Interpretable Multivariate Time Series Classification through Genetic Programming: CMAGEP-MTSC

Iulia Ilie[1], Luca Costabello[2], Freddy Lecue[3], and Georgiana Ifrim[1]

Insight Centre for Data Analytics, University College Dublin, Dublin, Ireland[1]
Accenture Labs, Dublin, Ireland[2]
Inria Sophia Antipolis, France[3]
iulia.ilie@insight-centre.org, luca.costabello@accenture.com,
freddy.lecue@inria.fr,georgiana.ifrim@insight-centre.org

Abstract. Multivariate Time Series Classification (MTSC) is the task of classifying numeric sequences to a target class label. In MTSC, each example time series has multiple dimensions, for example the data collected from multiple sensors describing the normal or abnormal functioning of a machine. Although the state-of-the-art for univariate Time Series Classification (TSC) has advanced significantly in recent years, the MTSC problem poses many challenges stemming from the dependency between different dimensions, noise associated with possibly redundant dimensions and scalability with increasing number of dimensions. Many of the existing MTSC algorithms are quite complex, and it is difficult to interpret the model and the classification decision. Notable such examples are deep learning approaches which are the state-of-the-art with regard to accuracy. In this paper we propose a genetic programming approach for modelling the MTSC task. Our approach is based on Covariance Matrix Adapted Gene Expression Programming for MTSC (CMAGEP-MTSC). We propose a framework for fitting symbolic regressions for MTSC, where explicit, possibly non-linear, regression functions are fitted to each class. The resulting CMAGEP-MTSC models are human readable mathematical models, allowing for further analysis and interpretation of the model and classification decision. We compare to recent state-of-the-art MTSC algorithms on a range of MTSC datasets, including some with more than 60 dimensions. We show that the proposed CMAGEP models outperform existing approaches with regard to accuracy, having the significant added advantage of interpretability.

Keywords: Multivariate Time Series Classification· Genetic Programming · Symbolic Regression · Interpretation.

1 Introduction

Time series (TS) data are temporally ordered sequences of numbers, while Multivariate Time Series (MTS) are TS with multiple dimensions. For example, where TS would represent a signal measured by one sensor over time, such as a patient's heartbeat, an MTS refers to more complex monitoring systems, with interconnected sensors measuring different aspects of a studied phenomenon at the same time. TS usually refer to numeric sequences, although measuring mixed types of features over time is also possible. In the current work we focus on numeric MTS.

MTS classification (MTSC) is the task of mapping an MTS to a target label. MTSC problems are found in a wide range of real-world domains, such as finance, medical sciences, industry, human activity recognition, etc. In recent years the univariate TSC field has seen tremendous advancement [3], due largely to increased TSC standardized data sets availability that allowed for better benchmarking and field progress assessment. MTSC on the other hand is still developing, with some of the main challenges of the MTSC field stemming from the "dimensionality curse" [16], with difficult to disentangle dependencies between dimensions, noisy MTS signals often caused by redundant dimensions, and scalability issues. The currently proposed methods in the MTSC field focus on improving classification accuracy [22], [13], although the interpretation aspect of the classification decision is often neglected or insufficiently explored. The leading approaches in the MTSC state-of-the-art often include very complex models, such as large ensembles or deep learning models, rendering the model interpretation as well as the interpretation of the classification decision a difficult task. Notably, [13] recently explored the interpretation aspect by explaining the MTSC decision made by deep learning approaches with a post-hoc visualisation technique. We propose a different perspective on the MTSC task and the interpretation aspect based on Genetic Programming (GP, [17]). We explore the potential of a generative modelling framework for MTSC by building automated symbolic regressions (SR, [2][24]) for each MTSC label independently and predicting labels for new MTS based on the regression fit scores of the trained GP models.

The GP system introduced here for MTSC is the Covariance Matrix Adapted Gene Expression Programming (CMAGEP) [11, 12]. CMAGEP is capable of returning short human readable mathematical equations for modelling each class type. It was initially proposed for symbolic regression problems (to predict numeric targets), and here we adapt that approach for a classification setting in MTSC, where we train a GP model for each class. The mathematical equations offer a starting base for the interpretation process. A further application of the learned classification models becomes immediate, since the mathematical equations can be easily deployed to any new problem (and dataset), or could even be considered building blocks to larger problem solving systems once they have been built.

In this paper we explore the potential of using CMAGEP for MTSC by studying accuracy and interpretation on a set of 6 binary MTSC problems from an established MTSC benchmark dataset [5]. We compare the label prediction accuracy with that of 11 state-of-the-art MTSC approaches that include deep learning, TS discretisation and random forests methods. We also include in our study results from linear and generalised linear model regression as generative models to study the need for a system capable of reconstructing non-linear components in describing the relationships across dimensions. We found that over these 6 studied problems CMAGEP-MTSC outperforms the state-of-the art in terms of accuracy score based rankings. We also illustrate and study the mathematical structures independently built by CMAGEP-MTSC for each of the binary classes for two of the problems studied.

The paper is structured as follows: in Section 2 we discuss some of the related work for MTSC and SR, in Section 3 we introduce the CMAGEP-MTSC algorithm, we continue with the experiments and data description in Section 4 and present and discuss results in Section 5.

2 Related work

2.1 MTSC State-of-the-Art

The current state-of-the-art is represented by the work of [23], and by deep learning methods. The work of [23] introduced MUSE+WEASEL, an MTS discretization and feature extraction technique for MTSC and is currently leading the board in terms of prediction accuracy scores over the 20 MTSC problems available in the UEA MTSC benchmark [5]. However, due to the fact that the extracted MTS features that build the classification model come from the frequency domain, and a way to directly map to the initial MTS is not yet proposed, WEASEL+MUSE lacks the interpretation aspect. Naturally, deep learning models also perform well for MTSC as they do in other classification tasks, with the most noteworthy methods here being MLSTMFCN [14] and ResNet[13]. Due to the model complexity, the two also lack direct interpretability.

To tackle the interpretation aspect, Fawaz et. al. [13] recently proposed the CAM method as a visualisation aid for understanding the classification decision making.

Generalized Random Shapelet Forest (gRSF, [15]) were shown to also perform well for MTSC over the 20 benchmark problems, however, once more, due to the complexity and size of the MTSC model, interpretation cannot be easily accomplished.

The established baseline for MTSC is currently considered to be the DTWi [21], having the advantage of speed and ease of deployment over other methods in the field.

2.2 Symbolic Regression

Symbolic Regression is the task of finding an accurate mapping from a dataset to a numeric target outcome. The SR task is different from the standard regression task. Standard linear regression require the optimization of variable weights, or parameters in an a-priori established linear regression. In SR on the other hand, the task is to automatically build the entire regression model from scratch, given a vocabulary of functions and variables. This includes feature selection, feature transformation and parameter optimization. GP systems were shown to perform very well in this task [11]. By building the entire regression based only on signals present in the data, without prior constraints, GP systems perform in fact a reverse engineering of the MTS inter-dimensional dependencies. In this paper we extend the CMAGEP framework introduced in [11] for symbolic regressions, to propose an interpretable GP-based classification algorithm for MTSC.

2.3 GP for MTSC

To the best of our knowledge, there is not much work done in the field of MTSC with GP based systems. There are however GP approaches proposed for TSC, among which, [1] and [25] are notable for their reported accuracy results. These approaches lack however the direct interpretability aspect proposed in our framework. The work shown in [25] uses GP to extract significant patterns for the classification decision, making it more suitable for to a shapelet based classification approach, whereas the GP system in [1] processes the TS serially, returning only a classification label.

3 Method

3.1 CMAGEP

CMAGEP [12] is a hybrid algorithm between a Genetic Programming (GP, [17]) approach for mathematical function discovery and an Evolutionary Strategy (ES, [7]) for local parameter calibration. The GP component of CMAGEP is Gene Expression Programming (GEP) [9], an evolutionary algorithm that, in the present implementation [11] , builds Symbolic Regressions (SR). The ES component of CMAGEP is the Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES, [10]), which performs an evolutionary search for an optimal parameter set to a given regression. Previous work [12], has shown that the inclusion of a dedicated constant calibration step through CMA-ES in the GEP evolution leads to significantly shorter SR solutions without loss of prediction performance. By returning parsimonious solutions, CMAGEP offers a better chance for user interpretation of the mathematical function structures with better chance of understanding the studied system responses.

CMAGEP automatically builds SR by mapping candidate variables and their functional transformations to a target variable over multiple evolution generations [12].

Over the course of a CMAGEP evolution the initial set of randomly generated candidate regressions are:

1. evaluated and selected for fitness
2. optimally calibrated by CMA-ES
3. transformed by genetic operators (mutation, inversion, recombination, etc.)

until a stop condition is reached. Stop conditions include: best possible fitness value, maximum number of generation without improvement of fitness, maximum time allowed for evolution, etc. The full CMAGEP evolution is described in Algorithm 1, and the necessary parameters that need to be set for each run are given in Table 2.

3.2 CMAGEP MTSC

In the MTSC framework our main assumption is that if there would be significant differences in the functioning of a system monitored by a MTS between different classes described by labels, those difference would appear in the mathematical relationships of the MTS dimensions as well. Although a proof has not yet been constructed to support our hypothesis, the initial results shown in this paper indicate that our assumptions seems to indeed hold in practice.

Under this framework, CMAGEP builds mathematical functions mapping a set of $n-1$ dimensions to 1 target for each class independently. Once models are built per class, these are applied to the test MTS, with the model showing a better fit predicting the class label.

In order to cover a larger range of MTSC problems, the initial CMAGEP implementation [12] has been expanded to CMAGEP HD (High Dimensionality, CMAGEP-MTSC), so that it would allow for building regressions with much higher dimensionality. The update comes from a change in the manner in which the encodings of the different regression components are stored in the evolution process, going from string encodings containing a unique character for each component, to C++ Vector type of encoding assigning a Vector element for each regression component. This allowed for a jump from a maximum dimensionality of 24 characters to as many as the machine memory allows (1000+ dimensions).

The proposed CMAGEP-MTSC algorithm is described in Algorithm 1.

3.3 Generating Lags

In order to keep the time aspect of the MTS which would be lost with a direct regression model for the inter-dimension relations, we have decided to add more dimensions to the original MTS built from time lags with a given window size. The lags are generated for all dimensions, including the target, starting from the assumption that if there would be auto-regressive behaviour even for the target, this might influence the assignment of the MTS to a class label or another.

3.4 Data and Experiment Design

In order to asses the accuracy and interpretation performance of CMAGEP for MTSC a set of 6 binary MTSC problems were studied, with the MTS number of dimensions ranging between 2 and 62. The MTSC benchmark contains 20 datasets [5, 22], out of which 6 are binary classification datasets. For the purpose of this study we only focus on these binary datasets, with the multi-class CMAGEP MTSC implementation still being work in progress.

The properties of the selected MTS for performing benchmarking experiments are illustrated in Table 1. The studied problems come from various domains such as health care (ECG), industry (Wafer, and NetFlow) and human activity recognition (CMUSubjects, KickVsPunch, WalkVsRun). The problem features (no. of dimensions, time series size, train/ test sample size) are sufficiently varied as well, allowing for real world extrapolation.

Algorithm 1 CMAGEP-MTSC framework

```
 1: procedure Prepare MTSC data for CMAGEP
 2:     Assign Time Series label to each Time Stamp
 3:     Concatenate all Time Series by dimensions
 4:     Determine dimension importance to label discrimination by Random Forest
 5:     Assign most important dimension as Target for regression
 6:     Assign all remaining dimensions to vector of input Variables for regression
 7:     Split data per class label
 8:     Sample training data for each run per class label
 9: procedure Generate Time Lags
10:     Generate $\lambda$ time step lags for each dimension
11: for Each class label and each train sub-sample do
12:     procedure Evolve symbolic regression
13:         Randomly generate population of $n$ regressions based on CMAGEP parameters
14:         while stop condition is not met do:
15:             - Evaluate each regression in population against target based on fitness value
16:             - Calibrate $k$ best regression by CMA-ES
17:             - Save best regression for next generation
18:             - Select regressions by roulette tournament based on fitness values
19:             - Generate $n$-$1$ regressions for next generation by genetically altering the selected regressions using genetic operators
20:         Return Symbolic Regression
21: procedure Select set of discriminative regressions
22:     for each model in class 1, $m_1$ do
23:         for each model in class 2, $m_2$ do
24:             for each MTS in train set do
25:                 Compute Fitness Value, $f_v$
26:                 if $f_v(m_1) > f_v(m_2)$ then
27:                     - MTS is assigned label 1
28:                 else
29:                     - MTS is assigned label 2
30:             compute model combination accuracy $MA$
31:     Return $max(MA)$
32: procedure Predict Class label
33:     if Model fitness score is larger for class 1 than for class 2 over a time series then
34:         - MTS is assigned label 1
35:     else
36:         - MTS is assigned label 2
```

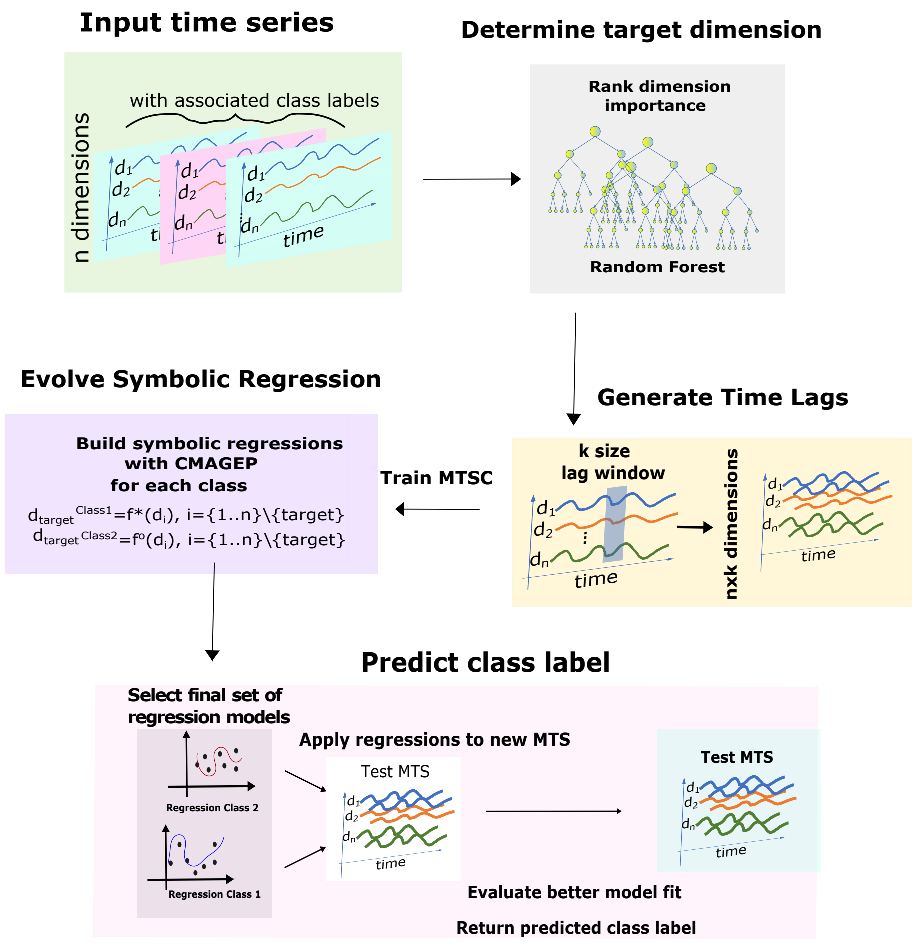
Input time series
with associated class labels
n dimensions
d1
d2
dn
time
Determine target dimension
Rank dimension importance
Random Forest
Evolve Symbolic Regression
Build symbolic regressions with CMAGEP for each class
d_{target}^{Class1}=f*(d_i), i={1..n}\{target}
d_{target}^{Class2}=f°(d_i), i={1..n}\{target}
Train MTSC
Generate Time Lags
k size lag window
nxk dimensions
Predict class label
Select final set of regression models
Regression Class 2
Regression Class 1
Apply regressions to new MTS
Test MTS
Evaluate better model fit
Return predicted class label

Fig. 1. CMAGEP for MTSC work flow.

One of the main critiques of this benchmark set can be that the included datasets have single splits between train and test samples, meaning that all reported results could be very well biased by sampling method. Nevertheless, for benchmarking MTSC methods the sampling bias should not be a strong concern, since the bias would then affect all method runs and results.

The small sub-set of studied MTSC problems could mean that the present results might not transfer easily to a larger set of problems, however we believe that for an initial exploration of the CMAGEP-MTSC framework capabilities, the CMAGEP accuracy and interpretation results would be sufficiently illuminating.

Table 1. Properties of 6 binary classification multivariate time series datasets described in [5, 22]

Problem	No. Classes	Dimensions	TS length	Train	Test
ECG	2	2	39-152	100	100
NetFlow	2	4	50-997	803	534
Wafer	2	6	104-198	298	896
KickvsPunch	2	62	274-841	16	10
CMUsubject16	2	62	127-580	29	29
WalkvsRun	2	62	128-1918	28	16

.

For all 6 problems in Table 1, 10 independent CMAGEP runs were performed for each class to avoid initial seed bias as described in Algorithm 1, with parameter settings given in Table 2. This means that for each problem, at the end of all CMAGEP runs we are left with 20 models, 10 per class. Of the 10 models per class, in order to generate class label prediction, 100 combinations of models were evaluated in terms of class label prediction accuracy over all MTS in the train set. The combination of 2 models with the highest label prediction accuracy was selected and used to predict class labels on the test MTS samples. The test label prediction accuracy is reported in Tables 3 and 4

Additionally, in order to study the effects of Min-Max Standardizations of the MTSC dimensions over class prediction accuracy, MTSC experiments were performed with the original data as found in the Baydogan repository [5] as well as with data transformed by standardizing values over all dimensions the 0-1 interval. The test label prediction accuracy are reported in Table 3.

In order to asses the CMAGEP performance in the context of the MTSC state-of-the-art approaches, the results of the best performing CMAGEP models by accuracy per problem were added to Table 4. The Table 4 also contains results of linear regression and glmnet models, applied for MTSC in the CMAGEP-MTSC framework, by replacing the CMAGEP symbolic regression component. The linear regressions were studied in order to asses the need for symbolic regressions with capacity to describe non-linear relations of the MTS dimensions in MTSC.

In the present study, the fitness measure is reported in terms of the Nash–Sutcliffe modelling efficiency (MEF) coefficient [20, 6] which is often used in the

Table 2. CMAGEP settings

Parameter	
Data sample size	1000
Lag window size	5
Number of chromosomes	250
Number of genes	3
Head length	6
Functions	$+, -, /, *, x^y, \sqrt{}, \ln, \exp$
Terminals	$x_1 ... x_n$, n MTS dimension number
Link function	$+$
Max run time	600 seconds
Fitness function	MEF
Selection method for replication	tournament[8]
Mutation probability	0.2
IS and RIS transpositions probabilities	0.05
Two-point recombination probability	0.3
Inversion probability	0.05
One point recombination probability	0.4
Number of individuals to optimize	20
Time to start optimization	0
Maximum CMA-ES iterations	50

context of quantifying the performance of physical models [19], [18]. The MEF is computed as

$$\mathrm{MEF} = 1 - \frac{\sum_{i=1}^{n} (o_i - p_i)^2}{\sum_{i=1}^{n} (o_i - \bar{o})^2} \tag{1}$$

where o_i is the observed value at time step i, p_i is the predicted value at step i and $\bar{o}$ is the mean of observed values. MEF values range between $-\infty$ and 1, where an MEF value of 1 corresponds to the case where the predicted and observed values are identical. A negative MEF value means that the predictions are worse than the mean of the observations in recreating the observed signal. MEF=0 indicates that the models prediction are as good as a prediction by $\bar{o}$.

3.5 Code and Data Availability

All code and data used to perform the aforementioned experiments can be found in a git-lab repository [1].

[1] `https://gitlab.insight-centre.org/iulia.ilie/emcl-2019-cmagep-mtsc.git`

4 Evaluation

4.1 CMAGEP version Accuracy

We start the study by comparing the two CMAGEP implementations over the studied problems, with CMAGEP HD as expected having better coverage of MTSC problems (see. Tab. 3). Likely due to the allowance of larger dimensional space, CMAGEP HD shows better performance for label prediction accuracy than CMAGEP when compared over the Non-Normalized data. The story is repeated as well on the Normalized data sets.

Table 3 illustrates as well the effects of applying normalisation over all dimensions, with the learning process seemingly greatly supported by the data transformation and the label prediction accuracy significantly improving after re-scaling. These findings differ from those of [4], where the accuracies of the DTW approaches don't seem to be improved by data transformations, on the contrary, with the rankings of the normalised runs being the lowest.

Table 3. Effects of normalization on the 2 versions of CMAGEP as evaluated over 6 datasets for MTSC after 10 independent runs per each class type with settings given in Table 2.

Dataset	CMAGEP	Norm CMAGEP	CMAGEP HD	Norm CMAGEP HD
ECG	0.67	0.98	0.78	0.98
NetFlow	0.782	0.99	0.765	0.996
Wafer	0.895	0.98	0.91	0.998
CMUsubject16	NaN	NaN	1	1
KickvsPunch	NaN	NaN	0.8	0.86
WalkvsRun	NaN	NaN	1	1

4.2 CMAGEP in the context of MTSC SOTA : Accuracy

We compared accuracy scores of CMAGEP-MTSC with those of 11 other state-of-the-art MTSC approaches as reported in the work of [22] and that of [13] and illustrate the results in Table 4 and Fig. 4.2. Both Table 4 and Fig. 4.2 show that the CMAGEP for MTSC outperforms the current state-of-the-art in terms of average rank per problem by label prediction accuracy scores, with the models learned by CMAGEP performing best, with MUSE+WEASEL and gRSF following.

Due to the low sample size of the data sets on which the benchmark is done however, from Fig. 4.2 we cannot state that the differences in accuracy scores are significantly better than those of the first group, but they do seem to be significantly outperforming the last group formed by the DTWi, the linear regressions and surprisingly one of the Deep Learning methods, ResNet.

Table 4. Accuracy results for 6 MTSC problems computed for 13 machine learning approaches. CMAGEP results come after 10 independent runs per class type with settings given in Table 2 and after a selection of the optimal set of 2 models for discrimination.

Dataset	ECG	NetFlow	Wafer	CMUsubject16	KickvsPunch	WalkvsRun
SMTS	0.818	0.977	0.965	0.997	0.82	1
LPS	0.82	0.968	0.962	1	0.9	1
mvARF	0.785	NaN	0.931	1	0.976	1
DTWi	0.79	0.976	0.974	0.93	0.6	1
ARKernel	0.82	NaN	0.968	1	0.927	1
gRSF	0.88	0.914	0.992	1	1	1
MLSTMFCN	0.87	0.95	0.99	1	0.9	1
FCN	0.872	0.89	0.982	1	0.54	1
ResNet	0.867	0.627	0.989	0.997	0.51	1
MUSE	0.88	0.961	0.997	1	1	1
LR	0.74	0.72	0.6	0.55	0.4	0.75
GLM	0.77	0.73	0.6	0.55	0.4	0.75
CMAGEP	0.98	0.996	0.998	1	0.86	1

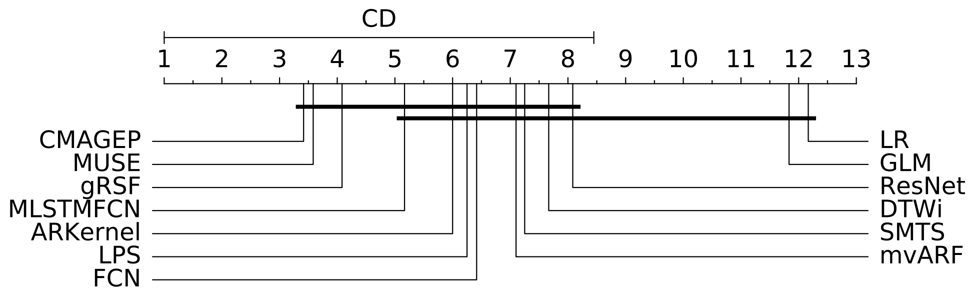

Fig. 2. Critical diagram showing the average accuracy ranks for 13 studied MTSC methods over 6 binary MTSC problems.The lower the rank, the better.

4.3 Number of Used Lags

From our experiments it seems that the window size of lag generation, practically, the number of time steps up to which CMAGEP can look behind when building the models is a very important parameter to set. Here we have observed that if no lags are included, the class prediction performance is very low, it increases with increasing lag window size, however, when a higher number of lags is generated, the prediction performance drops again. We believe that this could a problem of feature selection, since with increasing lag window, the dimensions of the MTS increase considerably as well, although according to literature , GP approaches are not sensitive to high dimensional spaces and are capable of performing relevant automatic feature selection. This is a very interesting aspect of the CMAGEP-MTSC framework that would need further investigation.

4.4 Interpretation of CMAGEP Model Structures

In the following section we illustrate mathematical functions automatically built for individual binary classes by CMAGEP. The two mathematical models generated for each problem were applied to test time series for predicting the validation class label. To reduce verbosity, all other structures from presented experiments are given in the repository.

ECG. The ECG problem is a MTSC application where the requirement is to classify the function of a patient's heartbeat as normal or abnormal. The heart function is monitored by 2 electrodes receiving signals after being placed in the upper and lower area of the chest. The data sets contain half-hour ECG recordings of normal and abnormal heartbeat signals.

After applying the CMAGEP-MTSC framework, two models were returned, one for each class.

$$s_2(t)_{an} = 0.96s_1(t-1) + 0.17s_2(t-3)^{0.97} \times (0.71s_2(t-3) - 1.1) + 0.082 \quad (2)$$

$$s_2(t)_n = 2.2exp(0.35s_1(t-1)) + 0.02s_2(t-3) - 2.1 \quad (3)$$

Where $s_1(t)$, $s_2(t)$ are the measured signals by the two sensors at time stamp t, with $s_2(t)_{an}$ and $s_2(t)_n$ denoting the signal measured in the abnormal and normal classes.

The CMAGEP models learned for the two individual classes have different mathematical structures, indicating that the two regression models do in fact show different responses for the two studied cases.

However, as the authors lack the expert knowledge necessary to understand if the described responses make sense from a physical and biological point of view, we could not draw clear conclusions regarding interpretation beyond the initial assessment of the structure aspects of the mathematical models.

According to the dataset description the upper signal will frequently show normal heartbeats, whereas normal beats are frequently difficult to discern in the lower signal, with abnormal beats often more prominent.

Wafer.

$$s_1(t)_{c1} = 0.66 s_3(t) - 0.24 s_4(t) + 0.72 s_2(t-1) s_4(t-3) \tag{4}$$

$$s_1(t)_{c2} = 1.1 s_3(t) s_5(t-3) - 0.22 s_1(t-3) s_5(t-3) \tag{5}$$

Although CMAGEP returns mathematical models, it is quite apparent that many of them are complex, and might pose difficulty for direct interpretation. In order to tackle these issues, conditions could be added in the fitness function used for training.

4.5 Non-linearity

It is worth mentioning that the best models for MTSC for the illustrated cases contain non-linear components as well as interactions between the selected variables, supporting the hypotheses that linear regressions are not sufficient to capture the differences in system states described by the studied MTS per individual class label, which was also confirmed by the label prediction accuracy scores. This is a very interesting finding and we believe it needs further investigation, although without expert knowledge it would be difficult to understand if these non-linearities are artefacts or do in fact represent real processes.

5 CMAGEP Run-times and Memory Requirements

The main disadvantage of the CMAGEP approach for MTSC is the long runtime needed to reach a sufficiently well performing solution. Even more, the runtime was shown to scale fast with increase in data. We tackle these points here by running independent evolutions in parallel on sub-samples of the train split, and putting pressure on non-evolving populations, with lack of change in fitness after 50 generation being a stop criterion. With these regression models were returned under 10 minutes. For MTSC, under these conditions, because multiple models generating per class label, a selection of the final regression model pair is needed. The selection process is quite slow as well, and scales fast with number of models as well as with number of sample MTS in the test split. For the present experiments the selection process was also in the order of minutes, always under 10 minutes.

The evolutions of CMAGEP under the current set-up have not needed more than 50 Mb of memory space. The main memory requirements for CMAGEP will come with the data storage during the evolution as well as for the population of evolution individuals. If the set of regressions to be evolved is set as much more than 250, then the space required would also naturally increase.

The CMAGEP models are mathematical functions that can be saved as strings, meaning that they always occupy space in the range of Kb, making them easy to deploy or to store.

6 Conclusion

In this paper we have proposed a genetic programming approach for modelling the MTSC task. Our approach is based on Covariance Matrix Adapted Gene Expression Programming for MTSC (CMAGEP-MTSC). We have proposed a framework for fitting symbolic regressions for MTSC, where explicit, possibly non-linear, regression functions are fitted to each class. The resulting CMAGEP-MTSC models are human readable mathematical models, allowing for further analysis and interpretation of the model and classification decision. We have compared to recent state-of-the-art MTSC algorithms on a range of MTSC datasets, including some with more than 60 dimensions. We have shown that the proposed CMAGEP models outperform existing approaches with regard to accuracy, having the significant added advantage of interpretability. Our proposed algorithm CMAGEP-MTSC produced models that both performed well in terms of class label prediction accuracy, delivering accuracy comparable to complex state-of-the-art MTSC approaches, as well as having the advantage of returning mathematical functions for further study and interpretation.

References

1. Alfaro-Cid, E., Sharman, K., Esparcia-Alcázar, A.I.: Genetic Programming and Serial Processing for Time Series Classification. Evolutionary Computation **22**(2), 265–285 (jun 2014). https://doi.org/10.1162/EVCOa00110, `http://www.ncbi.nlm.nih.gov/pubmed/24032750http://www.mitpressjournals.org/doi/10.1162/EVCO{\_}a{\_}00110`
2. Augusto, D., Barbosa, H.: Symbolic regression via genetic programming. In: Proceedings. Vol.1. Sixth Brazilian Symposium on Neural Networks. pp. 173–178. No. diagram C, IEEE Comput. Soc (2000). https://doi.org/10.1109/SBRN.2000.889734, `http://ieeexplore.ieee.org/document/889734/`
3. Bagnall, A., Bostrom, A., Large, J., Lines, J.: The Great Time Series Classification Bake Off: An Experimental Evaluation of Recently Proposed Algorithms. Extended Version (feb 2016), `http://arxiv.org/abs/1602.01711`
4. Bagnall, A., Dau, H.A., Lines, J., Flynn, M., Large, J., Bostrom, A., Southam, P., Keogh, E.: The UEA multivariate time series classification archive, 2018 pp. 1–36 (2018), `http://arxiv.org/abs/1811.00075`
5. Baydogan, M.G., Runger, G.: Learning a symbolic representation for multivariate time series classification. Data Mining and Knowledge Discovery **29**(2), 400–422 (mar 2015). https://doi.org/10.1007/s10618-014-0349-y, `http://link.springer.com/10.1007/s10618-014-0349-y`
6. Bennett, N.D., Croke, B.F., Jakeman, A.J., Newham, L.T.H., Norton, J.P.: Performance evaluation of environmental models. 2010 International Congress

on Environmental Modelling and Software Modelling for Environment's Sake pp. 1–9 (2010), http://www.iemss.org/iemss2010/papers/S20/S.20.01.Performanceassessmentofenvironmentalmodels-ANTHONYJAKEMAN.pdf
7. Beyer, H.G., Schwefel, H.P.: Evolution strategies – A comprehensive introduction. Natural Computing **1**(1), 3–52 (2002). https://doi.org/10.1023/A:1015059928466, http://link.springer.com/10.1023/A:1015059928466
8. Coello, C.A., Montes, E.M.: Constraint-handling in genetic algorithms through the use of dominance-based tournament selection. Advanced Engineering Informatics **16**(3), 193–203 (2002). https://doi.org/10.1016/S1474-0346(02)00011-3, http://www.sciencedirect.com/science/article/pii/S1474034602000113
9. Ferreira, C.: Gene Expression Programming in Problem Solving. In: WSC6. pp. 1–9. No. 1992 (2002)
10. Hansen, N., Müller, S.D., Koumoutsakos, P.: Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). Evolutionary computation **11**(1), 1–18 (jan 2003). https://doi.org/10.1162/106365603321828970, http://www.ncbi.nlm.nih.gov/pubmed/12804094
11. Ilie, I., Dittrich, P., Carvalhais, N., Jung, M., Heinemeyer, A., Migliavacca, M., Morison, J., Sippel, S., Subke, J.A., Wilkinson, M., Mahecha, D.: Reverse engineering model structures for soil and ecosystem respiration: The potential of gene expression programming. Geoscientific Model Development **10**(9) (2017). https://doi.org/10.5194/gmd-10-3519-2017
12. Ilie, I., Mahecha, M., Jung, M., Carvalhais, N., Dittrich, P.: Evolving compact symbolic expressions by a GEP CMA-ES hybrid approach (under review)
13. Ismail Fawaz, H., Forestier, G., Weber, J., Idoumghar, L., Muller, P.A.: Deep learning for time series classification: a review. Data Mining and Knowledge Discovery pp. 1–44 (2019). https://doi.org/10.1007/s10618-019-00619-1
14. Karim, F., Majumdar, S., Darabi, H., Harford, S.: Multivariate LSTM-FCNs for Time Series Classification (jan 2018), http://arxiv.org/abs/1801.04503
15. Karlsson, I., Papapetrou, P., Boström, H.: Generalized random shapelet forests. Data Mining and Knowledge Discovery **30**(5), 1053–1085 (sep 2016). https://doi.org/10.1007/s10618-016-0473-y, http://link.springer.com/10.1007/s10618-016-0473-y
16. Keogh, E., Mueen, A.: Curse of Dimensionality. In: Encyclopedia of Machine Learning and Data Mining, pp. 314–315. Springer US, Boston, MA (2017). https://doi.org/10.1007/978-1-4899-7687-1192, http://link.springer.com/10.1007/978-1-4899-7687-192
17. Koza, J.R.: Genetic programming as a means for programming computers by natural selection. Statistics and Computing **4**(2), 87–112 (1994). https://doi.org/10.1007/BF00175355
18. Migliavacca, M., Reichstein, M., Richardson, A.D., Mahecha, M.D., Cremonese, E., Delpierre, N., Galvagno, M., Law, B.E., Wohlfahrt, G., Andrew Black, T., Carvalhais, N., Ceccherini, G., Chen, J., Gobron, N., Koffi, E., William Munger, J., Perez-Priego, O., Robustelli, M., Tomelleri, E., Cescatti, A.: Influence of physiological phenology on the seasonal pattern of ecosystem respiration in deciduous forests. Global Change Biology pp. 363–376 (2015). https://doi.org/10.1111/gcb.12671
19. Mitchell, S., Beven, K., Freer, J.: Multiple sources of predictive uncertainty in modeled estimates of net ecosystem CO2 exchange. Ecological Modelling **220**(23), 3259–3270 (dec 2009). https://doi.org/10.1016/j.ecolmodel.2009.08.021, http://www.sciencedirect.com/science/article/pii/S0304380009006000

20. Nash, J., Sutcliffe, J.: River flow forecasting through conceptual models part I — A discussion of principles. Journal of Hydrology **10**(3), 282–290 (apr 1970). https://doi.org/10.1016/0022-1694(70)90255-6, http://www.sciencedirect.com/science/article/pii/0022169470902556
21. Rakthanmanon, T., Campana, B., Mueen, A., Batista, G., Westover, B., Zhu, Q., Zakaria, J., Keogh, E.: Data Mining a Trillion Time Series Subsequences Under Dynamic Time Warping. Tech. rep., https://www.ijcai.org/Proceedings/13/Papers/454.pdf
22. Schäfer, P., Leser, U.: Multivariate Time Series Classification with WEASEL+MUSE. In: Proceedings of ACM Conference. p. 11. Washington, DC, USA, (nov 2016), http://arxiv.org/abs/1711.11343
23. Schäfer, P., Leser, U.: Multivariate Time Series Classification with WEASEL+MUSE (nov 2017), http://arxiv.org/abs/1711.11343
24. Smits, G.F., Kotanchek, M.: Pareto-Front Exploitation in Symbolic Regression. In: Genetic Programming Theory and Practice II, pp. 283–299. Springer-Verlag, New York (2005). https://doi.org/10.1007/0-387-23254-017, http://link.springer.com/10.1007/0-387-23254-017
25. Xie, F., Song, A., Ciesielski, V.: Learning Time Series Patterns by Genetic Programming. In: Australasian Computer Science Conference (ACSC. Melbourne, Australia (2012), https://pdfs.semanticscholar.org/d624/100c4b2ff9298f51df4569879bda2c694d74.pdf