Dynamic signal segmentation for activity recognition

Simon Kozina, Mitja Luštrek, Matjaž Gams
Jozef Stefan Institute, Department of Intelligent Systems
Jamova cesta 39, 1000 Ljubljana, Slovenia
{simon.kozina, mitja.lustrek, matjaz.gams}@ijs.si

Abstract

Activity recognition is an essential task in many ambient assisted living applications. Activities are commonly recognized using data streams from on-body sensors such as accelerometers. An important subtask in activity recognition is signal segmentation: a procedure for dividing the data into intervals. These intervals are then used as instances for machine learning. We present a novel signal segmentation method, which utilizes a segmentation scheme based on dynamic signal partitioning. To validate the method, experimental results including 6 activities and 4 transitions between activities from 11 subjects are presented. Using a Random forest algorithm, an accuracy of 97.5% was achieved with dynamic signal segmentation method, 94.8% accuracy with non-overlapping and 95.3% with overlapping sliding window method.

1 Introduction

Activity recognition using on-body sensors is required for many ambient assisted living applications. This paper focuses on an important subtask in activity recognition: signal segmentation, the process of dividing the data into intervals. On-body sensors are collecting and continuously outputting streams of data. These streams are used to recognize the user's current activity.

The problem tackled in this paper is how to segment the data into intervals most suitable for activity recognition. Most approaches use overlapping and non-overlapping sliding windows, which means that the data is divided into intervals of fixed length. On each interval features are computed and then used as an instance for activity recognition. We present a novel method for signal segmentation, which attempts to match the intervals to the borders between different activities.

Dynamic signal segmentation method is based on searching for significant differences between consecutive data samples. A significant difference is determined by a dynamically computed threshold. It is updated whenever a new data sample is received, and adapts to changes in the data stream.

The paper is structured as follows. Section 2 gives an overview of related work on activity recognition with on body sensors. Section 3 describes two signal segmentation methods: the sliding window method and the novel dynamic signal segmentation method. Section 4 lists the attributes extracted from the input data that are fed into the machine learning algorithms. Section 5 presents the experiments in which the signal segmentation methods are compared. Finally, Section 6 concludes the paper and outlines the future work.

2 Related work

Various sensors are used for activity recognition: accelerometers and gyroscopes, real-time locating systems (RTLS) [Mircevska *et al.*, 2009], cameras [Qian *et al.*, 2004; Vishwakarma *et al.*, 2007] and environmental sensors [Zhan *et al.*, 2007]. Cameras pose a (real or perceived) threat to privacy, RTLS are expensive, and both require installation in the apartment as do environmental sensors. Because of that accelerometers and/or gyroscopes, which are inexpensive and portable, are most commonly for activity recognition, although in some situations they are not unobtrusive to the user.

Koskimaki et al. [2009] used a single wrist worn accelerometer to collect the acceleration and angular speed data. They defined four activities and one class value to denote "other" activities. Using the overlapping sliding window method with the window size of half a second, almost 90% accuracy was achieved. Ravi et al. [2005] tried to recognize eight activities, using one accelerometer placed on the abdominal area. Two of these activities are the same as in our testing scenario, others are similar. They divided the signal into overlapping five-second windows, and achieved accuracy of 73.3%. Mannini and Sabatini [2010] tried to recognize seven activities using five accelerometers placed on the body. Four of these seven activities are identical to the ones in our scenario. They achieved 98.5% accuracy when using the 6.7 second overlapping sliding window method. None of this researches had tackled the problem of recognizing transitions between activities.

Bifet and Gavalda [2007] have presented a segmentation algorithm that is recomputing the size of the sliding window accordingly to the rate of change observed from the data. The window is growing when the data is stationary, and shrinking when change is taking place. In order to work, the algorithm has to be integrated into a machine learning algorithm. Nunez et al. [2007] also presented an incremental decision tree algorithm, which is adapting sliding window size to portions of

the target concept. Each leaf of the decision tree holds a time window and a local performance measure. When the performance of a leaf decreases, the size of its local window is reduced. Some limitation may arise when dealing with large amount of data as the decision tree has to be updated when new examples are available.

3 Signal segmentation

Two methods are typically used to evaluate a stream of data for activity recognition. The first method is to use a single data point to determine the current activity. This method is not commonly used as the information gathered from a single data point is in most cases not sufficient for activity recognition. The second method involves signal segmentation. This means that consecutive sensor data are grouped. In contrast to the first method, multiple data points are used to determine the current activity. Using multiple data points allows more information to be extracted from the data, so the activities can be determined more accurately. However, the question of how exactly to group consecutive data needs to be tackled.

Some common methods for signal segmentation are overlapping and non-overlapping sliding windows, and signal spotting [Junker *et al.*, 2004; Benbasat *et al.*, 2000; Amft *et al.*, 2005]. In this section a new method for signal segmentation is proposed - dynamic signal segmentation method.

3.1 Sliding window method

The sliding window method is the most commonly used signal segmentation method for activity recognition with machine learning. The sliding window method accumulates sensor data over a fixed time window. Features are computed over one time window and are used as an instance for a learning/testing set. Two approaches are commonly used for data segmentation with sliding windows. The first approach is non-overlapping sliding windows, where consecutive time windows do not share common data samples. The second approach is overlapping sliding windows, which share common data samples between time intervals; for example, two consecutive time windows may have 50% of data samples in common.

3.2 Dynamic signal segmentation

Dynamic signal segmentation method is a novel method for signal segmentation. In principle the method can be used on any domain where a stream of sensor data has to be processed and the data has to be divided into segments. We tested the usability of the method on an acceleration-based domain for the purpose of activity recognition. We assume that, in addition to the acceleration data, the method can also be used for ECG or thermometer data, but it has not been tested yet.

The method searches for a significant change between consecutive data samples and divides the data into intervals at that point. The significant change is defined as a sequence of consecutive data samples where the values are in descending order, and the difference between the maximum and the minimum element in the sequence is larger than a threshold. The condition that the values should be in descending order is specific to our problem of accelerometer-based activity recognition, since each strong deceleration is typically quickly followed by an acceleration. Considering both would thus lead to dividing the data twice when a significant change occurs. For other types of data, both descending and ascending order should be considered.

Examples of sequences with the values in descending order are shown in Figure 1 denoted with a dotted line. When a set of descending data samples is found, the last element of this sequence is used as an ending point of one and starting point of the next interval. Therefore, the length of an interval is changing dynamically, as opposed to the sliding window, where it is set to a specific length. The features computed from each of the intervals are used as an instance for machine learning.

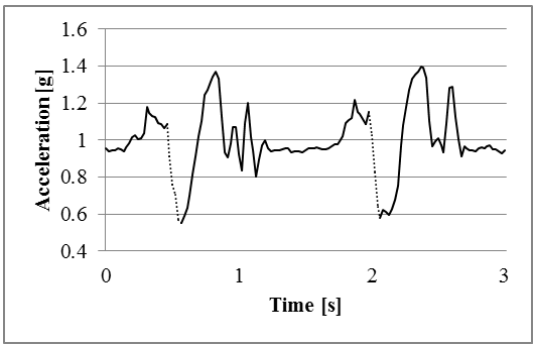

Figure 1: Descending sequence, denoted with dotted line, on a three-second time window.

The threshold, at each data sample, is computed from previous N data samples. Therefore, an initialization process of the algorithm uses N data samples to compute the first threshold. These data samples are used to compute the average minimum (avg_{min}) and average maximum (avg_{max}) values. The average minimum value is defined as the average of the first smallest ten percent of values in the last N data samples. The average maximum value is defined as the largest ten percent of values. In Figure 2, maximum values are denoted with circles and minimum values with squares. An average of each of these points is computed.

When these two values are obtained, the threshold can be computed:

$$threshold = (avg_{max} - avg_{min}) \cdot C$$

where $C \in [0, 1]$ is a constant selected prior to the start of the algorithm. This approach for setting the threshold is better than using only the minimum and the maximum values on an interval . For example, if there are some errors in the data, such as abnormal high or low peaks, these will be partially corrected with the other values for averaging minimum or maximum. The constant C can be computed from a learning dataset as follows:

$$C = \frac{\frac{1}{n} \cdot \sum_{i=1}^{n} a_i}{a_{max} - a_{min}}$$

The value n denotes the number of data samples in a learning dataset, a_{min} and a_{max} are the minimum and the maximum

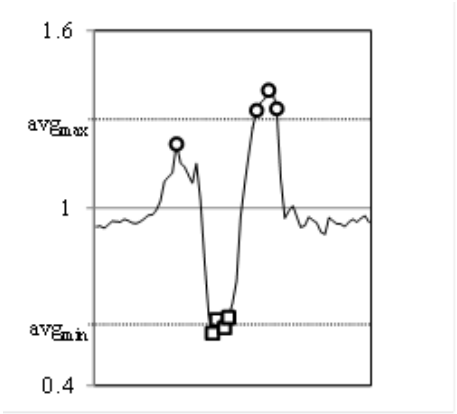

Figure 2: Four minimum and four maximum points on 40 data sample interval.

accelerations on the interval and a_i is the length of an acceleration vector at data sample i. Another way to set the constant C would be to tune it by running the dynamic signal segmentation on a separate dataset.

In addition to selecting the appropriate constant C, the developer has to determine which input signal should be used for threshold computation. This depends on a diversity of input signals and the domain. Our experiments were done using two 3-axial accelerometers attached to the left thigh and the chest. If, for example, we were driving a car, vertical acceleration would stay identical for almost all the time and same would apply for the threshold. On the other hand, if several activities, like walking, running, etc., were performed, the vertical acceleration would probably be the best choice as it would provide maximum information about activities.

A general solution when using one 3-axial accelerometer would be to use the length of the acceleration. However, when using more than one accelerometer, like in our example, the input signal for a threshold computation should be derived from multiple accelerometers. In our experiments the input signal for threshold computation was the arithmetic mean of lengths from both accelerometers and was derived as follows:

$$A = \frac{1}{2} \cdot \sqrt{a_x^2 + a_y^2 + a_z^2} \cdot \sqrt{b_x^2 + b_y^2 + b_z^2}$$

where $\vec{a} = [a_x, a_y, a_z]$ and $\vec{b} = [b_x, b_y, b_z]$ are acceleration vectors from both accelerometers.

4 Feature computation

In our experiments, once the stream of data was segmented either by the sliding window method or the dynamic signal segmentation, we used the same procedure to compute the features activity recognition by machine learning. Some additional information could be derived when using dynamic signal segmentation method, for example the time duration of an interval. However, in order to have comparable results, these additional attributes were not used.

As stated above, two accelerometers were used in our experiments. The following attributes were derived separately for the acceleration vectors from each of the accelerometers:

- The average length of the acceleration vector within the window, which could be of fixed size or computed with dynamic signal segmentation.
- The variance of the length of the acceleration vector. The variance within the window was defined as follows:

  $$\delta^2 = \frac{\sum_{i=1}^{N}(a_i - \overline{a})^2}{N}$$

  where N is the number of acceleration data within the window, is the length of the i-th acceleration vector and $\overline{a}$ is the average length of the acceleration of all previous samples.
- The average acceleration along the x, y and z axes.
- The maximum and the minimum acceleration along the x, y and z axes.
- The difference between the maximum and the minimum acceleration along the x, y and z axes.
- The angle of change in acceleration between the maximum and the minimum acceleration along the x, y and z axes. It was defined as follows:

  $$\Omega = \arctan\left(\frac{a_{max} - a_{min}}{t_{a_{max}} - t_{a_{min}}}\right)$$

  where a_{max} and a_{min} are the maximum and minimum acceleration along one axis within the window, and $t_{a_{max}}$ and $t_{a_{min}}$ are the times when they were measured. Figure 3 shows the principle of computing the angle of change in acceleration in one time window.If $t_{a_{max}} > t_{a_{min}}$ the angle is positive, otherwise the angle is negative.

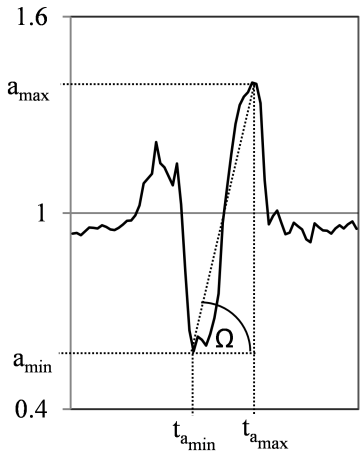

Figure 3: The angle of acceleration in a time window.

- The orientation of the accelerometer. We assumed that the acceleration vector $a = [a_x, a_y, a_z]$, which consists of the accelerations along the three axes of the accelerometer, generally points downwards (in the direction of the Earth's gravity). Let z be the axis pointing downwards when the accelerometer is in upright position. The angle ϕ between the acceleration vector and

the z axis thus indicates the person's orientation, and was computed as follows:

$$\phi = \arccos\left(\frac{a_z}{\sqrt{a_x^2 + a_y^2 + a_z^2}}\right)$$

To sum it up, 18 attributes were computed for each accelerometer. The final attribute was the angle between accelerometer vectors. It was obtained by computing the scalar product of vectors, normalized to their length:

$$\Theta = \arccos\left(\frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \cdot \|\vec{b}\|}\right)$$

Vectors $\vec{a}$ and $\vec{b}$ each represent the acceleration from both accelerometers. One instance in learning/testing set was thus represented with an attribute vector consisting of 37 attributes.

5 Experiments

We compared the performance of the signal segmentation methods on a scenario recorded by 11 healthy volunteers (7 male and 4 female), 5 times by each. Three of these recordings (2 male and 1 female) were used to create the training set and the other 8 were used to create the test set.

The scenario included 6 activities and 4 transitions. Transitions are defined as short actions between two activities. The activities and transitions are listed in Table 1.

	Activity		Transition
1.	standing	7.	falling
2.	walking	8.	sitting down
3.	on all fours	9.	standing up
4.	sitting	10.	lying down
5.	sitting on the ground		
6.	lying		

Table 1: Activities and transitions

To classify new instances we trained a classifier using the Random forest algorithm on our training set. The algorithm was implemented in Weka machine learning suite [Hall *et al.*, 2009]. The constant C, used by dynamic signal segmentation, was set to 0.4. This value was obtained by testing the dynamic signal segmentation algorithm on a different dataset than used for this paper. The same procedure was used to determine the value N for the number of data required for threshold computation. The value N was set to 100. The length of the sliding window was set to 1 second.

Each data sample in our training and test sets was labeled with an activity, whereas both the sliding window method and dynamic signal segmentation recognize the activity of a whole time interval. For training and testing purposes we thus considered the true label of an interval to be the majority if the labels of all the data samples in the interval.

5.1 Results

We compared the results of dynamic signal segmentation to overlapping and non-overlapping sliding window methods. We divided the scenario into two separate problems. The first problem was to recognize only the activities, and the second problem was to recognize both the activities and the transitions. Both problems were tested on the same dataset with one difference, in the first case the transitions were excluded from the training and test sets. The performance of the three methods was measured in terms of classification accuracy. Table 2 shows the results of all the methods.

	Methods		
	Non-overlapping sliding window	Overlapping sliding window	Dynamic signal segmentation
Activities	94.8%	95.3%	97.5%
Activities and transitions	89.0%	89.6%	92.9%

Table 2: All the methods compared using just activities and activities with transitions.

Based on these results we can conclude that there is a difference between non-overlapping and overlapping sliding windows, compared to dynamic signal segmentation method. On the other hand, the difference between the two sliding window methods and the dynamic segmentation method on the problem without transitions is 2.9 and 2.4 percentage points, and when the transitions are included it is 3.9 and 3.3 percentage points.

The quality of dynamic signal segmentation method depends on the threshold computation. For example, if the value N, which determines the number of previous samples used for threshold computation is set too high, the threshold does not update fast enough. As a consequence, transitions between activities are overlooked. On the other hand, if the number of previous samples is set too low, the threshold is over-fitted to the acceleration data. As a consequence data is fragmented and not appropriate for proper activity recognition. Figure 4 shows the threshold values in a single recording. We can notice that the threshold is changing according to activities. When a static activity, e.g. lying or sitting, is performed the algorithm updates the threshold to a small value, but while a person is walking, the threshold is updated to a higher value.

If the threshold computation was not implemented in the algorithm, the data stream would be divided using a simple, predefined threshold. Static activities like standing and sitting would not be separated using this approach. Another problem would be the determining the threshold.

As mentioned in section 3.2, the avg_{min} and avg_{max} values are used to compute the threshold. These values are obtained by computing the mean value of minimum and maximum ten percent of values. Other approaches can be used to obtain these values. Instead of the mean value, we could

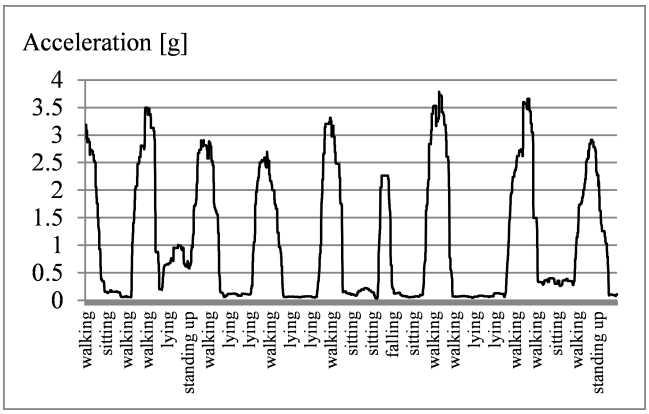

Figure 4: Changing of threshold values in a scenario.

compute a median value or use only minimum and maximum values. Results of these approaches are shown in Table 3.

	Mean	Median	Only min and max values
Accuracy	97.5%	96.9%	96.1%

Table 3: Accuracy of different approaches for computing avg_{min} and avg_{max} values.

By analyzing the confusion matrix [Kohavi and Provost, 1998] for the experiment excluding the transitions between activities (Table 4), we can conclude that the accuracy of all the activities except *on all fours* is above 90%. *On all fours* is usually confused with lying on the stomach, because the sensor orientation are the same (parallel with the ground and facing the ground). Another reason for poor performance, when recognizing *on all fours* activity, can be found in a small number of instances of this activity in the learning set (only 0.6 %). One of the solutions would be to change our scenario accordingly to extend the recording time when a person is on all fours.

	Sta	Walk	On4	Sit	SitG	Ly
Sta	95.2%	4.8%	0	0	0	0
Walk	2.0%	97.5%	0.4%	0	0	0.1%
On4	3.5%	10.6%	51.8%	0	0	34.1%
Sit	0	0.3%	0	95.9%	3.4%	0.4%
SitG	0	0.2%	0	5.2%	92.6%	2%
Ly	0	0.1%	0.1%	0.1%	0	99.7%

Table 4: Confusion matrix for activity recognition. Standing (Sta), walking (Walk), on all fours (On4), sitting (Sit), sitting on the ground (SitG), lying (Ly).

The results of activity recognition with transitions between activities are presented in Table 5. The overall accuracy of activities has decreased as there are four more classes that have to be predicted. Recognition of transitions is 62.2% accurate. This could have occurred due to the fact that the length of the transitions is much shorter than the length of activities; therefore the labels may not correspond perfectly to the data in these short intervals. This can happened because of several reasons: mislabeling, sometimes it is hard to determine the correct limit between transition and activity even by hand.

Activity/transition	Accuracy
standing	90.5%
walking	96.9%
on all fours	23.3%
sitting	96.9%
sitting on the ground	93.8%
lying	98.7%
falling	42.1%
sitting down	49.5%
standing up	69.7%
lying down	41.2%

Table 5: Accuracy of activity and posture recognition.

6 Conclusion

We have presented a novel method for signal segmentation, which is an important subtask in activity recognition. Signal segmentation is a process of dividing the stream of data into groups and is used for dividing of acceleration data, gyroscope data etc. Common methods for signal segmentation are overlapping and non-overlapping sliding windows. These methods divide the data into fixed time intervals. Our method, dynamic signal segmentation, is dividing the data based on patterns in data stream. The method is searching for significant changes in the data based on the threshold. The threshold is updated with every new data sample and is changing dynamically, according to the signal. When such a change is found in the data, it is used as a limit between consecutive intervals. An interval is then used as an input for machine learning.

We compared the performance of common signal segmentation methods with our dynamic segmentation method on a scenario recorded by 11 healthy volunteers (7 male and 4 female). Each scenario included six activities and four transitions between activities. Using the Random forest algorithm 97.5% accuracy was achieved with dynamic signal segmentation, 95.3% with overlapping and 94.8% with non-overlapping sliding window method. We have also showed that transition have negative effects on accuracy of activity recognition. All the methods had lower accuracy with transitions instances in learning/testing set.

There are several directions for future work. The first is the development of more acceleration related attributes and augment them with feature selection techniques. The second direction is automatic labeling of the data: an algorithm for semi-supervised learning which would group similar intervals together into clusters. User would only need to manually label these clusters after the experiments. The third direction is in improvement of the existing algorithm with techniques for statistical data analysis.

References

[Amft *et al.*, 2005] Oliver Amft, Holger Junker, and Gerhard Troster. Detection of eating and drinking arm gestures using inertial body-worn sensors. In *Proceedings of the Ninth IEEE International Symposium on Wearable Computers*, pages 160–163, Washington, DC, USA, 2005. IEEE Computer Society.

[Benbasat *et al.*, 2000] Ari Y. Benbasat, Joseph A. Paradiso, Stephen A. Benton, Ari Yosef Benbasat, and Ari Yosef Benbasat. An inertial measurement unit for user interfaces, 2000.

[Bifet and Gavalda, 2007] Albert Bifet and Ricard Gavalda. Learning from time-changing data with adaptive windowing. In *SIAM International Conference on Data Mining*, 2007.

[Hall *et al.*, 2009] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: An update. *SIGKDD Explorations*, 11, 2009.

[Junker *et al.*, 2004] Holger Junker, Paul Lukowicz, and Gerhard Trster. Continuous recognition of arm activities with body-worn inertial sensors. In *Proceedings of the Eighth International Symposium on Wearable Computers*, ISWC '04, pages 188–189, Washington, DC, USA, 2004. IEEE Computer Society.

[Kohavi and Provost, 1998] Ron Kohavi and Foster Provost. Glossary of terms. *Machine Learning*, 30(2-3), 1998.

[Koskimaki *et al.*, 2009] Heli Koskimaki, Ville Huikari, Pekka Siirtola, Perttu Laurinen, and Juha Roning. Activity recognition using a wrist-worn inertial measurement unit: A case study for industrial assembly lines. *Mediterranean Conference on Control and Automation*, 0:401–405, 2009.

[Mannini and Sabatini, 2010] Andrea Mannini and Angelo Maria Sabatini. Machine learning methods for classifying human physical activity from on-body accelerometers. *Sensors*, 10(2):1154–1175, 2010.

[Mircevska *et al.*, 2009] Violeta Mircevska, Mitja Lustrek, Igone Vlez, Narciso Gonzlez Vega, and Matjaz Gams. Classifying posture based on location of radio tags. *Ambient Intelligence and Smart Environments*, 5:85–92, 2009.

[Nunez *et al.*, 2007] Marlon Nunez, Ral Fidalgo, and Rafael Morales. Learning in environments with unknown dynamics: Towards more robust concept learners, 2007.

[Qian *et al.*, 2004] Gang Qian, Feng Guo, Todd Ingalls, Loren Olson, Jody James, and Thanassis Rikakis. A gesture-driven multimodal interactive dance system. In *Proceedings of IEEE International Conference on Multimedia and Expo*, pages 27–30, 2004.

[Ravi *et al.*, 2005] Nishkam Ravi, Nikhil Dandekar, Prreetham Mysore, and Michael L. Littman. Activity Recognition from Accelerometer Data. *American Association for Artificial Intelligence*, 2005.

[Vishwakarma *et al.*, 2007] Vinay Vishwakarma, Chittaranjan Mandal, and Shamik Sural. Automatic detection of human fall in video. In *Proceedings of the 2nd international conference on Pattern recognition and machine intelligence*, PReMI'07, pages 616–623, Berlin, Heidelberg, 2007. Springer-Verlag.

[Zhan *et al.*, 2007] Yi Zhan, Shun Miura, Jun Nishimura, and Tadahiro Kuroda. Human activity recognition from environmental background sounds for wireless sensor networks. In *Proceedings of the 2007 IEEE International Conference on Networking, Sensing and Control*, pages 307–312, 2007.