

Evolving Event Detectors in Multi-channel Sensor Data

Feng Xie, Andy Song, and Vic Ciesielski

RMIT University, Melbourne, VIC 3001, Australia
{feng.xie,andy.song,vic.ciesielski}@rmit.edu.au
<http://www.rmit.edu.au/compsci>

Abstract. Detecting events of interest in sensor data is crucial in many areas such as medical monitoring by body sensors. Current methods often require prior domain knowledge to be available. Moreover, it is difficult for them to find complex temporal patterns existing in multi-channel data. To overcome these drawbacks, we propose a Genetic Programming (GP) based event detection methodology which can directly take raw multi-channel data as input. By applying it to three event detection tasks with various event sizes and comparing with four typical classification methods, we can see that those detectors evolved by GP can handle raw data much better than other methods. With features manually defined based on domain knowledge, our method can also be comparable with others. The analysis of evolved detectors demonstrates that distinctive characteristics of the target events are captured by these GP detectors.

1 Introduction

In a dynamic environment sensors continuously produce data when monitoring objects. The sequences of data can be viewed as time series. As the state of an object changes, certain variations would occur in this temporal space. For instance, heart failure may be reflected as a sudden decrease of blood pressure, a feeling of fear may be shown due to intensive muscle contractions. In reality multiple channels of sensors are often used, each being a time series of its own. For example in the field of Activity Recognition(AR), human locomotion, e.g. walking, sitting and standing, and gestures, e.g. drinking water and closing a door, can be identified by the fluctuations in a group of sensor readings. These temporal variations of interest can be viewed as events of which the duration is the event size. Detection of these events in time series input from sensors is obviously important in many areas, and it is also the goal of this study.

The interest in detecting patterns from sequential recordings has been increasing in the past few years. A common approach is to extract distinctive features of an event and then to “query” matching subsequences from incoming input stream. In domains that the features are well established, this approach is widely adopted due to its effectiveness and simplicity in implementation [17,18]. However when the underlying knowledge of an event remains unknown, finding good features for that event would be a challenge. Furthermore features suitable

for one problem may not be appropriate for another problem since the knowledge is often domain dependent. There are no universal features.

In order to eliminate the requirements of priori knowledge, some researchers proposed to construct mathematical models based on original data. The objective then becomes detecting any significant changes of the model or the parameters of that model, therefore, it is also known as change detection problem [13,14]. This approach is highly effective for fields in which significant changes are unusual, for example astronomy. These change point detection techniques tend to use relatively simple models, for example linear models. It would be very difficult for such techniques to handle complex events which consist of not just a single changing point but a series of changing points. Moreover, this approach mainly aims at solving univariate time series problems and consequently not suitable for events occurring in multiple channels, for example a single sensor can hardly contain the information for recognizing a gesture.

To address the issues in these existing works, we propose a GP-based method that is capable of detecting events directly from raw time series data. The event can be simple as a change between two adjacent points or complex as variation in multiple variables during a period of time. The data we investigated in this study is time series stream. Each time step is labelled positive or negative indicating whether the event of interest occurred or not. Therefore, it is reasonable to use a supervised learning method. This approach seems similar to two-class time series classification [16]. However in time series classification the data has to be segmented into non-overlapped, fixed-length vectors in advance. On the contrary, our method receives continuous streaming input so it has to determine at which point an event occurs. The particular research questions are addressed here:

1. How can a suitable GP representation be established so event detectors for multi-sensor input can be evolved when features are not available?
2. How does the performance of GP working on raw data compare to typical classification algorithms when features are available?
3. How would GP-evolved programs detect events? Are there any insight to reveal their behaviours?

The performance of the proposed method is evaluated on three tasks, including both synthetic and real world data sets. Additionally, J48 decision tree classifier, Naïve Bayes classifier, IB1 nearest neighbour classifier and AdaBoost meta classifier are used for comparison due to their popularity in the area of classification.

2 Related Work

GP has been proven to be a powerful problem solving mechanism in many domains, especially in those where human knowledge is limited [1,2,3]. It is not surprising that GP has a long history of being adopted to handle time series data. A great number of works are devoted to prediction based on past observations [4,6,7], but few studies are on time series detection.

Previous work has shown that GP is capable of working on raw data, even in sophisticated domains such as Computer Vision, for example, texture segmentation [19], video shot detection [8,22] and motion detection [5].

Another related area is novelty detection [15] which is to find unseen time sequences. It uses a sliding window to examine through all the possible subsequences to identify the most distinctive ones from the rest of time series. It has some overlaps with our work since it essentially considers an abnormality as an event. Unfortunately, all novelty detection algorithms require parameters to be known beforehand, at least the length of a subsequence. Furthermore, only very limited of them are working on multi-channel time series.

3 GP Representations

We use tree-based GP in our study which requires a primary set of nodes including functions and terminals. Table 1 presents functions in GP representation. Apart from four basic mathematical operations, three extra functions are introduced to handle multi-channel sensor data. The two functions proposed for exploring temporal relationships are discussed in Section 3.1 and Section 3.2. Section 3.3 describes how to deal with multiple variables in time series. Terminals are listed in Table 2, each row presenting one terminal including its name, value ranges, return type and the specified functions it can be attached to. It should be noted that each terminal “Variable m ” corresponds to one variable in time series, representing one channel input. The value of m is the index of that channel, which is within the range from 0 to total channel numbers minus 1.

Table 1. Function Set of GP

Function	Parameters	Return Type
+	Double, Double	Double
+	Double, Double	Double
*	Double, Double	Double
/	Double, Double	Double
Window	Double i, Int operation, Int temporal-index	Double
Temporal_Diff	Double i	Double
Multi-Variable	Int multivariable-operation, Int variable-index	Double

3.1 Window Function

A function named “Window” is defined to cover and analyse a collection of points as it moves along the stream data or data stream passes through it. The window size S determines the maximum number of data points within that window.

Table 2. Terminal Set

Terminal	Value	Return Type	Function
Variable m	Current value of channel m	Double	General
Random	A random value	Double	General
Operation	AVG,STD,DIF,SKEWNESS	Integer	Window
Temporal Index	$[1, 2^{window-size} - 1]$	Integer	Window
Variable Index	$[1, 2^{num-of-variables} - 1]$	Integer	Multi-Variable
Multivariable Operation	MED,AVG,STD,RANGE	Integer	Multi-Variable

This “Window” function takes three parameters. The first one is “ i ”, the input of this function. Either a function or a terminal can be attached here as long as it returns a double value. The latest historical values of this input is stored in the function, denoted as $t_0, t_1, \dots, t_{S-1}$ from the earliest value to the current one. The value of S is set to 8 in this study.

The second parameter is “**Temporal Index**” which returns a random integer falling in the range from 1 to $2^S - 1$. The binary equivalent of this decimal number is mapped to the window, one bit to one data point. A data point will be selected only when its corresponding bit is “1”. Assume the value of “**temporal index**” is 11, of which the equivalent binary string is 00001011, it will select points t_4, t_6 and t_7 since bits at these three positions are “1”. Similarly, a value of 255 will take in all 8 points in that window. This mapping mechanism enables flexible point selection within the window and consequently helps to find the event size.

The third parameter “**operation**” randomly applies one operation on values selected by the second parameter. It is assigned to an integer value from 1 to 4 inclusively. Each value corresponds one of four operations: **AVG**, **STD**, **DIF** and **SKEWNESS**. These operations calculate the average value, the standard deviation, sum of absolute differences and skewness of the selected points respectively.

3.2 Temporal_Diff Function

It is not difficult to see that temporal changes between two consecutive points are important to characterise an event. Function “**Temporal_Diff**” is proposed to capture such temporal information. It takes one parameter as its input and returns the difference between the last value and the current value, therefore it essentially behaves as a sliding window of size 2.

3.3 Functions for Multiple-variable Time Series

“Window” function is capable to handle the temporal relationships in one sequence of values, however, it is infeasible to capture events occurring in multiple variables. To address this issue, function “Multi-Variable” is introduced to select arbitrary variables and summarise characteristics of these variables. It takes two parameters: **variable index** and **multiple-variable operation**. The first parameter works similarly as **temporal index** in “Window” function, the only difference is the range of **variable index** is within the range of $[1, 2^M - 1]$ (M is

the total number of variables). The value of second parameter can be any integer value from 1 to 4, corresponding to four multiple-variable operations: MED, AVG, STD and RANGE which return the middle value, the average, the standard derivation, and the distance between the maximum and minimum values of the selected variables.

4 Experiments

As shown in Table 3, a standard GP setting was used in the experiments. Moreover, dynamic range selection [20] was adopted to help GP process classification problems. We applied GP on three event detection tasks including both synthetic and real world data, which are described in Section 4.1. We are interested in the best solution that can be found. Therefore, for each task, GP was run for ten times and the best result among these runs was then considered as the final outcome. In Section 4.2, We compare the performance of GP to four traditional classifiers.

Table 3. GP Runtime Parameters

Population	30
Generation	300
Maximum Depth	8
Minimum Depth	2
Mutation Rate	5%
Crossover Rate	85%
Elitism Rate	10%
Number of Runs	10

4.1 Data Set

One simulated and two real world data sets were investigated in this study. All of these problems are with multiple variables and varying event sizes. Table 4 presents numbers of positive instances, numbers of negative instances in both training and testing phases. As shown in this table, the number of instances used for training is twice than the number used for test. In the first data set the number of positives and negatives are close while the last two data sets are unbalanced.

Synthetic Data: Changes in Any Four out of Six. In many real world situations, it is expected that sensor data will be smooth and any temporal change beyond a normal range should raise an alarm. The task is to simulate such situation, of which the objective is to detect noticeable changes in a six-variable time series. A noticeable change occurs when the absolute difference between two adjacent points is larger than 5. The current time point is considered as positive when such a change just occurred in any four variables simultaneously. Therefore,

Table 4. Training and Testing Data of Three Data Sets(%)

Data Set	Training			Test		
	total	positives	negatives	total	positives	negatives
1. Synthetic Data: Changes in Any Four out of Six	400	206	194	200	108	92
2.Activity Recognition: Walking	11368	1322	10046	5647	855	4792
3.Activity Recognition: Lying Still	11368	229	11139	5647	147	5500

concurrent changes happened in more than four variables are positive as well. Otherwise, it should be negative. Since two consecutive time points defines the event, the time span of this event is 2. To produce the artificial time stream, first each variable is initialised with a random value. Then, six values for next time point are generated randomly. For each variable, the probability of a notice change occurring at next time stamp is 0.9.

Activity Recognition: Walking and Lying Still. As mentioned in Section 1, human locomotion can be detected in wearable sensor data. The data we used in experiment comes from OPPORTUNITY data set [23]. The data set contains recordings of four healthy subjects, two male and two female; for each subject, six daily activity sessions were recorded. Over one hundred sensors were located on the jacket, limbs and feet of the subjects. Four locomotion, including walking, standing, lying still and sitting, are manually labelled by at least two researchers and further validated by video-recordings. Beyond these four classes, there is a none class indicating any other locomotions, for example, running.

There are two tasks for this data set: walking detection and lying still detection. To simplify the problem, it is sensible that only data obtained from accelerometers located on feet are considered since movements of both feet are enough to distinguish the two actions from others. Finally, six channels are investigated in this study. The original data is sampled every 33 milliseconds. Considering that walking and lying are high level locomotion, a lower sample rate is suitable. Therefore, we take data every 0.1 second as inputs. It should be noted that neither cross-session nor cross-subject validation are studied in this research. We only use the data of the first session of one subject, two third for training and the rest for testing. It should be noted that the event size and optimal features are unknown in these two tasks.

4.2 Results

Four conventional classifiers were applied to each task. The first three are *IB1* [11], *Naïve Bayes* [10] and *J48* [9]. The best one among these classifiers was then selected as the base classifier for the ensemble classifier *AdaBoost* [12]. Since conventional classifiers have no built-in sliding window mechanism, we manually segment raw data according to the event size. Since in Task 2 and Task 3 the event sizes are not available, we use 8, which is the max window size of GP, to

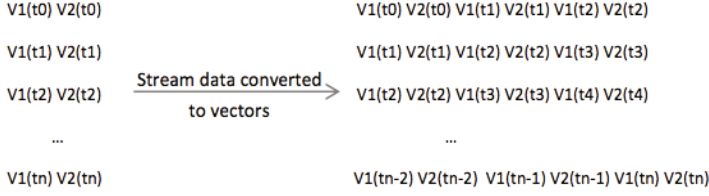


Fig. 1. Converting Raw Data Streams to Vectors for Conventional Classifiers

ensure that traditional classifiers receive the same amount of information as GP. Figure 1 illustrates the process of converting a two-variable time series stream into a six-attribute vectors, assuming an event size of 3.

Another group of experiments were conducted by applying the traditional classifiers to manually construct features. The features for all three data sets are the differences between two consecutive points of all variables. Given the same conditions in Figure 1, the extracted features can be shown in Figure 2.

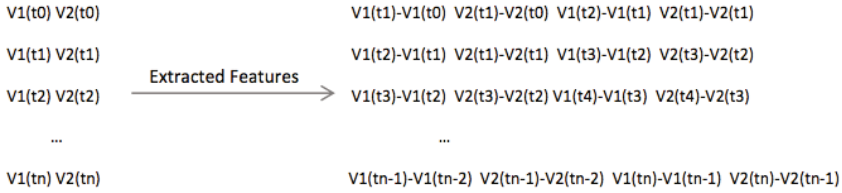


Fig. 2. Extracting Features for Conventional Classifiers

The results are presented in Table 5 and Table 6 which show the accuracies, true positive rates and true negative rates from test as test results are much more meaningful. With pre-defined features GP was slightly less effective but still comparable to other methods on Task 1. However, in the other two tasks GP can achieve better performance. As shown in Table 6, if the manually constructed features were not available, traditional classifiers suffered badly. On Task 1, J48 and Naïve Bayes classified all instances to be positive while IB1 and AdaBoost were not much better than random guessing (around 50%). In lying still detection task, traditional classifiers seemed to achieve similar accuracy as GP. However, it was because of the imbalance of the data that J48 and Naïve Bayes could appear accurate. The true positive rate were as low as 68.7% and 8.2% respectively. Only IB1 and Adaboost performed as good as GP. On Task 3, GP achieved the best performance as well. It should be noted here, in walking detection task, traditional classifiers gained benefits from features while in lying still detection task it was the contrary. With given features, classical classifiers can not successfully identify any positive instances except Naïve Bayes of which,

unfortunately, the overall accuracy was only around 31%. The same set of features giving opposite results in two tasks further proves that an appropriate feature set are highly problem dependent.

Table 5. Comparing with Conventional Methods on Pre-defined Features(%)

Tasks	J48	Naïve Bayes	IB1	AdaBoost	GP
1. Changes in Any Four out of Six	89.95 TP : 91.7 TN : 87.9	92.46 TP : 90.7 TN : 94.5	97.99 TP : 100 TN : 95.6	97.49 TP : 99.1 TN : 95.6	95.34 TP : 95.28 TN : 95.4
2. Lying Still	97.4 TP : 0 TN : 100	31.27 TP : 83.7 TN : 29.9	96.97 TP : 0.7 TN : 99.5	96.97 TP : 0 TN : 99.5	99.72 TP : 99.32 TN : 99.73
3. Walking	88.05 TP : 48.7 TN : 95.1	86.08 TP : 51.3 TN : 92.3	89.08 TP : 53.1 TN : 95.5	88.16 TP : 53.2 TN : 94.4	90.49 TP : 52.28 TN : 97.31

Table 6. Comparing with Conventional Methods on Raw Data(%)

Tasks	J48	Naïve Bayes	IB1	AdaBoost	GP
1. Changes in Any Four out of Six	54.27 TP : 100 TN : 0	54.27 TP : 100 TN : 0	53.27 TP : 63 TN : 41.8	52.76 TP : 65.7 TN : 37.4	95.34 TP : 95.28 TN : 95.4
2. Lying Still	98.81 TP : 68.7 TN : 99.6	97.43 TP : 8.2 TN : 99.8	99.79 TP : 94.6 TN : 99.9	99.79 TP : 94.6 TN : 99.9	99.72 TP : 99.32 TN : 99.73
3. Walking	78.6 TP : 52.3 TN : 83.3	55.89 TP : 75.2 TN : 52.4	70.82 TP : 53.9 TN : 73.8	60.1 TP : 53 TN : 61.4	90.49 TP : 52.28 TN : 97.31

5 Discussion and Analysis

Some of the evolved detectors have been analysed to give us more understanding of the problem and of GP itself. Based on this analysis we discuss the aspects of GP on building reusable blocks, reducing dimensionality, and deciding event size.

It has been found that GP tend to build common blocks which can be reused in other parts of the same tree. We give two program trees as examples. The best individuals in Task 1 and Task 3 are shown in Figure 3 and Figure 4 respectively. The identical sub-trees are enclosed by dashed lines in the figures. The values of these repeated components are actually important to characterise events; therefore they can be considered as genuine features for those events.

In Task 1, the variable index 63 can be converted to binary string “111111”, meaning that all variables should be taken into account. It is sensible since the

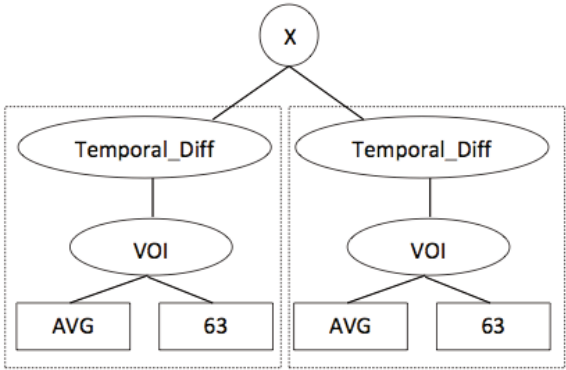


Fig. 3. Evolved Program(Task1)

changes may happen in sensor 1,2,3 and 4 as well as in sensor 1,3,5 and 6 or any combination of four. So no single sensor shall be ignored.

On detecting walking event, GP program took only 2 sensors among all six ones, the fourth and the sixth. However J48 classifier required all sensor inputs. Although only two channels were selected the performance of GP did not deteriorate. To verify the significance of these two channels of input, we applied feature extraction only on these two channels and fed the features to the traditional classifiers. The results are shown in Table 7. Comparing to all six features included, these two features still provided reasonable performance. It verifies that GP did find important inputs and was selective. It did not simply use all channels as other methods did.

Table 7. Conventional Methods on Dimensionality-Reduced Data of Task 3(%)

	J48		Naïve Bayes		IB1		AdaBoost	
Num of Channels	6	2	6	2	6	2	6	2
Accuracy	88.05	86.97	86.08	86.81	89.08	84.83	88.16	84.7
TP Rate	48.7	35.6	51.3	36.3	53.1	41.1	53.2	43.2
TN Rate	95.1	96.1	92.3	95.8	95.5	92.6	94.4	92.1

In addition, GP tries to identify the event duration as well. For example in Task 1, it is obvious that the event size of 2 was successfully captured by temporal function “Temporal_Diff” (see Figure 3). While in Task 3, “Window” functions were nested, suggesting data points more than the 8-point limit of a single window are needed (see Figure 4). GP found a way to go beyond that limit.

GP shows its flexibility on the choice of window size. To verify that, we applied GP to each task with different window size: 4,8,12 and 16. The results show that the performance deteriorated with a window size smaller than 8 while window

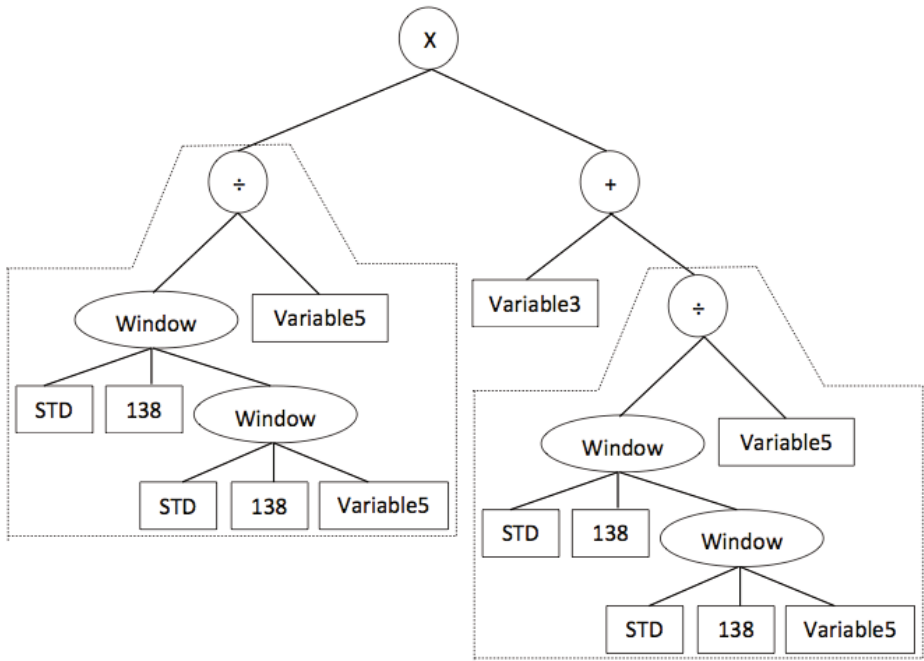


Fig. 4. Evolved Program(Task3)

sizes larger than 8 did not bring significant benefits in accuracy. GP is capable to find reasonable solutions with these window sizes. However, how to find the optimal window size is still an open question.

6 Conclusions and Future Work

In this study, we present a GP-based methodology for detecting event of interest in multi-channel sensor data. A number of specialised representations are introduced to process multi-variable time series. The method was applied to a range of detection tasks from changes in synthetic data to activity recognition problem to verify its performance. The experimental results show that on manually constructed features GP can achieve comparable or even better performance when comparing to the four traditional classifiers. Moreover when manually defined features are unavailable, GP is far superior to the others. The analysis on evolved event detectors demonstrates that GP successfully selected and extracted distinctive characteristics of the target events. On the synthetic data which is the only case that we know the event size, GP evolved programs determined the event size accurately.

At current stage, although intuitive analysis gives us some insights about the behaviour of GP detectors, however, they can not be fully understood. On the other hand, explanation of these models can be meaningful to the domain.

Therefore, in near future, we will further analyse the evolved programs. Another limitation of our work is that for window containing event boundaries, GP probably will make wrong decisions. In the next stage, we will try to address that issue. Additionally, as mentioned above, the system can be enhanced by enabling GP to search for the optimal window size in future.

References

1. Poli, R., Langdon, W., McPhee, N.: A Field Guide to Genetic Programming. Lulu.com (2008)
2. Koza, J.R.: Genetic programming I: On the programming of computers by means of natural selection. MIT Press (1996)
3. Lohn, J., Hornby, G., Linden, D.: An Evolved Antenna for Deployment on Nasa's Space Technology 5 Mission. In: Genetic Programming Theory and Practice II, pp. 301–315. Springer (2004)
4. Kaboudan, M.A.: Spatiotemporal Forecasting of Housing Price By Use of Genetic Programming. In: The 16th Annual Meeting of the Association of Global Business (2004)
5. Song, A., Pinto, B.: Study of GP representations for motion detection with unstable background. In: IEEE Congress on Evolutionary Computation (2010)
6. Hetland, M., Strom, P.: Temporal Rule Discovery using Genetic Programming and Specialised Hardware. In: Proceedings of the 4th International Conference on Recent Advances in Soft Computing (2002)
7. Wagner, N., Michalewicz, Z.: An analysis of adaptive windowing for time series forecasting in dynamic environments: further tests of the DyFor GP model. In: Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation, pp. 1657–1664 (2008)
8. Song, A., Xie, F.: Evolving automatic frame splitters. In: Proceedings of the 13th Annual Conference Companion on Genetic and Evolutionary Computation (2011)
9. Quinlan, R.: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers (1993)
10. George, H.J., Pat, L.: Estimating Continuous Distributions in Bayesian Classifiers. In: Eleventh Conference on Uncertainty in Artificial Intelligence, pp. 338–345 (1995)
11. Aha, D., Kibler, D.: Instance-based learning algorithms. *Machine Learning* 6, 37–66 (1991)
12. Freund, Y., Robert, E.S.: Experiments with a new boosting algorithm. In: Thirteenth International Conference on Machine Learning, pp. 148–156 (1996)
13. Guralnik, V., Srivastava, J.: Event detection from time series data. In: Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge, pp. 33–42 (1999)
14. Preston, D., Protopapas, P., Brodley, C.: Event discovery in time series. In: Proceedings of the SIAM International Conference on Data Mining, pp. 61–72 (2009)
15. Keogh, E., Lin, J., Fu, A.: Hot sax: Efficiently finding the most unusual time series subsequence. In: Fifth IEEE International Conference on Data Mining (2005)
16. Ratanamahatana, C.A., Lin, J., Gunopulos, D., Keogh, E., Vlachos, M., Das, G.: Mining time series data. In: *Data Mining and Knowledge Discovery Handbook* 2010, pp. 1049–1077 (2010)

17. Hunter, J., McIntosh, N.: Knowledge-Based Event Detection in Complex Time Series Data. In: Horn, W., Shahar, Y., Lindberg, G., Andreassen, S., Wyatt, J.C. (eds.) AIMDM 1999. LNCS (LNAI), vol. 1620, pp. 271–280. Springer, Heidelberg (1999)
18. Shahabi, C., Yan, D.: Real-time pattern isolation and recognition over immersive sensor data streams. In: Proceedings of The 9th Intl Conference on Multi-Media Modeling, pp. 93–113 (2003)
19. Song, A., Ciesielski, V.: Texture Segmentation by Genetic Programming. In: Evolutionary Computation, pp. 461–481 (2008)
20. Loveard, T., Ciesielski, V.: Representing classification problems in genetic programming. In: Proceedings of the 2001 Congress on Evolutionary Computation, pp. 1070–1077 (2001)
21. Wagner, N., Michalewicz, Z., Khouja, M., McGregor, R.R.: Time Series Forecasting for Dynamic Environments: The DyFor Genetic Program Model. *IEEE Transactions on Evolutionary Computation* 11(4), 433–452 (2007)
22. Xie, F., Song, A., Ciesielski, V.: Event detection in time series by genetic programming. In: IEEE World Congress on Computational Intelligence, pp. 2507–2514 (2012)
23. Roggen, D., Calatroni, A., Rossi, M., Holleczech, T., Trster, G., Lukowicz, P., Pirkel, G., Bannach, D., Ferscha, A., Doppler, J., Holzmann, C., Kurz, M., Holl, G., Chavarriaga, R., Sagha, H., Bayati, H., del R. Milln, J.: Collecting complex activity data sets in highly rich networked sensor environments. In: Seventh International Conference on Networked Sensing Systems (2010)