Human Action Recognition from Multi-Sensor Stream Data 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. This paper presents an approach to recognition of human actions such as sitting, standing, walking or running by analysing the data produced by the sensors of a smart phone. The data comes as streams of parallel time series from 21 sensors. We have used genetic programming to evolve detectors for a number of actions and compared the detection accuracy of the evolved detectors with detectors built from the classical machine learning methods including Decision Trees, Naïve Bayes, Nearest Neighbour and Support Vector Machines. The evolved detectors were considerably more accurate. We conclude that the proposed GP method can capture complex interaction of variables in parallel time series without using predefined features.

1 Introduction

The widespread penetration of smart phones with various sensors onboard has opened up opportunities for applications of human action recognition. Even a consumer phone can be easily turned into a sensor platform which can both constantly transmit and process sensor input about the person who carries the phone. Accurate recognition of the person's actions can enable a wide range of applications such as timing an activity, creating a sport profile and assisting living and healthcare [15].

The data stream produced by the sensors can be viewed as time series as each sensor generates a sequence of observations at regular intervals. Furthermore data are received from multiple channels such as accelerometer readings in x, y and z axes. Patterns of human locomotion such as walking and running, likely exist in these multi-channel time series. Since one sensor is not sufficient to make sense of a person's action, the ability of handling multiple data streams is crucial in this domain. Another difficulty in human action recognition is the lack of prior knowledge on the correlation between certain data patterns and human actions. As a result, manually constructing suitable models or features for different actions is not very feasible.

We present a Genetic Programming (GP) based method to address the aforementioned issues. More specially, our research questions are:

1. How can a suitable GP based methodology be established to evolve human action recognition programs which can handle raw input from multiple body sensors?

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 418–427, 2013.

2. How can this method handle multi-class of human actions?
3. How does the GP method compare to conventional classification algorithms?

Action detection problems can be formulated in two different ways: (1) As binary problems, for example, is the person running or not, or (2) As multi-class problems, for example, is the person the person running, or walking or standing, or none of these. Binary problems are easier to formulate and accuracies are higher, however, multi-class problems are more important in practice. In this paper we address both formulations.

2 Related Work

There is a massive body of literature on time series analysis. Most of the work is focused on the prediction of future values. However, there has been some work on other aspects such as detecting shapes and patterns in time series [6]. Most of the work on time series has been on a single time series, there has been very limited work on multi-channel time series.

There are existing studies using GP in time series analysis. Most of these are for forecasting. Kaboudan [4], for example, used GP for housing price prediction. The researcher found that the results produced by GP were more reliable and logically acceptable than those from neural networks. Wagner and Michalewicz [14] proposed GP with adaptive windowing for forecasting in a dynamic environment. Song and Pinto [13] applied GP to a sequence of video frames for motion detection problems. Hetland et al. [2] combined a pattern matching chip and GP to discover temporal rules useful for prediction. Xie et al. [16] used a GP-based framework to detect events of interest in a time series with background noise. Although these works have a different goal to ours, they do show the potential of GP to find rules in time series data without much human intervention.

Our work is similar to time series classification problem [12]. However, they are eventually different. The data mining algorithms usually require time series stream to be segmented into non-overlapped, fixed-size vectors as inputs. The proposed method is capable to deal with overlaps and detect action patterns with any length within a maximum search window size.

GP is most naturally used for binary classification, a negative output denoting one class and a positive output the other. There have been a number of proposals for extending GP to multiple classes. Kishore et al. [7] models an n-class classification problem as n binary classification problems. One Genetic Programming Classifier Expression(GPCE) is trained for each class which can recognise its own class. A strength of association is calculated for each GPCE to address conflicts where one instance is claimed by two or more GPCEs. Muni et al. [9] reported a novel approach in which one GP tree consists of n subtrees, each representing one of the n classes. Loveard et al. [8] proposed five ways of multi-class classification in GP: Binary Decomposition, Static Range Selection, Dynamic Range Selection (DRS), Class Enumeration and Evidence Accumulation. Of these methods, DRS is proved to be the best, and this is the method

used in our work. In this approach each evolved individual carries with it a mapping of the output values to classes and the mapping is generated as part of the evolutionary process.

3 GP Representations

We use tree based genetic programming. The function set is shown in Table 1 which lists the parameters and return types for each function. In addition to the four basic arithmetic functions, there are three extra functions for multi-channel time series. Two of these, `Window` and `Multi-Channel`, take the special terminals such as `Operation` and `Temporal Index`, listed in the terminal set (Table 2). Others take double types as input. These are from the sensor streams or returned values of other functions. In Table 2, each `Channel` m terminal denotes one value from a channel. These functions and terminals are explained below.

Table 1. Function Set

Functions	Parameters	Return Type
$\{ +, -, *, / \}$	1. Double, 2. Double	Double
Window	1. Double, 2. Temporal Index, 3. Operation	Double
Temporal_Diff	1. Double	Double
Multi-Channel	1. Channel Index, 2. Multivariable Operation	Double

Table 2. Terminal Set

Terminal	Value	Return Type	For Function
Channel m	Current value at Channel m	Double	*Any*
Operation	AVG,STD,DIF,SKEWNESS	Integer	Window
Temporal Index	$[1, 2^{window\_size} - 1]$	Integer	Window
Channel Index	$[1, 2^{num\_of\_channels} - 1]$	Integer	Multi-Channel
Multivariable Operation	MED,AVG,STD,RANGE	Integer	Multi-Channel

Function Window. This function is responsible for taking a selection of points from a time series. It has three parameters. The first one is the input in double. The value changes as data is continuously fed in. To "remember" past values, this function has a memory list which keeps the most recent S readings as $t_0, t_1, ..., t_{S-1}$, from the earliest to current. S is set to 12 in this study. The second parameter selects data points from the memory list. It reads the return value from terminal `Temporal Index` which randomly generates an integer in the range 1 to $2^{12} - 1$. The binary equivalent of this number is then used to determine which data points in the memory list will be selected. For example, a value of 11 of "`temporal index`" will be converted to binary giving 000000001011. Points t_8, t_{10} and t_{11} are selected as the eighth, the tenth and the eleventh bits are `true`. A value of 4095 would result in all 12 points in the memory list being selected. This mapping mechanism enables flexible point selection inside a

window and consequently helps find the duration of an action.The third parameter randomly selects one of four operations: `AVG`, `STD`, `DIF` and `SKEWNESS` on the points selected by the second parameter `Temporal Index`. These operations correspond to the calculation of average, standard deviation, sum of absolute differences and skewness of the selected points respectively.

Function Temporal_Diff. Function `Temporal_Diff` captures temporal differences. It takes one value from the stream input as its parameter and returns the difference between the current value and the previous one. It is effectively a window of size 2 so it can be considered as a special case of function `Window` which applies the subtraction operation on the last two points in memory list.

Function Multiple-Channel. Functions `Window` and `Temporal_Diff` handle only one sequence of values, namely readings from one channel. They can not capture patterns occurring across multiple channels. Hence function `Multi-Channel` is introduced which has two parameters: `Channel Index` and `Multivariable Operation`. The first parameter works in a similar fashion to `Temporal Index` in function `Window`. The range of index is in [1,$2^M - 1$] (M is the total number of channels). The second parameter randomly selects one of four multiple-variable operations: MED, AVG, STD and RANGE which operate on the current values of the selected variables and return the median, the average, the standard derivation and the distance between the maximum and minimum of these values.

4 Multi-class Classification

In this work we have used two ways to implement multi-class classification: (1) A single multi-class classifier. (2) An ensemble classifier based on binary classifiers. Given a classification problem with a set of classes $C = \{c_1, c_2, ..., c_n\}$, for each class a classifier $Classifier_j$ is evolved to classify class c_j (positive) against all other classes (negative). In total, $n - 1$ classifiers are generated. When one instance is claimed by multiple classifiers as positive, the classifier with higher accuracy in training will win.

5 Experiments

The GP runtime settings in experiments for this study are fairly standard. The population size is 1000. The maximum and minimum tree sizes are 8 and 2 respectively. The crossover, mutation and elitism rates are 85%, 10% and 5%. The evolution stops at a maximum of 50 generations. Each run is repeated 10 times and best individual is used for testing.

Table 3. Number of Instances in the Three Data Sets

Data Set	Class	Training	Test
Synthetic Data 1	Positive	200	103
	Negative	199	96
Synthetic Data 2	Positive	193	110
	Negative	206	89
Human Action Data	Sitting	1697	311
	Standing	1074	1345
	Walking	950	892
	Running	819	549
	Others	85	51

5.1 Data Sets

Three sets of multi-channel streams were used for evaluation. The first two are synthetic, containing two and five channels of input respectively. They are to validate the aforementioned methodology. The third is human action data, containing input from 21 channels. Table 3 shows the number of instances in each class. They have been split into training set and test set.

Synthetic Data. Both of the two synthetic data sets are designed for binary classification purposes. In Synthetic Data 1, there are two variables of streaming data input. A positive is defined as the presence of significant simultaneous changes(> 0.5) in both variables. To generate this data, each variable was initialised randomly. The value at each time interval was also random. The probability of a noticeable change in one variable occurring is 0.7.

Synthetic Data 2 is more complex as it has five variables. If any two of five channels simultaneously changed by more than 5 over two consecutive time intervals, then that is a positive. Similarly to the first data set, for each channel the next point is randomly generated with a probability of change 0.3. As we understand the data sets, the suitable features for them would be the differences between two adjacent points on each channel.

Human Action Data. This real world data set was collected from the built-in accelerometer, gyro, and magnetometer sensors of an iPhone4. An application was developed to read triaxial physical movement measurements from inertial units at a frequency of 30Hz. In total, there are 21 measurements (channels) available for recording. This data set involved one subject and no cross-subject validation was included in this research. The subject was asked to put the iPhone 4 in the waist pocket and to perform four actions in an order, `Sitting`, `Standing`, `Walking` and `Running`. The duration of each action was arbitrary.

To label the ground truth, the subject go "GO" to mark the end point of the previous action and the starting point of a new one. The voice recording is synchronised with the sensor data recording. The "GO" command also occurred at the same time as the subject changed action. The class labels of the recorded data are relatively accurate.

Data for Comparisons. Using the above data sets, the proposed GP method was compared with four conventional classification methods: *J48* (Decision Trees) [11],*Naïve Bayes* [3] , *IB5* (Nearest Neighbours) [1] and *SVM* (Support Vector Machine) [5, 10]. However, these methods can not directly work on time series data as they have no built-in sliding window mechanism, so we manually segmented the three data sets. For each channel, a window of fixed-length of W_s is used to build an instance. The value W_s is set to 2 for the two synthetic data sets and 12 for action data. This is to ensure that these non-GP methods will receive the same amount of information as GP.

These methods treat one instance as one row of values. However in multi-channel streams, there are multiple rows in one instance, so we flattened them into one row just like representing a matrix in a one-dimensional array. These are the raw stream inputs. In addition, two feature sets Set A and Set B are constructed for conventional classifiers. Set A calculates the temporal difference, that is the difference between two consecutive points. Therefore, $(W_s - 1)$ features are extracted for one channel. Set B contains the averages and the standard deviations of points of each channel at one window position . The number of attributes in the data for conventional methods are shown in Table 4. Set B is only used on human action data.

Table 4. Data Converted for Conventional Methods

Data Set	Window size	No. of Attributes (Raw Input)	No. of Attributes (Features)
Synthetic Data 1	2	4	$2 \times (2-1) = 2$
Synthetic Data 2	2	10	$5 \times (2-1) = 5$
Human Action Data	12	252	Set A $= 21 \times (12-1) = 231$ Set B $= 21 \times 2 = 42$

5.2 Results

All the methods for comparison use default parameters from the WEKA package. IB5 was used because we found 5 nearest neighbour often gave the best results on these data sets. The other algorithms has been tuned to achieve their best results.

Binary Classification. Table 5 presents the average accuracies, true positive rates (TP) and true negative rates (TN) on each test data by various methods. Each experiment is a binary classification, so there are 6 rows of results as there are 4 actions in the human action data. When manually constructed features are not available, the performance of the conventional classifiers is very poor. In the case of sitting, *J48* and *Naïve Bayes* appear to have a good accuracy of 90.12%. However, this is deceptive as their true positive rates are zero. Effectively they recognized nothing. The high accuracy is merely due to the dominance of negative cases in the data set. Most of the other results are not much better. The only good result from non-GP methods is obtained by IB5 on running and SMO on sitting.

Table 5. Test Results - Conventional Methods vs. GP- both on Raw Input (%)

Data Set	J48	Naïve Bayes	IB5	SVM	GP
Synthetic Data 1	51.8 TP : 100.0 TN : 0	49.8 TP : 27.2 TN : 74.0	51.8 TP : 58.3 TN : 44.8	48.2 TP : 0 TN : 100.0	**100.0** TP : **100.0** TN : **100.0**
Synthetic Data 2	55.3 TP : 0 TN : 100.0	57.3 TP : 10.1 TN : 95.5	50.3 TP : 48.3 TN : 51.8	44.7 TP : 100.0 TN : 0	**100.0** TP : **100.0** TN : **100.0**
3. Sitting	90.1 TP : 0 TN : 100.0	90.1 TP : 0 TN : 100.0	40.2 TP : 0 TN : 44.6	99.6 TP : 100 TN : 99.5	**99.7** TP : **100.0** TN : **99.7**
4. Standing	57.0 TP : 0 TN : 99.6	57.3 TP : 0 TN : 100.0	57.3 TP : 0 TN : 100.0	57.0 TP : 0 TN : 99.6	**96.6** TP : **92.1** TN : **99.9**
5. Walking	76.3 TP : 21.7 TN : 97.9	74.4 TP : 9.9 TN : 100.0	85.7 TP : 52.9 TN : 98.6	81.2 TP : 52.6 TN : 92.5	**A:97.7** TP : **97.1** TN : **98.0**
6. Running	34.2 TP : 88.5 TN : 22.7	68.2 TP : 100.0 TN : 61.4	96.4 TP : 94.4 TN : 96.8	20.8 TP : 84.9 TN : 7.3	**99.5** TP : **98.0** TN : **99.9**

Table 6. Test Results - Conventional Methods on Features vs. GP on Raw Input (%)

Data Set	J48	Naïve Bayes	IB5	SVM	GP
Synthetic Data 1	100.0 TP : 100.0 TN : 100.0	100.0 TP : 100.0 TN : 100.0	100.0 TP : 100.0 TN : 100.0	100.0 TP : 100.0 TN : 100.0	**100.0** TP : **100.0** TN : **100.0**
Synthetic Data 2	98.0 TP : 97.8 TN : 98.2	87.9 TP : 100.0 TN : 78.2	100.0 TP : 100.0 TN : 100.0	100.0 TP :100.0 TN : 100.0	**100.0** TP : **100.0** TN : **100.0**
3. Sitting	76.9 TP : 25.4 TN : 82.6	60.8 TP : 100.0 TN : 56.3	63.1 TP : 89.4 TN : 60.2	90.1 TP : 0 TN : 100.0	**99.7** TP : **100.0** TN : **99.7**
4. Standing	72.9 TP : 54 TN : 87.0	85.2 TP : 89.3 TN : 82.1	64.4 TP : 21.1 TN : 96.6	57.3 TP : 0 TN : 100.0	**96.6** TP : **92.1** TN : **99.9**
5. Walking	90.4 TP : 74.3 TN : 96.8	40.5 TP : 78.6 TN : 25.4	93.8 TP : 84.8 TN : 97.4	71.7 TP : 0 TN : 100.0	**A:97.7** TP : **97.1** TN : **98.0**
6. Running	96.1 TP : 96.0 TN : 96.1	77.4 TP : 100.0 TN : 72.6	97.9 TP : 88.0 TN : 100.0	82.6 TP : 0 TN : 100.0	**99.5** TP : **98.0** TN : **99.9**

Table 6 presents the results from these conventional methods on Set A features, comparing with GP. This is a somewhat unfair comparison between conventional methods using temporal features and GP using raw data. The rightmost column in the table is for GP, which is consistently the best performer in every task. Although the conventional methods operate on temporal differences, they still can only achieve comparable results to GP on the synthetic data. In particular, the true positives of these methods are rather poor, for example SVM on all four actions, J48 on sitting and standing and IB5 on standing. Their performance on four human action recognition tasks is much worse than GP. It should be noted that by using Set A features SMO result in worse performance than raw data. Set B features help SMO to achieve better accuracy on running(98.6%) and on walking(85.3%). In case of sitting detection, the result is slightly worse than using raw data but is still reasonable with an accuracy of 98.7%. However, it still failed to recognise any standing action. This feature set B did not bring any benefit to other non-GP methods. The details are not presented due to the space constraints.

Multi-class Classification. Table 7 shows a comparison of these methods treating human action data a multi-class problem instead of a set of binary problems. For the conventional methods, we can see that the use of features is effective and different methods react to features differently. *J48*, *Naïve Bayes* and *IB5*, achieved better results on Set A features, while *SVM* benefits from Set B features. Nevertheless, their performance is much worse than that of GP on raw data which is 93.7%, almost 14% higher than the best from these non-GP methods (80%).

Table 7. Test Accuracies from Multi-class Classification(%)

	J48	Naïve Bayes	IB5	SVM	GP
Raw Input	20.7	29.0	33.1	35.6	
Set A: Temporal Diff	63.9	80.0	58.1	20.6	**93.7**
Set B: AVG and STD	28.3	62.5	40.1	50.2	

Table 8. Ensemble of Conventional Classifiers

	Sitting	Standing	Walking	Running	Accuracy
Raw Input	SVM	IB5	IB5	IB5	42.5%
Set A: Temporal Diff	J48	Naïve Bayes	IB5	IB5	78.7%
Set B: AVG and STD	SVM	Naïve Bayes	IB5	IB5	47.9%

Ensemble Approach for Multi-class Classification. From the above experiments, we selected the best binary classifiers generated by conventional methods on each action, either using features, or on raw inputs. They are listed in Table 8. All classifiers in one row are combined together to form an ensemble. The process is described in Section 4. When an instance is classified as positive by

multiple classifiers e.g. Sitting and Standing. The most accurate classifier will label the instance. If none of the classifiers recognize the instance, then it is marked as *Others*. The accuracy of each ensemble is presented on the rightmost cell on that row.

For GP, the best classifiers trained for each action (as shown in rightmost column in Tables 5 and 6) can also work together as an ensemble to perform multi-class classification. The accuracy was improved slightly to 94.5% compared to 93.7% achieved by a single GP classifier. Compared to the best result of each row in Table 7, we can see that the ensemble approach did not bring benefits to conventional methods, but marginally helped GP to be more accurate.

Table 9. Confusion Matrix for GP Classifier Ensemble: (Accuracy:94.5%)

	Sitting	Standing	Walking	Running	*Other*
Sitting	311	0	0	0	0
Standing	0	1238	15	0	92
Walking	0	0	862	4	26
Running	0	0	11	538	0
Other	9	0	16	0	26

Table 9 shows the confusion matrix of using an ensemble of the best evolved binary classifiers for the five-class problem. A major advantage of this ensemble is that an outcome of "None of the above" (*Other* in Table 9) is possible. While this is an error for the current task, it would be a perfectly good outcome if the person was lying down. In the current task we think that most of these errors come from transitions between states.

6 Conclusions and Future Work

The results of the above investigation have clearly demonstrated the advantages of the GP approach to human action recognition from multi-channel data stream. We conclude that with a proper function set and terminal set, GP can evolve multivariable time series pattern recognition programs to differentiate various human actions based on a collection of body sensor input. The methodology does not require manually designed time series features and can handle raw input, so it can be applied to scenarios where domain knowledge about the actions is not available. This method can handle multi-class human actions either by a direct multi-class approach or by an ensemble of binary classification programs. In comparison with conventional methods, the high accuracies of our GP method are evident. It outperforms these methods even when they operate on temporal features rather than on raw input.

In our future work, we will analyse evolved individuals to gain some insight into the evolved rules. Another future adaptation of this work is to take transitions between different actions into account.

References

1. Aha, D., Kibler, D., Albert, M.: Instance-based learning algorithms. Machine Learning 6(1), 37–66 (1991)
2. Hetland, M.L., Sætrom, P.: Temporal rule discovery using genetic programming and specialized hardware. In: Proc. of the 4th Int. Conf. on Recent Advances in Soft Computing, pp. 182–188 (2002)
3. John, G.H., Langley, P.: Estimating continuous distributions in bayesian classifiers. In: Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence, UAI 1995, pp. 338–345. Morgan Kaufmann Publishers Inc., San Francisco (1995)
4. Kaboudan, M.: Spatiotemporal forecasting of housing prices by use of genetic programming. In: The 16th Annual Meeting of the Association of Global Business (2004)
5. Keerthi, S.S., Shevade, S.K., Bhattacharyya, C., Murthy, K.R.K.: Improvements to platt's smo algorithm for svm classifier design. Neural Comput. 13(3), 637–649 (2001)
6. Keogh, E., Lin, J., Fu, A.: Hot sax: Efficiently finding the most unusual time series subsequence. In: Proceedings of the Fifth IEEE International Conference on Data Mining, ICDM 2005, pp. 226–233. IEEE Computer Society, Washington, DC (2005)
7. Kishore, J.K., Patnaik, L.M., Mani, V., Agrawal, V.K.: Application of genetic programming for multicategory pattern classification. Trans. Evol. Comp. 4(3), 242–258 (2000)
8. Loveard, T., Ciesielski, V.: Representing classification problems in genetic programming. In: Proceedings of the 2001 Congress on Evolutionary Computation, vol. 2, pp. 1070–1077. IEEE (2001)
9. Muni, D.P., Pal, N.R., Das, J.: A novel approach to design classifiers using genetic programming. Trans. Evol. Comp. 8(2), 183–196 (2004)
10. Platt, J.C.: Fast training of support vector machines using sequential minimal optimization. In: Advances in Kernel Methods, pp. 185–208. MIT Press, Cambridge (1999)
11. Quinlan, J.R.: C4.5: programs for machine learning. Morgan Kaufmann Publishers Inc., San Francisco (1993)
12. Ratanamahatana, C., Lin, J., Gunopulos, D., Keogh, E., Vlachos, M., Das, G.: Mining time series data. In: Data Mining and Knowledge Discovery Handbook, pp. 1049–1077 (2010)
13. Song, A., Pinto, B.: Study of gp representations for motion detection with unstable background. In: 2010 IEEE Congress on Evolutionary Computation (CEC), pp. 1–8. IEEE (2010)
14. 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, GECCO 2008, pp. 1657–1664. ACM, New York (2008)
15. Wang, L., Gu, T., Tao, X., Chen, H., Lu, J.: Recognizing multi-user activities using wearable sensors in a smart home. Pervasive Mob. Comput. 7(3), 287–298 (2011)
16. 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 (June 2012)