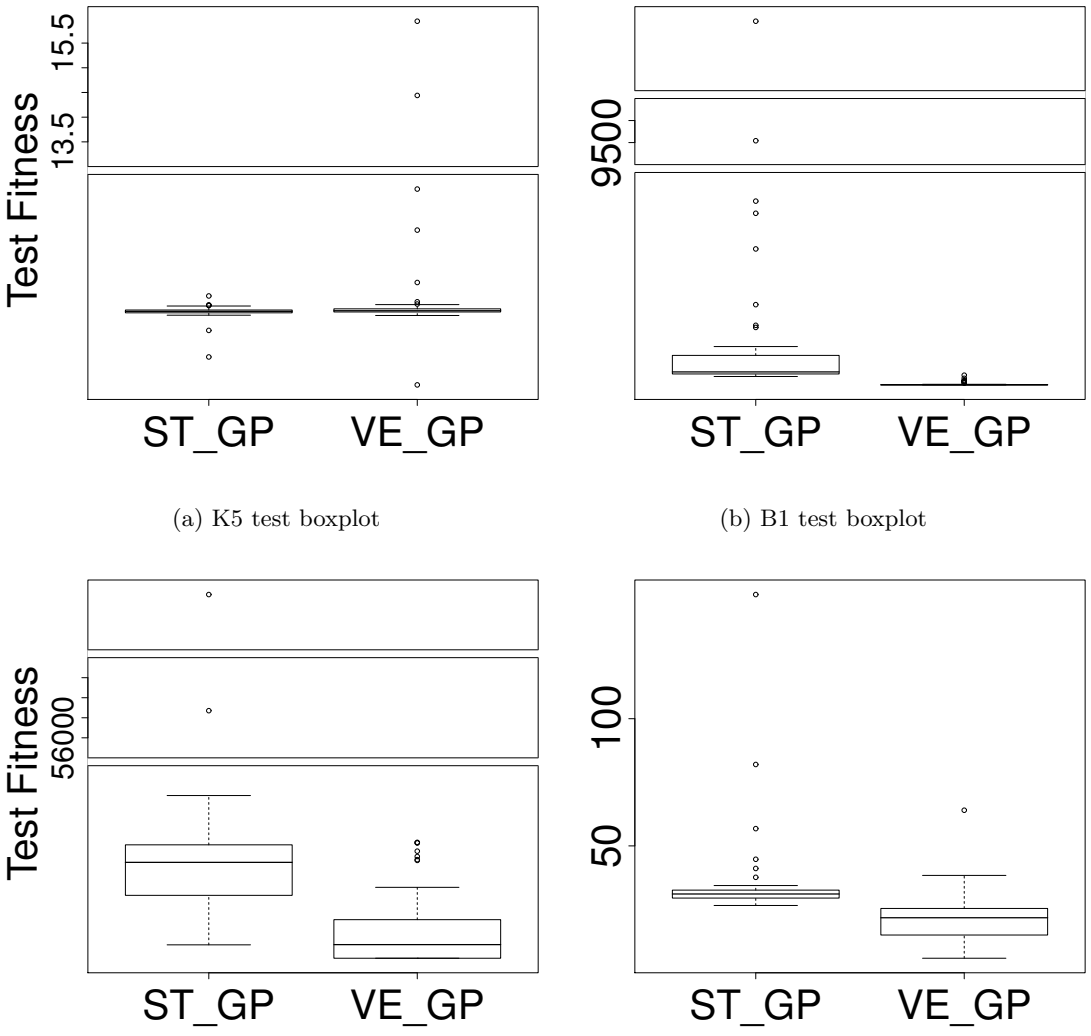
(e) B2 train (f) B2 test (g) B3 train (h) B3 test

Figure 6.2: GP and VE_GP fitness evolution plots. The bars represent the first and the third quartile. K5 through a) and b), B1 through c) and d), B2 through e) and f) and B3 through g) and h) .

Table 6.9: Statistical results of final test fitness comparison.

K5	**B1**	**B2**	**B3**
$p = 0.14$	$p = 6.79 \cdot 10^{-18}$	$p = 1.08 \cdot 10^{-9}$	$p = 1.34 \cdot 10^{-12}$
$A = 0.41$	$A = 1$	$A = 0.85$	$A = 0.91$

Firstly, if we consider the K5 benchmark, there is no significant disparity in performance between the algorithms. This confirms our expectation since the target of the problem does not involve the new functions; the difference between techniques, thus, it is just in data representation. The VE_GP algorithm moreover is not affected by unnecessary improvement of the initialization step and by the extension of the primitive set. Table 6.9 and Figure 6.3 shows differently that VE_GP outperforms GP for B1, B2, and B3. Moreover,



(a) K5 test boxplot

(b) B1 test boxplot

(c) B2 test boxplot

(d) B3 test boxplot

Figure 6.3: GP and VE_GP fitness boxplots for test set.

Figure 6.2 reveals an increasing error for the GP test set on both B1 and B2 problems which means that overfitting is occurring. Therefore GP is not able to understand the underlying relationship between the data. This phenomenon does not happen to VE_GP that increases in fitness during generation for both training and test data. A notable observation that emerges from the B2 evolution plot is the growing amplitude of percentiles. This consideration stresses the fact that every time GP tries to extract the implicit relationship between data it fails, remaining stuck to high error levels. Concerning B3, the difficulty of finding the correct window of time emerges even from VE_GP, where percentiles show the presence of runs not able to overcome GP in 50 generations. Nevertheless, at the end of the evolutionary process, every VE_GP model outperforms the GP ones that stabilize at high error suggesting the idea of no future improvements.

6.5 Next step: VE_GP real application

The technique of VE_GP turned out to be a powerful approach to deal with panel data. The key feature that distinguish VE_GP is the vectorial representation of time series that preserves the true nature of sequential variables. The main consequence of this new representation is the innovative capability of the algorithm to extract the most informative features of a time series variable during the evolution.

In order to characterize suitable problems for VE_GP we chose benchmark problems in which the algorithm resulted with better performance. However, the idea of VE_GP approach was suggested by panel datasets that represents many eco-epidemiological problems such as the mosquito abundance prediction. Therefore, to claim that VE_GP reveals advantages and overcome the issues emerged in 5, we are now going to apply it on mosquito abundance problem treated in 5, comparing the new results with the ones of the previous works.

Published Original Research Article Azzali, I., Vanneschi L., Silva, S., Bakurov I., Giacobini M., A vectorial approach to genetic programming. *Genetic Programming. EuroGP 2019. Lecture Notes in Computer Science*, vol 11451. Springer, Cham (2019).

Chapter 7

Vectorial Genetic Programming Stalks Mosquitoes

Vectorial Genetic Programming (VE_GP) was developed to solve the main drawback of Genetic Programming (GP) on mosquito prediction: the mistreating of time series. In Chapter 5 we highlighted the importance given to the *SIN* variable in the GP model, as the only informer about the time flow. Time series predictors were, in fact, expanded in different fitness cases and therefore treated independently. The vectorial representation of VE_GP allow these time series to be kept in one single fitness case, avoiding the use of *SIN* to indicate the time of the season in which a fitness case was collected.

The implemented approach of VE_GP is even provided by aggregating functions, parametric or not, which role is to extract information about the behaviour of a time series. These aggregating functions gives to the evolution the task of inferring the most meaningful characteristics of time series directly from data, instead of using experts prior knowledge. To clarify, we consider the mosquito problem under analysis. The time series predictors *TWEEK, RAIN, NDVI* consists of aggregated values starting from the daily ones: *TWEEK* is an average of daily temperatures registered during a precise week before the collection day, *RAIN* is a cumulative sum of daily rainfall again registered during a precise week and *NDVI* is a 16 day average. Instead of keeping together these already aggregated values, in VE_GP we could consider daily values and evolve aggregations and windows of time. This idea agrees with the choice of eliminating *SIN* from the predictors, as a variable reflecting experts knowledge on mosquito seasonality.

In this Chapter therefore we will apply VE_GP on mosquito prediction modyfing the previous dataset in the following ways. Firstly, we exclude *SIN* from the predictors, secondly we group togheter daily values of the time series predictors, determining as the new fitness cases the observations from a trap during a year. The main goal is to treat properly time series, relying on evolution as the process able to detect meaningful behaviours and windows of time of the time series. To give a threshold to VE_GP results and

to highlight the advantages of this innovative approach we compare its performance against the ones of the techniques involved in 5 deprived of the *SIN* variable. Moreover, we insert in the comparison the long short-term memory (LSTM) network, which is one of the current state of art regarding machine learning for time series prediction. This technique was suitable for application on the problem, when we realized there was a deficit in dealing with time series. However, VE_GP overcomes two limitations of LSTM network. First, LSTM network can not deal with time series of different length between the predictors and the target, while VE_GP is totally able to learn from daily values and measure the quality of the learning trough comparison on specific days. Second, LSTM network does not return a readable model which is instead an added value of VE_GP, fundamental for interpretation in the field of eco-epidemiological problems.

7.1 The new datasets

To perform the experiments on VE_GP we start by considering the same scalar predictive variables selected in Chapter 5. They are *ELEV, DISTU, DISTR, DISTW, RICEA*. Regarding the time series predictors considered in Chapter 5, we introduce the novelties mentioned before. First, we discard the variable *SIN*. Second we remove the prior aggregations of the time series predictors, therefore, VE_GP handles daily values of land surface temperatures, normalized difference vegetation index, and rainfalls which are the environmental time series predictors. Since mosquitoes are not active during the whole year we consider the daily values of the variables from the 1^{st} of April until the 20^{th} of September. We keep the same name for the daily time series variable, therefore all the predictors are *ELEV, DISTU, DISTR, DISTW, RICEA, TWEEK, NDVI, RAIN*.

Regarding the classical techniques of Random forest (RF), Extreme Gradient Boosting (XGBoost), Multilayer Perceptron (MLP) and standard GP, we involve the same dataset of Chapter 5 simply deprived of the variable *SIN*. We do not consider the generalized linear mixed model (GLMM), because removing *SIN* would mean repeating the whole procedure to find out the most meaningful predictors among all the ones proposed in [11] minus *SIN* which may cause the identification of variables never considered in our investigation.

Concerning LSTM network we start from the dataset used by the classical techniques and we group together the yearly values of collections corresponding to the same trap.

To sum up the three dataset structures we have:

- VE_GP: the dataset is a matrix of 180 rows and 9 columns. Columns one to eight indicate a predictor while the rightmost is the target; time series variables (*NDVI*, *TWEEK* and *RAIN*) are represented as vectors of length 173 since they contain daily values from April 1^{st} (37 days before the first collection of the year) to September 20^{th} (the last collection day of the year). The target *Mosq* is instead a

vector of length 20 representing the 20 collections per year from each trap. Each row corresponds to the collections from a trap during a year.

- RF, XGBoost, MLP, GP: the dataset is a matrix of 3600 rows and 9 columns. Columns one to eight indicate a predictor while the rightmost is the target; each row corresponds to a day of collection.
- LSTM: the dataset is a matrix of 180 rows and 9 columns. Columns one to eight indicate a predictor while the rightmost is the target; time series variables (*NDVI*, *TWEEK* and *RAIN*) are represented as vectors of length 20 corresponding to the values associated at each collection day in a year (the aggregated values used in an untied form in Chapter 5). The target *Mosq* is instead a vector of length 20 representing the 20 collections per year from each trap. Each row corresponds to the collections from a trap during a year.

7.2 Experiments

We perform 30 runs of all the techniques dividing the dataset into learn and test set. Collections from 2002 to 2005 determine the learning set, while collections of 2006, according to the temporal order of years, form the test set. The learning set is then divided randomly in 70% fitness cases as the train set and the remaining 30% as the validation set. This division is used by the different techniques to tune some parameters that have to be manually inserted in the implementation. Precisely, the number of trees in RF, the number of iterations in XGBoost, the number of hidden nodes in MLP and LSTM. The goal of this exploration is undoubtely a first evaluation of VE_GP performance involving the comparison with other techniques, thus no optimization of parameters is performed. However, the tuning of very few parameters not provided at default values by the software involved is mandatory to return at least reasonable results. Concerning VE_GP and GP the division into train and validation set is instead used to select the predictive model. At the end of evolution, in fact, we have a population of individuals evolved by learning from the train set. The individual with the best predictive accuracy on the unseen validation test is chosen as the best model.

The parameters used by LSTM and VE_GP are reported in Table7.1. Since VE_GP deals with simultaneous time series between the input and the target (same flowing year) and we would like to discover new aggregations and ranges of the time series, we consider as the functions set $\mathbf{F}$ ={$\texttt{V\_Cmean}_{p,q}$, $\texttt{V\_Cmax}_{p,q}$, $\texttt{V\_Cmin}_{p,q}$, $\texttt{V\_Csum}_{p,q}$}, where each function is the one describe in Chapter 6. To find out the best windows of time parameters are subject to the mutation of parameters, Chapter 6. The tuning of RF, XGBoost, MLP returned the same parameters as Chapter 5, thus for these techniques and GP we refer to 5.1.

The performance of each technique is evaluated by means of Root Mean Squared Error(RMSE) between the predicted abundance and the real collections of 2006. VE_GP predicts daily values of abundances, thus

Table 7.1: Parameters used to set LSTM and VE_GP.

LSTM Parameters	
learning algorithm	Adam
hidden neurons LSTM layer	200
epochs	50
batch size	1
VE_GP Parameters	
population size	500
max number of generations	100
initialization	Ramped Half and Half with rules [61]
selection method	Lexicographic parsimony pressure [70]
elitism	Best individual kept
crossover rate	0.5
mutation rate	0.1
mutation param rate	0.4
max tree depth	17

we include in RMSE calculation only the abundances corresponding to the collection days. This problem belongs to minimization problems, thus the lower RMSE, the better performance.

The comparison is primarily conducted between VE_GP, LSTM and RF, XGBoost, MLP, GP without *SIN*. However we even examine VE_GP and LSTM in contrast with RF, XGBoost, MLP, GP with *SIN* (the results of Chapter 5), in order to reveal potential better performance simply related to time series well representation. Be aware that to reduce the computational times, increased by the vectorial representation, we perform only 30 runs. From previous GP results therefore, we extract randomly 30 runs.

Statistical significance of the null hypothesis of no difference in performance between all the techniques is determined with Kruskal-Wallis non parametric ANOVA at $\alpha = 0.05$. In case of statistical difference in performance we use a Wilcoxon Rank Sum test to detect difference in performance between VE_GP and the other techniques at $\alpha = 0.05/6 = 0.008$ after Bonferroni correction.

7.3 Results

Table 7.2 reports the p-values of the statistical tests comparing the test results of all the techniques and the test performance of VE_GP against the other methods. The techniques of RF, XGBoost, MLP, GP, in

these first results, use *SIN* as a predictive variable. To have a visual feeling of which method is the best in case of difference in performance, we show in Figure 7.1 the boxplot of both learn and test errors.

Table 7.2: Statistical significance of the difference in performances between the methods. RF, XGBoost, MLP, GP consider *SIN*.

Kruskal-Wallis test $p = 6.9 \cdot 10^{-7}$				
VE-GP vs RF	VE-GP vs XGBoost	VE-GP vs MLP	VE-GP vs GP	VE-GP vs LSTM
$p = 3.7 \cdot 10^{-9}$	$p = 0.5$	$p = 1.3 \cdot 10^{-8}$	$p = 7.7 \cdot 10^{-6}$	$p = 0.00028$

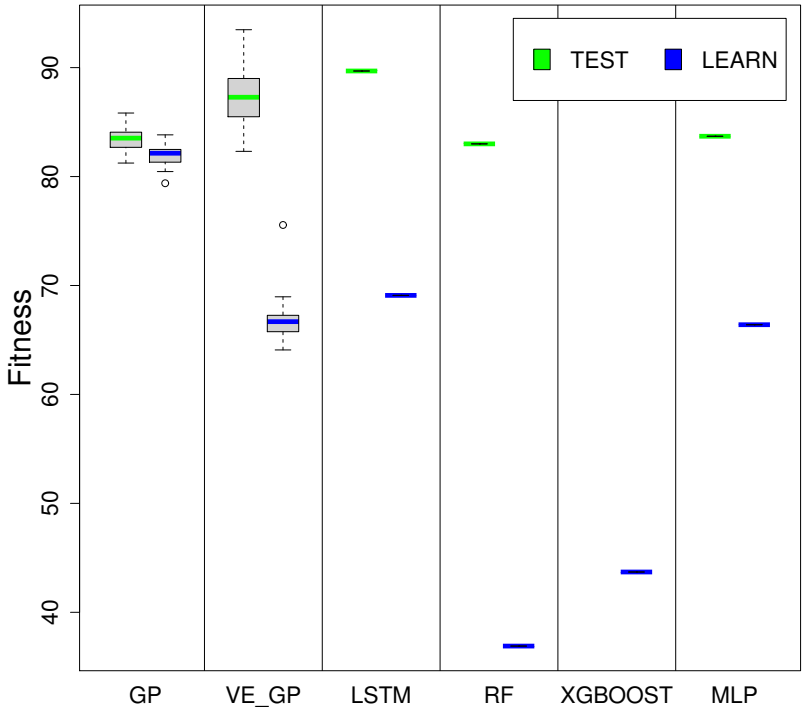


Figure 7.1: RMSE boxplots. RF, XGBoost, MLP, GP consider *SIN*.

The variable *SIN* is a very influencing predictor and, despite the use of vectors and evolving aggregations, VE_GP is not able to outperform all the classical techniques on 2006 collections.

Considering instead RF, XGBoost, MLP and GP without *SIN*, Table 7.3 and Figure 7.2 report the statistical test results on the test set and the test and learn boxplots respectively.

In this scenario VE_GP is the outperforming technique in terms of accuracy. In particular it has better performance even respect to LSTM which is one of the current state of art ML technique to deal with time series. The capability of VE_GP to extract meaningful windows of time is therefore a great added value of this approach. LSTM in fact, deals only with time series of the same length, thus with correspondence of

Table 7.3: Statistical significance of the difference in performances between the methods. RF, XGBoost, MLP, GP consider *SIN*.

Kruskal-Wallis test $p = 1.3 \cdot 10^{-6}$				
VE-GP vs RF	VE-GP vs XGBoost	VE-GP vs MLP	VE-GP vs GP	VE-GP vs LSTM
$p = 5.7 \cdot 10^{-7}$	$p = 1.9 \cdot 10^{-9}$	$p = 1.8 \cdot 10^{-9}$	$p = 1.6 \cdot 10^{-9}$	$p = 0.00028$

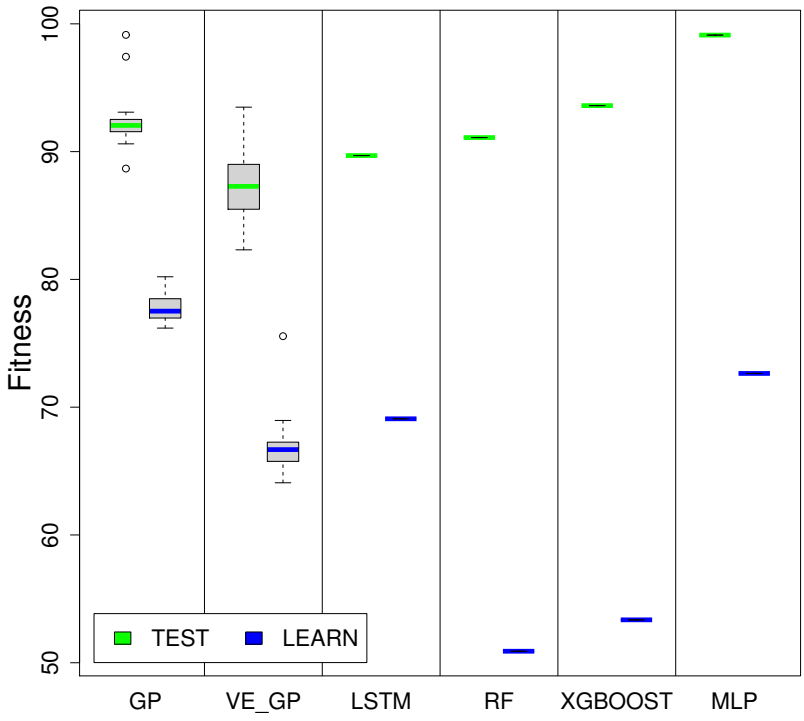


Figure 7.2: RMSE boxplots. RF, XGBoost, MLP, GP do not consider *SIN*.

time instants.

Having established the potential of VE_GP we can now exploit one of its important feature: the readability of the model. Table 7.4 reports the median appearance of each predictive variables in the best models returned. As justified in Chapter 5, these values represent a subjective measure of the importance of each variables in target approximation.

Comparing these frequencies with those of 5.4, where 2 out of 3 time series predictors were never used (*NDVI* and *RAIN*), we can highlight how the absence of *SIN* leads the evolution towards the understanding of the environmental dynamics responsible of mosquito abundances. The variables *TWEEK* and *RAIN*, in fact, are the second and third most used variables in VE_GP models. Nonetheless, a static feature such as *DISTU* seems to be fundamental to reveal mosquito abundance.

The supporting tool we would provide to inform the monitoring plan on mosquito is the best predictive

Table 7.4: Median frequency of each variable in the best models.

Variable	Frequency
TWEEK	6
RAIN	5
NDVI	3
DISTU	4
DISTR	5
DISTW	10
RICEA	1
ELEV	0

model. We therefore include the formula of this model, selected as the best performing one on the validation set over the 30 runs, and we try to give a simple interpretation to it, extracting the information we can provide to vector control officies. We remind that, the despite the complexity of GP models, they are always readable and the main effect of variables on the target could be captured for interpretation purposes. Equation (7.1) represents the expression of the best VE_GP model. The symbols used are the ones classically associated with the primitive functions reported in Section 7.2; when no symbol is found, a multiplication is occurring.

$$
\begin{aligned}
\#\text{Mosquitoes} = & \texttt{V\_Cmean}_{71,29}\Big[RAIN - DISTR \cdot DISTU \cdot RAIN + \texttt{V\_Cmax}_{46,31}\Big(\\
& \texttt{V\_Cmax}_{46,35}(TWEEK - 3DISTR + 2RAIN) - TWEEK + DISTR \cdot \\
& DISTU^2 - 3DISTR - \texttt{V\_Cmean}_{66,35}\Big(2TWEEK - DISTR \cdot DISTU \cdot \\
& RAIN - 3RAIN + \texttt{V\_Cmax}_{46,28}(2RAIN - DISTR) + \texttt{V\_Cmax}_{46,31}\Big(\\
& 5RAIN - 2DISTR - \texttt{V\_Cmax}_{25,2}(DISTR \cdot RAIN \cdot \texttt{V\_Cmin}_{158,14}(RAIN)) \\
& \Big)\Big) - \texttt{V\_Cmax}_{25,2}(DISTR \cdot DISTU \cdot RAIN) + \frac{0.84RAIN}{NDVI + DISTR} + RAIN\Big) \\
& \Big] - DISTR
\end{aligned}
\tag{7.1}
$$

Based on 7.1 we can make the following observations:

- *DISTW*, the most frequent variable in median, is absent in this model. This characteristic may be one of the reason of such good performance. The model, in fact, rely more on time series variable.

- *RAIN* is the most frequent variable involved. This predictor, modified by multiplication with other variables, is mostly applied to a maximum or a minimum. Differently from the prior aggregation therefore, VE_GP has detected as meaningful the peaks of rainfall.
- The variable *RAIN*, modified by multiplication with *DISTR* and *DISTU* is always found at the numerator with minus sign. This results suggest that the more it rains far from river and urban center, the less abundant are mosquitoes.

To understand whether the best model (7.1) is able to estimate the seasonality of abundance without *SIN*, in Figure 7.3 we plot the predicted median abundance of 2006 over the traps.

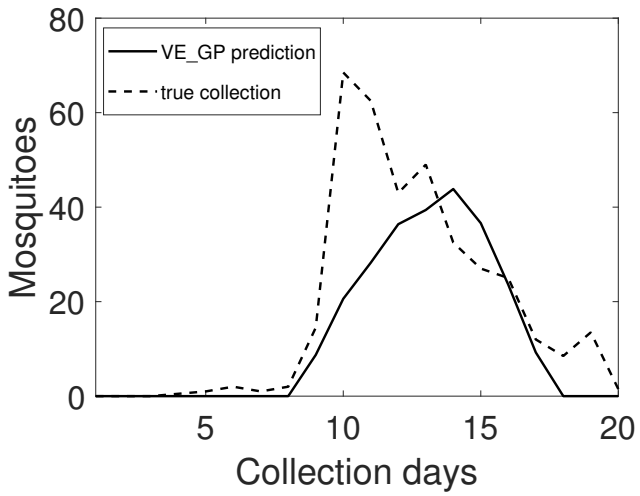


Figure 7.3: Median number of mosquito abundance in 2006. The solid line represents the abundances predicted by VE_GP, the dashed line represents the collected abundances.

VE_GP is able to identify the period of mosquitoes highest activity, which was attributed by experts to mid-summer, therefore from the collection number 10 (first collection of July) to collection number 18 (last collection of August). The seasonality is thus reconstructed, but VE_GP model does not reach the real abundance of the peak. The consequences of this under estimation are highlighted in the vector maps. The vector maps show the potential spatial distribution of mosquito abundances, dividing the abundance values into categories. These maps are fundamental to design vector control interventions during years in which virus transmitted by vectors are known to circulate. Figure 7.4 reports the vector maps of mosquitoes during 2006, considering both VE_GP predictions and real collections. We plot 3 different maps according to 3 different periods in which mosquitoes are active: spring (collection 1 to collection 7), early summer (collection 8 to collection 15) and late summer (collection 16 to collection 20). For each period we normalize the median abundance of each trap with respected to the real median abundance of mosquitoes during the warm season (all collections). The normalized values are then placed into 4 categories: 1) “None to low” for normalized

abundances <2% the value of the warm season median; 2) “Low to moderate” (2-25%); 3) “Moderate to high” (25-75%); and 4) “High” (>75%). These thresholds are chosen to provide a representative of the lower quartile, the interquartile range, and the upper quartile with respect to warm season values. The “None to low” cut-point of 2% rules out extremely low abundance values that are unlikely to be sufficient to generate substantial risk for humans.

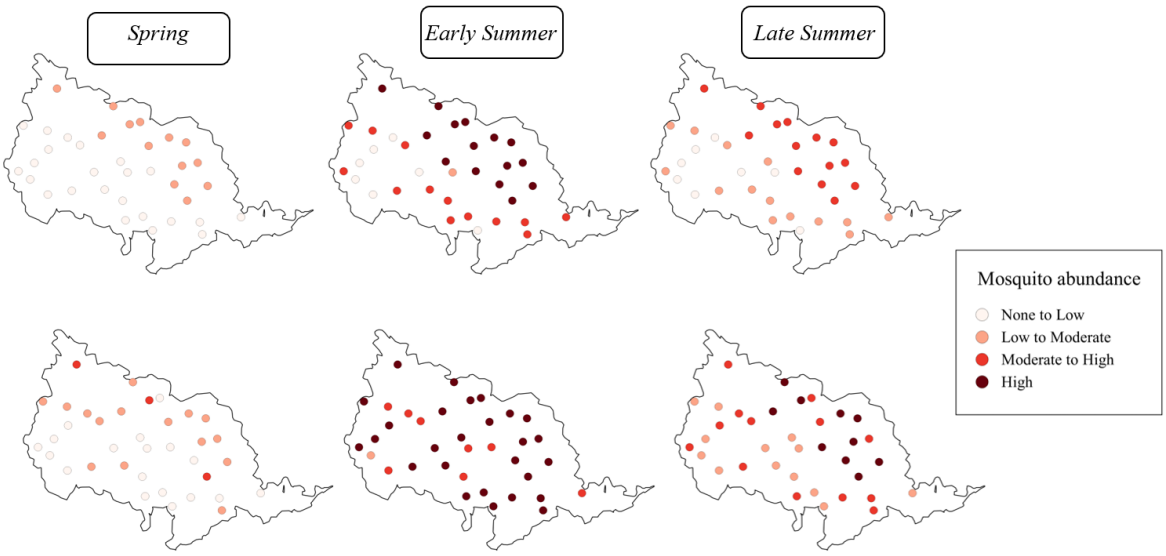


Figure 7.4: Vector map of mosquito abundances during 2006. The upper maps report VE_GP forecasted abundances, the lower maps report the real abundances.

The seasonality recognized by VE_GP is evident even in these maps: spring has the lowest abundances that increase and peak in early summer to decrease, not at spring level, in late summer. Moreover the model well marks the upper part of the studied area as the one with the highest abundances. Despite these good properties, VE_GP model under estimates mosquito levels in early summer, and in late summer does not identify high abundances. Since vector maps are used to guide virus control strategies, this limitation of the model worsen the surveillance and intervention activities.

7.4 Conclusion and next steps

Understanding the relationship between mosquitoes and their environment can provide valuable information to identify West Nile virus, and other vector-borne pathogens, introduction and spread. The previous works on the topic included among environmental variables the artificial indicator of mosquito seasonality, *SIN*. The role of this predictor was fundamental as revealed by results of Chapter 5. The variable *SIN* in fact, provides the time dimension used by the approaches not specific for time series. The use of *SIN*, however, hides the complex environmental dynamics responsible of the abundances, which can instead be used to

understand how to act to control mosquitoes. This requirement lead us to the development of vectorial genetic programming (VE_GP) that by means of vectorial terminals and aggregating functions keep time series in their natural structure and extract from them interesting and unusual features. The time dimension is therefore maintained in the vectorial representation, thus the specific problem of mosquito in Piedmont region could be treated without the use of *SIN*. The aim of this Chapter was therefore the application of VE_GP on mosquito problem, avoiding of course *SIN* and using daily values of the predictor time series instead of previously aggregated ones. This latter characteristic makes the evolution discover from data which aggregation and windows of time are meaningful and informative about the target.

The use of VE_GP on the problem at hand has revealed great advantages. With respect to the other techniques deprived of *SIN*, VE_GP outperformed all other counterpart machine learning algorithms on predicting the mosquito abundance in the year 2006. Being moreover able to see and partially interpret the best predictive model has been a great added value of VE_GP. We remind, in fact, the we want to provide a useful tool to mosquito management. This model reveals the use of interesting and different aggregations of predictor time series to approximate the target. Considering therefore daily values has given the possibility to discover these aggregations directly from data, without the need of experts knowledge.

There is however a drawback of VE_GP that should be overcome. The predictive model of VE_GP is able to catch the seasonality of abundances, which is positive since we cut off *SIN*. However the model is not able to reach the peak value, thus under estimating the abundances. This fact compromise the information the model provides, since it does not well recognize the areas of the region at high level of mosquitoes and therefore at high risk for WNV dynamics. Possible reasons underlying this issue are the lack of an environmental predictor or the lack of an innovative aggregating functions. More investigation on how to solve the problem should be carried out in order to return an even more performing model.

Besides the peak problem that has not yet deserved a proper attention, we move on with further works on VE_GP. Since mosquito problem is the only real scenario in which we applied VE_GP, we involve another real available dataset to validate even more the method. Furthermore, we investigate which techniques enhance VE_GP performance and which instead should not be included to fully exploit VE_GP characteristics. The following Chapters are dedicated to these mentioned further analysis.

Chapter 8

Further Works on Vectorial Genetic Programming: Prediction of Physiological Time Series

8.1 Introduction to the problem

Vectorial Genetic Programming (VE_GP) has revealed great advantages in mosquito prediction, thus it needs to be tested on other real scenarios involving time series. The health-care domain is the right place where to look for other possible VE_GP applications. The prediction of physiological time series, in fact, is taking the place of classical monitoring. The reasons behind this preferred approach are multiples. The acquisition of some physiological data can, in fact, be very physically demanding [18] and in field monitoring may be difficult where there is a lack of portable instruments to directly register data [69]. Several techniques can be used to perform these predictions, from classical linear regression to machine learning (ML) algorithms [79]. All these techniques forecast time series by means of a training phase in which they learn the relationships among predictors and target from previously collected data. Since in the health-care domain data are complex and heterogeneous, machine learning may provide more efficient methods for the purpose rather than traditional regression method that do not catch complex relationship between different variables.

However, an important issue is that none of these basic methods takes into account the temporal component. To clarify, the problem that these techniques have to tackle is the prediction of time series based on other time series, therefore some inputs and the output are sequences of values collected/predicted at regular intervals. To present admissible dataset to the algorithms these sequences are usually split into different observations implying the loss of information regarding the history of the series. The problems described so far are therefore suitable for VE_GP application.

The specific problem we tackle through VE_GP is the prediction of ventilation flows of running people based on heart rate and other physiological variables, in order to monitor the inhaled load of pollutants. As previously done in Chapter 7, we analyse VE_GP performance against six other different techniques. We consider therefore Linear Regression (LR), Random Forest (RF), Multilayer Perceptron (MLP) and Genetic Programming (GP) that use a dataset where time series are split in different fitness cases. Moreover, we investigate Long Short Term Memory network (LSTM) that is frequently used to process time series.

The aim of the comparison is a first evaluation of the potential of VE_GP in the field of time series regression, therefore we do not strongly tune the techniques involved, but we rather consider general reasonable values for the parameters. Once selected through this preliminary analysis the preferred technique to tackle the problem, we should optimize its parameter set to improve the accuracy of the forecast.

8.2 The dataset

The problem was proposed by the Centre of Preventive Medicine and Sport - SUISM - University Structure of Hygiene and Sport Sciences, Centre of excellence of the University of Turin. The goal is to predict ventilation flow during outdoor activities based on physiological variables in order to monitor the intake of air pollution.

To train and test the algorithms for predictions on real time collected values, data were recorded during an indoor trial conducted by the centre. The relationship between variables and respiratory rate, in fact, does not change in function of the conditions in which physical exercises are performed [58].

A group of 262 volunteers underwent an aerobic exercise on a trade mill in order to measure some cardio-respiratory variables through a portable miniaturized ergospirometer (K4, Cosmed, Italy, [63]). Tests were: basal metabolic rate tests with a constant heart rate, threshold tests (in which the heart rate increases up to the maximum of the lactic acid threshold, then suddenly decreases), and altitude tests (in which the air pressure is simulated in a definite altitude level). The test started with a speed of 5 km/h that was incremented of 1 km/h every minute. The participant could suspend the test when he/she felt exhausted. Heart rate was recorded every 10 seconds as well as ventilation, which is needed to evaluate models ability in prediction.

The methods LR, RF, MLP and GP work on a dataset M of 20496 rows (instances, observations or fitness cases) and 5 columns (features or variables). The features selected as predictors are: the age (AGE), the biological sex (a binary value, equal to 1 for female) (SEX), the body mass index (BMI) and the heart rate (HR). The rightmost column of M contains the collected ventilation values, which are considered as the target of our regression problem. We remark that each row of M contains the heart rate and the ventilation of an individual at a precise time instant. The dataset is drawn differently for RNN and VE-GP according to the sequential representation that they admit. Starting from M, we group together in vectors the values

of the same time series so that the new dataset consists of 262 rows and 5 columns. Each row contains the age, the biological sex, the BMI, the heart rate series and the ventilation series of a person.

8.3 Methodologies

We now report all the algorithms enrolled to face the problem, focusing the attention on the parameters we manually set. A deep explanation of how each technique works could be found in Chapter 2.

8.3.1 Genetic programming

In GP we have firstly to declare the primitive set. We used as the functions set set $F = \{+, -, *, /\}$, where / is the division operator protected as in [61], while as the terminal set the the four variables of our dataset (sex, age, BMI and heart rate), plus a random constant. To evaluate the trees we first perform a linear scaling of the predicted output. Linear scaling is a technique introduce in [50] to improve the performance of GP in difficult symbolic regression problems. Instead of applying the fitness measure directly on the predicted output y, we perform a linear regression of the target t on y to find out a and b such that the sum of squared errors between t and $a + by$ is minimized. Thus, we calculate the fitness as the root mean squared error (RMSE) between predicted and scaled output and the real target. In this way GP evolves trees so that the shape of their expression is more similar to the shape of the target function. This prevents GP from spending too many generations in finding the range of the output before adapting the shape. A sum up of GP parameters, kept mainly at the default values proposed by the system, is reported in Table 8.1.

8.3.2 Random forest

The implementation used to run RF requires the number of trees in the forest. Following the main conclusion of [53], we fix the number of trees in the forest equal to 100. The other parameters are kept at the values offered by the system and reported in Table 8.1.

8.3.3 Multilayer perceptron

Concerning MLP, we only set the number of hidden nodes to 3, following the rule of thumb that suggests to choose a number of hidden neurons between 1 and the number of input variables. In order to train and test the network on the same data as the other techniques, we do not consider a validation set to stop the learning phase. All the other parameters are reported in Table 8.1.

8.3.4 Linear Regression

LR does not require any parameters and is not considered at all a ML methods. However we include it in the analysis as the simplest technique to catch relationship and to make predictions.

8.3.5 Long short-term memory network

LSTM network is the only technique receiving the same data representation as VE_GP. However, in case of time series of different length among fitness cases, LSTM network is forced to group in batch observations and pad time series to the longest one of the batch. Since VE_GP is able to deal with time series of different length without completing them, to make fair the comparison we set the batch size of LSTM network equal to 1. In Table 8.1 we summarize the parameters setting for LSTM.

8.3.6 Vectorial genetic programming

VE-GP parameters are set according to default values which are mainly the same as the standard GP and which allows an as fair as possible comparison with the other techniques. The terminal set is composed by the four scalar variables age, sex, BMI and a random constant and by the vectorial heart rate. The functions set is formed by `VSUMW`, `V_W`, `VprW`, `VdivW` and by the cumulative `C_mean` and `C_min` since ventilation and heart rate are collected simultaneously. All the functions are defined in 6. Fitness is calculated applying RMSE on linear scaled outputs as described in 8.3.1. Table 8.1 reports the parameters setting.

Table 8.1: Parameters used to set ML algorithms.

GP Parameters	
population size	500
max number of generations	300
initialization	Ramped Half and Half [61]
selection method	Lexicographic parsimony pressure [70]
elitism	Best individual kept
crossover rate	0.9
mutation rate	0.1
max tree depth	17
RF Parameters	
number of trees	100
MLP Parameters	
learning algorithm	LM backpropagation
hidden neurons	3
μ increase factor	0.1
μ decrease factor	10
initial μ	0.001
epochs	1000
LSTM Parameters	
learning algorithm	Adam
hidden neurons LSTM layer	200
epochs	50
batch size	1
VE_GP Parameters	
population size	500
max number of generations	300
initialization	Ramped Half and Half with rules [61]
selection method	Lexicographic parsimony pressure [70]
elitism	Best individual kept
crossover rate	0.9
mutation rate	0.1
max tree depth	17

8.4 Experiments: Results and Discussion

To estimate how accurately the predictive techniques perform, we adopt a stochastic cross validation approach. Differently from mosquito problem, in fact, we can not ascribe a fixed reasonable group of observations to the test set. We therefore perform 50 runs of each algorithm considering 50 random splits of the dataset into training and test set. The training set is the dataset provided to the algorithm in order to learn, while the test set comprehends unseen data used to test the algorithm in predicting the target. The training set always contains 70% of the participants (183 people) randomly selected, while the test set includes the remaining 30% (79 people).

The measure of comparison selected is the median RMSE between the predicted and the real ventilation values over the 50 test sets. We choose the median rather than the mean because of the presence of stochastic methods such as GP and VE_GP which are more inclined to have outliers. Boxplot and statistics concerning the test errors can be found in Figure 8.1 and Table 8.2. In Figure 8.1 the errors are represented in logarithmic scale.

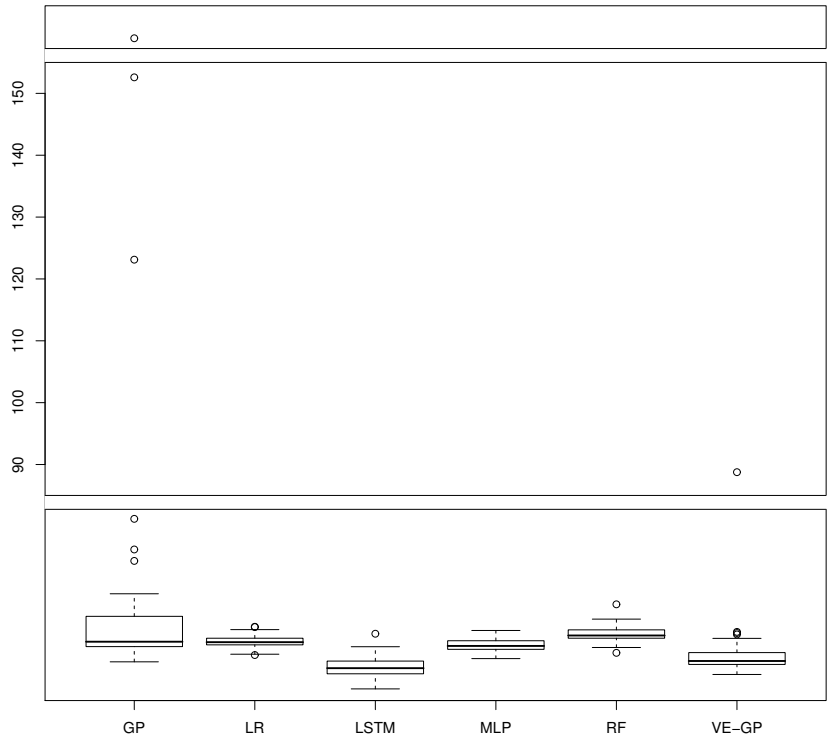


Figure 8.1: RMSE test boxplots

Boxplot of test RMSE in logarithmic scale for the different algorithms.

Table 8.2: Statistics about the RMSE of the different techniques on the test set.

	GP	**LR**	**LSTM**	**MLP**	**RF**	**VE_GP**
mean	129.8	21.5	17.2	20.8	22.5	20.1
standard deviation	715.3	1.0	1.6	1.1	1.3	10.0
median	21.4	21.3	17.1	20.7	22.4	18.2

Table 8.2 reveals that VE_GP outperforms all the other techniques except LSTM. To assess statistical significance of difference in performance between VE-GP and the other techniques, we firstly perform a Kruskal-Wallis non-parametric ANOVA test with a significance level of $\alpha = 0.05$. The resulting p-value reported in Table 8.3 shows the significant difference in median performance between the methods. Moreover, we apply pairwise Wilcoxon tests with $\alpha = 0.05/5 = 0.01$ after Bonferroni correction.

The test p-values are reported in Table 8.3. Statistical analysis indicates that VE_GP consistently outperforms LR, RF, GP and MLP, but it is beaten by LSTM.

From these results we can infer the importance of keeping together ordered sequences such as ventilation and heart rate, in a structure like for instance a vector to improve the accuracy of predictions.

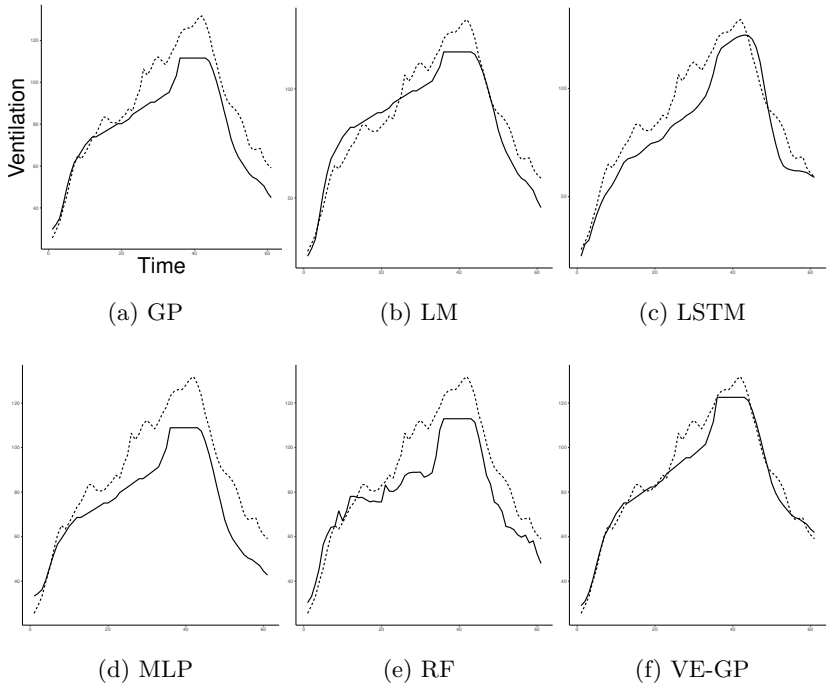
Table 8.3: Statistical significance of the difference in performances between the methods.

Kruskal-Wallis test $p < 10^{-16}$				
VE_GP vs GP	VE_GP vs LR	VE_GP vs LSTM	VE_GP vs MLP	VE_GP vs RF
$p = 1.3 \cdot 10^{-11}$	$p = 9.5 \cdot 10^{-12}$	$p = 5.8 \cdot 10^{-6}$	$p = 2.0 \cdot 10^{-9}$	$p = 7.3 \cdot 10^{-14}$

Besides RMSE, performances evaluation should consider the capability of methods in catching the shape of ventilation flow. For this reason, we selected three different persons with the common feature of being poorly represented in the whole dataset so that we can even show the ability of the methods in generalization. One person has a BMI greater than 25 (16% of the entire dataset), one person is a female (5% of the entire dataset) and the last one is greater than 50 years old (7% of the entire dataset).

Figures 8.2, 8.3, 8.4 show the collected and the predicted ventilations of the three people for all the methods involved on a random run that included them in the test set.

All the methods are good in finding intriguing characteristics of ventilation shape such as peaks and monotony and reveal good capabilities in generalization. Moreover, VE_GP and LSTM shows smooth ventilation series. A possible reason of this behaviour is the fact that they receive the whole heart rate series as input, therefore they have the possibility to remove noise and irregular roughness from it and highlight



(a) GP (b) LM (c) LSTM

(d) MLP (e) RF (f) VE-GP

Figure 8.2: Predicted vs observed ventilation for a person with BMI > 25. The dashed line represents the collected values of ventilation, while the other line represents the predicted values of ventilation by the corresponding method.

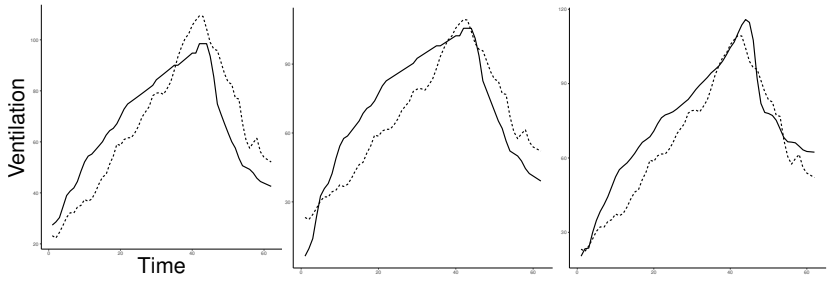
meaningful features.

8.4.1 Analysis of the best solutions

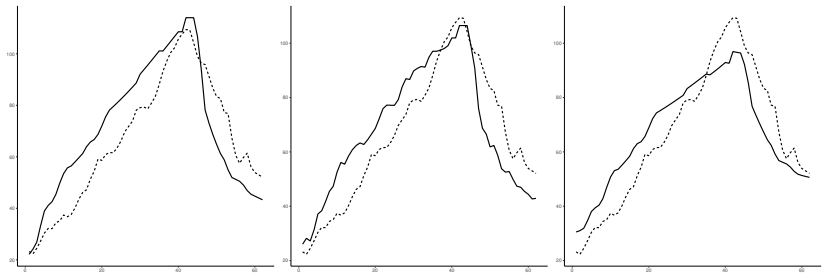
Among all the techniques used in this paper, GP, VE_GP and LR are the most used for producing interpretable solutions. This property let us further investigate how time series treatment influences the performance of a method. Due to the large size and complexity of the best solutions provided by the genetic programming approaches, we base our analysis on the features selection characteristic. The aim is to find out how the time component may change which variables are preferred or ignored more often. Unfortunately, this analysis is not possible for black box methods such as RF, MLP and LSTM.

The standardized coefficients in LR give a measure of the change in the target (in standard deviations) for every standard deviation change in the predictor variables. Since the higher standardized coefficient is the one of heart rate (0.80), we assume that the linear model already view this variable as an impacting one, missing however some information.

Concerning GP and VE_GP we measure the median occurrence of the variables in the 50 best models. Table 8.4 shows the frequencies. The most recurrent feature in GP is BMI while in VE_GP the most frequent one is HR. We deem therefore that GP is not able to give the right importance to the flow of heart rate.



(a) GP (b) LM (c) LSTM



(d) MLP (e) RF (f) VE-GP

Figure 8.3: Predicted vs observed ventilation for a female. The dashed line represents the collected values of ventilation, while the other line represents the predicted values of ventilation by the corresponding method.

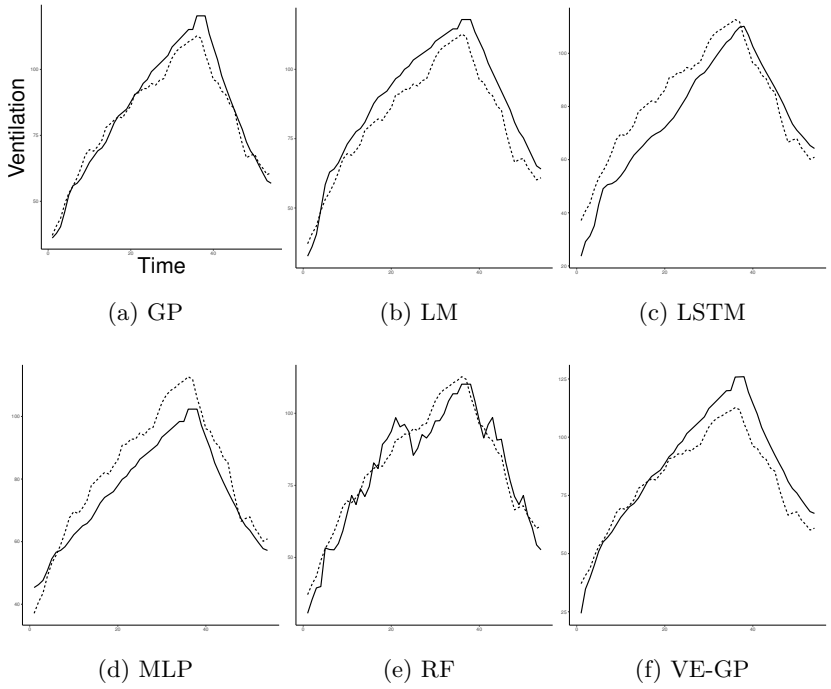
These observations highlight the importance of keeping together ordered sequences which is the key feature of VE_GP.

Table 8.4: Median occurrence of variables in the solutions found by genetic programming approaches.

	HR	**SEX**	**AGE**	**BMI**
GP	49.5	29	47	63
VE_GP	78.5	10.5	11.5	26

8.5 Conclusion

In the health-care domain, prediction of physiological time series relies on machine learning (ML) techniques that automatically discover insightful relationships between variables. However, the sequence of measures of a physiological variable over time is presented to classical ML methods as a group of different fitness cases. This representation may cause a loss of useful information about the behaviour of time series. Thus, prediction of physiological time series may require more advanced ML techniques such as long short-term memory network (LSTM), that consider the time relationship among data. Beside these, the vectorial genetic



(a) GP (b) LM (c) LSTM

(d) MLP (e) RF (f) VE-GP

Figure 8.4: Predicted vs observed ventilation for a person of age > 50. The dashed line represents the collected values of ventilation, while the other line represents the predicted values of ventilation by the corresponding method.

programming (VE_GP) seems suitable for the problem in analysis due to its capability of treating time series as vectors. We have therefore carried out a preliminary comparison of ML techniques that deal differently with the time component on the problem of predicting ventilation flow from other physiological variables including heart rate series.

VE_GP turned out to be a promising technique in the field of machine learning for time series prediction. This approach not only considers time series as vectors, but is able to manage time series of different length and scalar variables without forced padding. Moreover, VE_GP is able to extract meaningful features from the predictor time series (heart rate) that improves the target prediction (ventilation), which is not possible for classical ML methods. VE_GP, in addition, still allows the interpretability of the solution which may provide meaningful information about the problem. Although VE_GP performances are overcame by LSTM, we have to remind that there is no particular tuning of the technique. Since we purposely do not optimize VE_GP implementation, we can expect improvements in performances when VE_GP parameters are properly tuned. The use of VE_GP therefore becomes encouraging even on a real problem, reinforcing its potential and hinting its application on other similar problems of the health-care domain.

Published Original Research Article Azzali I., Vanneschi L., Bakurov I., Silva S., Ivaldi M., Giaconini M., Towards the use of vector based GP to predict physiological time series. *Applied Soft Computing*, Volume 89, (2020).

Part III

Improving Genetic Programming

Chapter 9

An Attempt to Improve Vectorial Genetic Programming Performance: the Inclusion of Geometric Semantic Operators

9.1 Introduction

Vectorial Genetic Programming (VE_GP) has already revealed advantages in benchmark problems, chapter 6, but noteworthy are the results on the real prediction of time series data, Chapters 7, 8. The predictive accuracy and the generalization ability of VE_GP is always ascribed to the key feature of keeping together ordered sequences in vectors. This representation, in fact, lets the evolution discover the most informative aggregating functions to be used in the predictive model, which are responsible of inferring information on the time series behaviour.

Nonetheless, one of the major advantages of VE_GP is claimed to be its ability to evolve the window of time where the new aggregating functions are applied. VE_GP, in fact, adds to all the aggregating functions their parametric version, so that they can be applied only on a portion of the whole vector. The search for the best parameters, the ones that determine the most informative portion of the vector, is part of the evolutionary process, thanks to the introduction of a parameter mutation operator.

In recent years, the use of geometric semantic operators (GSOs) in GP [9] became popular and showed some interesting advantages with respect to GP with classical genetic operators [46, 49, 48, 43]. GSOs, thus, deserve to be explored even in VE_GP approach to see if they still bring advantages in time series forecasting, although they can not include a semantic parameter mutation.

We thus investigate the use of GSOs in VE_GP by presenting a comparative study of GP techniques on two time series forecasting problems. The first one is the well known mosquito problem already implied

in Chaptesr 5, 7. This dataset demands for the inclusion of parametric aggregating functions as primitives, as shown in Chapter 7, offering the possibility to explore the role of the windows evolution for the accuracy of predictions. The second dataset is the one regarding the prediction of people ventilation, described in Chapter 8. In this case, the fact that the time series among different subjects have different lengths suggests the use of aggregation functions without parameters. Further explanations on the different use of aggregation functions will be provided in Section 9.2. The methods we compare are VE_GP and classical GP, both using classical and semantic genetic operators.

9.1.1 Geometric Semantic Operators

Geometric semantic operators (GSOs) are genetic operators recently introduced for GP [9] to replace the traditional syntax-based crossover and mutation. The term *semantic* in GP community indicates the vector of outputs an individual produce on the training instances. Thus, any GP individual can be identified as a point (its semantic) in a multidimensional space (dimension equal to the number of observations) called *semantic space*. While traditional crossover and mutation manipulate individuals just randomly changing their syntax, GSOs define transformation on the syntax of individuals that correspond to the genetic algorithms operators of geometric crossover and ball mutation in the semantic space. Geometric crossover generates an offspring that stand on the segment joining the parents; ball mutation is a weak perturbation of the coordinates of an individual. We report the definition of the GSOs as given in [9] for individuals with real domain and considering Euclidean distance as the fitness function, since these are the operators we are going to use in the experimental phase.

Geometric semantic crossover (GSXO) returns, as the offspring of the parents $T_1, T_2 : \mathbb{R}^n \to \mathbb{R}$, the individual:

$$T_{X0} = (T_1 \cdot T_R) + ((1 - T_R) \cdot T_2)$$

where T_R is a random number in $[0, 1]$. *Geometric semantic mutation* (GSM) transforms the individual $T : \mathbb{R}^n \to \mathbb{R}$ according to the expression:

$$T_M = T + m_s \cdot (T_{R1} - T_{R2})$$

where T_{R1} and T_{R2} are random real individuals with codomain in $[0, 1]$ and m_s is the mutation step. We refer to [9] for a proof of the fact that GSXO corresponds to geometric crossover in the semantic space, while GSM corresponds to ball mutation in the semantic space. The main advantage of these semantic operators is that they induce a unimodal fitness landscape, thus an error surface characterized by the absence of locally suboptimal solution, on every supervised learning problems. This property should enhance GP evolvability on all these problems. The main drawback that afflicts GSOs is that the size of the offsprings is larger than

the one of their parent(s). To overcome this problem we use the implementation of GSOs proposed by [44] and the strategy of elitist replacement suggested in [47].

Unfortunately it is not possible to define the semantic equivalent of parameter mutation as described in Chapter 6. To clarify, let us assume that this operator exists, we call it geometric semantic parameter mutation (GSPM). GSPM has to change one of the parameter of a parametric aggregation function determining, as a result, a weak perturbation of the semantic of T, the individual containing the parametric aggregation function. However, modifying the window in which the aggregation function is applied means considering different values of the time series observed, thus the perturbation of the semantic of T depends on the semantic of T itself. Surely this fact is in contrast with the hypothesis of weak perturbation.

9.2 Experiments

9.2.1 The datasets

Concerning mosquito problem (P_Mosq) the datasets involved are the ones described in Chapter 7. GP therefore considers all the environmental predictors except *SIN* while VE_GP considers the environmental predictors with daily values vectors of time series predictors.

Regarding ventilation problem (P_Physio),instead, the datasets are fully described in Chapter 8.

9.2.2 Experimental Settings

We have extended GP and VE_GP implementations in order to include GSOs. The methods involved in the experiments are therefore GP, VE_GP, GP with GSOs (GSGP) and VE_GP with GSOs (GSVEGP). With each technique we have performed a total of 50 runs on both P_Mosq and P_Physio. Here we lay out the design of the experiments conducted on both problems.

P_Mosq

In each experimental run we have considered the same partition of training and test sets that follows the natural order of years: collections from 2002 to 2005 are used as the training set, while collections of 2006 form the test set.

Fitness is calculated as the Root Mean Square Error (RMSE) between the output and the target. In case of vector based GP (VE_GP and GSVEGP) the output are the predictions of mosquito abundance over 173 days (April 1^{st}-September 20^{th}), thus for the evaluation of fitness we have considered as the actual output the predictions corresponding to the collection days. Since the output of trees built by VE_GP and GSVEGP is supposed to be a vector, for these latter algorithms we have calculated the RMSE vertically

disbanding both output and target; in this way the measures of fitness are ensured to be comparable among all the techniques.

All the runs used population of 100 individuals and the evolution stopped after 50 generations. GP and GSGP initialized populations using the Ramped Half-and-Half (RHH) method [60] with a maximum initial depth equal to 6, while VE_GP and GSVEGP initialized populations using the process proposed in Chapter 6 based on RHH with maximum initial depth again equal to 6. The functions set for GP and GSGP contains the four binary arithmetic operators $+, -, \times$ and / protected as in [60]. The days of mosquitoes collection are the same across years and traps, thus it is reasonable to look for common informative windows of time among all the observations. For this reason, the functions set for VE_GP contains the binary operators `VSUMW`, `V_W`, `VprW`, `VdivW` plus the parametric cumulative aggregating functions $\texttt{C\_max}_{p,q}$, $\texttt{C\_min}_{p,q}$, $\texttt{C\_mean}_{p,q}$, $\texttt{C\_sum}_{p,q}$. GSVEGP can not handle parametric functions, thus its functions set consisted of the binary operators `VSUMW`, `V_W`, `VprW`, `VdivW` plus the cumulative aggregating functions `C_max`, `C_min`, `C_mean`, `C_sum`. All the functions of the vectorial approaches are defined in Chapter 6. The terminal sets contains the 8 variables as described in Chapter 7 plus random constants r between 0 and 1 generated in run time when building individuals. To select parents we use a tournament selection involving 4 individuals. To create new individuals, GP uses standard crossover and subtree mutation [60] with probabilities equal to 0.9 and 0.1 respectively. Besides crossover and mutation, VE_GP uses parameter mutation with probabilities respectively 0.5, 0.1 and 0.4. The semantic algorithms of GSGP and GSVEGP, instead, uses GSXO and GSM with probabilities respectively 0.1, 0.9 and 0.7, 0.3; the mutation steps are respectively 1 and 0.01. The different probabilities and mutation rates depend on a preliminary experimental study performed to find the best parameter setting. Survival of individuals is elitist for GP and VE_GP, while we use the elitist replacement [47] for GSGP and GSVEGP. Maximum tree depth is fixed at 17 for GP and VE_GP while no depth limit is imposed in GSGP and GSVEGP.

P_Physio

Differently from the previous problem, a distinct partition of the training and test sets has been considered in each run. In particular, 70% of the data instances are randomly selected at the beginning of each run as training set, while the remaining 30% were used as the test set.

Fitness is calculated as the RMSE between the output and the target. In case of vector based GPs we follow the procedure described above to guarantee comparable measures.

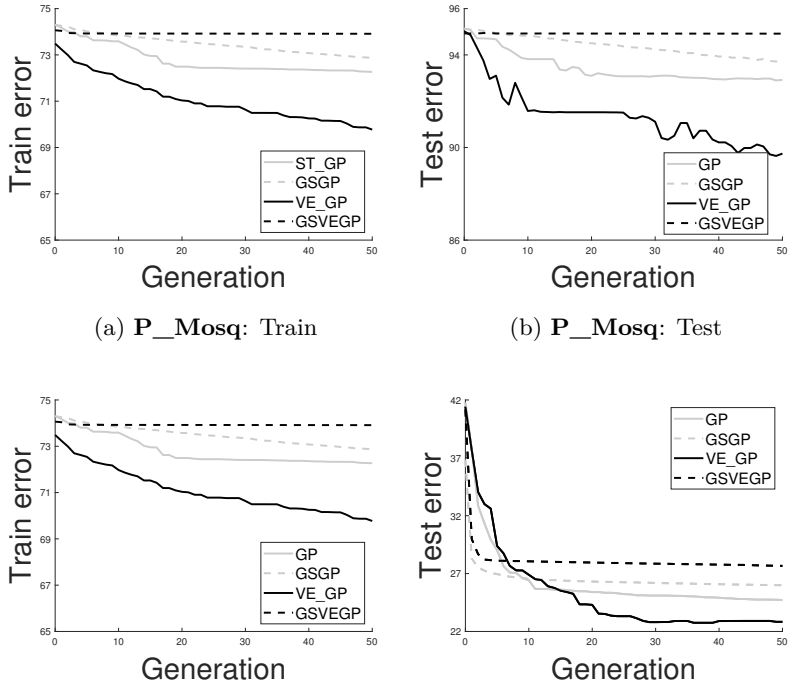
All the runs used population of 100 individuals and the evolution stopped after 50 generations. GP and GSGP initialize populations using the RHH with a maximum initial depth equal to 6, while VE_GP and GSVEGP initialize populations using the process proposed in 6 based on RHH with maximum initial depth

again equal to 6. The functions set for GP and GSGP contain the four binary arithmetic operators $+, -, \times$ and / protected as in [60]. The subjects of the trial ran on the trade mill as long as they could, thus the HR and VE series have different lengths among the people. Looking for a common informative window of time across all the people may weaken the learning phase. In fact, some time windows may be more adequate for long time series compared to shorter ones, causing a loss of generalization ability. For this reason, the functions set for both VE_GP and GSVEGP contains the binary operators `VSUMW`, `V_W`, `VprW`, `VdivW` plus the cumulative aggregating functions `C_min` and `C_mean` as in Chapter 6. All the terminal sets contain the 4 variables as described in Chapter 8 plus random constants r between 0 and 1 generated in runtime when building individuals. To select parents we use a tournament selection involving 4 individuals. To create new individuals, both GP and VE_GP uses standard crossover and subtree mutation [60] with probabilities equal to 0.9 and 0.1 respectively. The semantic algorithms of GSGP and GSVEGP, instead, use GSXO and GSM with probabilities respectively 0.3, 0.7 and 0.5, 0.5; the mutation steps are respectively 1 and 0.1. Also in this case, the different probabilities and mutation rates depends on a preliminary experimental study performed to find the best parameter setting. Survival of individuals is elitist for GP and VE_GP, while we use the elitist replacement [47] for GSGP and GSVEGP. Maximum tree depth is fixed at 17 for GP and VE_GP while no depth limit have been imposed in GSGP and GSVEGP.

9.2.3 Experimental Results

In this section, we report the results obtained in terms of training and test RMSE. In particular, at each generation we store the value of RMSE on the training and on the test set of the best individual in the population, i.e. the one with the smallest RMSE on the training data. The curves report the median over the 50 runs of all these values collected at each generation. The median was preferred over the mean due to its robustness to outliers which are common in stochastic methods. Figure 9.1 reports the training and test errors for P_Mosq and P_Physio.

These plots clearly show that VE_GP in both problems is the fastest in learning, with perspective of further improvement going on with generations, at least for the P_Mosq problem. Moreover, the fast decreasing of the test error confirms that VE_GP is learning with generalization ability. On the contrary, both GSGP and GSVEGP exhibit a slow and almost static (GSVEGP in particular) learning phase. We claim that the main reason behind this fact is the huge size of the semantic space. Considering in fact GSVEGP, in P_Physio problem the semantic space has dimension $(\text{length}(p_1) \times \cdots \times \text{length}(p_{183}))$ where 183 is the number of people in the training set (70% of data instances) and $\text{length}(p_i)$ is the length of the time series recorded for person p_i; in P_Mosq the size is still huge, being $(20)^{144}$ where 20 is the number of mosquitoes collections over a year and 144 is the number of collections in the training set (36 traps$\times$4 year).



(a) **P_Mosq**: Train

(b) **P_Mosq**: Test

(c) **P_Physio**: Train

(d) **P_Physio**: Test

Figure 9.1: GP, GSGP, VE_GP and GSVEGP fitness evolution plots. In grey GP technique, in black VE_GP technique. The dashed lines refer to GSOs, while the solid lines refer to standard genetic operators.

Since the goal of the analysis is to understand how parametric functions influence the performance, we compare the RMSE on the test set of the models found out in the 50 runs by all the techniques. We consider as a model the best individual on the training set at the end of the evolution. Statistical significance of the null hypothesis of no difference among the methods is determined with pairwise Kruskal-Wallis non-parametric ANOVAs at $p = 0.05$. In both problems the resulting p-value states that there is a significant difference in performance among techniques, thus we perform multiple two-sample Wilcoxon signed rank tests to understand which method differs from the other. The significance level for each test depends on the Bonferroni correction. We report the values of the statistical tests in Table 9.1, as well as the boxplots of models test fitness in Figure 9.2.

According to the statistical tests, VE_GP performance differs from all the other methods for both problems. Moreover, boxplots in Figure 9.2 show that VE_GP is outperforming all the other techniques. This outcome confirms that VE_GP is the better GP approach when dealing with panel data, rather than classical GP approach.

Regarding P_Mosq, the results confirm our intuition on the benefit of evolving time windows to discover the most informative ones without prior fixing them just by means of experts knowledge. Surely GSVEGP's

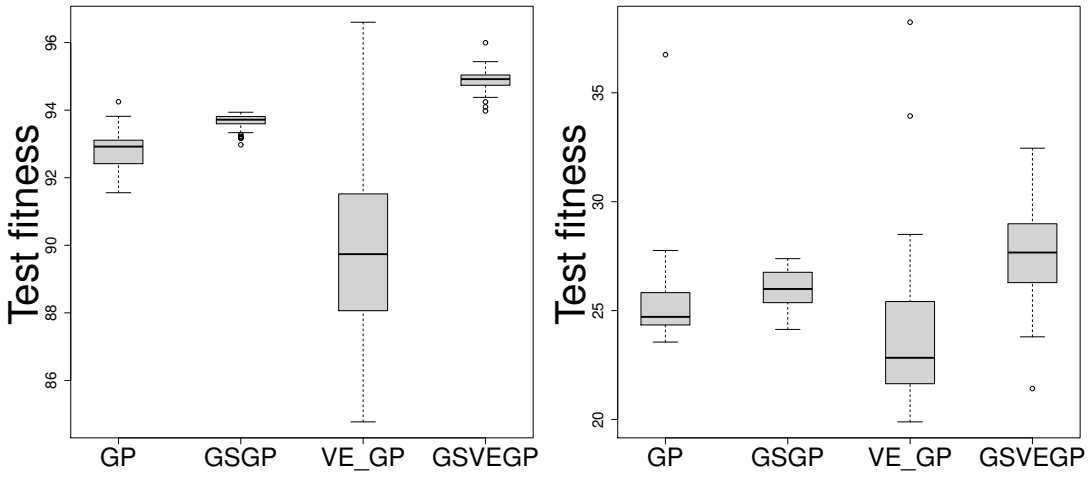
Table 9.1: Results of comparison between techniques on P_Mosq and P_Physio. Significance level of Wilcoxon test after Bonferroni correction $p = 0.05/3 = 0.02$.

P_Mosq: Kruskal-Wallis ANOVA $p < 10^{-16}$				
VE_GP	vs	VE_GP	vs	VE_GP vs GP
GSVEGP		GSGP		
$p < 10^{-16}$		$p = 5.7 \cdot 10^{-16}$		$p = 2.2 \cdot 10^{-14}$
P_Phisio: Kruskal-Wallis ANOVA $p < 10^{-16}$				
VE_GP	vs	VE_GP	vs	VE_GP vs GP
GSVEGP		GSGP		
$p = 2.6 \cdot 10^{-10}$		$p = 2.1 \cdot 10^{-7}$		$p = 1.1 \cdot 10^{-5}$

slow learning is due to the semantic space dimension, but we claim that considering always all the data points of previous collections (classical cumulative functions) rather than an evolving windows over previous times may cause a loss in population diversity and thus be another reason of slow learning. In fact, in VE_GP we find individuals containing different aggregations that span different time series portion, while in GSVEGP we find surely individuals containing different aggregations, but they all span the same time series portion. To confirm this observation we report the median (over the 50 runs) diversity curves along generations for GSVEGP and VE_GP. We use as a subjective measure of diversity the standard deviation of the fitness values in the population at each generation.

Figure 9.3 clearly shows that GSVEGP is unable to keep good diversity levels which is a key feature of a successful search process. We try to give an explanation of other plausible reasons for this GSVEGP diversity drop. GSOs seems to quickly direct the diversified initial population towards the target; however, after the individuals have converged, the improvements are thinner and thinner and the weak perturbation of one of the components of one of the output time series results in a weak perturbation of the individual fitness. At a certain point, thus, GSOs seems to be less efficient due to the high dimension of the semantic space. In addition, the elitist replacement used to control individual growth [47], at that certain point, causes more frequently the replication of individuals instead of the offspring replacements. In fact, if the weak perturbations are not efficient (produce offsprings with bigger fitness) parents are preferred rather then their offsprings. All these behaviours are feasible reasons of GSVEGP loss of diversity.

Concerning P_Physio results, we expected GSVEGP to be to be the best performing method, since parametric functions are not involved in any primitive set. However, statistical tests and the boxplots reveal that VE_GP is the method with the best performance. These results confirm that GSOs are not suitable



(a) P_Mosq (b) P_Physio

Figure 9.2: Test fitness boxplots of models found out by each technique. Figure (a) refers to P_Mosq, while figure (b) refers to P_Physio.

to deal with time series variables, probably because of the high dimension of the semantic space induced.

9.3 Conclusions

This analysis is an investigation on the usefulness of evolving parametric aggregation functions for time series forecasting. Aggregations of values may return informative features of predictors time series for the target, however aggregations on all historical times of predictors may not be needed to forecast the target series. The behaviour of the predictors over a window of time may in fact be more meaningful for the target. To clarify, let us consider the P_Mosq problem of predicting the abundance of mosquitoes during a year: mosquitoes collected at day t are more likely to be affected by the rainfalls over the week before t rather than on all the rainfalls of the days before t; mosquitoes growth in fact, lasts more or less one week, thus rainfalls over a week may cause the loss of eggs and thus adult mosquitoes at day t.

Vectorial genetic programming (VE_GP) includes in the functions set aggregating function depending on parameters to define time window in which to apply the function. The genetic operator of parameter mutation, moreover, gives the possibility to parameters to evolve in order to catch the most informative windows. By this work we therefore want to highlight the benefits of tackling time series forecasting using VE_GP. In particular, we compare VE_GP performance against VE_GP with geometric semantic operators (GSVEGP) on two problems. While the first one, P_Mosq, demands for parametric aggregating functions in the functions set, the second problem, P_Physio, does not require evolving time windows. We choose GSVEGP as a benchmark because although geometric semantic operators should improve the performance,

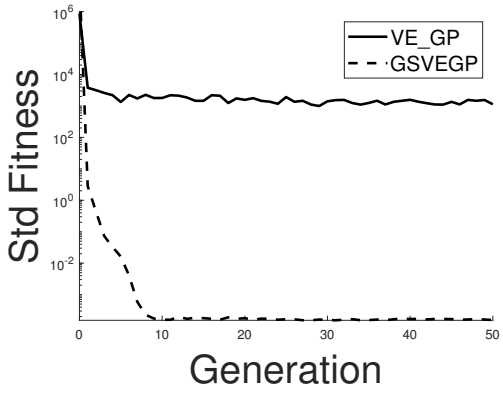


Figure 9.3: Diversity evolution for VE_GP and GSVEGP on P_Mosq. Curves are plotted in logarithmic scale.

the geometric semantic awareness does not allow parameter mutation as a genetic operator.

The main contribution of this analysis consists in showing that parametric aggregating functions can further improve the performance when the dataset hypothesis allow for their inclusion (P_Mosq). Moreover we find out that considering all the history of time series may influence the maintenance of diversity in the evolving population. Surprisingly, however, results achieved on P_Physio reveals a weakness of GP algorithms with geometric semantic operators. We impute this result to the high dimension of the semantic space caused by time series variables that slow the learning process.

The outcomes pave the way for future works on the design of more efficient geometric semantic operators for problems involving time series, able to perform more dynamical evolutions. A wider result is, however, the highlight of VE_GP as a successful approach in time series forecasting, making the GP community aware of its value.

Published Original Research Article Azzali I., Vanneschi L., Giaconini M., Investigating the use of geometric semantic operators in vectorial genetic programming. *Genetic Programming. EuroGP 2020. Lecture Notes in Computer Science*, vol 12101. Springer, Cham (2020).

Chapter 10

A Coevolutionary Approach Towards Assumption Free Genetic Programming

10.1 Introduction

Natural evolution developed species, suitable to the environment in which they live, apparently without any use of prior knowledge. Why therefore not challenging evolutionary algorithms, so that they can also work in a totally autonomous way, without requiring any prior knowledge from humans? Vectorial Genetic Programming (VE_GP) was introduced exactly to avoid as much as possible the use of prior knowledge and improve GP's autonomy in search processes. In case of problems dealing with time series VE_GP keeps these variables in their natural structure, without pre-applying any aggregating function (like its mean or maximum), suggested by expert knowledge, that collapse them into a scalar value. There is still however, a reminder of prior knowledge in VE_GP: to deal with time series, VE_GP needs to be furnished with a predefined set of aggregating functions, that catch informative windows of time and time series variables during evolution. These aggregating functions are typically the classical ones involved in time series analysis (for instance, mean, sum, maximum, minimum, etc.), and they can be embedded in VE_GP as they are known to return meaningful features of the time series. Complex problems, however, may be solved by other aggregating functions, defined as combinations of vector entries, not yet know or defined. Therefore, why not let evolution itself search for new aggregating functions, instead of manually embedding a set of predefined ones?

This question led us to propose and investigate an approach where even the nature of aggregations is explored, Coevolutionary Vectorial Genetic Programming (coevo_VE_GP). In order to avoid the use of a predefined set of known aggregating functions, a population of aggregating functions is coevolved in parallel with the standard VE_GP. The main idea is, therefore, the parallel evolution of two populations, one of

standard VE_GP individuals and one of aggregating functions. These two populations are connected by means of their fitness functions: the better the individuals of VE_GP that use an aggregating function, the higher the aggregating function fitness value.

Coevolution is a well-known concept, but what makes our approach original is its purpose of applying co-evolutionary dynamics to remove human embedded prior knowledge. The reader familiar with VE_GP may think that the introduction of a parametric terminal function that picks vector entries allows the standard VE_GP to achieve the same result as the coevolutionary approach. This terminal function, by means of the parameter evolution, can be used to build any combination of vector entries, thus any aggregating function. Surely this is right, but using coevolution we expect to reduce the computational effort and to speed up the learning process. In this chapter, we define coevo_VE_GP to solve time series problems, assessing its performance through artifical benchmark problems, designed to explore the emergence of different kinds of aggregation functions. Surely we expect that the added complications will cause less performance in some environments, but the overall objective is a preliminary investigation of coevo_VE_GP, in order to state its feasibility and to elucidate the framework in which the technique would be helpful.

10.2 Formulation of coevo_VE_GP

The algorithm of coevo_VE_GP is based on the simultaneous evolution of two populations. One population, called *POP1*, evolves a VE_GP system, whose individuals are feasible solutions to the problem at hand. The other population, called *POP2*, evolves functions to be used by *POP1* to transform vector variables into scalar values. In particular, the individuals of *POP2* are aggregating functions, that is to say they define aggregations of vector entries.

We now explain in detail this coevolutionary process, using an example for clarity. We assume we are tackling a problem described by $\{X_1, X_2, x_3\}$ as the input variables, where X_1 and X_2 are vectorial variables, and x_3 is a scalar variable. For simplicity we consider both X_1 and X_2 vectors of length 3.

10.2.1 Initialization

The first step of coevo_VE_GP is the initialization of *POP2*. The individuals of *POP2* define relationships among vector entries, where the vectors are actually the vectorial variables involved in the problem. Considering the example problem, the individuals are built using the terminals $\{v_1, v_2, v_3\}$ which represent the general 3 entries of the vectorial variables at hand. The function set could include any kind of arithmetical function as in classical GP. However, for the sake of simplicity, we consider the standard $\{+, -, \times, /\}$. The operator / returns 1 when the denominator is equal to 0, and the result of the division otherwise.

The initial population could be built with any initialization technique, but we decided to always include

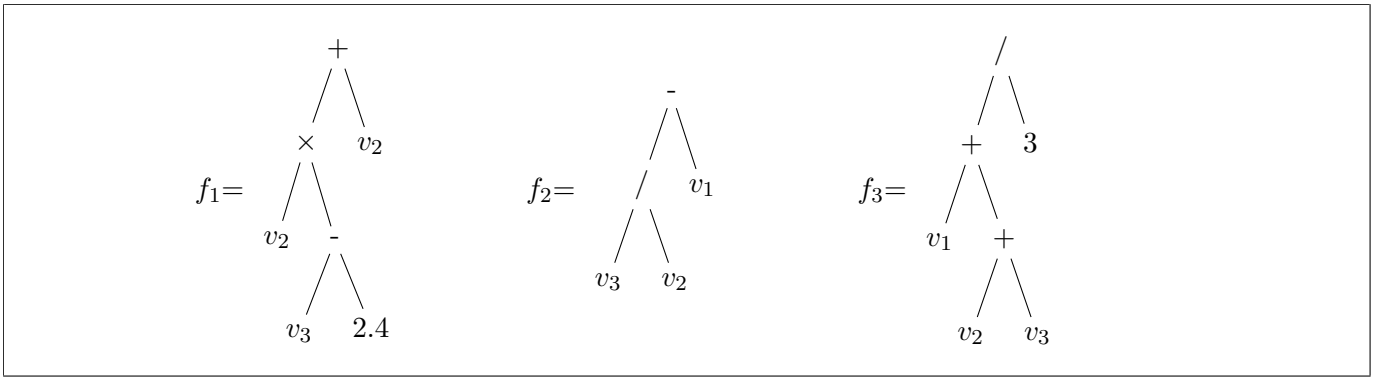


Figure 10.1: Generation 0: *POP2*=$\{f_1, f_2, f_3\}$.

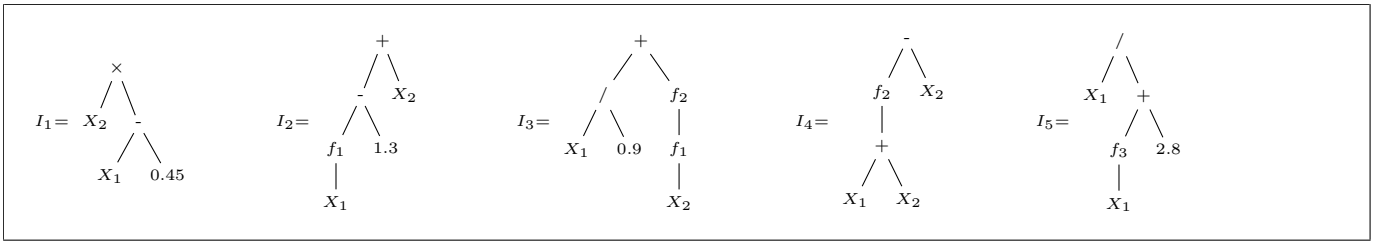


Figure 10.2: Generation 0: *POP1* = $\{I_1, I_2, I_3, I_4, I_5\}$.

some driven individuals. These individuals represent classical aggregations (e.g., mean, sum) which return meaningful values of the vector in consideration. Results achieved in Chapters 7, 8 made us realize the key role of these functions in the predictive model of real-world problems, thus they can potentially be the building blocks of more sophisticated aggregations. These functions should be therefore included as individuals in the initial *POP2* and let the evolution discard or rely on them. Figure 10.1 provides an example of initial *POP2* composed by 3 individuals, including the individual representing the mean.

After *POP2* initialization, we can build the initial *POP1*. *POP1* is a standard population of VE_GP. For example, following our example, the terminal set would be $\{X_1, X_2, x_3\}$. The functions set is the key element of the coevolutionary process. Besides any kind of operations available in VE_GP it contains all the individuals of the current *POP2*, considered as functions on the vectorial input variables. The functions set at the base of *POP1* consists, therefore, of $\{+, -, \times, /, f_1, f_2, f_3\}$. The operators $+, -, \times, /$ are extended to vectors as defined in Chapter 6.

All the VE_GP techniques of initialization can be used to initalize *POP1*. However, following a common rule, every element of *POP2* is used at least once as a terminal function in the initial *POP1*. This guideline guarantees a well defined fitness for every individual of the *POP2* initial population. The previous assertion in clarified in subsection 10.2.2 where we present how fitness is calculated. Figures 10.2 shows an example of *POP1* initial population.

The number of *POP2* initial individuals should not be too high in order not to force too many inclusions of aggregating functions in *POP1*. Furthermore, the size of *POP2* increases during the evolution, as explained in subsection 10.2.4. A reduced dimension is likely to imply a lack of diversity in the population and for this reason we adopt a strategy, explained in subsection 10.2.3, to avoid the problem.

10.2.2 Fitness evaluation

The core of the coevolutionary process is the fitness evaluation in the two populations. Since the problem at hand is solved by *POP1* individuals, the first fitness values to be calculated are the ones of the *POP1* individuals. Nodes corresponding to individuals of *POP2* are always treated as aggregating functions on *POP1* terminals.

Regarding *POP2*, instead, the fitness assigned to each individual depends on how good are the individuals in *POP1* that contain them. Given an individual f in *POP2*, the fitness of f is therefore the mean of the fitness of all the individuals in *POP1* containing f. This definition implies the possibility of individuals in *POP2* without a fitness value.

10.2.3 Genetic operations

The genetic phase starts in *POP2*. Besides classical crossover and mutation [61], we decided to introduce the random generation of individuals, in order to maintain diversity. There are several techniques for maintaining diversity, which typically reduce selection pressure, selection noise or operator disruption [14]. These techniques operate on the existing individuals of the population. Since we start from a small *POP2*, whose size is dynamical (see subsection 10.2.4), the simple introduction of new random individuals is preferable over the other existing techniques to keep diversity and to avoid a premature collapse of *POP2*.

Figure 10.3 shows an example of *POP2* offspring. We use the following short forms: $c_{p,q}(f,g)$ denotes the crossover swapping the subtree rooted in node p in f with the subtree rooted in node q in g; $rg()$ denotes a randomly generated individual. Once the genetic phase is done in *POP2*, *POP1* is able to generate its own offspring. Crossover and mutation are used normally, but we added another genetic operation to exchange individuals with *POP2*. This new genetic operator is called mutation_pop2. Mutation_pop2 looks for individuals of *POP2* in the nodes of a *POP1* individual. After selecting randomly one of these *POP2* individuals, the operator of mutation_pop2 changes it randomly with one of the offspring of *POP2*.

In Figure 10.4, we represent an example of *POP1* offspring. We use the following short form: $c_{p,q}(I,Y)$ denotes the crossover swapping the subtree rooted in node p in I with the subtree rooted in node q in Y; $m_p(I)$ denotes the mutation of the subtree of I rooted in p; $mpop2_p(I)$ denotes the mutation of the *POP2* individual rooted in node p of I; $rep(I)$ denotes the replication of I.

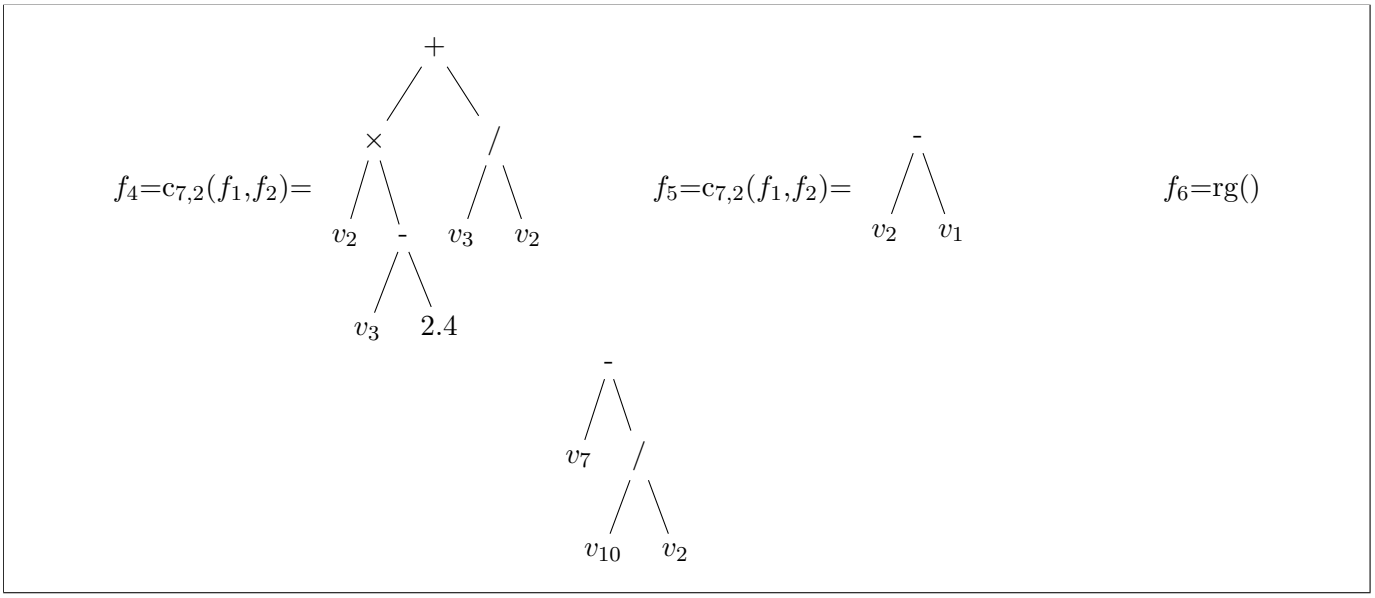


Figure 10.3: *POP2*_offsprings=$\{f_4, f_5, f_6\}$.

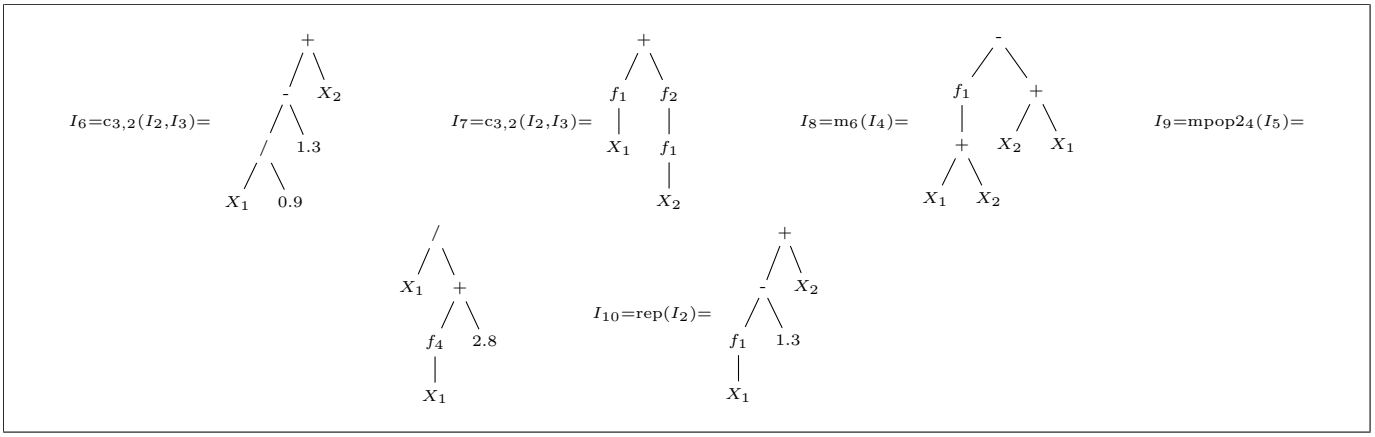


Figure 10.4: *POP1*_offsprings=$\{I_6, I_7, I_8, I_9, I_{10}\}$.

10.2.4 New generation

To create the new generation and continue the coevolution process, we start by calculating the fitness of the individuals in *POP1*. After applying a classical survival operator [61], the new generation of *POP1* is established. In our example: *POP1*=$\{I_2, I_{10}, I_9, I_7, I_8\}$. Having the updated *POP1*, we can calculate the fitness of all the available individuals of *POP2*. It is important to calculate again the fitness of the individuals of the previous generation, since we have a new *POP1*. In case an individual f of *POP2* is not included in any individual of *POP1*, we can not ascribe a fitness to it. In this case, the survival is applied by simply removing from the whole *POP2* (previous *POP2* and offspring) the individuals without a fitness. This process of selection implies a dynamical size of *POP2*. The new generation of *POP2*, in our example, is therefore *POP2*=$\{f_1, f_2, f_4\}$.

The reader must be aware that a possible outcome of coevo_VE_GP is the emptying of *POP2*. In the case, in fact, when the updated *POP1* does not use any aggregating functions, no individual of *POP2* survives. In this situation, the coevolutionary process is stopped since it does not involve anymore two simultaneously evolving populations.

10.2.5 Algorithm

To summarize the description of the coevolutionary technique, we report the pseudo-code of the coevo_VE_GP in Algorithm 1.

Algorithm 1: Pseudo-code of coevo_VE_GP.

```
START;
    GENERATE: POP2={f_1, f_2, f_3}
    GENERATE: POP1={I_1, I_2, I_3, I_4, I_5}
    FITNESS_POP1: F_{I_s} = fitness(I_s) for all s ∈ 1,...,5
    FITNESS_POP2: F_{f_t} = mean(F_{I_r}) : f_t ∈ I_r for all t ∈ 1,2,3
for g=1,...,G do
    SELECTION_POP2
    GENETIC OPERATIONS POP2
    SELECTION_POP1
    GENETIC OPERATIONS POP1
    compute FITNESS_POP1
    POP1_generation(g)=SURVIVAL(POP1+offspring)
    compute FITNESS_POP2
    POP2_generation(g)=SURVIVAL(POP2+offspring)
end
```

10.3 Experiments

We begin the experimental validation of coevo_VE_GP using 4 artificial benchmark problems. These problems are randomly designed in order to fulfil the first goal of the exploration: show that the coevolution is actually working, and that *POP2* is bringing an active contribution to the evolution of *POP1*. Here we describe how these 4 datasets are structured. Note that by X_i^j we denote the j-th entry of vector X_i.

Benchmark #1 (bench1)

- X_1 = vector of length 10 of numbers randomly drawn with uniform probability from $[1, 10]$.
- X_2 = vector of length 10 of numbers randomly drawn with uniform probability from $[50, 100]$.
- x_3 = number randomly drawn with uniform probability from $[-5, 5]$.
- *Target* $= ({X_1}^1 + 2 \cdot {X_1}^2) \cdot x_3 + X_2$.

Benchmark #2 (bench2)

- X_1 = vector of length 10 of numbers randomly drawn with uniform probability from $[1, 100]$.
- *Target* $= \frac{({X_1}^1 + ... + {X_1}^{10}) + {X_1}^1}{X_1}$.

Benchmark #3 (bench3)

- X_1 = vector of length 10 of numbers randomly drawn with uniform probability from $[1, 100]$.
- *Target* $= (\frac{{X_1}^1 + ... + {X_1}^{10}}{10}) + X_1$.

Benchmark #4 (bench4)

- X_1 = vector of length 10 of numbers randomly drawn with uniform probability from $[1, 10]$.
- X_2 = vector of length 10 of numbers randomly drawn with uniform probability from $[1, 100]$.
- X_3 = vector of length 10 of numbers randomly drawn with uniform probability from $[-10, 10]$.
- x_4 = number randomly drawn with uniform probability from $[0, 1]$.
- *Target* $= x_4 \cdot X_1 + \frac{X_2}{{X_2}^1 - \frac{{X_3}^4}{{X_3}^5}}$.

Table 10.1: Parameters setting of coevo_VE_GP used in the benchmark problems.

	POP1	POP2
generations	50	50
initial pop size	100	20
genetic operators	crossover, mutation, mutation_pop2	crossover, mutation, random generation
genetic operators probs	0.5, 0.1, 0.4	0.9, 0.1, rate=0.1
initialization	Ramped Half-and-Half with *POP2* control	Ramped Half-and-Half
functions set	`VSUMW, V_W, VprW, VdivW`	+,-,×,koza division
terminals	input variables + random numbers	10 general inputs + random numbers
fitness	RMSE	average RMSE
parents selection	Lexicographic Parsimony Pressure	Lexicographic Parsimony Pressure
elitism	keep best	keep all indiv with fitness
pop size	fixed	dynamic
max depth	17	17

Each dataset consists of 1000 instances, 70% randomly used as the training set, the remaining 30% used as the test set. We performed 50 exploratory runs of coevo_VE_GP without any parameter tuning, remarking that these experiments are meant just as a first evaluation of the method. Table 10.1 reports the parameter setting used for each problem.

To evaluate the performance of coevo_VE_GP we compared it with standard VE_GP applied on the same problems. The standard technique of VE_GP is, in fact, totally capable of evolving a population towards each benchmark target. Aggregations of vector entries can be built by means of a terminal function that extract from a vector one of its entries. To clarify, let us consider the function f_j that returns from vector V the j-entry (V^j). The aggregating function $F(V) = V^1 + V^3$, for example, can be found in a VE_GP tree as $VSUMW(f_1(V), f_3(V))$.

We expect, however, in VE_GP a less efficient evolution because good aggregations can be easily destroyed by a crossover or a mutation. Differently, coevo_VE_GP keeps good aggregations in nodes.

To perform VE_GP experiments we introduced a parametric terminal function `V_entryp` that returns the p-th entry of a vector; in case the input is a scalar, the output will be the scalar itself, in case p is greater

than the length of the vector, the output will be zero. To properly handle this function, we introduced a new genetic operator, `mutation_p` that randomly mutates the parameter p of a `V_entryp` function. Since the function `V_entryp` is specific for vectors, we applied the forced initialization of standard VE_GP, as described in 6.

We performed the usual 50 runs keeping using the parameters summarized in Table 10.2.

Table 10.2: Parameters setting of the algorithm of VE_GP used in all benchmark problems.

generations	50
initial pop size	100
genetic operators	crossover, mutation, mutation_p
genetic operators probs	0.5, 0.1, 0.4
initialization	Ramped Half-and-Half with rules
functions set	`VSUMW, V_W, VprW, VdivW, V_entryp`
terminals	input variables + random numbers
fitness	RMSE
parents selection	Lexicographic Parsimony Pressure
elitism	keep best
pop size	fixed
max depth	17

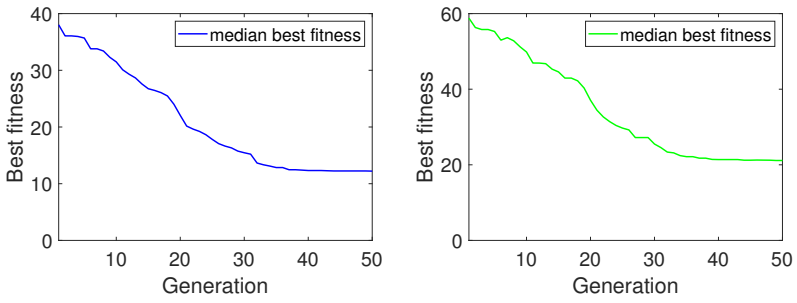
10.4 Results and discussion

This section presents the results achieved by coevo_VE_GP on the benchmark problems. We firstly focus the attention on the validity of the technique as a coevolutionary algorithm. Then we prove the effectiveness of the specific strategies inserted in coevo_VE_GP to finally highlight the advantages and disadvantages of the coevolutionary process against a standard approach.

10.4.1 Is coevo_VE_GP a real learning process?

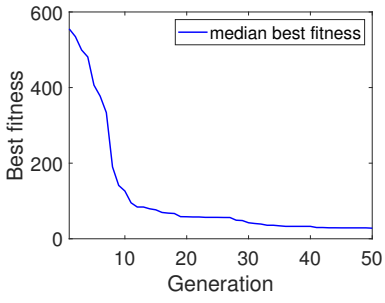
The first results we show are the evolutions of the fitness of the best individuals of *POP1* on both the training and the test set, for each benchmark problem. These results are reported in Figure 10.5. The lines plotted in the figures report the median values calculated over the 50 runs.

These plots highlight the capability of *POP1* to gain knowledge through the coevolutionary process on every benchmark problem. The learning process, moreover, exhibits a good capability of generalization since

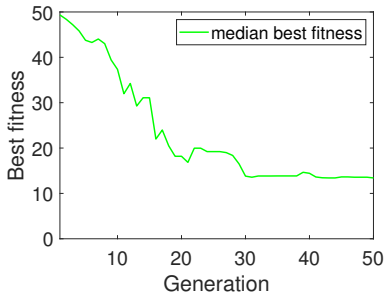


(a) Train bench1

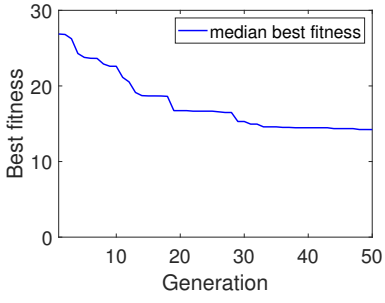
(b) Test bench1



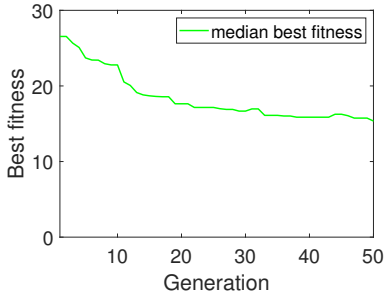
(c) Train bench2



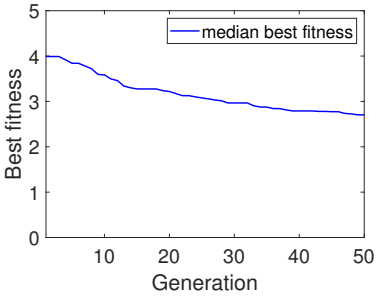
(d) Test bench2



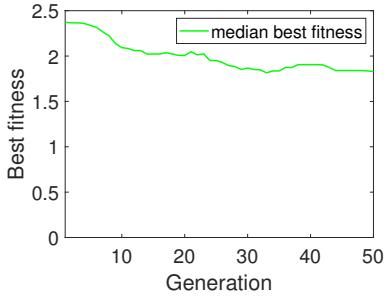
(e) Train bench3



(f) Test bench3



(g) Train bench4



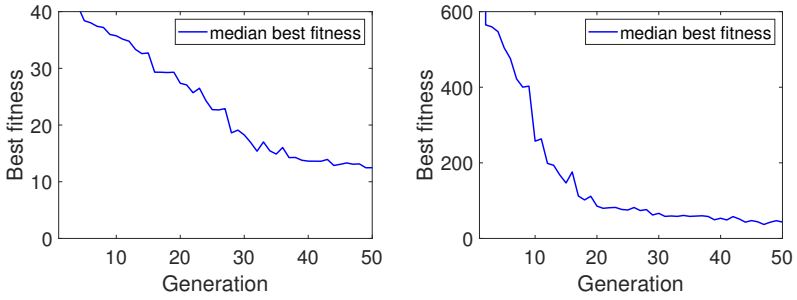
(h) Test bench4

Figure 10.5: Evolution of the median best fitness of *POP1* on bench1, bench2, bench3 and bench4. In blue the training set, in green the test set.

the test errors are generally decreasing, thus overfitting is not occurring.

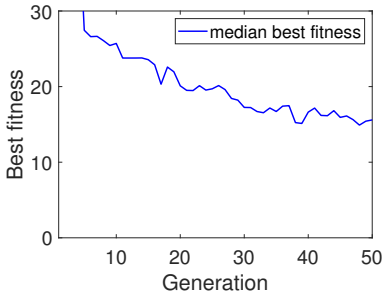
10.4.2 The (true) evolution of aggregations

As we are developing a coevolutionary algorithm, we want to be sure that both populations are really evolving through generations. For this reason, we report the evolution of the fitness of the best individual of *POP2*. We remind that, in case of a run where coevolution ends prematurely, the best individual is the last best individual found for every missing generation. The lines again show the median best fitness over the 50 runs performed.

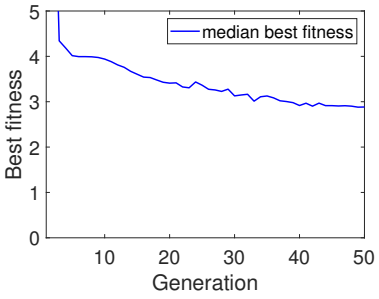


(a) Train bench1

(b) Train bench2



(c) Train bench3

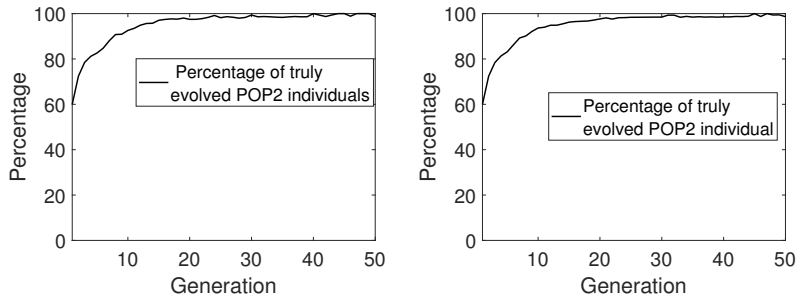


(d) Train bench4

Figure 10.6: Evolution of the median best fitness of *POP2* on bench1, bench2, bench3 and bench4.

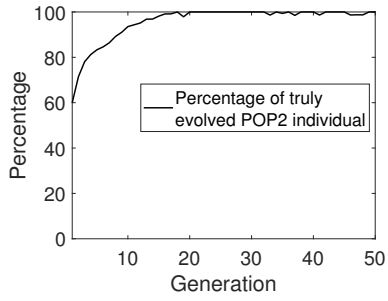
Figure 10.6 shows a decreasing fitness through generations, thus the best aggregations are used by better and better individuals of *POP1* through generations. These good individuals, however, may have not been originated through crossover and mutation. We remind, in fact, that during the genetic phase random individuals are added into *POP2* to keep diversity. To ensure that evolution through crossover and mutation is the actual process enhancing the performance, in Figure 10.7 we report the evolution of the percentage of individuals in *POP2* originated by a genetic operation.

As usual, the line shows the median percentage over the 50 runs. As Figure 10.7 shows, the non-random individuals increase in the population through generations, thus genetic evolution is really taking place.

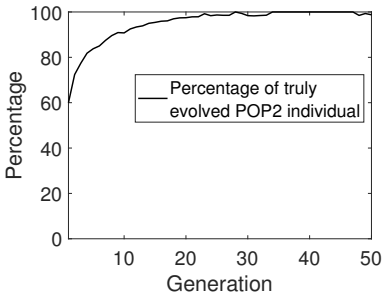


(a) Train bench1

(b) Train bench2



(c) Train bench3



(d) Train bench4

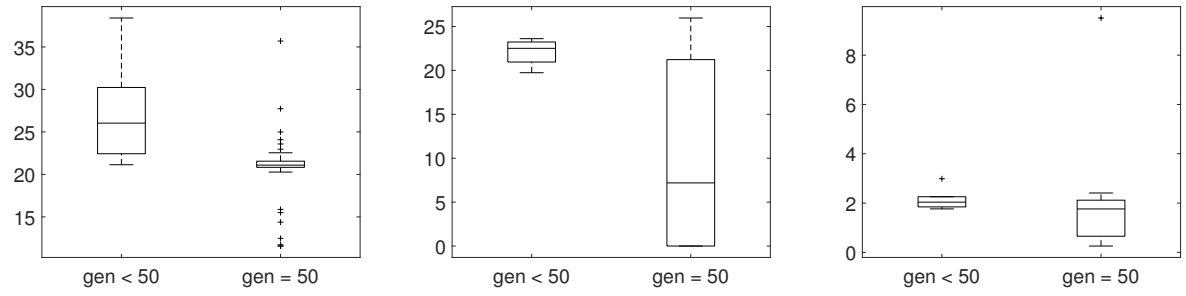
Figure 10.7: Median percentage of not random individuals in *POP2* on bench1, bench2, bench3 and bench4.

10.4.3 Coevolution prematurely ended

As already described, some runs may end the coevolution before the stopping criteria, because *POP1* individuals do not use anymore the aggregations of *POP2*. To understand the role of the coevolutionary process in approximating the aggregations, we compare the sample of test fitness of the best individuals in *POP1* of the two class of runs. The comparison is carried out by means of Wilcoxon signed rank test at $\alpha = 0.05$ significance level, with null hypothesis of no difference in performance between the samples. Table 10.3 reports the p-values of the test and Figure 10.8 shows the boxplot of the two samples in order to get a visual feeling, in case of difference in performance, of the sample that returns the best performance. Concerning bench2, the statistical test and the consequent boxplot are not reported because every run reaches the last generation.

Table 10.3: Wilcoxon rank-sum test p-values comparing runs that reach the last generation respect to runs that end the coevolution before the last generation. The significative values are represented in bold.

Bench1	$\mathbf{3.6 \cdot 10^{-4}}$
Bench3	**0.026**
Bench4	0.11



(a) Bench1 (b) Bench2 (c) Bench4

Figure 10.8: Boxplots of the test fitness of the best individuals in *POP1* for every benchmark problem. On the left the sample of coevo_VE_GP runs that end before generation 50, on the right the sample of coevo_VE_GP runs that reach generation 50.

Bench4 is the only problem where we fail to reject the null hypothesis of no difference in performance, therefore there is no sufficient evidence to conclude that one sample performs differently from the other.

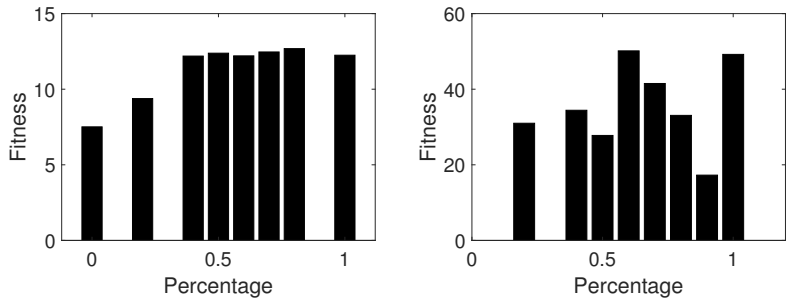
We should, therefore, use with caution models returned by coevolutionary processes ended before the last generation.

10.4.4 The driven inizialization is beneficial

The decision of including precise known aggregations (called forced individuals) in the initial pool of *POP2* should be demonstrated as an appropriate idea. We would like to observe, in fact, that individuals having these forced ones as ancestors have better performance with respect to individuals having only random trees as ancestors. This result indicates that known aggregations drive the evolutionary process towards the right direction. We therefore perform an analysis of the ancestors of the best individuals of *POP2* with respect to the fitness. After the first generation we store the ancestors of each individual. Moving on with the evolution we store for each individual the ancestors of its parents. This process allows the identification of forced individuals in the history of each individual.

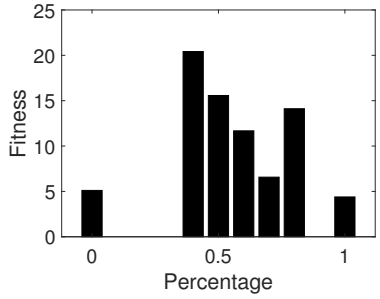
Figure 10.9 shows the fitness of the best individuals for each percentage of forced individuals in its genealogical tree. The bars represent the median fitness of the best individuals over the 50 best individuals whose percentage of forced ancestors is the same. We considered just runs that actually reach generation 50, because they are more representative of the coevolutionary process and they still form a sample of size greater than the minimum 30.

The barplots suggest that precise percentages of these forced ancestors determines the lowest fitness, thus the best performance. Unfortunately in bench1 and bench4 the best performance are achieved by individuals without forced ancestors. The other 2 problems instead prove that having a huge number of forced ancestors

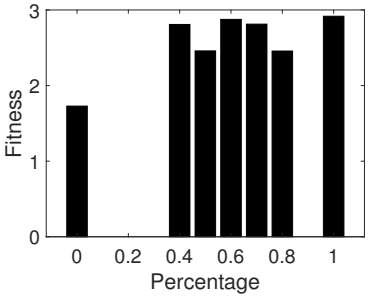


(a) Bench1

(b) Bench2



(c) Bench3



(d) Bench4

Figure 10.9: Barplots of the median best fitness in *POP2* for every benchmark problem. On the horizontal axis the percentage of forced individuals in the ancestors history, on the vertical axis the median fitness.

determines better performance. We can therefore conclude that the inclusion of meaningful aggregation during *POP2* initialization should be a problem dependent choice. This result actually agrees with the leitmotif of the whole work: some problems may be solved by innovative and unusual aggregations.

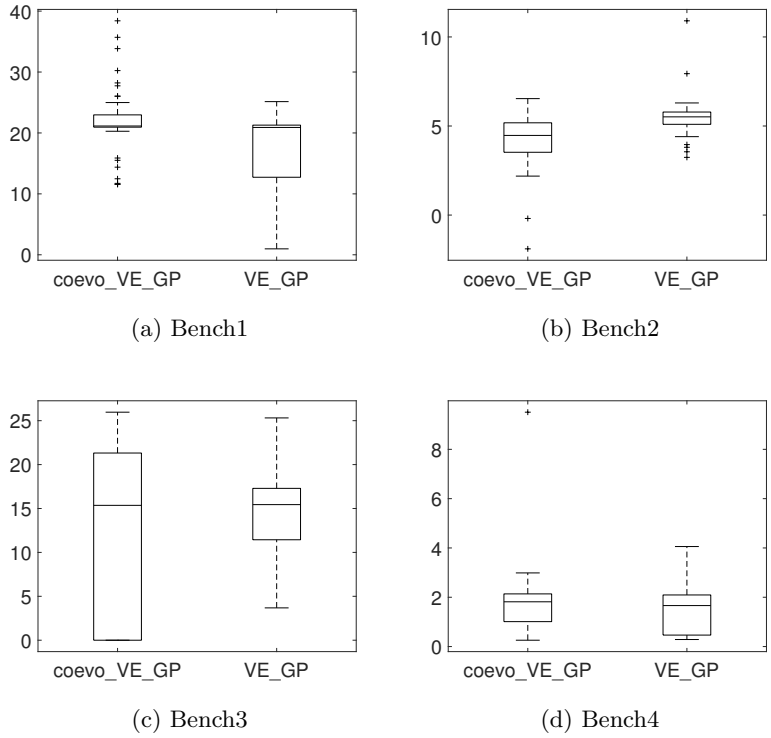
10.4.5 Comparison with VE_GP

To evaluate the difference in performance between coevo_VE_GP and VE_GP, we use a Wilcoxon rank-sum test, with confidence level $\alpha = 0.05$, comparing the 50 test fitness of the best individuals for each problem. The null hypothesis is that there is no difference in test performance between the two approaches. Concerning coevo_VE_GP, the best individual is always the best individual of the last generation coevolved. The resulting p-values are reported in Table 10.4. In Figure 10.10, we represent the boxplots of the two samples, provided to understand which technique achieves the best performance in case of significative statistical difference. The boxplot regarding bench2 is reported in logarithmic scale, due to the presence of outliers.

The results of subsection 10.4.3 revealed that runs that do not reach the last generation have usually worse performance respect to run that end properly the coevolutionary process. These runs therefore may obscure the good performance of coevo_VE_GP respect to standard VE_GP. Consequently we compare

Table 10.4: Wilcoxon rank-sum test p-values comparing coevo_VE_GP respect to VE_GP. The significative values are represented in bold.

Benchmark 1	**0.0011**
Benchmark 2	$\mathbf{5.6 \cdot 10^{-7}}$
Benchmark 3	0.37
Benchmark 4	0.39



(a) Bench1 (b) Bench2

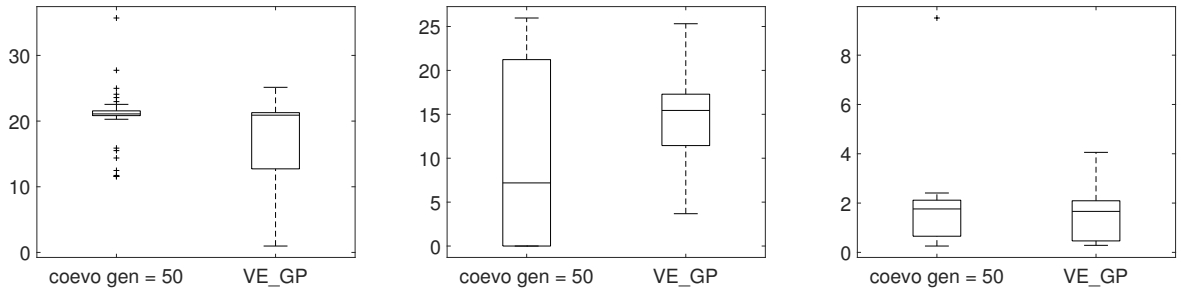
(c) Bench3 (d) Bench4

Figure 10.10: Boxplots coevo_VE_GP vs VE_GP.

VE_GP only against the runs of coevo_VE_GP that reach generation 5. Table 10.5 reports the p-values of the comparison, while Figure 10.11 shows the boxplots. Bench2 does not appear in this latter analysis because every coevo_VE_GP run of bench2 reaches properly generation 50.

Table 10.5: Wilcoxon rank-sum test p-values comparing coevo_VE_GP runs that reach generation 50 respect to VE_GP. The significative values are represented in bold.

Benchmark 1	**0.036**
Benchmark 3	0.11
Benchmark 4	0.72



(a) Bench1 (b) Bench3 (c) Bench4

Figure 10.11: Boxplot coevo_VE_GP vs VE_GP considering only coevolution runs that reach the final generation.

The results suggest that the previous considerations are not related to the worse performance of prematurely ended coevolution runs. In VE_GP, therefore, the aggregations seems not be totally destroyed during the genetic phase, and there is no evidence of a general predominance of one technique.

Besides the fact that these results depend on the problems involved, there is an aspect where coevo_VE_GP may exhibit its potential: computational effort. The key feature of coevo_VE_GP is the parallel evolution of two populations, that makes the identification of the best aggregation a separate sub-problem from the main one. This split determines a different computational overhead between coevo_VE_GP and classical VE_GP. We therefore decided to compare the fitness of the two approaches against the computational efforts, defined as the total number of nodes the algorithm has evaluated in each population for a given number of generations. At generation g the computational effort is:

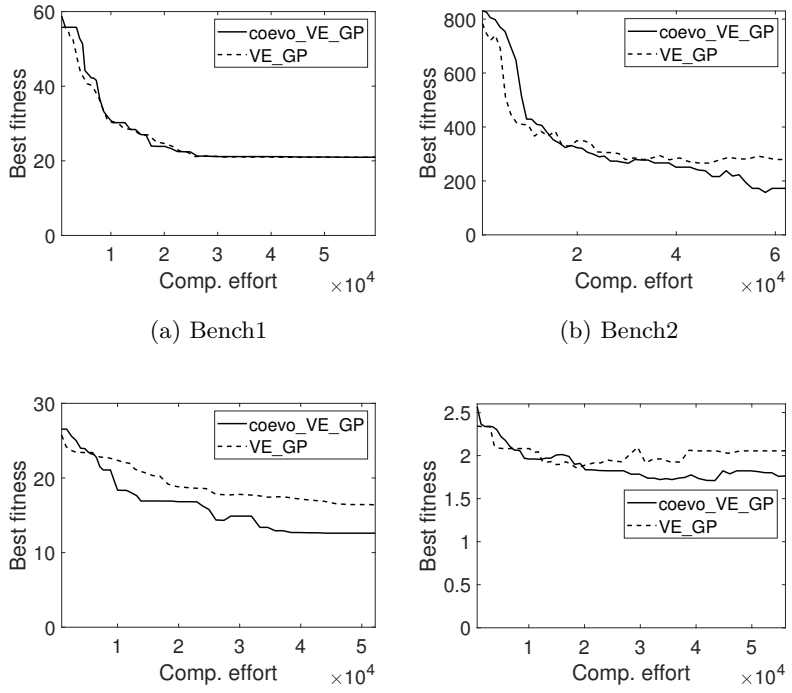
$$CE_g = PE_g + PE_{g-1} + \cdots + PE_0,$$

where

$$PE_g = \sum_{k=1}^{N} p\_k_g \cdot avg\_nodes\_k_g,$$

with N being the number of coevolving populations, p_k_g the number of individuals in population k at generation g and $avg\_nodes\_k_g$ the average number of nodes in population k at generation g. Figure 10.12 shows the median test fitness curves of the best individuals against computational effort, for every benchmark.

It can be seen that the our coevolutionary approach always require less computational effort to achieve the same performance.



(a) Bench1

(b) Bench2

(c) Bench3

(d) Bench4

Figure 10.12: Best median fitness against computational effort for every benchmark problem. The solid lines refer to coevo_VE_GP while the dashed lines refer to VE_GP.

10.5 Conclusions

The work was motivated by the idea of checking if evolution is able to extract knowledge from data and solve problems in an autonomous way. The technique of VE_GP was developed based on this trust, by restoring the natural structure of time series and avoiding prior aggregations. The coevolutionary approach we propose (coevo_VE_GP) removes from VE_GP the known aggregations used during evolution by evolving them in a parallel population, thus making a further step towards autonomy. The population of main programs relies on a population of aggregating functions by using these individuals as terminal functions. The evolution of this latter population is determined by the evolution of the main population: the better the individuals that include an aggregating function, the better the aggregating function. This subordinate fitness evaluation is the key feature of coevo_VE_GP. The algorithm we present fits perfectly in the coevolutionary field, but is built on innovative goals and techniques: the inclusion of the VE_GP evolutionary technique and the search for unusual aggregations of time series.

We explored the effectiveness and the efficiency of coevo_VE_GP by means of four benchmark problems based on different hand-tailored aggregations. We compared the coevo_VE_GP performance against the only method we are aware of, having the same ability to discovery innovative aggregations: standard VE_GP.

The primary result achieved is the validity of the proposed approach. In both populations, new evolved populations are better than previous ones. Moreover, when the coevolution stops prematurely because aggregating functions are no more used by the main population, the proposed solutions are worse compared to those evolved by complete coevolutionary processes. These outcomes confirm the correct working of the technique, but there are reasons to believe in an improvement, in terms of the quality of the evolved solutions, with respect to VE_GP. While in coevo_VE_GP, in fact, the aggregating functions are nodes, in VE_GP they are subtrees; since subtrees are subject to genetic manipulation, in standard VE_GP it is, in principle, much easier to destroy a good aggregation, compared to coevo_VE_GP, weakening therefore the evolutionary process. The results presented show that the approach of coevo_VE_GP is not outperforming VE_GP in terms of accuracy of the solutions. The strength of coevo_VE_GP is indeed revealed when gaining more insight into the behaviour of the techniques. Coevo_VE_GP exhibits a lower computational effort to achieve the same performance, compared to VE_GP. This feature is fundamental considering the area of application of these technique. The problems to tackle are complex and difficult, involving time series and aggregation. Reduce the computational effort is thus an indispensable goal to reduce the length of already expansive runs.

The introduction of coevo_VE_GP must be followed undoubtedly by a further exploration of the main behaviours emerged, and by an application of the technique on real problems. Since standard VE_GP has already been applied on real problems in 7, 8, one of the first steps of our future work is the use of coevo_VE_GP to tackle these problems. These investigations may find out innovative aggregations to approximate the target and may give an unusual interpretation of the final models to better understand the dynamic underlying the problems.

Part IV

Conclusions

Conclusions

Are genetic programming approaches an improvement of the current state of art in vector abundance modelling? At the end of this dissertation we can definitely say yes. The original goal of this work was an investigation of genetic programming (GP) in the field of vector abundance modelling through the specific problem involving mosquito. As described in Chapter 2, Machine Learning (ML) is not a frequent choice when dealing with vector abundance prediction, despite the good results achieved. In addition, GP, belonging to a subfield of ML, has only been applied on problems of marine ecology. There is, therefore, a clear lack of ML applications in ecological modelling that GP can start to fill as a suitable technique to combine accuracy and epidemiologist interest of readability. The proposed investigation of GP consisted not only in the application of the technique, but mainly on the enhancement of GP to properly deal with the data structures regarding vector abundance modelling. We can therefore analyse the fulfilments of the goals dividing them into two points: GP application and GP enhancement.

GP application

The dataset of mosquito collections in Piedmont region gave us the possibility to explore GP capabilities in the field. GP has revealed all its deemed advantages: ability to catch more complex variable interactions rather than classical statistical modelling, ability to perform an implicit features selection and the ability to provide a readable model. Since machine learning (ML) methods do not assume any fixed combination of variables to predict the target, the property of finding complex interaction of variable is shared by all ML techniques. The impossibility to have, however, a model to be interpreted in order to understand the main causes of high abundance has put ML methods aside in the field of vector abundance modelling. Thanks to this work, we have provided awareness to the veterinary science community about a ML technique, GP, that can join predictive accuracy and generalization ability with readability, probably the main reason of immediate statistical modelling application.

GP enhancement

Our main contribution, however, is the development of vectorial genetic programming (VE_GP). Thanks to the issues raised from GP application to mosquito abundance prediction, we built a technique, VE_GP, suitable for vector abundance modelling, but open to other application involving similar data structures. We claim with ambition, in fact, that VE_GP is a general technique and should be used with high degree of reliability when repeated observations of a problem have their natural representation in a vector. To give a preliminary proof this statement we tested VE_GP not only on mosquito prediction but even on the ventilation problem. Despite the increased complexity of the data format VE_GP can deal with, this new techniques still embrace the good properties of models' readability and easy structure of classical GP. Moreover, VE_GP is a winning approach without any further tentative of enhance its performance. When we tried, in fact, to tune the genetic operators by including the geometric semantic ones, the complexity of the data strongly emerged and compromised VE_GP predictive accuracy.

The overall path of this work, reflecting the next moves to each emerging issues, provides a major insight: data left revealing knowledge are more powerful rather than experts knowledge. The arguments that led us to the development of VE_GP were originated by artificial constructions based on experts knowledge on the problem at hand. Helping the evolutionary process through prior knowledge seemed reasonable, nonetheless this knowledge was hindering the discovery of the real variables interactions. The idea of an assumption-free evolutionary process arised in VE_GP, but became strongly evident in the coevolutionary vectorial genetic programming (coevo_VE_GP). While VE_GP removes prior aggregations from time series but still assumes known aggregations to be the building blocks of variables interactions, coevo_VE_GP let even the nature of aggregations evolve. This idea always suggested us how to move on. Trying to enhance VE_GP by simply enhancing its predictive ability using geometric semantic operators was not the winning approach, instead trying to enhance VE_GP philosophy following the prior knowledge removal gave birth to the powerful coevo_VE_GP. The whole dissertation journey is actually a process towards the awareness that data hide all the knowledge about a problem, we have only to find the best way to extract this knowledge.

10.6 Future works

All the concluding assertions should take into account one main limitation of the research: dataset availability. We inferred the potential of VE_GP applying the technique to two real problems, with only one belonging to the field of ecological modelling. The conclusions we made therefore are undoubtedly problem specific, but other vector abundance dataset are likely to have the same structure of the one regarding mosquito. A possible difference may be the choice of the time instants thus months or hours rather than days. In addition,

coevo_VE_GP, due to time restriction has just been explored through artificial benchmark problem and not jet on real ones.

In order to overcome the limitation of the study a recommended future work is the application of VE_GP and coevo_VE_GP on real problems regarding vector abundance modelling. Moreover, this study was oriented to vector dynamics in time. VE_GP is a technique that consider vectors as terminals, without strictly assuming that vector representation should be used only for time series. The vectorial terminal can be implied for any other ordered series that requires not to be collapsed. Predicting vector abundance in altitude provides an example of ordered series where the sequential values do not correspond to consequent values in time but rather consequent values in space. It would be interesting therefore to apply VE_GP even to vector abundance problem concerning spatial prediction rather time prediction to highlight the potential of the technique in dealing with any kind of ordered structure.

Bibliography

[1] Arpa Piemonte. `http://www.arpa.piemonte.it`.

[2] INLA. `https://inla.r-inla-download.org/R/stable`.

[3] Istituto per le piante da legno e l'ambiente. `ww.ipla.org`.

[4] Kaggle competitions. `https://www.kaggle.com`.

[5] Random forest, 2002. `https://CRAN.R-project.org/doc/Rnews/`.

[6] XGBoost, R package version 0.82.1, 2019. `https://CRAN.R-project.org/package=xgboost`.

[7] Ministero della Salute. West Nile Disease - Notifica alla Commissione europea e all'OIE - Piano di sorveglianza straordinaria. Gazzetta Ufficiale della Repubblica Italiana, N. 277, 26/11/2008.

[8] Blum A. *Neural Networks in C++: An Object-oriented Framework for Building Connectionist Systems*. 1992.

[9] Moraglio A., Krawiec K., and Johnson C. Geometric semantic genetic programming. In *Parallel Problem Solving from Nature - PPSN XII*, volume 7491 of *Lecture Notes in Computer Science*, pages 21–31, 2012.

[10] Vargha A. and Delaney H. D. A critique and improvement of the CL common language effect size statistics of McGraw and Wong. *Journal of Educational and Behavioral Statistics*, 25(2):101–132, 2000.

[11] Bisanzio D., Giacobini M., and Bertolotti L. Spatio-temporal patterns of distribution of West Nile virus vectors in eastern Piedmont Region, Italy. *Parasites Vectors*, 4, 2011.

[12] Cianci D., Hartemink N., and Ibáñez Justicia A. Modelling the potential spatial distribution of mosquito species using three different techniques. *Int J Health Geogr*, 14, 2015.

[13] Dermofal D. Time-series cross-sectional and panel data models. *Spatial Analysis for the Social Sciences*, 32:141–157, 2015.

[14] Gupta D. and Ghafir S. An overview of methods maintaining diversity in genetic algorithms. *IInternational Journal of Emerging Technology and Advanced Engineering*, 2, 2012.

[15] Roiz D., Ruiz S., Soriguer R., and Figuerola J. Landscape effects on the presence, abundance and diversity of mosquitoes in Mediterranean wetlands. *PlOs One*, 2015.

[16] M. S. Diamond. *West Nile Encephalitis Virus Infection*. Springer, 2008.

[17] Papworth D.J., Marini S., and Conversi A. A novel, unbiased analysis approach for investigating population dynamics: A case study on *Calanus finmarchicus* and its decline in the North Sea. *PLoS ONE*, 11(7), 2016.

[18] Baralis E., Cerquitelli T., Giordano A., Mezzani A., Susta D., and Xiao X. Predicting cardiopulmonary response to incremental exercise test. In *28th IEEE International Symposium on Computer-Based Medical Systems (CBMS 2015)*, pages 135–140. IEEE, 2015.

[19] Cid Alfaro E., Sharman K., and Esparcia Alcázar A. Genetic programming and serial processing for time series classification. *Evolutionary Computation*, 22:265–285, 2013.

[20] Sinka M. E., Rubio-Palis Y., and Manguin S. The dominant *anopheles* vectors of human malaria in the Americas: occurrence data, distribution maps and bionomic précis. *Parasites Vectors*, 3(72), 2010.

[21] Rizzo C. et al. West Nile virus transmission: results from the integrated surveillance system in Italy, 2008 to 2015. *Euro Surveill 2016*, 21(37), 2016.

[22] Chaves L. F., Hamer G. L., Walker E. D., Brown W. M., Ruiz M. O., and Kitron U. D. Climatic variability and landscape heterogeneity impact urban mosquito diversity and vector abundance and infection. *Ecosphere*, 2(6):1–21, 2011.

[23] Monaco F., Lelli R., Teodori L., Pinoni C., Di Gennaro A., Polci A., Calistri P., and Savini G. Re-emergence of West Nile virus in Italy. *Zoonosis public health*, 57:476–486, 2010.

[24] European Centre for Disease Prevention and Control. Guidelines for the surveillance of native mosquitoes in Europe. `https://www.ecdc.europa.eu/sites/default/files/media/en/publications/Publications/surveillance-of%20native-mosquitoes%20-guidelines.pdf`, 2014.

[25] Benelli G. and Melhorn H. *Mosquito-borne diseases*. 2018.

[26] Marini G., Arnoldi D., and Baldacchino F. et al. First report of the influence of temperature on the bionomics and population dynamics of*aedes koreicus*, a new invasive alien species in Europe. *Parasites Vectors*, 12(524), 2019.

[27] Marini G., R. Rosà, and Pugliese A. et al. West Nile virus transmission and human infection risk in Veneto (Italy): a modelling analysis. *Sci Rep*, 8(14005), 2018.

[28] Braks M. A. H., van Wieren S. E., Takken W., and Sprong H. *Ecology and control of vector-borne diseases*. Wageningen Academic Publishers, 2016.

[29] Guo H., Jack L. B., and Nandi A. K. Automated feature extraction using genetic programming for bearing condition monitoring. In *Proceedings of the 14th IEEE Signal Processing Society Workshop Machine Learning for Signal Processing*, pages 519–528, 2004.

[30] Yoo E. H. Site-specific prediction of West Nile virus mosquito abundance in Greater Toronto Area using generalized linear mixed models. *International Journal of Geographical Information Science*, 28:296–313, 2013.

[31] Yoo E. H., Chen D., Diao C., and Russell C. The effects of weather and environmental factors on West Nile virus mosquito abundance in Greater Toronto area. *Earth Interactions*, 2016.

[32] De Falco I., Della Cioppa A., and Tarantino E. A genetic programming system for time series prediction and its application to el Niño forecast. *Soft Computing: Methodologies and Applications*, 32:151–162, 2005.

[33] Elman J. Finding structure in time. *Cognitive Science*, 14:19–211, 1990.

[34] Fox J. *Applied Regression Analysis and Generalized Linear Models*. 2015.

[35] McDermott J., O'Reilly U. M., Luke S., and White D. A community-led effort towards improving experimentation in genetic programming.

[36] Levemberg K. A method for the solution of certain non-linear problems in least squares. *Quarterly of Applied Mathematics*, 2(2):164–168, 1944.

[37] Okuneye K., Abdelrazec A., and Gumel A. B. Mathematical analysis of a weather-driven model for the population ecology of mosquitoes. *Mathematical Biosciences and Engineering*, 15, 2018.

[38] Autorino G. L., Battisti A., Deubel V., Ferrari G., Forletta R., Giovannini A., Lelli R., Murri S., and Scicluna M.T. West Nile virus epidemic in horses, Tuscany region, Italy. *Emerging Infect Dis*, 8(1):372–1378, 2002.

[39] Breiman L. Random forests. *Machine Learning*, 45:5–32, 2001.

[40] Breiman L., Friedman J., Stone C. J., and Olshen R. A. *Classification and Regression Trees*. The Wadsworth and Brooks-Cole statistics-probability series. Taylor & Francis, 1984.

[41] Holladay K. L. and Robbins K. A. Evolution of signal processing algorithm using vector based genetic programming. In *15th International Conference on Digital Signal Processing*, pages 503–506, 2007.

[42] Pecoraro H. L., Day H. L., Reineke R., Stevens N., Withey J. C., Marzluff J. M., and Meschke J. S. Climatic and landscape correlates for potential West Nile virus mosquito vectors in the Seattle region. *J. of Vector Ecology*, 32(1):22–28, 2007.

[43] Vanneschi L., Castelli M., Manzoni L., and Silva S. A new implementation of geometric semantic GP and its application to problems in pharmacokinetics. In *Genetic Programming- 16th European Conference EUROGP 2013*, volume 7831, pages 205–216, 2013.

[44] Vanneschi L., Silva S., Castelli M., and Manzoni L. Geometric semantic programming for real life applications. *Genetic Programming Theory and Practice*, 11:191–209, 2014.

[45] Kramer L.D., Styer L.M., and Ebel G.D. A global perspective on the epidemiology of west nile virus. *Annual review of entomology*, 53:61–81, 2008.

[46] Castelli M., Castaldi D., Giordani I., Silva S., Vanneschi L., Archetti F., and Maccagnola D. An efficient implementation of geometric semantic genetic programming for antcoagulation level prediction on pharmacogenetics. *Progress in Artificial Intellingence*, pages 78–89, 2013.

[47] Castelli M., Vanneschi L., and Popovic A. Controlling individuals growth in semantic genetic programming through elitist replacement. *Computational Intellingence and Neuroscience*, 2016.

[48] Castelli M., Vanneschi L., and FeliceM. D. Forecasting short-term electricty consumption using a semnatics-based genetic programming framework: the south Italy case. *Energy Economics*, 47:37–41, 2015.

[49] Castelli M., Vanneschi L., and Silva S. Prediction of the unified parkinson's disease rating scale assessment using a genetic programming system with geometric semantic genetic operators. *Expert Systems with Applications*, 41(10):4608–4616, 2014.

[50] Keijzer M. Improving symbolic regression with interval arithmetic and linear scaling. In *Genetic Programming-EuroGP 2003*, Lecture Notes in Computer Science, pages 70–82. Springer, 2003.

[51] Khormi H. M., Kumar L., and Elzahrany L. A. Regression model for predicting adult female *Aedes aegypti* based on meteorological variables: A case study of Jeddah, Saudi Arabia. *J Earth Sci Clim Change*, 5(1), 2014.

[52] Lutambi A. M., Penny M. A., Smith T., and Chitnis N. Mathematical modelling of mosquito dispersal in a heterogeneous environment. *Mathematical Biosciences*, 241(2):198–216, 2013.

[53] Oshiro T. M., Perez P. S., and Baranauskas J. A. How many trees in a random forest? *Machine Learning and Data Mining in Pattern Recognition*, 7376:154–168, 2012.

[54] Shone S. M., Curriero F. C., Lesser C. R., and Glass G. E. Characterizing population dynamics of *Aedes sollicitans* (Diptera: Culicidae) using meteorological data. *Journal of Medical Entomology*, 43(2):393–402, 2006.

[55] The MathWorks. MATLAB Neural Network Toolbox, 2018.

[56] Engler O., Savini G., and Papa A. et al. European surveillance for West Nile virus in mosquito populations. *Int J Environ Res Public Health*, 10(10):4869–4895, 2012.

[57] World Health Organization. Vector-borne diseases. `https://www.who.int/news-room/fact-sheets/detail/vector-borne-diseases`, 2020.

[58] Aspinall P., Mavros P., Coyne R., and Roe J. The urban brain: analysing outdoor physical activity with mobile EEG. *Br J Sports Med*, 49(4):272–276, 2015.

[59] Bartashevich P., Bakurov I., Mostaghim S., and Vanneschi L. Evolving PSO algorithm design in vector fields using geometric semantic GP. In *GECCO 2018: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 262–263, 2018.

[60] Koza J. R. *Genetic Programming*. The MIT Press, 1992.

[61] Poli R., Langdon W. B., and McPhee N. F. *A field guide to genetic programming*. Published via `http://lulu.com` and freely available at `http://www.gp-field-guide.org.uk`, 2008. (With contributions by J. R. Koza).

[62] Rosà R., Marini G., and Bolzoni L. *et. al.* Early warning of West Nile virus mosquito vector: climate and land use models successfully explain phenology and abundance of *Culex pipiens* mosquitoes in north-western Italy. *Parasites Vectors*, 7, 2014.

[63] Ross R., Alduishy A., and Gonzáles-Haro C. Validation of the cosmed k4b2 portable metabolic system during running outdoors. *Journal of strength and conditioning research*, 2019.

[64] Trawinski P. R. and MacKay D. S. Meteorologically conditioned time-series predictions of West Nile virus vector mosquitoes. *Vector-Borne and Zoonotic Diseases*, 8(4):505–521, 2008.

[65] Chen S., Whiteman A., and Li A. *et. al.* An operational machine learning approach to predict mosquito abundance based on socioeconomic and landscape patterns. *Landascape Ecol*, 34:1295–1311, 2019.

[66] Haykin S. *Neural Networks: a comprehensive foundation.* 1999.

[67] Hochreiter S. and Schmidhuber J. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[68] Kwon Y. S., Bae M. J., Chung N., Lee Y. R., Hwang S., Kim S. A., Choi Y. J., and Park Y. S. Modeling occurrence of urban mosquitos based on land use types and meteorological factors in Korea. *Int J Environ Res Public Health*, 12(10):13131–13147, 2015.

[69] Liu S., Gao R., Staudenmayer J., and Freedson P. Improved regression models for ventilation estimation based on chest and abdomen movements. *Physiological Measurement*, 33(1):79–93, 2012.

[70] Luke S. and Panait L. Lexicographic parsimony pressure. In *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 829–836, 2002.

[71] Marini S. and Conversi A. Understanding zooplankton long term variability through genetic programming. In *Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics*, EvoBIO 2012. Lecture Notes in Computer Science, 2012.

[72] Silva S. GPLAB - A Genetic Programming Toolbox for MATLAB. `http://gplab.sourceforge.net/index.html`.

[73] Walsh A. S., Glass G. E., Lesser C. R., and Curriero F. C. Predicting seasonal abundance of mosquitoes based on off-season meteorological conditions. *Environ Ecol Stat*, 15:279–291, 2008.

[74] Chen T. and Guestrin C. Xgboost: A scalable tree boosting system. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, 2016.

[75] ORNL DAAC Oak Ridge TennesseeUSA. ORNL DAAC 2018 MODIS and VIIRS land products global subsetting and visualization tool. `https://modis.ornl.gov/`.

[76] Banzhaf W., Nordin P., Keller R. E., and Francone F. D. *Genetic Programming: An Introduction.* 1998.

[77] Stroup W. W. *Generalized Linear Mixed Models: Modern Concepts, Methods and Applications.* CLC Press, 2012.

[78] Lee K. Y., Chung N., and Hwang S. Application of an artificial neural network (ANN) model for predicting mosquito abundances in urban areas. *Ecological Informatics*, 36:172–180, 2016.

[79] Obermeyer Z. and Emanuel E. J. Predicting the future — big data, machine learning, and clinical medicine. *New England Journal of Medicine*, 375(13):1216–1219, 2016.