Applied Soft Computing Journal 89 (2020) 106097

Contents lists available at ScienceDirect

Applied Soft Computing Journal

journal homepage: www.elsevier.com/locate/asoc

Towards the use of vector based GP to predict physiological time series

Irene Azzali [a,*], Leonardo Vanneschi [b,c], Illya Bakurov [b], Sara Silva [c], Marco Ivaldi [d], Mario Giacobini [a]

[a] DAMU - Data Analysis and Modelling Unit, Department of Veterinary Sciences, University of Torino, L.go Braccini 2, 10095, Turin, Italy
[b] NOVA Information Management School (NOVA IMS), Universidade Nova de Lisboa, Campus de Campolide, 1070-312 Lisboa, Portugal
[c] LASIGE, Departamento de Informática, Faculdade de Ciências, Universidade de Lisboa, 1749-016 Lisboa, Portugal
[d] SUISM, University Structure of Hygiene and Sport Sciences, University of Turin, Corso Trento 13, 10129 Turin, Italy

ARTICLE INFO

Article history:
Received 30 November 2018
Received in revised form 24 December 2019
Accepted 13 January 2020
Available online 24 January 2020

Keywords:
Ventilation
Physiological data
Machine learning
Genetic programming
Time series

ABSTRACT

Prediction of physiological time series is frequently approached by means of machine learning (ML) algorithms. However, most ML techniques are not able to directly manage time series, thus they do not exploit all the useful information such as patterns, peaks and regularities provided by the time dimension. Besides advanced ML methods such as recurrent neural network that preserve the ordered nature of time series, a recently developed approach of genetic programming, VE-GP, looks promising on the problem in analysis. VE-GP allows time series as terminals in the form of a vector, including new strategies to exploit this representation. In this paper we compare different ML techniques on the real problem of predicting ventilation flow from physiological variables with the aim of highlighting the potential of VE-GP. Experimental results show the advantage of applying this technique in the problem and we ascribe the good performances to the ability of properly catching meaningful information from time series.

1. Introduction

Prediction of physiological time series 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 [1] and in field monitoring may be difficult where there is a lack of portable instruments to directly register data [2]. Several techniques can be used to perform these predictions, from classical linear regression to machine learning (ML) algorithms [3]. 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 creating a panel dataset [4]. This strategy unfortunately implies the loss of information regarding the history of the series (peaks and regularities) that may be useful to rebuild the corresponding target series. Due to this lack, we decide to investigate the recently presented vectorial genetic programming (VE-GP) [5] on the prediction of physiological time series. VE-GP is a new approach of standard genetic programming that allows vectors as terminals. Time series are therefore handled by VE-GP as vectors, without splitting. This technique looks promising on these problems because it can manage both vectors and scalars together, it can treat vectors of different lengths and it introduces changes in the initialization and primitive set with the goal of exploiting at best the new allowed representation of terminals.

In this paper we compare several techniques that treat differently the time dimension on a real physiological dataset. The specific involved problem is the prediction of ventilation flows of running people based on heart rate, already suggested as ventilation predictor [2,6], and other physiological variables, in order to monitor the inhaled load of pollutants. Six different techniques are applied on physiological parameters to evaluate ventilation. On one hand we consider 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

* Corresponding author.
E-mail addresses: irene.azzali@unito.it (I. Azzali), lvanneschi@novaims.unl.pt (L. Vanneschi), ibakurov@isegi.unl.pt (I. Bakurov), sara@fc.ul.pt (S. Silva), marco.ivaldi@unito.it (M. Ivaldi), mario.giacobini@unito.it (M. Giacobini).

https://doi.org/10.1016/j.asoc.2020.106097
1568-4946/

cases. On the other, we investigate recurrent neural network (RNN) that is frequently used to process time series and VE-GP. These last two take into account the sequential nature of data, so they work on a dataset where time series are vectors. While LR, RF, MLP, GP and RNN are simply chosen as state of the art on regression problems as the one examined here, VE-GP is chosen to exhibit its potential on this suitable problem.

The aim of the comparison is a first evaluation of the performances of the methods according to data representation, therefore we do not strongly tune each technique, but we rather consider general reasonable values for the parameters to make them perform at a reasonable level.

The paper is organized as follows: in Section 2 we discuss the most relevant previous and related works concerning physiological time series prediction. Section 3 describes the dataset involved. Sections 4 and 5 present the techniques that we explore with a broader description of the new VE-GP. Section 6 shows the analysis of the comparison between the different method and discusses the obtained results. Finally, Section 7 concludes the paper.

2. Previous and related works

Study on the literature about ML and regression methods to predict physiological time series is necessary to highlight the main drawbacks of these basic approaches.

Linear regression is usually the first modelling algorithm applied to a problem. However, it works well on plain dataset and relationships which are not intrinsic features of physiological time series. In [7], LR models are applied to predict ventilation series only from the heart rate series. Regardless of the simple dataset with only two variables involved, the authors conclude that further evaluations are demanded.

ML algorithms, instead, carried out a wide range of predictions dealing with physiological time series with very good results. Rather than panel data representation, basic ML techniques frequently manage time series in other forms, depending on the problem in analysis. However, these other data representations still cannot capture all the information about the series. In [8] an artificial neural network is implemented to predict blood glucose levels of diabetic people using only continuous glucose monitoring data. The time series in this case are divided into different variables. Despite the accurate performances, the approach has one major restraint: the number of input neurons is small (4) and fixed, therefore the information concerning the history of blood monitoring is very limited. The same approach of splitting the series in variables is found in [9]. In this paper, GP is used to predict characteristics of the ECG R–R interval variation from 9 ECG interval signals. Another common technique to manage time series is the feature extraction. Meaningful statistical values are derived from raw time series and used as predictors rather than considering the time series itself. A RF is developed in [10] to predict sepsis using features extracted from streaming physiological data in real-time.

This a priori processing of time series prevents the algorithms to learn meaningful characteristics of the series directly from data and may cause a loss of information. Therefore, methods that use the natural ordered representation of time series are preferred. In [11] a RNN is involved to predict epileptic seizures based on electroencephalography and again a RNN in [12] is applied to predict dynamic respiratory state starting from heart rate, oxygen saturation and respiration series. To enhance the capacity to learn long-term dependencies, deeper RNN such as long-short-term memory (LSTM) are chosen [13]. Regarding VE-GP, it has only been tested on benchmark time series problems [5]. However, its capability of dealing with different length time series and at the same time with constants encourage the further investigation of this technique on real problems such as physiological time series prediction.

3. The dataset

The ML techniques are investigated on a real problem 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 [14].

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, [15]). 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 s 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 20 496 rows (instances, observations or fitness cases) and 5 columns (features or variables). The features selected as predictors are: the age (AGE), the gender (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 gender, the BMI, the heart rate series and the ventilation series of a person.

4. State of art of machine learning in the health-care domain

In this section, we briefly introduce the ML algorithms usually enrolled when facing the problem of prediction of physiological time series. While GP, RF, MLP and LR handle time series by splitting them into different fitness cases, RNN exploits the sequential nature of the time series.

4.1. Genetic programming

Genetic programming is an evolutionary technique that breeds a population of feasible models towards the fittest model of data through a process that mimics the principles of evolution stated by C. Darwin [16]. Individuals that well fit data are probabilistically selected and recombined using a set of variation operators, called genetic operators, usually crossover and mutation. The performance of an individual is measured through a specific function, called fitness function, that formalizes the goal of the whole process.

For our problem, we opt for the classical tree-based representation of the GP individuals, built by combining primitive functions from the function set $F = \{+, -, *, //\}$, where $//$ is the division operator protected as in [17], with the elements of the terminal set T composed by 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 [18] 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.

We use a public GP toolbox called GPLab [19], that consists in a quite user-friendly Matlab implementation of GP. Since the purpose of the paper is not to optimize a precise modelling technique, we keep the default parameters of GP proposed by the system and reported in Table 1.

4.2. Random forest

Random Forest denotes an ensemble of Regression Trees presented by Breiman in 2001 [20]. Starting from the classical bagging technique, Brieman introduced a new learning phase for each of the trees in the forest. At each node, the best split is produced involving only n variables, randomly selected from the total N features. Differently from the ordinary technique, trees are unlikely to be correlated since they do not select any more the same strong predictors. The output of the forest is the mean of the individual trees predictions.

In this work, we use the Brieman model implemented in R. Following the main conclusion of [21], 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 1.

4.3. Multilayer perceptron

Multilayer Perceptron is a feed-forward neural network [22]. It consists of at least an input layer, a hidden layer of non linearly activated hidden nodes and an output layer. We choose this method rather than simple perceptron because it can capture more complex relationships that are inherent in the data.

We consider the MLP implemented in the Matlab neural-network toolbox. The network is trained using the Levenberg–Marquardt (LM) back-propagation learning algorithm [23]. 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 1.

4.4. Linear regression

Linear regression is the technique of establishing a linear relationship between the predictors and the target [24]. We perform LR using the available function in R. Coefficients are determined by means of the least squares procedure. Since our problem deals with 5 predictors and 1 target each fitness case is a point in the 5 dimension space and a LR model is a hyperplane in the same space. The training consists therefore in finding the hyperplane in 5 dimension space that minimizes the sum of squared residuals (a residual being: the difference between a collected value, and the predicted value provided by a model).

Table 1
Parameters used to set ML algorithms.

GP parameters	
Population size	500
Max number of generations	300
Initialization	Ramped Half and Half [17]
Selection method	Lexicographic parsimony pressure [28]
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

4.5. Recurrent neural network and long short-term memory

Recurrent neural network is a type of artificial neural network designed to handle sequence of data [25]. 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 [26].

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 the network can predict ventilation at instant t using information about the heart rate values collected from the beginning of the activity until instant t.

LSTM is performed using the available implementation in Matlab deep learning toolbox. The core components of a LSTM network are a sequence input layer and a LSTM layer that learns long-term dependencies between time steps of sequential data. The network is trained using the default Adam optimization [27], with the number of epochs equal to 50. We set the batch size equal to 1 so that different observations are not forced to be padded. In Table 1 we summarize the parameters setting for LSTM.

5. Vectorial GP

VE-GP is a recently introduced version of GP, developed to improve performance when time series are inherent in the data [5]. This new technique does not only consider a representation of the data directly in a vectorial form, but also contains a set of new strategies, specialized on vectors. Here, we describe the main novelties of this approach.

Primitive set

In GP, the primitive set is usually composed by terminals and functions that are combined to build a model in a tree structure. In VE-GP, besides allowing vectors as terminals, new functions

Table 2
Aggregate functions of arity 1. The first column identifies the primitive function, the second column briefly describes it and the third column returns the outcome of the function applied on a given vector v (in this example, v = [3, 1.5, 2.1, 4]).

Primitive function (pf)	Description	pf(v)
`V_mean`	Returns the mean of a vector	`V_mean`([1, 2.5, 4.3, 0.7]) = 2.1
`V_max, V_min`	Returns the maximum/minimum of a vector	`V_max`([3, 1.5, 2.1, 4]) = 4 `V_min`([3, 1.5, 2.1, 4]) = 1.5
`V_sum`	Returns the sum of a vector	`V_sum`([3, 1.5, 2.1, 4]) = 10.6
`C_mean`	Returns the cumulative mean of a vector	`C_mean`([3, 1.5, 2.1, 4]) = [3, 2.3, 2.2, 2.7]
`C_sum`	Returns the cumulative sum of a vector	`C_sum`([3, 1.5, 2.1, 4]) = [3, 4.5, 6.6, 10.6]
`C_max, C_min`	Returns the cumulative maximum/minimum of a vector	`C_max`([3, 1.5, 2.1, 4]) = [3, 3, 3, 4] `C_min`([3, 1.5, 2.1, 4]) = [3, 1.5, 1.5, 1.5]

Table 3
Functions of arity two. The first column identifies the primitive function, the second column briefly describes it, the third column returns the outcome of the function applied on a given scalar s and vector v1 and the fourth column returns the outcome of the function applied on two vectors v1 and v2 (in this example, s = 1.2, v1 = [3, 2.1, 4.6], v2 = [1.2, 3.6, 0.8, 2.5]).

Primitive function (pf)	Description	pf(s, v1)	pf(v1, v2)
`VSUMW`	Returns the element-wise sum between two vectors, completed if necessary according to 5	`VSUMW`(1.2, [3, 2.1, 4.6]) = [4.2, 3.3, 5.8]	`VSUMW`([3, 2.1, 4.6], [1.2, 3.6, 0.8, 2.5]) = [4.2, 5.7, 5.4, 2.5]
`V_W`	Returns the element-wise difference between two vectors, completed if necessary according to 5	`V_W`(1.2, [3, 2.1, 4.6]) = [−1.8, −0.9, −3.4]	`V_W`([3, 2.1, 4.6], [1.2, 3.6, 0.8, 2.5]) = [1.8, −1.5, 3.8, −2.5]
`VprW`	Returns the element-wise product between two vectors, completed if necessary according to 5	`VprW`(1.2, [3, 2.1, 4.6]) = [3.6, 2.5, 5.5]	`VprW`([3, 2.1, 4.6], [1.2, 3.6, 0.8, 2.5]) = [3.6, 7.6, 3.7, 2.5]
`VdivW`	Returns the element-wise division *(protected version as [17]) between two vectors, completed if necessary according to 5	`VdivW`(1.2, [3, 2.1, 4.6]) = [0.4, 0.6, 0.3]	`VdivW`([3, 2.1, 4.6], [1.2, 3.6, 0.8, 2.5]) = [2.5, 0.6, 5.8, 0.4]

are included as primitives. In all cases we define scalars as vector of length 1.

Functions of arity 1. In order to capture the behaviour of a vector, aggregate functions are involved in the primitive set. These functions combine together multiple entries of a vector into one value of possible more significant meaning. Two possible versions of aggregate functions are available in VE-GP: standard and cumulative. While standard versions collapse the whole vector, cumulative versions collapse different portions of the vector. To clarify, when cumulative aggregate functions receive a vector $V = [V_1, V_2, \ldots, V_n]$ as input, they return another vector $W = [W_1, W_2, \ldots, W_n]$ as output, where, for each $j = 1, 2, \ldots, n$, W_j is the value of the standard aggregate function applied to $[V_1, \ldots, V_{j-1}, V_j]$. Arity 1 primitives are fully explained in Table 2.

The choice between standard and cumulative versions depends on the recording time of the vectorial variables. Some problems in fact wish to predict a sequence of values based on other simultaneous time sequences so that the entries of the target refer to the same time instants as the entries of the predictors. In this case, since standard aggregate functions put together all the values of a sequence, they are not allowed among primitives because target values cannot depend on future predictor values. Therefore only cumulative aggregate functions take part into the primitive set.

Functions of arity 2. Classical arity 2 functions are adapted and extended in order to manage different kinds of inputs. When two vectors are involved, the shortest is completed with the null-element of the considered function, with an exception in case one of the two vectors has length 1, thus it is a scalar. In this last case in fact, to highlight the static nature of scalars, the scalar is replicated up to the length of the vector. To clarify, Table 3 describes in detail arity 2 functions.

Initialization

The new functions of the primitive set are inclined to be frequently misused, especially in the initialization step. In fact, when aggregate functions are too frequently applied on scalars the result is a huge tree full of useless branches. Moreover some trees may have a scalar output when the goal is instead to predict a time series. Therefore, to ensure a pool of proper initial trees as the starting point for the evolutionary process, VE-GP proposes a new initialization strategy:

- $n1$ individuals, during the generation with one of the classical techniques [17], are forced to apply aggregate functions only to vectorial variables;
- $n2$ individuals are generated with one of the classical techniques [17] and checked in their output. If individual t returns a scalar, a vectorial terminal X and an arity 2 function f are randomly selected and the tree t is replaced with the following tree, using post-fix notation, *(f t X)*;
- the remaining $n3$ individuals are generated with one of the classical techniques [17].

The classical generation method is the same for all the three groups of individuals and the subdivision depends on the user.

Fitness evaluation

In order to evaluate the fitness of trees with a scalar output, VE-GP replicates the value up to the length of the corresponding target, transforming the output into a constant vector. To clarify, let T = `V_mean`(V) be the explicit form of a tree where V is a vectorial variable. Let $v = [1.2, 3.5, 4.6, 0.5]$ be the collected value of V for the first fitness case and let $t = [2.3, 4.5, 6.7, 1.8]$ be the corresponding value of the target. The evaluation of T on the first fitness case is therefore 2.45, but we expect a vector of length 4, thus we assume $[2.45, 2.45, 2.45, 2.45]$ as the actual prediction of T on the first fitness case.

Table 4
Parameters used to set VE-GP algorithm.

VE-GP parameters	
Population size	500
Max number of generations	300
Initialization	Ramped Half and Half with rules [17]
Selection method	Lexicographic parsimony pressure [28]
Elitism	Best individual kept
Crossover rate	0.9
Mutation rate	0.1
Max tree depth	17

Table 5
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

Nonetheless, these trees are penalized by multiplying their fitness with a huge constant. This suggest to the algorithm to not select them as they are unlikely to be a good predictive model.

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. Fitness is calculated applying RMSE on linear scaled outputs as described in 4.1 and [18]. Table 4 reports the parameters setting.

6. Experiments: Results and discussion

To estimate how accurately the predictive techniques perform, we adopt a stochastic cross validation approach. We 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 Fig. 1 and Table 5. In Fig. 1 the errors are represented in logarithmic scale.

Table 5 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 6 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 6. Statistical analysis indicates that VE-GP consistently outperforms LR, RF, ST-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.

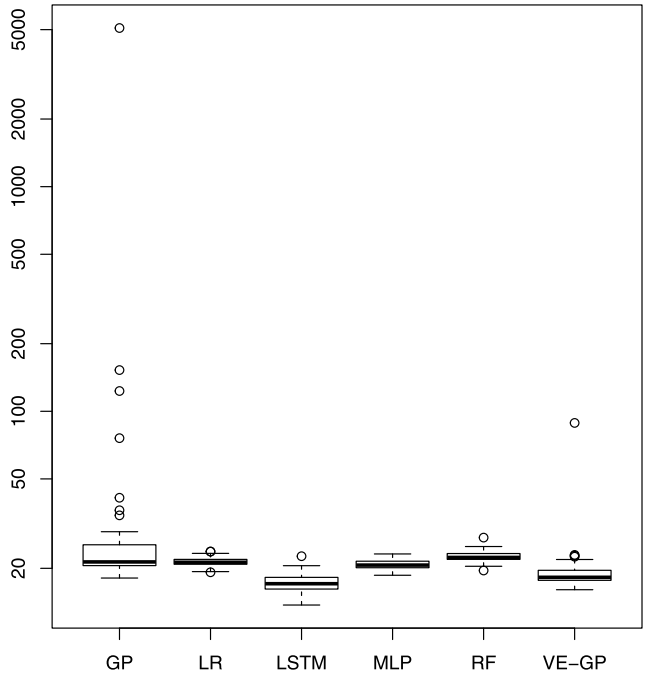


Fig. 1. RMSE on the test set in logarithmic scale for the different algorithms. Ticks values on y-axis refer to the untransformed error values.

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).

Figs. 2, 3, 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.

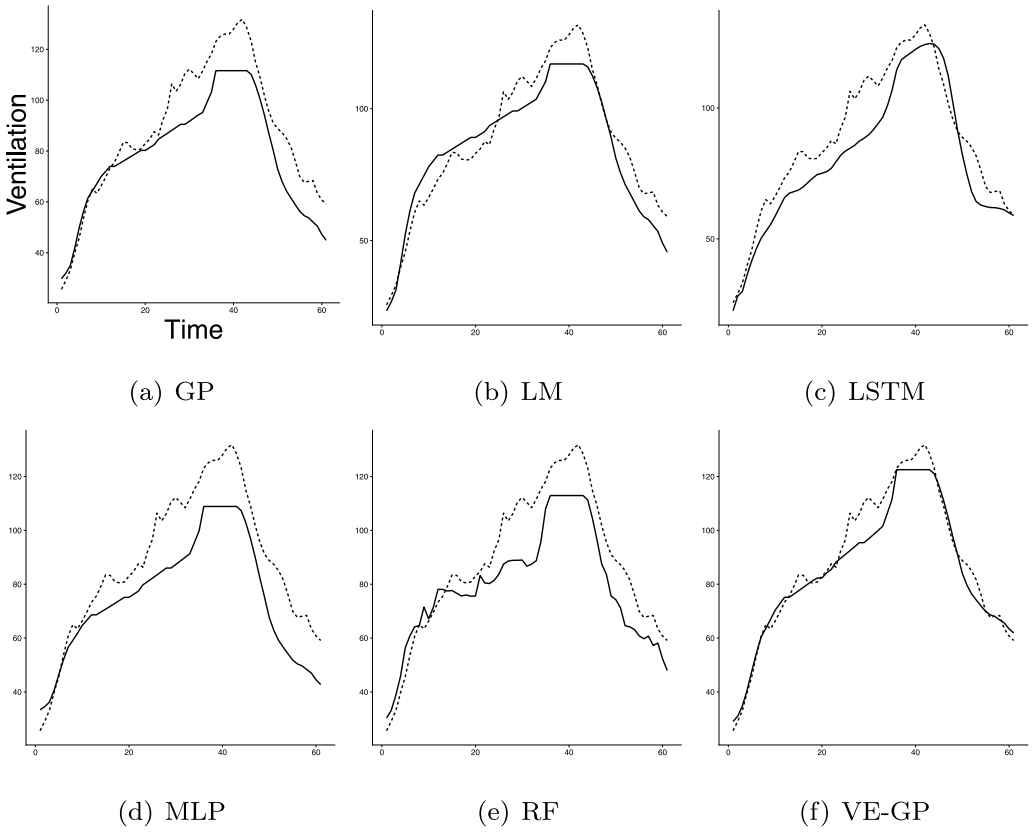
All the methods are good in finding intriguing characteristics of ventilation shape such as peaks and monotony and reveal good capabilities in generalization. Moreover, VE-GP and LSTM shows smooth ventilation series. A possible reason of this behaviour is the fact that they receive the whole heart rate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

In Table 7 we report the training CPU time of a random run of each algorithm.

Clearly there are large differences in times which depend on the software and hardware used for the implementations, and moreover on the fact that the codes are not optimized. These differences could be of importance in context where the training needs to be performed in real time. In this application, however, each ML method has its own learning phase which is surely time consuming, but once done, the resulting predictive model is quickly executable on all unseen data. The running time, furthermore, does not compromise the quality of the solution. In our concern, the quality of the predictive model depends on the accuracy and the generalization ability.

6.1. Analysis of the best solutions

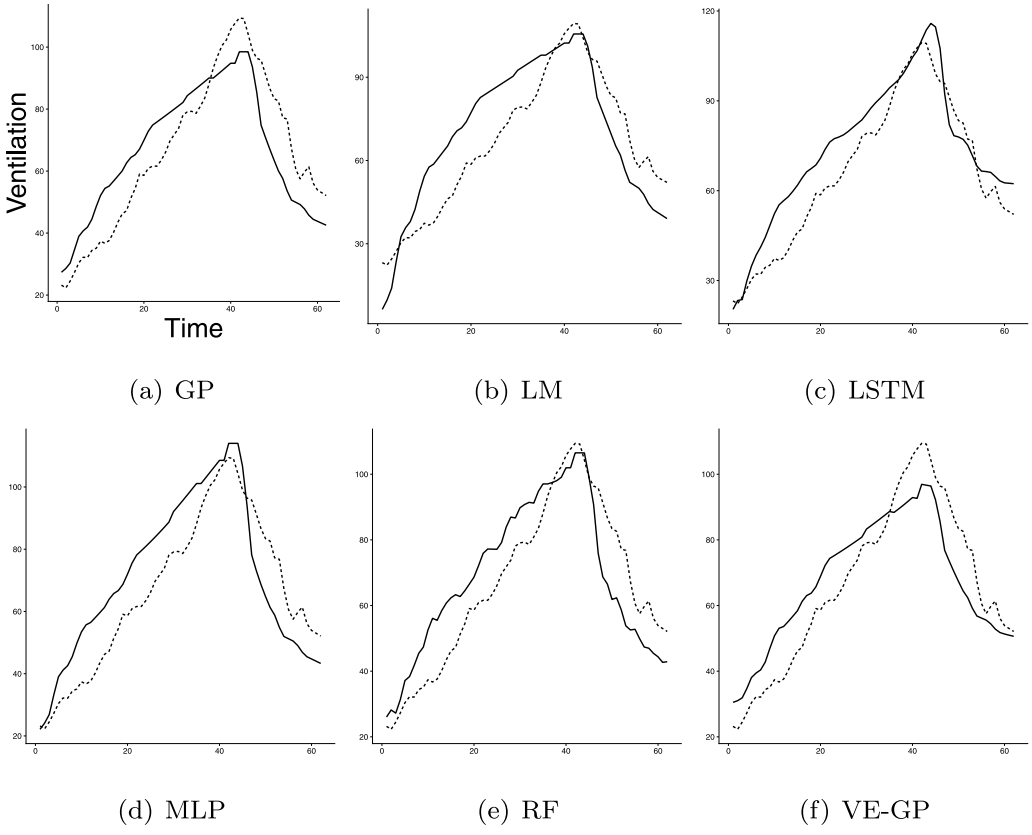
Among all the techniques used in this paper, GP, VE-GP and LR are the most popular for producing interpretable solutions. This property let us further investigate how time series treatment



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

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

Fig. 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.



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

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

Fig. 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.

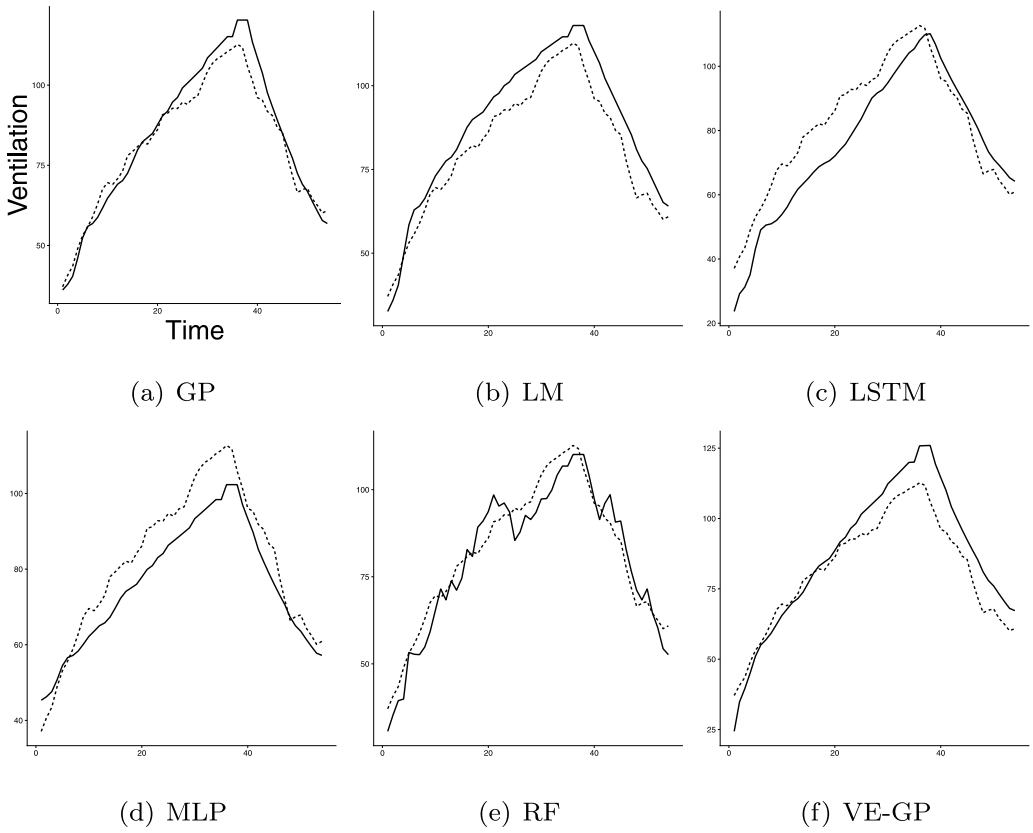


Fig. 4. Predicted vs observed ventilation for a person of age > 50. The dashed line represents the collected values of ventilation, while the other line represents the predicted values of ventilation by the corresponding method.

Table 6
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
$p = 1.3 \cdot 10^{-11}$	$p = 9.5 \cdot 10^{-12}$	$p = 5.8 \cdot 10^{-6}$	$p = 2.0 \cdot 10^{-9}$	$p = 7.3 \cdot 10^{-14}$

Table 7
CPU training times in seconds.

GP	LR	LSTM	MLP	RF	VE-GP
19131.348	0.242	2068.032	3.124	1.264	44020.415

Table 8
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

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 shows the frequencies.

As Table 8 shows, 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. These observations highlight the importance of keeping together ordered sequences which is the key feature of VE-GP.

7. 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 newly introduced vectorial genetic programming (VE-GP) seems suitable for the problem in analysis due to its capability of treating time series as vectors. The goal of this paper is therefore

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 consider 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. Moreover, VE-GP 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.

Declaration of competing interest

No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.asoc.2020.106097.

Acknowledgements

This work was partially supported by FCT, Portugal, through funding of LASIGE Research Unit (UIDB/00408/2020) and projects INTERPHENO (PTDC/ASP-PLA/28726/2017), PERSEIDS (PTDC/EMS-SIS/0642/2014), OPTOX (PTDC/CTA-AMB/30056/2017), BINDER (PTDC/CCI-INF/29168/2017), AICE (DSAIPA/DS/0113/2019), GADgET (DSAIPA/DS/0022/2018), and PREDICT (PTDC/CCI-CIF/29877/2017). This study was also supported by Ministero dell'Istruzione, dell'Università e della Ricerca (MIUR) under the programme "Dipartimenti di Eccellenza ex L.232/2016" to the Department of Veterinary Science, University of Turin.

References

[1] E. Baralis, T. Cerquitelli, A. Giordano, A. Mezzani, D. Susta, X. Xiao, Predicting cardiopulmonary response to incremental exercise test, in: 28th IEEE International Symposium on Computer-Based Medical Systems, CBMS 2015, IEEE, 2015, pp. 135–140.
[2] S. Liu, R. Gao, J. Staudenmayer, P. Freedson, Improved regression models for ventilation estimation based on chest and abdomen movements, Physiol. Meas. 33 (1) (2012) 79–93.
[3] Z. Obermeyer, E.J. Emanuel, Predicting the future — Big data, machine learning, and clinical medicine, New Engl. J. Med. 375 (13) (2016) 1216–1219.
[4] D. Dermofal, Time-series cross-sectional and panel data models, Spat. Anal. Soc. Sci. 32 (2015) 141–157.
[5] I. Azzali, L. Vanneschi, S. Silva, I. Bakurov, M. Giacobini, A vectorial approach to genetic programmig, in: Genetic Programming- 22nd European Conference, EUROGP 2019, in: Lecture Notes in Computer Science, Springer, 2019, pp. 213–227.
[6] C.M. Mermier, M.J. Samet, W.E. Lambert, T.W. Chick, Evaluation of the relationship between heart rate and ventilation for epidemiologic studies, Arch. Environ. Health Int. J. 48 (4) (1993) 263–269.
[7] I.C. Cozza, D.M.T. Zanetta, F.L.A. Fernandes, F.M.M. da Rocha, P.A. de Andre, M.L.B. Garcia, R.B. Paceli, G.F. Prado, M. Terra-Filho, P.H. do Nascimento Saldiva, U. de Paula Santos, An approach to using heart rate monitoring to estimate the ventilation and load of air pollution exposure, Sci. Total Environ. 520 (1) (2015) 160–167.
[8] J. Ben Ali, T. Hamdi, N. Fnaiech, V. Di Costanzo, F. Fnaiech, J.M. Ginoux, Continuous blood glucose level prediction of Type 1 Diabetes based on Artificial Neural Network, Biocybern. Biomed. Eng. 38 (4) (2018) 828–840.
[9] B.Y. Kim, Y.S. Chang, K.S. Park, A nonlinear model for predicting ECG R-R interval variation based on the evolutionary computation approach, in: ICCS'03 Proceedings of the 1st International Conference on Computational Science: PartI, in: Lecture Notes in Computer Science, Springer, 2003, pp. 378–386.
[10] F. van Wick, R. Kamaleswaran, R.L. Davis, A. Khojandi, Physiomarkers in real-time physiological data streams predict adult sepsis onset earlier than clinical practice, 2018, BioArXiv, the preprint server for biology.
[11] Arthur Petrosian, Danil V. Prokhorov, Richard Homan, Richard Dasheiff, Donald C. Wunsch, Recurrent neural network based prediction of epileptic seizures in intra- and extracranial EEG, Neurocomputing 30 (2000) 201–218.
[12] J. Bock, D. Gough, Prediction of physiological state signals in sleep apnoea, IEEE Trans. Bio-Med. Eng. 45 (1998) 1332–1341.
[13] Qingnan Sun, Marko V. Jankovic, Lia Bally, Stavroula G. Mougiakakou, Predicting blood glucose with an LSTM and bi-LSTM based deep neural network, 2018, CoRR, abs/1809.03817.
[14] P. Aspinall, P. Mavros, R. Coyne, J. Roe, The urban brain: analysing outdoor physical activity with mobile EEG, Br. J. Sports Med. 49 (4) (2015) 272–276.
[15] R. Ross, A. Alduishy, C. Gonzáles-Haro, Validation of the cosmed K4b2 portable metabolic system during running outdoors, J. Strength Cond. Res. (2019).
[16] J. Koza, Genetic Programming: On the Programming of Computers by Means of Natural Selection, 1992.
[17] R. Poli, W.B. Langdon, N.F. McPhee, A Field Guide to Genetic Programming, 2008.
[18] M. Keijzer, Improving symbolic regression with interval arithmetic and linear scaling, in: Genetic Programming, EuroGP 2003, in: Lecture Notes in Computer Science, Springer, 2003, pp. 70–82.
[19] S. Silva, GPLAB a genetic programming toolbox for MATLAB, 2007, http://gplab.sourceforge.net/index.html.
[20] L. Breiman, Random forests, Mach. Learn. 45 (2001) 5–32.
[21] T.M. Oshiro, P.S. Perez, J.A. Baranauskas, How many trees in a random forest? Mach. Learn. Data Min. Pattern Recognit. 7376 (2012) 154–168.
[22] S. Haykin S., Neural Networks: A Comprehensive Foundation, 1999.
[23] D. Marquardt, An algorithm for least-squares estimation of nonlinear parameters, SIAM J. Appl. Math. 11 (2) (1963) 431–441.
[24] J. Fox, Applied Regression Analysis and Generalized Linear Models, 2015.
[25] J. Elman, Finding structure in time, Cogn. Sci. 14 (1990) 19–211.
[26] S. Hochreiter, J. Schmidhuber, Long Short-Term memory, Neural Comput. 9 (8) (1997) 1735–1780.
[27] D. Kingma, J. Lei Ba, Adam: A method for stochastic optimization, 2014, arXiv:1412.6980v9.
[28] S. Luke, L. Panait, Lexicographic parsimony pressure, in: GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO'02, Morgan Kaufmann Publishers Inc., 2002, pp. 829–836.