Learning Patterns of States in Time Series by Genetic Programming

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. A state in time series can be referred as a certain signal pattern occurring consistently for a long period of time. Learning such a pattern can be useful in automatic identification of the time series state for tasks like activity recognition. In this study we showcase the capability of our GP-based time series analysis method on learning different types of states from multi-channel stream input. This evolutionary learning method can handle relatively complex scenarios using only raw inputs requiring no features. The method performed very well on both artificial time series and real world human activity data. It can be competitive comparing with classical learning methods on features.

Keywords: Genetic Programming, Pattern Recognition, Time Series.

1 Introduction

Time series pattern refers to certain regularities in time series that may be of user interests. There are in general two types of time series patterns. One is event patterns which indicate the occurrence of an event, for example a heart beat on EEG readings. Another type is patterns of states which indicate the time series reading stabilizing during a relatively short period of time.

The main three differences between *state* or an *event* are that:

1. Occurrence: A *state* is usually a reflection of a stable condition, for example, a person being standing or sitting. An *event* usually happens when transiting from one condition to another, for example, a person sitting down (changing from standing to sitting).
2. Data Characteristic: A *state* may show a certain form of repetition of segments over a time period, for example, a person can be in a walking state with the repetition of leg movement. The data is often homogenous. On contrary, an *event* is heterogeneous along the time axis. Figure 1 demonstrates the transition from standing to sitting. The 3 regions divided by 2 dotted grey line show the subject be standing, sitting down and be sitting sequentially. We can see that in both two states, the readings are similar but it is not the case in transitions.

G. Dick et al. (Eds.): SEAL 2014, LNCS 8886, pp. 371–382, 2014.

3. Detection: The detection mechanism would be different as the occurrence of an event can not be decided before the completion of the event while that is not the case for detecting a state. An event usually has a certain duration while a state may last indefinitely.

The boundary between *state* or an *event* are however subtle. As shown in Fig 2, the short walking period is composed of 4 steps. Each step can be viewed as an event. The repetition of such "step" events forms the a walking state.

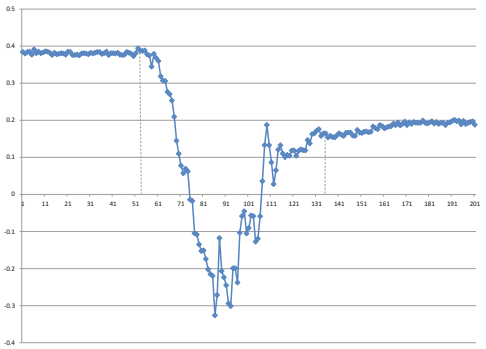


Fig. 1. A Person transiting from Standing to Sitting

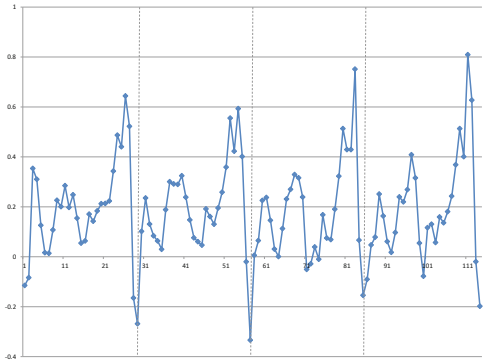


Fig. 2. Illustration of a walking state (4 steps)

A drawback of existing works on classifying time series patterns including events and states is that they often require to know the pattern size in advance [10, 8, 5, 6]. In case of state detection, the pattern size refers to the *state length*, the minimum period that a state can be existing. [1] Such information is not

[1] State length will be used with *pattern size* interchangeably through this paper.

always available. Moreover, a suitable set of features has to be defined for each particular task, which makes the solution usually domain-dependent. In addition, a great number of techniques cannot work well on multi-channel time series.

Given the above complexity, a method that can automatically search the state length and extract useful features is highly beneficial. In [12], Genetic Programming (GP) has been shown its effectiveness in solving event detection problem for raw, multi-channel time streams. We therefore propose to use this GP based method for state detection problems. In particular, the three research questions we are addressing in this paper are:

1. Is the GP-based event detection method applicable to state detection problems?
2. How does GP perform on a range of multivariate synthetic state detection tasks?
3. How does GP perform when applied to real scenario e.g. Activity Recognition tasks?

2 Related Work

There are mainly two categories of methods for classifying time series patterns: 1) Similarity-based techniques and 2) Features-based techniques. In the first category, the class of a time series segment can be determined by its similarity between segments from all classes. Nearest Neighbour, a typical distance-based classifier is the most popular similarity-based time series classifier [10]. Another popular choice is decision tree which uses similarity measure for the partition of trees [8]. The key factor affecting the performance of such classifiers is the effectiveness of that similarity measure [13]. The commonly used measures include Euclidean Distance [7, 3] and Dynamic Time Warping (DTW) [1]. Similarity-based methods assume that a time series pattern always appears similarly which may not be true. Feature-based methods carry out classification based on time series features. However feature extraction may be very time consuming and are often highly problem-specific [5, 6, 9, 4, 2].

Methods of both categories mentioned above have to know the pattern size beforehand and use it to define the window size for sampling segments. Our approach is different as it does not require such information. Moreover, the majority of aforementioned methods are designed for single channel time series. Patterns over several parallel time series are very difficult to be captured by those methods, because redundant or irrelevant channels have to be ruled out from decision process and the dependencies between relevant channels are sometimes complex. Our proposed method can handle multiple channel of time series.

3 A GP-Based State Detection Methodology

In this section, we present our methodology which is based on Genetic Programming. The description mainly focuses on the function set which includes

the window function, temporal difference function and multi-channel function. These functions are internal nodes on a GP constructed program tree which in this case a classifier. The higher classification accuracy the better fitness the tree will receive.

3.1 Window Function

The Window Function defines the incoming sequential inputs, selects data points inside the window, and applies the operations on the select data points. It takes three parameters: `i`, `temporal index` and `operation`.

The first parameter (i.e. i) is the input of this function which samples a data point at every time step. It keeps the subsequence of historical values of that input in memory. The length of this subsequence is called "Window" function size (denoted as S), which is manually adjustable. The reserved data points are marked from the earliest point to the most recent one as $t_0, t_1, ..., t_{S-1}$. The value of S is set as 8 in this study. Greater values are not used so that the evolved programs can be less complex for analysis. Moreover, this value does not deteriorate the performance.

The second parameter is from terminal "`temporal index`" which returns an random integer within the range of $[1, 2^S - 1]$. First the integer is converted into its binary form. In case that the binary is shorter than S and not sufficient to mark all elements in the subsequence, it will be left padded with 0. For example, assuming S is 8 and the parameter value is 5 then the binary string should be 00000101, in which the first five 0s come from padding. This binary is then mapped to the subsequence of time series data under the window. A bit with "1" indicated the data point with the same index will be selected while a bit with "0" will be discarded.

The third parameter (i.e. operation) is a randomly generated integer valued from a range $[1, 4]$. Each value corresponds to one of the four operations: AVG, STD, DIF and SKEWNESS. They are used for calculating the average, the standard deviation, the sum of absolute differences and the skewness of the selected points under the window. The return value is the final output of the Window Function.

3.2 Temporal-Difference Function

Temporal-Difference Function (noted as *Temporal_Diff*) is introduced to capture temporal change between adjacent points as it is obviously important for identifying the occurrence of events.

It only takes one double value parameter i which defines the input. It stores the value t_{i-1} which is one time stamp earlier and returns the difference between t_{i-1} and the current value t_i. It consequently can be considered to have an effective window size of 2. Eventually, it calculates the first derivative of the time series, as temporal changes can be more revealing. Higher order derivatives can be achieved as well if this function is used iteratively.

3.3 Multi-channel Function

The two functions mentioned earlier only handle the temporal dependence, that is, they only work on a sequence along time axis by themselves. They can hardly be aware of any relationship cross channels at a particular time point. Consequently, a state occurring in multiple channels would not be captured by those functions. To address this problem, Multi-Channel Function is introduced. The function selects arbitrary collection of channels and computes characteristics of these channels. It takes two integers as its parameters: `channel index` and `channel operation`. No input parameter needs to be specified as the whole set of channels are treated as input. The parameter `channel index` works in a similar way as to `temporal index` in the Window Function, except its range is from 1 to $2^M - 1$, where M is the total number of channels. So assuming the channel number is 6 in total, a binary form of 13, 001101, would tell the function to operate on the 3rd, the 4th and the 6th channels. The parameter for channel operation also returns an integer from 1 to 4, which corresponds to the following functions: median value which is the middle value of the selected variables (MED), their average (AVG), their standard derivation (STD) and the distance between the maximum and minimum values (RANGE).

The Window Function can be integrated with Multi-Channel Function by taken the output of the latter as input data. Such combination enables GP to find both temporal relationships and variable dependence simultaneously.

4 Synthetic State Detection

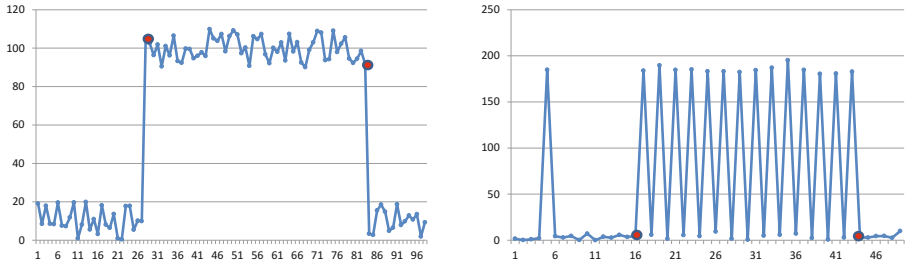
In this session, we introduce six synthetic data sets with increasing complexity. They are used to verify the capability of the proposed method for states detection. These data sets vary in the state size and the number of channels.

4.1 Single-Channel Time Series

In single-channel time series tasks, there is only one channel involved in the time series. The time series data and the tasks are explained in following.

Box Functions. The task is to identify a state of signal at certain level. An example is shown in Figure 4.1. The starting point and the end point of a state are marked with red dots. It is the same in all other graphs in this section. This stimulates voltage or temperature maintaining at a certain level with minor fluctuations.

Oscillation. In a range of applications, constant oscillation may be viewed as a certain state, such as vibration of a spring, which may indicate the normal working condition of the spring. In this time series, a state is defined as consecutive peaks of which the top value is above 180 and the bottom value is bellow 10 (shown in Figure 4.1). Note that the state should last at least for a period of p samples ($p = 4$) given each sample taking 12 time points.

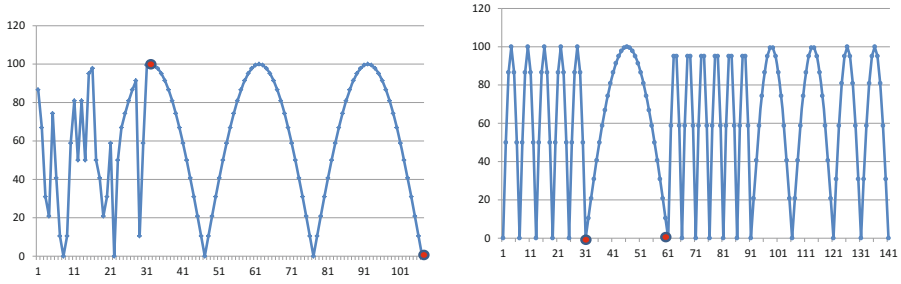


(a) An Example of Box Function (b) An Example of Oscillation

Fig. 3. Two Non-periodical Synthetic States

Sine Wave vs. Random Numbers. A state can be not just a constant value. It can also be a regular signal. This task is to distinguish a signal being generated through a periodical function versus at random. The regulated signal is produced $y = |100 * sin(x)|$ which is sampled at every $\frac{\pi}{30}$. We define the state size as 8. An example is given in Figure 4.1.

Sine Waves. The positive state is defined the same as last problem. The negatives are however consisted of other sine waves, instead of random numbers. These variants are generated by several similar periodical function $y = |100 * sin(x * f)|$ (where $f = 2, 3, 4, 5, 6$), sampled at the same rate as target function (shown if Figure 4.1). The aim is to investigate whether our method can discriminate similar regularities. The state size is also set as 8 for this task.



(a) Sine Wave VS. Random Numbers (b) Sine Wave VS. Other Sine Waves

Fig. 4. Two Periodical Synthetic States

4.2 Multi-channel Time Series

In the following two tasks, there are more than one channels in the time series. The time series data and the tasks are explained in following.

Sine Wave in Two Channels. The sine wave is again $y = 100 * sin(x)$, sampled at every $\frac{\pi}{7}$. However, in this task, positives are only when time series in both channels are sine waves. If one channel is random numbers the state will be considered as negative. As shown in Figure 5, only the middle section is considered positive.

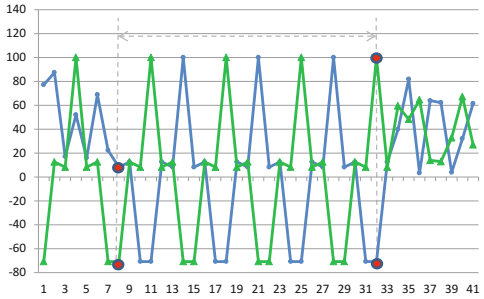


Fig. 5. An Example of Two-Channel Sine Waves

Box Functions in Four Out of Five Channels. The time series is in the positive state if there are more than four channels receiving signal value above 90 for at least 8 points. There is no constrain that at which channels the high reading may occur.

5 Experiments and Results

GP was applied on the six synthetic tasks described in Section 4. For comparison purposes five non-GP classifiers were also applied on those tasks, including *OneR*, *J48*, *Naive Bayes* and *IB1*. In addition AdaBoost was used to combine multiple classifiers as an ensemble to boost accuracy. For each task, the best conventional classifier from the four was selected as the base classifier in AdaBoost. The experimental settings for GP and Non-GP classifiers are shown in Section 5.1 and Section 5.2. The experimental results are presented in Section 5.3

5.1 The Experimental Settings for GP

The GP runtime parameter setting for synthetic data sets is: Population (300), Generation (50), Maximum Depth (8), Minimum Depth (2), Mutation Rate (5), Crossover Rate (85), Elitism Rate (10) and Window Function Size (8). In activity recognition task, a larger population size of 1000 and a greater Window Function Size of 12 are used due to the complexity of the problem. Each run is repeated 10 times and the best run is taken as GP's result.

5.2 The Experimental Settings for Non-GP Classifiers

The time series streams are converted into a list of segments as inputs for non-GP classifiers. For each tasks, the segments are extracted by a sliding window of which the size equals to the state length. This ensures that each segment contains sufficient amount of information while redundant information is eliminated. The segments containing raw data can be used as inputs to classifiers directly. We call such a segment raw input vector. Alternatively, a set of features can be extracted based on each raw input vector and called feature set. We use both types of inputs for non-GP classifiers. The processes of obtaining these inputs are demonstrated in following by an example where the time series has 2 channels and the pattern size is 3.

Raw Input Vector: A. A sliding window is moving through time series to extract raw input vectors. For multi-channel time series, all the readings are flattened into one row just like representing a matrix in a one-dimensional array as shown in Figure 6.

Feature Set B: Wave Length. This feature is uniquely designed for sine functions, which is calculated by equation $\sum_{i=1}^{3} |t_i - t_{i-1}|$. This features is not effected by the phases of the sine wave. Therefore the feature at any time point should have the identical feature values, hence a good feature for finding a state of waves.

Feature Set C: Temporal Average and Variance. The feature set provides the average and standard deviation over the length of a pattern. So the size of this feature set is the double of the number of channels.

Feature Set D: Channel Average. This feature takes averages calculations on different points at each single channel. This feature set enumerate the average value of all channels at each time point. The number of features should be equal to the pattern size.

Table 1 summarizes, for each task, the state length, the numbers of attributes in raw input vector, the type of feature set used in that task, and the numbers of attributes in the feature set.

Stream Data	Converted Vectors
$C_0(t_0)$ $C_1(t_0)$	$C_0(t_0), C_0(t_1), C_0(t_2), C_1(t_0), C_1(t_1), C_1(t_2)$
$C_0(t_1)$ $C_1(t_1)$	$C_0(t_1), C_0(t_2), C_0(t_3), C_1(t_1), C_1(t_2), C_1(t_3)$
$C_0(t_2)$ $C_1(t_2)$	$C_0(t_2), C_0(t_3), C_0(t_4), C_1(t_2), C_1(t_3), C_1(t_4)$
...	...
$C_0(t_n)$ $C_1(t_n)$	$C_0(t_{n-2}), C_0(t_{n-1}), C_0(t_n), C_1(t_{n-2}), C_1(t_{n-1}), C_1(t_n)$

Fig. 6. An Example showing converting a Two-Channel Time Series Stream To Raw Data Vectors for Conventional Classifiers (Pattern Size: 3)

Stream Data	Feature Vector B
$C_0(t_0)$ $C_1(t_0)$	$\|(C_0(t_1) - C_0(t_0)) + (C_0(t_2) - C_0(t_1))\|$, $\|(C_1(t_1) - C_1(t_0)) + (C_1(t_2) - C_1(t_1))\|$
$C_0(t_1)$ $C_1(t_1)$	$\|(C_0(t_2) - C_0(t_1)) + (C_0(t_3) - C_0(t_2))\|$, $\|(C_1(t_2) - C_1(t_1)) + (C_1(t_3) - C_1(t_2))\|$
$C_0(t_2)$ $C_1(t_2)$	$\|(C_0(t_3) - C_0(t_2)) + (C_0(t_4) - C_0(t_3))\|$, $\|(C_1(t_3) - C_1(t_2)) + (C_1(t_4) - C_1(t_3))\|$
...	...
$C_0(t_n)$ $C_1(t_n)$	$\|(C_0(t_{n-1}) - C_0(t_{n-2})) + (C_0(t_n) - C_0(t_{n-1}))\|$, $\|(C_1(t_{n-1}) - C_1(t_{n-2})) + (C_1(t_n) - C_1(t_{n-1}))\|$

Fig. 7. Illustration of Extraction Feature Type B (Pattern Size: 3)

Stream Data	Feature Vector C
$C_0(t_0)$ $C_1(t_0)$	$Average\{C_0(t_0), C_0(t_1), C_0(t_2)\}$, $Average\{C_1(t_0), C_1(t_1), C_1(t_2)\}$, $STD\{C_0(t_0), C_0(t_1), C_0(t_2)\}$, $STD\{C_1(t_0), C_1(t_1), C_1(t_2)\}$
$C_0(t_1)$ $C_1(t_1)$	$Average\{C_0(t_1), C_0(t_2), C_0(t_3)\}$, $Average\{C_1(t_1), C_1(t_2), C_1(t_3)\}$, $STD\{C_0(t_1), C_0(t_2), C_0(t_3)\}$, $STD\{C_1(t_1), C_1(t_2), C_1(t_3)\}$
$C_0(t_2)$ $C_1(t_2)$	$Average\{C_0(t_2), C_0(t_3), C_0(t_4)\}$, $Average\{C_1(t_2), C_1(t_3), C_1(t_4)\}$, $STD\{C_0(t_2), C_0(t_3), C_0(t_4)\}$, $STD\{C_1(t_2), C_1(t_3), C_1(t_4)\}$
...	...
$C_0(t_n)$ $C_1(t_n)$	$Average\{C_0(t_{n-2}), C_0(t_{n-1}), C_0(t_n)\}$, $Average\{C_1(t_{n-2}), C_1(t_{n-1}), C_1(t_n)\}$, $STD\{C_0(t_{n-2}), C_0(t_{n-1}), C_0(t_n)\}$, $STD\{C_1(t_{n-2}), C_1(t_{n-1}), C_1(t_n)\}$

Fig. 8. Illustration of Extraction Feature Type C (Pattern Size: 3)

Stream Data	Feature Vector D
$C_0(t_0)$ $C_1(t_0)$	$Average\{C_0(t_0), C_1(t_0)\}$, $Average\{C_0(t_1), C_1(t_1)\}$, $Average\{C_0(t_2), C_1(t_2)\}$
$C_0(t_1)$ $C_1(t_1)$	$Average\{C_0(t_1), C_1(t_1)\}$, $Average\{C_0(t_2), C_1(t_2)\}$, $Average\{C_0(t_3), C_1(t_3)\}$
$C_0(t_2)$ $C_1(t_2)$	$Average\{C_0(t_2), C_1(t_2)\}$, $Average\{C_0(t_3), C_1(t_3)\}$, $Average\{C_0(t_4), C_1(t_4)\}$
...	...
$C_0(t_n)$ $C_1(t_n)$	$Average\{C_0(t_{n-2}), C_1(t_{n-2})\}$, $Average\{C_0(t_{n-1}), C_1(t_{n-1})\}$, $Average\{C_0(t_n), C_1(t_n)\}$

Fig. 9. Illustration of Extraction Feature Type D (Pattern Size: 3)

5.3 Experimental Results on Synthetic Tasks

Table 2 shown the results of 6 classifiers on six state detection tasks. All the results are from test only. Considering the overall performance, GP outperformed other classifiers. In particular, in Task 3 and 5, GP significantly outperformed other counterparts. The performance gaps between GP and other classifiers are not as wide as what we found in event detection.

Table 3 presents the results of conventional classifiers on pre-defined feature sets B, C, D. The results from GP runs on **raw data** are also listed. Obviously these well designed features can help the classifiers to achieve better results. However the superior performance of GP can still be observed.

Table 1. Training and Test Data of the Six Synthetic State Detection Tasks

Tasks	Training	Test	State Size	Numbers of Attributes (No Features)	Feature Set	Numbers of Attributes (Features)
1. Box Functions	263:249	122:133	3	3	C	2
2. Oscillation	217:280	88:178	7	7	B	1
3. Sine Wave vs. Random Numbers	279:150	112:133	8	8	B	1
4.Sine Waves	219:201	69:141	8	8	B	2
5. Sine Waves in Two Channels	140:276	46:159	8	16	B	2
6. Box Functions in Four out of Five Channels	203:309	92:163	8	40	D	8

Table 2. Synthetic State Detection: Comparing GP with Non-GP Methods on Raw Input Vector%

Tasks	OneR	J48	NB	IB1	AdaBoost	**GP**
1	92.09 TP: 98.4 TN: 86.3	100 TP: 100 TN: 100	100 TP: 100 TN: 100	100 TP: 100 TN: 100	100 TP: 100 TN: 100	**100** TP : **100** TN : **100**
2	59.84 TP: 63.7 TN: 56.8	98.46 TP: 99.1 TN: 97.9	90.73 TP: 100 TN: 83.6	99.23 TP: 100 TN: 98.6	99.23 TP: 100 TN: 98.6	**100** TP : **100** TN : **100**
3	56.3 TP: 92.9 TN: 23.8	91.18 TP: 100 TN: 83.3	61.76 TP: 74.1 TN: 50.8	92.86 TP: 100 TN: 86.5	91.18 TP: 100 TN: 83.3	**99.58** TP : **99.11** TN : **100**
4	66.5 TP: 65.2 TN: 67.2	96.55 TP: 100 TN: 94.8	66 TP: 82.6 TN: 57.5	98.52 TP: 100 TN: 97.8	97.04 TP: 100 TN: 95.5	**98.52** TP: **100** TN: **97.76**
5	65.15 TP : 100 TN : 54.6	87.88 TP : 56.5 TN : 97.4	82.83 TP : 100 TN : 77.6	74.75 TP : 0 TN : 97.4	85.86 TP : 43.5 TN : 98.7	**100** TP: **100** TN: **100**
6	74.6 TP: 51.1 TN: 88.5	96.77 TP: 97.8 TN: 96.2	90.73 TP: 79.3 TN: 97.4	99.19 TP: 100 TN: 98.7	97.18 TP: 98.9 TN: 96.2	**100** TP : **100** TN : **100**

The results shown in Table 2 and Table 3 demonstrate that GP has the capability to extract features that can distinguish a state from the rest of time series, even when a state pattern is relying in several channels. This is because with the given functions and terminals, GP is able to combine and operate on raw numeric values. This is actually an implicit feature construction process.

5.4 Experimental Results on a Real-world Task

To further evaluate the performance of our method, we tested it on a benchmark data set [11] for mobile-based activity recognition [2], which includes 21-channel sensory data collected from 5 subjects. There are four state detection tasks: sitting, walking, running and lying flat. Note that the walking state includes different gaits, including going upstairs and going downstairs. A leave-one-person-out validation is conducted in this study. That is, for each detection task, the

[2] Data can be download at
`http://yallara.cs.rmit.edu.au/~s3268719/AR/data.html`

Table 3. Synthetic State Detection: Comparing GP with Non-GP Methods on Feature Sets %

Tasks	OneR	J48	NB	IB1	AdaBoost	**GP**
1	100 TP: 100 TN:100	100 TP: 100 TN:100	100 TP: 100 TN:100	100 TP: 100 TN:100	100 TP: 100 TN:100	**100** TP : **100** TN : **100**
2	100 TP: 100 TN:100	100 TP: 100 TN:100	100 TP: 100 TN:100	100 TP: 100 TN:100	100 TP: 100 TN:100	**100** TP : **100** TN : **100**
3	100 TP: 100 TN: 100	100 TP: 100 TN: 100	99.58 TP: 74.1 TN: 99.2	100 TP: 100 TN: 100	100 TP: 100 TN: 100	A:99.58 TP: 99.11 TN: 100
4	93.6 TP: 100 TN: 90.3	98.52 TP: 100 TN: 97.8	91.13 TP: 100 TN: 86.6	98.52 TP: 100 TN: 97.8	98.52 TP: 100 TN: 97.8	**98.52** TP: **100** TN: **97.76**
5	77.78 TP: 100 TN: 71.1	98.99 TP: 100 TN: 98.7	100 TP: 100 TN: 100	99.49 TP: 100 TN: 99.3	100 TP: 100 TN: 100	**100** TP: **100** TN: **100**
6	87.1 TP: 93.5 TN: 83.3	98.79 TP: 97.8 TN: 99.4	100 TP: 100 TN: 100	100 TP: 100 TN: 100	100 TP: 100 TN: 100	**100** TP : **100** TN : **100**

Table 4. Leave-one-person-out: Accuracies, true Positive and true Negative rates (trained and tested on data from the right front pant pocket)(%)

	Sitting	Walking	Running	Lying
Subject 1	91.5 TP: 100 TN: 90.1	93.7 TP: 97.5 TN: 88.5	99.7 TP: 96.9 TN: 99.9	99.6 TP: 99.2 TN: 99.6
Subject 2	97.1 TP: 79.8 TN: 99.3	94.1 TP: 96.9 TN: 87.5	99.5 TP: 94.2 TN: 99.7	97.7 TP: 98.2 TN: 97.7
Subject 3	88.7 TP: 93.4 TN: 88.1	91.2 TP: 97.8 TN: 83.6	86.0 TP: 90.3 TN: 83.3	99.6 TP: 96.4 TN: 99.9
Subject 4	95.9 TP: 94.3 TN: 96.3	93.4 TP: 95.4 TN: 91.6	96.4 TP: 94.0 TN: 96.6	98.1 TP: 94.7 TN: 98.6
Subject 5	98.5 TP: 96.3 TN: 98.9	91.0 TP: 91.2 TN: 91.0	96.0 TP: 97.5 TN: 95.7	99.7 TP: 99.1 TN: 99.8

classification is conducted for five times. For each time, the records from one subjects are used for testing and the rest for training.

Table 4 shows the results from all four tasks on 5 subjects. Our method did achieve consistently good accuracy over different scenarios of state detection. These results show that GP can detect states not only from synthetic time series but also in a complex, real-world scenario.

6 Conclusions

State and event are two main types of time series patterns. In this study, we proposed a GP-based method for state detection from multi-channel time series. This method requires no manual feature extraction. Our experiments show GP-based method can achieve significantly better results on raw inputs and

competitive results when non-GP methods are provided with pre-defined features. The good performance of the proposed method is consistent on a set of synthetic problems as well as on real-world activity recognition problems. We conclude that GP based time series classification method is suitable for state detection.

References

1. Bernad, D.J.: Finding patterns in time series: a dynamic programming approach. In: Advances in Knowledge Discovery and Data Mining (1996)
2. Brooks, R.R., Ramanathan, P., Sayeed, A.M.: Distributed target classification and tracking in sensor networks. Proceedings of the IEEE 91(8), 1163–1171 (2003)
3. Chan, K.-P., Fu, A.W.-C.: Efficient time series matching by wavelets. In: Proceedings of the 15th International Conference on Data Engineering, pp. 126–133. IEEE (1999)
4. Englehart, K., Hudgins, B., Parker, P.A., Stevenson, M.: Classification of the myoelectric signal using time-frequency based representations. Medical Engineering & Physics 21(6), 431–438 (1999)
5. Garrett, D., Peterson, D.A., Anderson, C.W., Thaut, M.H.: Comparison of linear, nonlinear, and feature selection methods for eeg signal classification. IEEE Transactions on Neural Systems and Rehabilitation Engineering 11(2), 141–144 (2003)
6. Nanopoulos, A., Alcock, R., Manolopoulos, Y.: Feature-based classification of time-series data. International Journal of Computer Research 10(3) (2001)
7. Ralanamahatana, C., Lin, J., Gunopulos, D., Keogh, E., Vlachos, M., Das, G.: Mining time series data. Data Mining and Knowledge Discovery Handbook, 1069–1103 (2005)
8. Rasoul Safavian, S.: David Landgrebe. A survey of decision tree classifier methodology. IEEE Transactions on Systems, Man, and Cybernetics 21(3), 660–674 (1991)
9. Subasi, A.: Eeg signal classification using wavelet feature extraction and a mixture of expert model. Expert Systems with Applications 32(4), 1084–1093 (2007)
10. Way, M.J., Scargle, J.D., Ali, K.M., Srivastava, A.N.: Advances in Machine Learning and Data Mining for Astronomy. CRC Press, Boca Raton (2012)
11. Xie, F., Song, A., Ciesielski, V.: Genetic programming based activity recognition on a smartphone sensory data benchmark. In: 2014 IEEE Congress on Evolutionary Computation (CEC), pp. 2917–2924. IEEE (2014)
12. Xie, F., Song, A., Ciesielski, V.: Event detection in time series by genetic programming. In: 2012 IEEE Congress on Evolutionary Computation (CEC), pp. 1–8. IEEE (2012)
13. Xing, Z., Pei, J., Keogh, E.: A brief survey on sequence classification. ACM SIGKDD Explorations Newsletter 12(1), 40–48 (2010)