Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 32 (2014) 553 – 560

Procedia Computer Science

5th International Conference on Ambient Systems, Networks and Technologies (ANT-2014)

Long-term Activities Segmentation using Viterbi Algorithm with a k-minimum-consecutive-states Constraint

Enrique Garcia-Ceja[a,*], Ramon Brena[a]

[a]*Tecnológico de Monterrey, Campus Monterrey, Av. Eugenio Garza Sada 2501 Sur, Monterrey, N.L., México*

Abstract

In the last years, several works have made use of acceleration sensors to recognize simple physical activities like: walking, running, sleeping, falling, etc. Many of them rely on segmenting the data into fixed time windows and computing time domain and/or frequency domain features to train a classifier. A *long-term activity* is composed of a collection of simple activities and may last from a few minutes to several hours (e.g., shopping, exercising, working, etc.). Since *long-term activities* are more complex and their duration varies greatly, generating fixed length segments is not suitable. For this type of activities the segmentation should be done dynamically. In this work we propose the use of the Viterbi algorithm on a Hidden Markov Model with the addition of a k-minimum-consecutive-states constraint to perform the *long-term activity* recognition and segmentation from accelerometer data. This constraint allows the algorithm to perform a more informed search by incorporating prior knowledge about the minimum duration of each *long-term activity*. Our experiments showed good results for the activity recognition task and it was demonstrated that the accuracy was significantly increased by adding the k-minimum-consecutive-states constraint.

Keywords: activity-recognition, viterbi-algorithm, segmentation, constrained-viterbi, context-awareness

1. Introduction

With the recent advent of new devices that include a collection of sensors it is now possible to collect data that can help to automatically understand the user's surrounding situation and self state. Examples of such devices are smartphones, smartwatches[1], fitness bracelets[2], etc. Many of them now include sensors to measure acceleration, rotation, humidity, position, magnetic fields, light intensity, among other physical phenomena. In the last years, several works have made use of acceleration sensors to recognize simple physical activities like: walking, running, sleeping, falling[3,4,5,6,7]. Many of them rely on segmenting the data into fixed time windows of 2-10 seconds. Then, there is a process of feature extraction which generally consists of computing time domain and/or frequency domain features[8] from the fixed time windows. Each set of features of the corresponding time window is known as an instance. Those instances are used to train and test classifiers like Decision Trees, Naïve Bayes, Support Vector Mchines, among others[9]. Aside from simple activity recognition, there has also been research on complex and *long-*

* Corresponding author. Tel.: +52-811-692-3923
E-mail address: A00927248@itesm.mx

doi:10.1016/j.procs.2014.05.460

term activities which are composed of a collection of simple activities and may include additional information like time of the day, location, interactions between other persons and objects, etc. Examples of this type of activities are: cooking, sporting, commuting, taking medication, etc. The recognition of these activities generally requires more sensors and a fixed infrastructure (video cameras, RFID tags, several accelerometers, magnetic sensors) [10,11,12].

The fixed time window approach for recognizing simple activities is appropriate since they last only a few seconds. However, for *long-term activities*, generating fixed length segments is not suitable. For example, the long-term activity *commuting* could last from a few minutes to several hours. Another problem is that different long-term activities may share the same simple activities. For example, when *commuting*, a person may drive and walk while for *exercising* he/she may run, do some intense movements but also walk. With a supervised learning approach with fixed time windows, if the person is walking, the *long-term activity* may be recognized as *commuting* when in fact the user may be in the middle of exercising. In our previous work [13] we used the Bag of Features approach [14] to recognize long-term activities, however the activities were already segmented. In real life, long-term activities occur continuously, i.e., they overlap from each other and occur one after the other. Given a sequence of accelerometer data, the task is to recognize which *long-term activities* were performed by the user and the order in which they occurred, i.e., their segmentation.

In this work we propose the use of Viterbi algorithm [15,16] on a Hidden Markov Model (HMM) with the addition of a k-minimum-consecutive-states constraint to perform the long-term activity segmentation from accelerometer data. The *long-term activities* will be modeled as a collection of *simple activities* called *primitives* which are automatically discovered from the data [13]. A *primitive* is a *simple activity* like *walking, running, jumping, etc.* and each of them is uniquely identified by a numerical id. Within the HMM context, each *primitive* is an observation at time t so the problem can be stated as follows: Given a sequence of T observations, find the most likely sequence of states(i.e., long-term acitivies) of length T that produced those observations. And that is precisely what Viterbi algorithm does. Suppose each *primitive* represents data from 10 seconds and for a specific sequence of observations we get the following most likely sequence of states:

...11111111111222222224422222222222...

where *1,2* and *4* represent the long-term activities *commuting, working* and *shopping*, respectively. According to the resulting state sequence, there was a *shopping* activity that lasted 20 seconds but it is very unlikely that someone will take that time to go shopping. Thus, we may want to include prior knowledge about the minimum duration of each long-term activity and that is what the k-minimum-consecutive-states constraint does. For example, we can restrict the *shopping* activity to have at least 30 consecutive states in the resulting sequence to indicate that it requires at least 5 minutes to be considered.

Section 2 presents the background of HMM's. In Section 3 we describe the proposed constraint for the Viterbi algorithm. Section 4 details the data collection process and Section 5 describes the training phase. In Section 6 we detail the experiments and present the results. Finally, in Section 7 we draw conclusions and propose the future work.

2. Background

In this Section we describe the basics of Hidden Markov Models and then we explain the Viterbi Algorithm which is the core for the segmentation process.

2.1. Hidden Markov Models

A Hidden Markov Model (HMM) is a probabilistic graphical model consisting of a set of observable(observations) and unobservable(sates) random variables that describe the state of the modeled world at a particular time t. In this work we focus on first-order HMM's with discrete state and observation variables. For the rest of this paper we will use the notation from Rabiner and Juang [17]. The total number of distinct states is N. The time instants associated with state changes are denoted as $t = 1, 2, ...$, and the actual state at time t as q_t. A first-order Markov process is one in which the current state depends only on the previous state:

$$P(q_t = j \mid q_{t-1} = i, q_{t-2=k,...}) = P(q_t = j \mid q_{t-1} = i) \tag{2.1}$$

A HMM can be defined as a 5-tuple $\langle N, M, A, B, \pi \rangle$ where

- N is the number of states in the model which are indexed by $\{1, 2, ..., N\}$
- M is the number of distinct observation symbols which are denoted as $V = \{\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_M\}$.
- A is the state transition probability distribution. $A = \{a_{ij}\}$ where

$$a_{ij} = P(q_{t+1} = j \mid q_t = i), \quad 1 \leq i, j \leq N \tag{2.2}$$

  and $a_{ij} \geq 0 \quad \forall j, i; \sum_{j=1}^{N} a_{ij} = 1 \quad \forall i$
- B is the observation symbol probability distribution. $B = \{b_j(k)\}$ in which

$$b_j(k) = P(\mathbf{o}_t = \mathbf{v}_k \mid q_t = j), \quad 1 \leq k \leq M, \tag{2.3}$$

  defines the symbol distribution in state j, $j = 1, 2, ..., N$.
- π is the initial state distribution $\pi = \{\pi_i\}$ in which

$$\pi_i = P(q_1 = i), \quad 1 \leq i \leq N. \tag{2.4}$$

The complete parameter set of the model is thus defined as $\lambda = (A, B, \pi)$

2.2. Viterbi Algorithm

What Viterbi algorithm does is to uncover the hidden part of the model(i.e., the states) given the observations and the model λ. It does so by maximizing $P(\mathbf{q} \mid \mathbf{O}, \lambda)$ using dynamic programming. The highest probability along a single path at time t for the first t observations and that ends in state i can be defined as:

$$\delta_t(i) = \max_{q_1, q_2, ... q_{t-1}} P(q_1, q_2, ..q_{t-1}, q_t = i, \mathbf{o}_1 \mathbf{o}_2 ... \mathbf{o}_t \mid \lambda) \tag{2.5}$$

and by induction

$$\delta_{t+1}(j) = \left[\max_i \delta_t(i) a_{ij}\right] \cdot b_j(\mathbf{o}_{t+1}) \tag{2.6}$$

The dynamic programming procedure can be divided into the following four steps:

1. Initialization

$$\delta_1(i) = \pi_i b_i(\mathbf{o}_1), \quad 1 \leq i \leq N \tag{2.7}$$

$$\psi_1(i) = 0. \tag{2.8}$$

2. Recursion

$$\delta_t(j) = \max_{1 \leq i \leq N} \left[\delta_{t-1}(i) a_{ij}\right] b_j(\mathbf{o}_t), \quad \begin{matrix} 2 \leq t \leq T \\ 1 \leq j \leq N \end{matrix} \tag{2.9}$$

$$\psi_t(j) = \arg\max_{1 \leq i \leq N} \left[\delta_{t-1}(i) a_{ij}\right], \quad \begin{matrix} 2 \leq t \leq T \\ 1 \leq j \leq N. \end{matrix} \tag{2.10}$$

3. Termination

$$P^* = \max_{1 \leq i \leq N} [\delta_T(i)] \tag{2.11}$$

$$q_T^* = \arg\max_{1 \leq i \leq N} [\delta_T(i)]. \tag{2.12}$$

4. Backtracking

$$q_t^* = \psi_{t+1}(q_{t+1}^*), \quad t = T - 1, T - 2, ..., 1. \tag{2.13}$$

Here, $\psi_t(j)$ is a table that keeps track of the argument that maximized Eq. (2.6). Section 3 will present how to add the k-minimum-consecutive-states constraint.

3. Constrained Viterbi

The constrained Viterbi algorithm will allow us to set the minimum number of consecutive states that can occur for each state i provided that i is in the resulting state sequence. Additionally to the functions defined in Section 2.2, we require:

$$\kappa(i) = n \in \mathbb{N}_{>0}, \quad 1 \leq i \leq N \tag{3.1}$$

which is an array containing the minimum number of consecutive states for each state i. Having $\kappa(i) = 1$ for $i = 1, 2, ..., N$ is the same as having no constraints. We also define $\zeta_t(i)$ which is a table that keeps track of the number of consecutive times that a particular state i has occurred at time t. The *Termination* and *Backtracking* steps remain the same and the *Initialization* and *Recursion* steps become:

- Initialization

$$\delta_1(i) = \pi_i b_i(\mathbf{o}_1), \quad 1 \leq i \leq N \tag{3.2}$$

$$\psi_1(i) = 0 \tag{3.3}$$

$$\zeta_1(i) = 1 \tag{3.4}$$

- Recursion

$$\delta_t(j) = \left\{ \begin{array}{ll} \delta_{t-1}(j)a_{jj}b_j(\mathbf{o}_t) & \text{if } \zeta_{t-1}(j) < \kappa(j) \\ \max\limits_{1\leq i\leq N} \left[\delta_{t-1}(i)a_{ij}\right] b_j(\mathbf{o}_t) & \text{otherwise} \\ \quad \text{subject to:} & \\ \quad i = j \text{ or} & \\ \quad \left[\zeta_{t-1}(i) \geq \kappa(i) \text{ and} \right. & \\ \quad \left. T - t + 1 \geq \kappa(j)\right] & \end{array} \right\}_{\substack{2\leq t\leq T \\ 1\leq j\leq N}} \tag{3.5}$$

$$\psi_t(j) = \left\{ \begin{array}{ll} j & \text{if } \zeta_{t-1}(j) < \kappa(j) \\ \arg\max\limits_{1\leq i\leq N} \left[\delta_{t-1}(i)a_{ij}\right] & \text{otherwise} \\ \quad \text{subject to:} & \\ \quad i = j \text{ or} & \\ \quad \left[\zeta_{t-1}(i) \geq \kappa(i) \text{ and} \right. & \\ \quad \left. T - t + 1 \geq \kappa(j)\right] & \end{array} \right\}_{\substack{2\leq t\leq T \\ 1\leq j\leq N}} \tag{3.6}$$

$$\zeta_t(j) = \left\{ \begin{array}{ll} 1 & \text{if } j \neq \psi_t(j) \\ \zeta_{t-1}(j) + 1 & \text{otherwise} \end{array} \right\}_{\substack{2\leq t\leq T \\ 1\leq j\leq N}} \tag{3.7}$$

The only change in the *Initialization* step is the addition of Eq. (3.4) which initializes all counts at time 1 to 1. In the *Recursion* step, we check if the constraint for state j is already satisfied. If not, a transition from j to j must be made until the constraint is satisfied. If the constraint for state j is already satisfied, it can be the case that transitioning from i to j for $i \neq j$ causes a constraint violation because we will reach the end of the sequence before the constraint $\kappa(j)$ can be met in which case we must transit from j to j. If that is not the case, then we can choose to make a transition from state i to j but ruling out $i's$ for which their respective constraint is not yet satisfied. The other addition is Eq. (3.7) which updates the counts in table ζ. If there is a transition from j to j then $\zeta_t(j)$ is incremented by 1. If there is a transition from i to j for $i \neq j$, then the count for j at time t is restored to 1. Note that in actual code implementation, Eq. (3.6) must occur before Eq. (3.7) since $\zeta_t(j)$ depends on $\psi_t(j)$.

Figure 1 shows an example run of the algorithm depicted as a trellis for two states A and B with $\kappa(A) = 2$, $\kappa(B) = 1$ and $T = 5$. Dotted arrows represent possible paths, whereas solid arrows are the actual chosen paths that maximized Eq. (3.6). Each node has a label representing the state and the number below it is the total count of consecutive states,

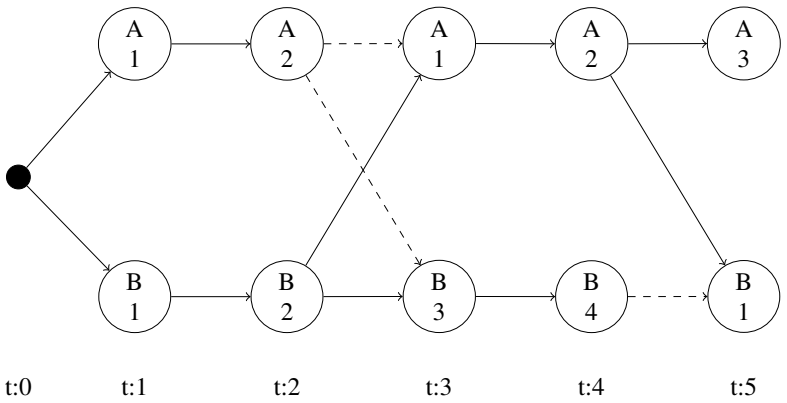

Fig. 1. Trellis showing the execution of the algorithm for two states: A and B with $\kappa(A) = 2$, $\kappa(B) = 1$. It is assumed that: $\psi_3(A) = B$, $\psi_3(B) = B$ and $\psi_5(B) = A$

i.e., $\zeta_t(i)$. At t:1, $\zeta_1(A) < \kappa(A)$ so the only possible path is going again to state A. At t:1, $\zeta_1(B) = \kappa(B)$ but we cannot have an edge going from A at t:1 to B at t:2 because $\kappa(A)$ is not yet satisfied at this point, so the only possible path is going to B. Arriving to A at t:3 can be done through A or B since both constraints are already satisfied. Assuming that $\psi_3(A) = B$, then the path coming from B was chosen and the process continues until t:5. Note that there is not an edge from B at t:4 to A at t:5 because it would not be possible to satisfy $\kappa(A)$ since it would require one more state but we are already at the end of the observation sequence. The complexity of the constrained Viterbi algorithm is $O(N^2T)$ which is the same as the unconstrained version since the conditions introduced in Eq. (3.5) and (3.6) take constant time.

4. Data Collection

A smartphone was used to collect the data from a triaxial accelerometer with a sampling rate of 50Hz. The sensor returns the acceleration value for each of the axes (x,y,z). Its maximum range is $\pm 19.60 m/s^2$. The x-axis runs parallel to the width of the smartphone, the y-axis parallel to the height of the phone and the z-axis perpendicular to its face. The smartphone was placed in the user's belt and the data collection consisted of 5 long-term activities: *commuting, working, performing home tasks, shopping and exercising*. The duration of the activities varies from about 5 minutes to a couple of hours. The total number of collected instances was 70. The data was collected by the same user since some activities are user-dependent, e.g., for person A, *working* may involve being at the office most of the time while for person B it may be more physical demanding. As a pre-processing step, the data was smoothed using an average filter with a window length = 15 Eq. (4.1).

$$v_s(t) = \frac{1}{n} \sum_{i=t-n}^{t-1} v(i) \tag{4.1}$$

where v is the original vector, v_s is the smoothed vector and n is the window length.

Then, in order to account for minor rotation variations the Euclidean Norm was computed over the three axes:

$$norm(t) = \sqrt{a_x(t)^2 + a_y(t)^2 + a_z(t)^2}, \tag{4.2}$$

where $a_x(t)^2$, $a_y(t)^2$ and $a_z(t)^2$ are the accelerations at time t.

5. Primitives Generation and Training

Primitives are the building blocks of long-term activities. A long-term activity will be represented as a sequence of primitives. These primitives will be automatically 'discovered' from the data. In the context of HMM's they will be

the observation symbols V. The process starts by segmenting the data into windows of fixed length l (we used $l = 200$ which is ≈ 4 seconds.) with an overlap of 33%. For each segment, a set of features were extracted: mean, variance and average derivative. Now each segment is represented as a feature vector also known as an instance. At this point, the instances do not have a label or class. What we want, is to put all instances into different groups such that instances that are very similar to each other belong to the same group. To accomplish this we used k-means clustering algorithm where k specifies the number of clusters in which the instances will be grouped in. Once the clustering is done, the center (also known as the centroid) of each cluster will correspond to a primitive with a unique id. This clustering step is just performed over the training data set. To assign a label/class to an instance, we get the id of the closest centroid an assign it as the instance's label.

To train the model λ, π_i was set to $1/5$ for all i. Then, activities from the training set of the same class were concatenated and B was set as:

$$b_j(k) = \frac{\text{number of primitives of class } k \text{ in } j}{\text{total number of primitives in } j}, \begin{matrix} 1 \le j \le N \\ 1 \le k \le M \end{matrix}$$

The transition probability distribution A was set by hand assuming that it is very unlikely to go from one activity to the other without passing through *commuting*. That is, if the user is doing home tasks then it is very unlikely that he will immediately start shopping without commuting first. For future work, we will use data collected during the entire day (here, we simulated an entire day by concatenating several activities) so state transition matrix A can be learned from the data. In the following transition probability matrix, a transition of one state to itself has a very high probability. A transition from a state u to *commuting* where $u \neq$ *commuting*, has a relatively high probability but going from u to another activity other than *commuting* has very low probability.

$$A = \begin{array}{c} \text{from/to} \\ \textit{commuting} \\ \textit{working} \\ \textit{home} \\ \textit{shopping} \\ \textit{exercising} \end{array} \begin{array}{c} \begin{array}{ccccc} \textit{commuting} & \textit{working} & \textit{home} & \textit{shopping} & \textit{exercising} \end{array} \\ \begin{pmatrix} 0.95 & 0.0125 & 0.0125 & 0.0125 & 0.0125 \\ 0.04 & 0.95 & 0.0033 & 0.0033 & 0.0033 \\ 0.04 & 0.0033 & 0.95 & 0.0033 & 0.0033 \\ 0.04 & 0.0033 & 0.0033 & 0.95 & 0.0033 \\ 0.04 & 0.0033 & 0.0033 & 0.0033 & 0.95 \end{pmatrix} \end{array}$$

6. Experiments and Results

Recall that a long-term activity is represented as a sequence of primitives. A *run* will be defined as a concatenation of long-term activities that occur consecutively. For the experiments, a *simulation* consists of training the model with the training set as described in Section 5 and generating random *runs* from the test set. Then, each *run* is segmented using the constrained and unconstrained Viterbi algorithm. For each *simulation*, 30% of the long-term activity instances are chosen at random to form the training set. Each *run* contains between 4 and 8 concatenated long-term activities chosen at random. The *runs* are generated by randomly choosing an instance (until the test set is empty) such that there is an activity of type *commuting* between any pair of activities. In case there are no more instances of type *commuting* they will be reused. An example *run* would be: *working-commuting-exercising-commuting-home*. The k-minimum-consecutive-sates constraint was set to $\kappa(i) = 30$ for all activities and 10 *simulations* were performed (with a total of 136 randomly generated *runs*).

The accuracy was computed as the percentage of correct state predictions. The average accuracy over all simulations when not using the constraint was 85.94% whereas it was 88.18% when using it (a difference of 2.23%). Figure 2 shows an example run. It can be seen how using the k-minimum-consecutive-states constraint helped to reduce false predictions, however there are still some errors between activities transitions since they have no clear defined boundaries.

The results were statistically validated with a paired Student's t-test where the null hypothesis is that $\mu_{constrained} - \mu_{uncontrained} = 0$ (The mean accuracy when using the constraint and when not using it is the same). The alternative hypothesis is that $\mu_{constrained} - \mu_{unconstrained} > 0$ and the significance level was set to $\alpha = 0.01$. The resulting p-value of the test was 9.8×10^{-5} which is smaller than our α, thus the null hypothesis is rejected concluding that there is a

significant difference in the segmentation accuracy when using the proposed constraint. Figure 3 shows boxplots for the simulations. Each simulation is represented as a blue dot connected to its corresponding pair.

7. Conclusions and Future Work

In this work we performed the segmentation of *long-term activities* that occur consecutively. When recognizing simple activities like walking, running, jumping, etc. it is sufficient to divide the data into fixed length windows and compute a set of features to train a classifier. For *long-term activities* this is not a feasible solution because of the great variance in their duration and the overlapping between consecutive activities. We used the Viterbi algorithm over a HMM to do the segmentation. Additionally, we added a constraint to the algorithm to take into account the prior knowledge we may have about the duration of each *long-term activity*. This addition, drives the algorithm to find state sequences that use that prior knowledge in order to discard implausible sequences. The experiments demonstrated that by adding this constraint the noise over the sate sequences was reduced and the recognition accuracy was improved. For future work, we plan to use data collected during the entire day (instead of simulating an entire day by concatenating several activities). We also plan to add more users to the experiments to see how the results are influenced because two users may tag an activity with the same label even though the activities may be completely different. One approach would be to first identify groups of similar users and then build different models for each group of users.

Acknowledgements

Enrique would like to thank Consejo Nacional de Ciencia y Tecnología (CONACYT) and the AAAmI research group at Tecnológico de Monterrey for the financial support in his PhD. studies.

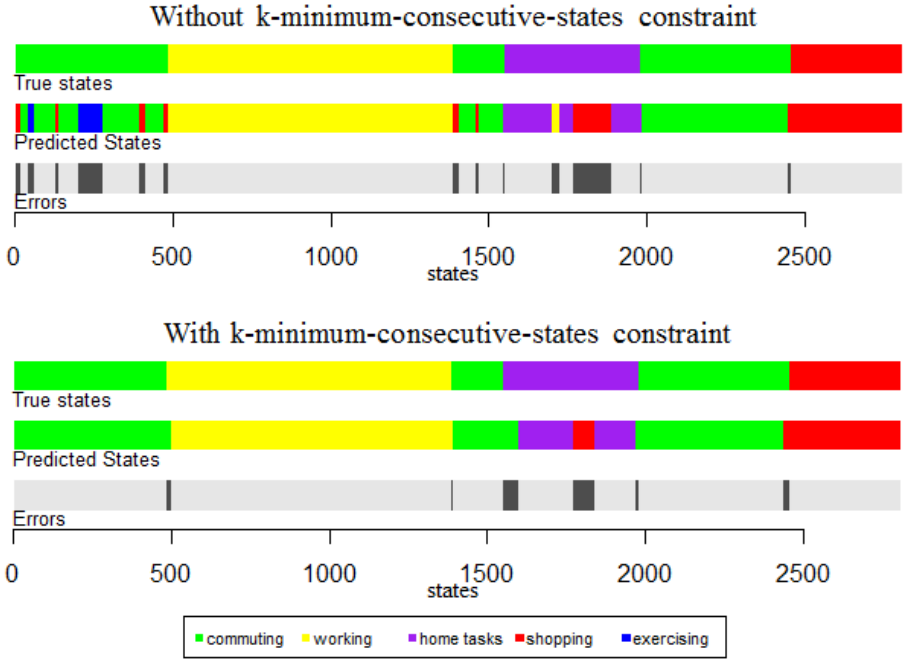

Fig. 2. Segmentation comparison without/with k-minimum-consecutive-states constraint for a given run

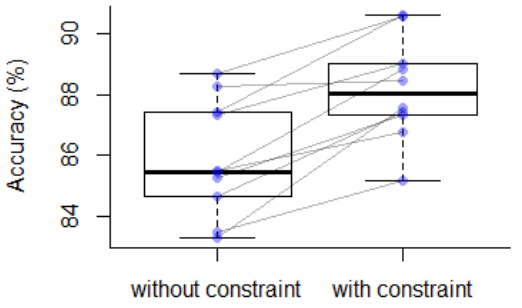

Fig. 3. Boxplot showing the accuracy for each of the simulations and its pairings

References

1. Pebble, http://getpebble.com/, accessed: 2013-09-05.
2. Jawbone up, http://jawbone.com/up/, accessed: 2013-09-05.
3. N. Ravi, N. Dandekar, P. Mysore, M. L. Littman, Activity recognition from accelerometer data, in: Proceedings of the 17th conference on Innovative applications of artificial intelligence - Volume 3, IAAI'05, AAAI Press, 2005, pp. 1541–1546.
   URL http://dl.acm.org/citation.cfm?id=1620092.1620107
4. D. Karantonis, M. Narayanan, M. Mathie, N. Lovell, B. Celler, Implementation of a real-time human movement classifier using a triaxial accelerometer for ambulatory monitoring., IEEE TRANSACTIONS ON INFORMATION TECHNOLOGY IN BIOMEDICINE 10 (1) (2006) 156 – 167.
5. T. Brezmes, J.-L. Gorricho, J. Cotrina, Activity recognition from accelerometer data on a mobile phone, in: S. Omatu, M. Rocha, J. Bravo, F. Fernández, E. Corchado, A. Bustillo, J. Corchado (Eds.), Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living, Vol. 5518 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2009, pp. 796–799.
   URL http://dx.doi.org/10.1007/978-3-642-02481-8_120
6. A. Mannini, A. M. Sabatini, Machine learning methods for classifying human physical activity from on-body accelerometers, Sensors 10 (2) (2010) 1154–1175. doi:10.3390/s100201154.
   URL http://www.mdpi.com/1424-8220/10/2/1154
7. J. R. Kwapisz, G. M. Weiss, S. A. Moore, Activity recognition using cell phone accelerometers, SIGKDD Explor. Newsl. 12 (2) (2011) 74–82. doi:10.1145/1964897.1964918.
   URL http://doi.acm.org/10.1145/1964897.1964918
8. E. J. Rechy-Ramirez, H. Hu, Stages for developing control systems using emg and eeg signals: A survey, Tech. rep., Technical Report: CES-513 in School of Computer Science and Electronic Engineering, University of Essex, United Kingdom (2011).
9. I. Witten, E. Frank, M. Hall, Data Mining: Practical Machine Learning Tools and Techniques, The Morgan Kaufmann Series in Data Management Systems, Elsevier Science, 2011.
   URL http://books.google.com.mx/books?id=5FIEAwyn9aoC
10. T. Huynh, U. Blanke, B. Schiele, Scalable recognition of daily activities with wearable sensors, in: J. Hightower, B. Schiele, T. Strang (Eds.), Location- and Context-Awareness, Vol. 4718 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2007, pp. 50–67, 10.1007/978-3-540-75160-1_4.
    URL http://dx.doi.org/10.1007/978-3-540-75160-1_4
11. T. Huynh, M. Fritz, B. Schiele, Discovery of activity patterns using topic models, in: Proceedings of the 10th international conference on Ubiquitous computing, UbiComp '08, ACM, New York, NY, USA, 2008, pp. 10–19. doi:10.1145/1409635.1409638.
    URL http://doi.acm.org/10.1145/1409635.1409638
12. T. Gu, Z. Wu, X. Tao, H. K. Pung, J. Lu, epsicar: An emerging patterns based approach to sequential, interleaved and concurrent activity recognition, Pervasive Computing and Communications, IEEE International Conference on 0 (2009) 1–9. doi:http://doi.ieeecomputersociety.org/10.1109/PERCOM.2009.4912776.
13. E. Garcia-Ceja, R. Brena, Long-term activity recognition from accelerometer data, Procedia Technology 7 (0) (2013) 248 – 256, 3rd Iberoamerican Conference on Electronics Engineering and Computer Science, CIIECC 2013. doi:10.1016/j.protcy.2013.04.031.
    URL http://www.sciencedirect.com/science/article/pii/S2212017313000327
14. M. Zhang, A. A. Sawchuk, Motion primitive-based human activity recognition using a bag-of-features approach, in: ACM SIGHIT International Health Informatics Symposium (IHI), Miami, Florida, USA, 2012, pp. 631–640.
15. A. Viterbi, Error bounds for convolutional codes and an asymptotically optimum decoding algorithm, Information Theory, IEEE Transactions on 13 (2) (1967) 260–269. doi:10.1109/TIT.1967.1054010.
16. G. D. Forney Jr, The viterbi algorithm, Proceedings of the IEEE 61 (3) (1973) 268–278.
17. L. Rabiner, B.-H. Juang, Fundamentals of speech recognition, Prentice hall, 1993.