Journal of Ambient Intelligence and Smart Environments 8 (2016) 681–695
DOI 10.3233/AIS-160406
IOS Press

A low-complexity activity classification algorithm with optimized selection of accelerometric features

Matteo Giuberti [a,*,**] and Gianluigi Ferrari [b]

[a] *Xsens Technologies B.V., Pantheon 6a, 7521 PR, Enschede, The Netherlands*
E-mail: matteo.giuberti@xsens.com

[b] *Department of Information Engineering, University of Parma, Parco Area delle Scienze, I-43123, Parma, Italy*
E-mail: gianluigi.ferrari@unipr.it

Abstract. Activity classification consists in detecting and classifying a sequence of activities, choosing from a limited set of known activities, by observing the outputs generated by (typically) inertial sensor devices placed over the body of a user. To this end, machine learning techniques can be effectively used to detect meaningful patterns from data without explicitly defining classification rules. In this paper, we present a novel Body Sensor Network (BSN)-based low complexity activity classification algorithm, which can effectively detect activities performed by the user just analyzing the accelerometric signals generated by the BSN. A preliminary (computationally intensive) training phase, performed once, is run to automatically optimize the key parameters of the algorithm used in the following (computationally light) online phase for activity classification. In particular, during the training phase, optimized subsets of nodes are selected in order to minimize the number of relevant features and keep a good compromise between performance and time complexity. Our results show that the proposed algorithm outperforms other known activity classification algorithms, especially when using a limited number of nodes, and lends itself to real-time implementation.

Keywords: Activity classification, machine learning, Body Sensor Networks, accelerometers, automatic feature selection

1. Introduction

Wireless Sensor Networks (WSNs) are attracting a relevant interest in many applications, typically associated with monitoring of particular environments. Body Sensor Networks (BSNs) are a special class of WSNs, where wireless nodes are applied to a user body in order to monitor and detect some activities, e.g., activities of daily living (ADL), performed by the user. Relevant applications of these systems include long-term remote monitoring (e.g., at home) of the activities performed by a user (e.g., elderly people or post-rehabilitation patients), typically for medical purposes [15].

Past work on BSN activity classification algorithms has relied on accelerometers placed in multiple locations over the body [2,10]. A performance improvement can be observed using multiple types of sensors [1,11,16,21]. Since the involved data are characterized by a high dimensionality and large variability, there is an inherent difficulty in determining exact classification rules. For such reason, machine learning and data mining techniques have gained an increasing interest due to their strength in "learning" ad-hoc rules and detecting significant patterns, provided that some data are given to the algorithm for "training" purposes. Regardless of the considered type of sensor, an activity classification algorithm is indeed generally composed of two phases: a *training* phase, typically used for cal-

*Corresponding author. E-mail: matteo.giuberti@xsens.com.

**Matteo Giuberti is with Xsens Technologies B.V. since April 2014. This work was performed while he was at the University of Parma.

ibration and parameters estimation purposes; and an *online* (classification) phase, possibly executed in real time. The training phase aims at identifying activity-specific features from the signals generated at each sensor, after manual [11] or automatic [9,20,22] signal segmentation. Regarding classification, most of the works in the literature tend to adopt thresholding or to use k-Nearest Neighbors (k-NN) algorithms, because of their simplicity and applicability on low-cost mobile devices [10,16]. However, more sophisticated machine learning techniques have also been considered, such as those based on the use of decision trees [2], support vector machines [6], or hidden Markov models [13,21]. Furthermore, machine learning techniques for activity classification typically require also the development of robust methods to address issues such as feature selection and classification [17], decision fusion and fault-tolerance [3,19,23].

In this work, we design a novel low-complexity automatic BSN-based activity classification algorithm, which aims at detecting and classifying a sequence of activities, choosing from a list of known activities, by observing accelerometric data. A preliminary training phase is used to automatically optimize key parameters of the algorithm. The goal of the training phase is that of selecting a proper subset of nodes in order to minimize the number of relevant features, yet guaranteeing an accurate activity classification degree. The classification performance of the proposed algorithm is analyzed using publicly available experimental data [14] (in part generated in the context of the Opportunity Challenge [4,18]), thus providing a valid and unbiased benchmark for comparisons with other algorithms. In particular, the classification algorithm is tested on four activities, which correspond to different locomotion modes: stand, walk, sit, and lie. The proposed algorithm outperforms, especially when using a limited number of nodes, other known low-complexity algorithms, such as the k-NN, the Nearest Centroid Classifier (NCC), the Linear Discriminant Analysis (LDA), and the Quadratic Discriminant Analysis (QDA) [4,5,12]. The obtained results are very promising, making the proposed algorithm suitable for real-time activity monitoring applications.

The rest of this paper is structured as follows. In Section 2, the experimental setup and the performance metric are preliminary introduced, followed by the derivation of the proposed algorithm. Section 3 is dedicated to performance analysis. Finally, in Section 4 concluding remarks are given.

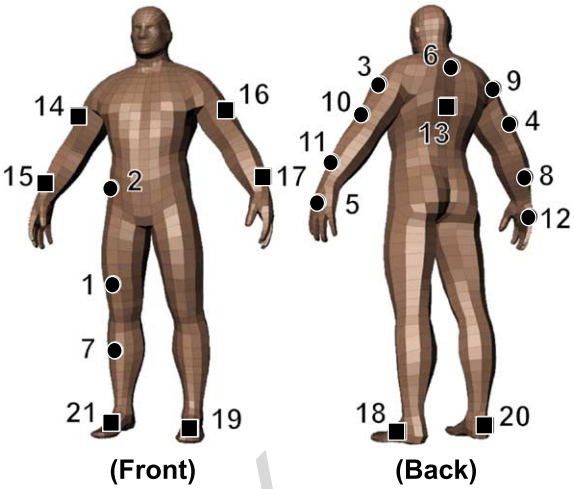


Fig. 1. Opportunity Challenge setup [4,18]. The position of accelerometers (●) and inertial measurement units (■) is highlighted.

2. Method

2.1. Experimental setup and performance metrics

As anticipated in Section 1, the experimental data used to test our algorithm are shared data collected in the context of the European project Opportunity and provided for the so-called Opportunity Challenge [4,14,18]. Figure 1 shows the experimental configuration of the sensor nodes in the considered BSN. The output of the BSN consists of mainly accelerometric data, integrated, for some nodes, with gyroscopic and magnetometric data – in this work, in order to guarantee a low hardware complexity (and, consequently, a low cost and large battery life of the whole activity classification system), only accelerometric data will be used.[1] For data collection, different users were asked to perform sequences of consecutive predefined movements and naturally executed daily activities. This allowed to generate highly realistic data upon which robust and flexible algorithms could be developed and trained. Data were recorded with a sampling rate $f_{\text{samp}} = 30$ Hz.

Given a discrete set of predefined activities $\mathcal{A} = \{a_1, a_2, \ldots, a_A\}$ (with cardinality $|\mathcal{A}| = A$), the metric which will be used to evaluate the performance of

[1] To this end, note that the cost of a gyroscope (or of a magnetometer) chip is typically at least twice that of an accelerometer chip. Similar considerations hold for power consumption requirements, as accelerometers require indeed much less power than gyroscopes and magnetometers do. This has obviously a strong impact on the battery lifetime.

the proposed classification algorithm is the weighted f1 score [4], denoted as $F1^{\mathrm{w}}$ and evaluated as follows:

$$F1^{\mathrm{w}} = \sum_{a=1}^{A} 2 * \left(\frac{n_a}{N}\right) * F1_a$$

$$= \sum_{a=1}^{A} 2 * \left(\frac{n_a}{N}\right) * \left(\frac{prec_a * rec_a}{prec_a + rec_a}\right)$$

where: a is the considered activity; n_a is the number of samples of the inertial data sequence in correspondence to which a user is performing activity a; N is the total number of samples of the collected inertial data sequence; $F1_a$ is the f1 score computed for activity a; and $prec_a$ and rec_a are, respectively, the precision (defined as TP/(TP + FP), where TP and FP are the numbers of true positives and false positives) and the recall (defined as TP/(TP + FN), where FN is the number of false negatives) evaluated for activity a. In practice, $F1^{\mathrm{w}}$ is the average f1 score over all activities. Note that, beside these metrics, in the *refinement* step of the proposed algorithm (which is described in Section 2.2.4) two other metrics will also be considered, namely: the accuracy, defined as (TP + TN)/(P + N), where TP, TN, P, and N are, respectively, the numbers of true positives, true negatives, positives, and negatives; and the specificity, defined as TN/(FP + TN), where TN and FP are the numbers of true negatives and false positives. For reproducibility purposes, the performance of the proposed algorithm is evaluated for multiple datasets collected from different subjects. To this end, in the Opportunity dataset, for each subject, golden labels are provided, containing a single stream of (manually) labeled activities.[2] In the proposed algorithm, which is described in the following subsections, we will only make use of accelerometric data.

2.2. Algorithm description

Generally, an activity classification problem leads to the design of an algorithm that can estimate the sequence of occurrences of specific activities choosing from a discrete set of predefined activities $\mathcal{A}$, directly working on (typically) inertial signals (e.g., accelerometric and gyroscopic signals). The proposed algorithm tries to concentrate most of its complexity in a (offline) *training* phase, which is performed once and aims at selecting the smallest subset of nodes to extract the smallest, yet sufficient, number of accelerometric time features to guarantee a good performance (high $F1^{\mathrm{w}}$). Then, upon optimized setting of proper thresholds, the *online* phase is relatively light (in terms of time complexity), making real-time activity classification applications feasible.

In the following, a detailed description of the operational steps of the proposed algorithm are presented. After preliminaries on data preprocessing, the online and training phases are presented. In order to run properly, the online activity classification algorithm needs some parameters that have to be estimated and optimized during the training phase. Even though, practically, the training phase precedes the online phase, in the remainder of this subsection, after preliminaries on data processing and feature extraction, we first describe in detail, for ease of presentation, the online phase. In the training phase, the same steps of the online phase are considered, with the only difference that known (labeled) data are used to tune the key parameters of the algorithm, which are then kept constant in the (following) online phase.

2.2.1. Preliminaries on data preprocessing and feature extraction

At each node, an accelerometer outputs a stream of three-dimensional data, which corresponds to the accelerations measured by the sensor in its three reference axes. More formally, let us define the three-dimensional acceleration vector, measured at the i-th epoch, as

$$\widetilde{\boldsymbol{\alpha}}_{\boldsymbol{i}} = (\widetilde{\alpha}_{xi}, \widetilde{\alpha}_{yi}, \widetilde{\alpha}_{zi}) \quad i \in \{1, 2, \ldots, N\}$$

where N is the number of samples in the stream. For the sake of simplicity, let us assume that the accelerometer is already calibrated and, thus, $\widetilde{\alpha}_{xi}$, $\widetilde{\alpha}_{yi}$, and $\widetilde{\alpha}_{zi}$ are expressed in g units. Furthermore, the accelerometric data are low-pass filtered in order to deal with smoother data in both training and online phases. To this end, the recursive low-pass filtering used for each accelerometric data stream is the same used in [7] and characterized by the following equation:

$$y_i = \begin{cases} x_1 & i = 1 \\ 0.8 \cdot y_{i-1} + 0.1 \cdot (x_i + x_{i-1}) & 2 \leqslant i \leqslant N \end{cases} \quad (1)$$

[2] Note that, while evaluating the algorithm performance, the samples containing "undefined" activities (e.g., transitions between classifiable activities), because of the temporal continuity of the collected data, are not taken into account for the evaluation of $F1^{\mathrm{w}}$. For the sake of completeness, we remark that this does not mean that the algorithm is not dealing with data coming from "unknown" classes. In fact, the algorithm is not forced to always choose at least one of the considered classes and it may actually happen that for some time epoch no activities are detected at all. This will be clearer by reading the algorithm description in Section 2.2.

where x_i is the i-th sample of the data stream ($x \in \{\widetilde{\alpha}_x, \widetilde{\alpha}_y, \widetilde{\alpha}_z\}$), y_i is the i-th sample of the filtered data stream ($y \in \{\alpha_x, \alpha_y, \alpha_z\}$). The cut-off frequency f_{co} of the low-pass filter defined in (1) is $f_{\text{co}} \approx \frac{f_{\text{samp}}}{30} = 1$ Hz. Therefore, we denote the filtered signal as

$$\boldsymbol{\alpha_i} = (\alpha_{xi}, \alpha_{yi}, \alpha_{zi}) \quad i \in \{1, 2, \ldots, N\}$$

and the norm of $\boldsymbol{\alpha_i}$ as

$$\overline{\alpha}_i = |\boldsymbol{\alpha_i}| = \sqrt{\alpha_{xi}^2 + \alpha_{yi}^2 + \alpha_{zi}^2} \quad i \in \{1, 2, \ldots, N\}.$$

Starting from the filtered signal, at the i-th epoch, two types of simple features are of interest and can be extracted. The first one, denoted as *p-feature* (where "p" stands for "parallel") and indicated with $acc_i^{(\text{p})}$, is a properly chosen component of the normalized[3] (to unity) acceleration vector, i.e., $acc_i^{(\text{p})} \in \{\alpha_{xi}/|\boldsymbol{\alpha_i}|, \alpha_{yi}/|\boldsymbol{\alpha_i}|, \alpha_{zi}/|\boldsymbol{\alpha_i}|\}$. In particular, the chosen component is the one parallel to the "main" axis of the related body segment. For instance, considering Fig. 1: for node 1, $acc_i^{(\text{p})}$ is the acceleration value measured along the femur direction; for node 7, the one measured along the tibia. Observe that, due to the normalization, $acc_i^{(\text{p})}$ can only assume real values in $[-1, +1]$.

The second considered feature, denoted as *dev-feature* and indicated with $\sigma_i(s)$, is the standard deviation of the norm of the acceleration computed on a sliding window (with fixed length $L = 2 \cdot s + 1$ and centered at the i-th sample[4]) that runs over all the acceleration samples. Neglecting border effects (the extension is straightforward, as shown in [8]), $\sigma_i(s)$ can be expressed as follows:

$$\sigma_i(s) = \sqrt{\frac{\sum_{k=i-s}^{i+s} [\overline{\alpha}_k - \mu_k(s)]^2}{L-1}} \tag{2}$$

where

$$\mu_k(s) \triangleq \frac{\sum_{k=i-s}^{i+s} \overline{\alpha}_k}{L-1}. \tag{3}$$

It can be observed that $\sigma_i(s)$ is always larger than or equal to 0 – typically, it is not considerably larger than 1 (due to the expression of the accelerometric data in g units). While evaluating the classification performance of the proposed algorithm, s is set to 7 samples and, therefore, the dev-features are computed on a window of length $L = 15$ samples (i.e., 0.5 s).

2.2.2. *The general idea*

Considering a single activity $a \in \mathcal{A}$, the output of an activity classification algorithm is given by a binary sequence $\boldsymbol{w}(a) = (w_1(a), w_2(a), \ldots, w_N(a))$ where $w_i(a) = 0$ if a is not detected at epoch i (the index i runs over the samples of the collected accelerometric data sequence) or $w_i(a) = 1$ if a is detected. In particular, $\boldsymbol{w}(a)$ contains "activity windows" (disjoint groups of consecutive "1"s), within which the activity a has been detected. The length of each activity window is recursively determined – using three parameters ℓ_1, ℓ_2, and ℓ_3, optimized independently for each activity. This process will be described later.

For each considered activity, the proposed activity classification algorithm aims at automatically selecting the best nodes in the BSN and the corresponding most significant feature types, that can best discriminate the occurrence of the considered activity (e.g., the thigh node is intuitively one of the best nodes to estimate a "sit" activity, whereas the feet nodes give relevant information about the "walk" activity). More formally, let us define as $\mathcal{F} = \{f_1, f_2, \ldots, f_F\}$ (with cardinality $|\mathcal{F}| = F$) a set of features where the generic feature $f \in \mathcal{F}$ corresponds to a feature "type" (p-feature or dev-feature) associated with a specific node. As an example, referring to Fig. 1, $f =$ 1p is the p-feature extracted from the thigh node (i.e., node 1) and $f =$ 7d is the dev-feature extracted from the tibia node (i.e., node 7). In the following, when referring to the value of a feature, we will implicitly refer to the value of the "embedded" feature type. Furthermore, given a specific activity a, $\mathcal{C}_a^* \subset \mathcal{F}$ represents the subset of the most significant features used to classify activity a.[5]

The proposed algorithm identifies activities by properly thresholding the selected features. In particular, referring to a given feature f, an activity a is consid-

[3] Note that, by normalizing the acceleration vector (for the p-feature), we are implicitly assuming that no linear acceleration is present (i.e., the device is still) and only the gravity acceleration contribution (i.e., 1 g) is measured by the accelerometer. Note that, although this assumption is not always true, in the majority of human movements it is true that the magnitude of the linear acceleration is rather lower than that of the gravity component.

[4] More details about the definition of the window, especially at the borders of the acceleration signal, are given in [8].

[5] Note that, throughout this paper, the term "optimal" and the superscript "*" are equivalently used with reference to the tunable parameters considered in the proposed algorithm. Furthermore, if not stated otherwise, the optimality is always intended in terms of classification performance through the f1 score (when activities are independently considered) and the weighted f1 score (when activities are combined together).

ered as detected at the samples in correspondence to which the feature (type) is between the (lower and upper) thresholds $t_1(a, f)$ and $t_2(a, f)$, which are specifically derived and optimized independently for each activity a and for each feature f. The optimal values $\{t_1^*(a, f)\}_{a\in\mathcal{A}, f\in\mathcal{C}_a^*}$ and $\{t_2^*(a, f)\}_{a\in\mathcal{A}, f\in\mathcal{C}_a^*}$ are determined in the training phase, as will be described later.

In order for the activity classification algorithm to output a single sequence $\boldsymbol{w} = (w_1, w_2, \ldots, w_N)$ (where $w_i = 0$ if no activities are detected and $w_i = a$ if activity a is detected, i.e., $w_i(a) = 1$), since $w_i(a_j) = w_i(a_k) = 1$ may happen for some $i \in \{1, 2, \ldots, N\}$ and for some $j \neq k$, with $j, k \in \{1, 2, \ldots, A\}$, different priorities need to be assigned to the activities. The priorities are denoted as $\boldsymbol{q} = (q(a_1), q(a_2), \ldots, q(a_A))$, where each element (which indicates the priority of each considered activity) can assume real values in [0, 1]. The priorities' list $\boldsymbol{q}$ must be interpreted as follows: if $q(a_i) > q(a_j)$, with $i, j \in \{1, 2, \ldots, A\}$, activity a_i has a higher priority than activity a_j. The priority assigned to an activity is strongly related to the confidence in correctly detecting that activity and is based on the evaluation of specific performance metrics, which are evaluated in the training phase, as will be described later. In this way, the output sequence $\boldsymbol{w}$ is an $(A+1)$-ary sequence of activity labels, where the label of the activity with highest priority is selected in the cases where more than one activity is detected at the same epoch. The optimal priorities' list $\boldsymbol{q}^*$ is determined in the training phase, as described later.

2.2.3. Activity classification

The online phase of the proposed algorithm is based on three main steps: (i) a first *coarse classification* step; (ii) a *refinement* step; (iii) and a final *priority-based activity combination* step.

Concerning the first coarse classification step, which is executed independently for each activity to be classified, a specific activity a is detected at epoch i (i.e., $w_i(a) = 1$) if *all* the features belonging to $\mathcal{C}_a^*$ are comprised between the corresponding optimal thresholds $\{t_1^*(a, f)\}_{f\in\mathcal{C}_a^*}$ and $\{t_2^*(a, f)\}_{f\in\mathcal{C}_a^*}$.

In the first coarse classification step, a single occurrence of an activity can be missed (because of little pauses or random movements). This can be avoided, or at least mitigated, by applying to the detected activity windows a refinement step which takes into account the length of the estimated activity windows. More specifically, given a specific activity a, in our implementation the refinement step is based on the following sequential operations: (1) every "null window" (a group of consecutive "0"s), with a length (in terms of number of samples) shorter than ℓ_1^*, is switched to an activity window (and incorporated in the preceding and following activity windows, which are then fused together); (2) every activity window with a length shorter than ℓ_2^* is turned into a null window; (3) step 1 is repeated considering now every null window with a length shorter than ℓ_3^*.

Finally, the priority-based activity combination step consists in combining the sequences $\boldsymbol{w}(a_1), \boldsymbol{w}(a_2), \ldots,$ and $\boldsymbol{w}(a_A)$ into a single sequence $\boldsymbol{w}$ of activity labels, on the basis of the optimal priorities $\boldsymbol{q}^*$ assigned to the activities.

2.2.4. Algorithm training

In order to work effectively, the proposed algorithm needs to be properly trained by exploiting the part of collected data for which the occurrences of the activity of interest are correctly (manually) labeled. The training phase aims at estimating the optimal values of the key parameters that will be used in the online phase. In particular, *for each activity a* the following parameters need to be estimated: (i) the optimal subset of features $\mathcal{C}_a^*$, along with the corresponding optimal thresholds $\{t_1^*(a, f)\}_{f\in\mathcal{C}_a^*}$ and $\{t_2^*(a, f)\}_{f\in\mathcal{C}_a^*}$; (ii) the optimal thresholds ℓ_1^*, ℓ_2^*, and ℓ_3^*, used to refine the estimated activity windows; (iii) the optimal activities' priorities list $\boldsymbol{q}^*$, *over all activities*.

A detailed flow diagram of the implementation steps of the training phase of the proposed algorithm is shown in Figs 2, 3, and 4 and will be now described, distinguishing between its three component blocks (the same of the online phase): (i) the first *coarse classification* step; (ii) the *refinement* step; and (iii) the final *priority-based activity combination* step.

Coarse classification At the beginning of the coarse classification step, shown in Fig. 2 and executed for each activity a, the set $\mathcal{F} = \{f_1, f_2, \ldots, f_F\}$, whose elements are all the initial features, must be (manually) defined. Similarly, the set $\mathcal{S}$ (with cardinality $|\mathcal{S}| = S$) accounts for different ways (each way is identified by a partition $s = (\mathcal{S}_\mathrm{T}/\mathcal{S}_\mathrm{O})$) of separating the (training) dataset into the two subsets $\mathcal{S}_\mathrm{T}$ and $\mathcal{S}_\mathrm{O}$ (s will be denoted as "separation"). Finally, $\mathcal{R}$ (with cardinality $|\mathcal{R}| = R$) accounts for possible values of "confidence," which will be given in terms of a given percentage of the Probability Mass Function (PMF) characterizing the values of the selected feature for the activity of interest, considered to derive the thresholds $\{t_1(a, f)\}_{a\in\mathcal{A}, f\in\mathcal{F}}$ and $\{t_2(a, f)\}_{a\in\mathcal{A}, f\in\mathcal{F}}$ – more de-

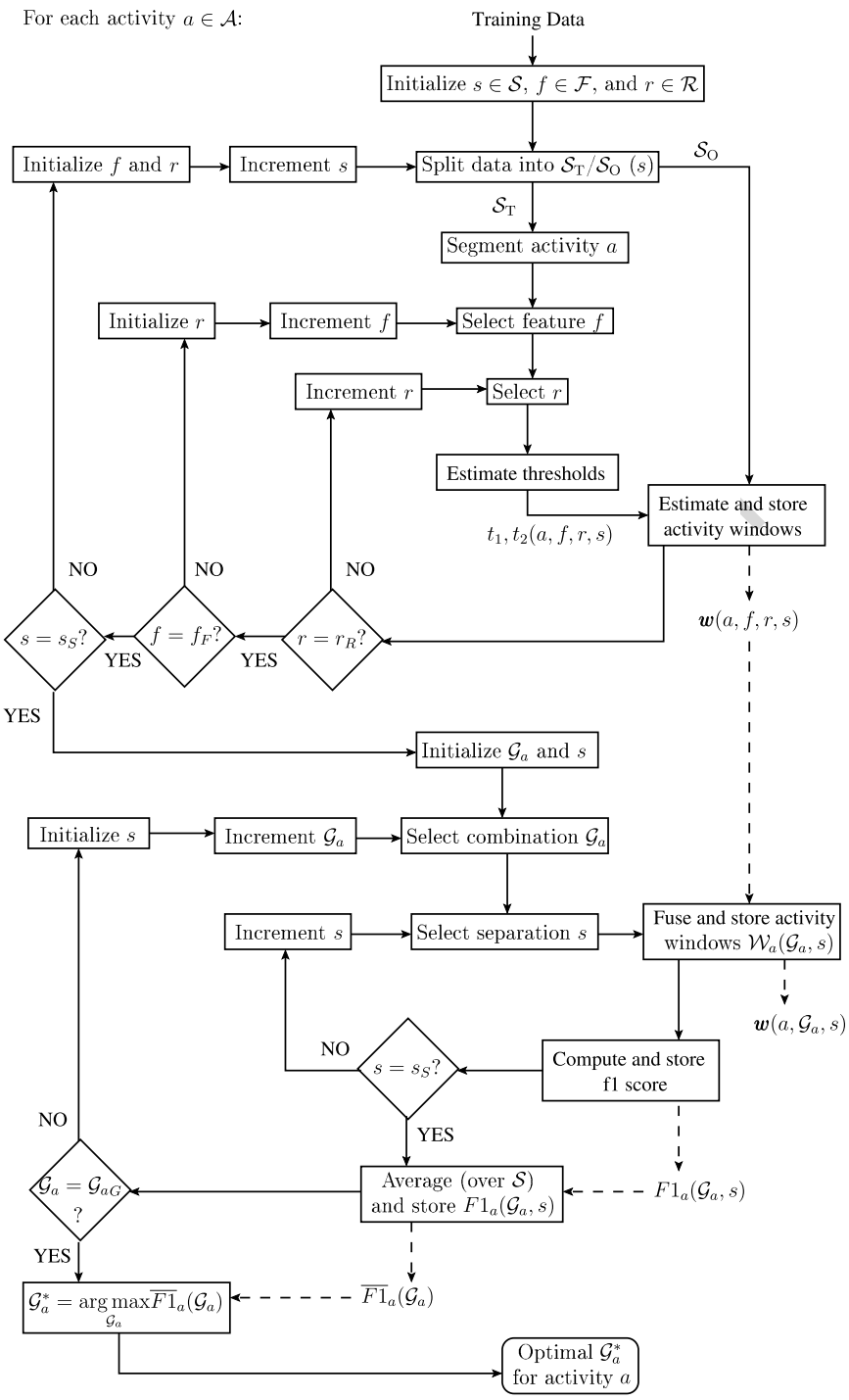


Fig. 2. Detailed flow diagram of the implementation steps of the proposed algorithm's training phase: the first *coarse classification* step.

tails about confidence are provided below with an example. Note that the separation into the two subsets $\mathcal{S}_\mathrm{T}$ and $\mathcal{S}_\mathrm{O}$ is done in order to cross-validate the statistical performance of the algorithm, i.e., to assess how the algorithm's performance will generalize with an independent dataset. In the machine learning field, this is a

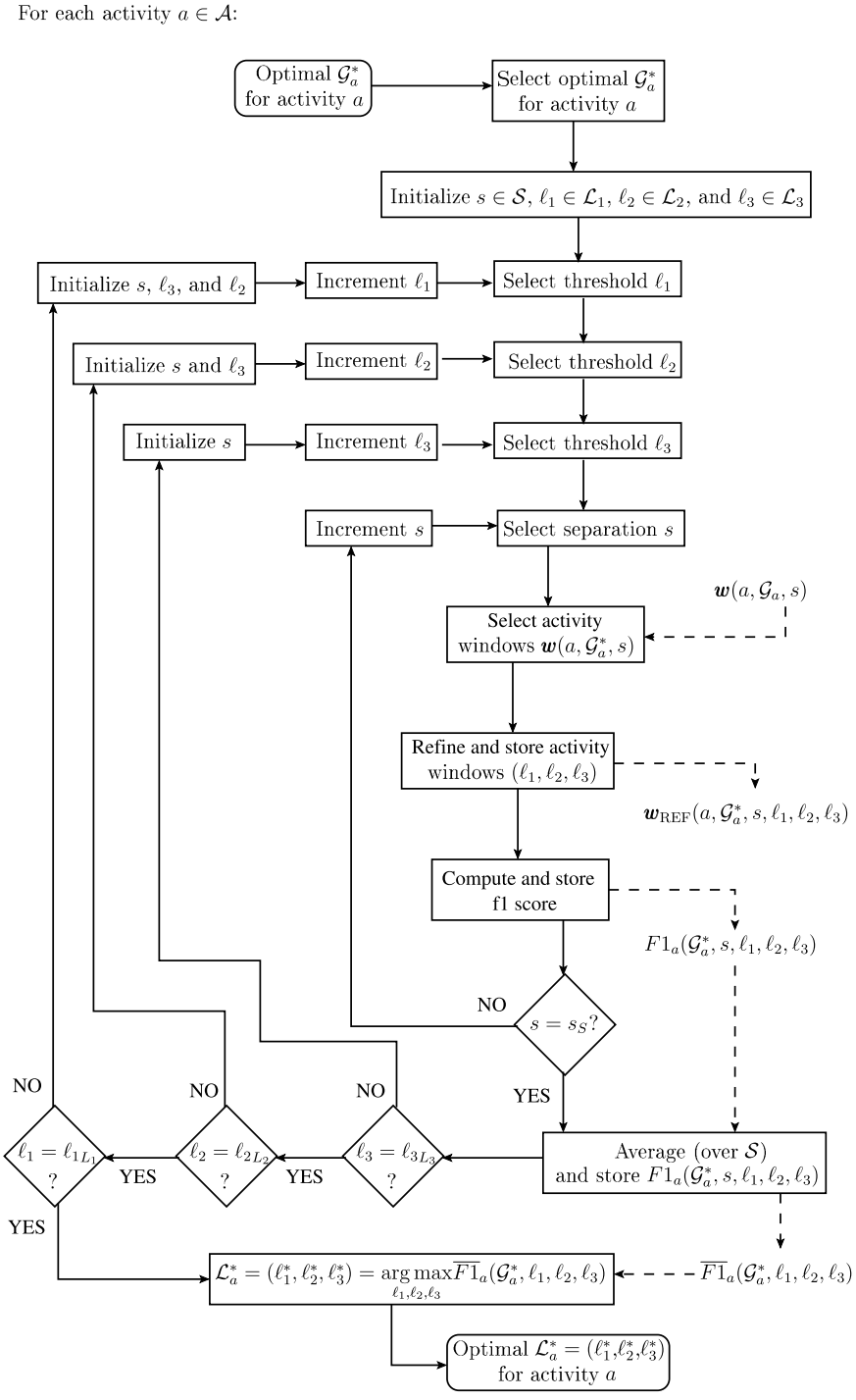


Fig. 3. Detailed flow diagram of the implementation steps of the proposed algorithm's training phase: the *refinement* step.

common practice used to derive unbiased performance of an algorithm, especially when only training data are available [5].

The goal of the first part of the coarse classification step is to estimate and store the activity windows $\{\boldsymbol{w}(a, f, r, s)\}_{a \in \mathcal{A}, f \in \mathcal{F}, r \in \mathcal{R}, s \in \mathcal{S}}$. As already ex-

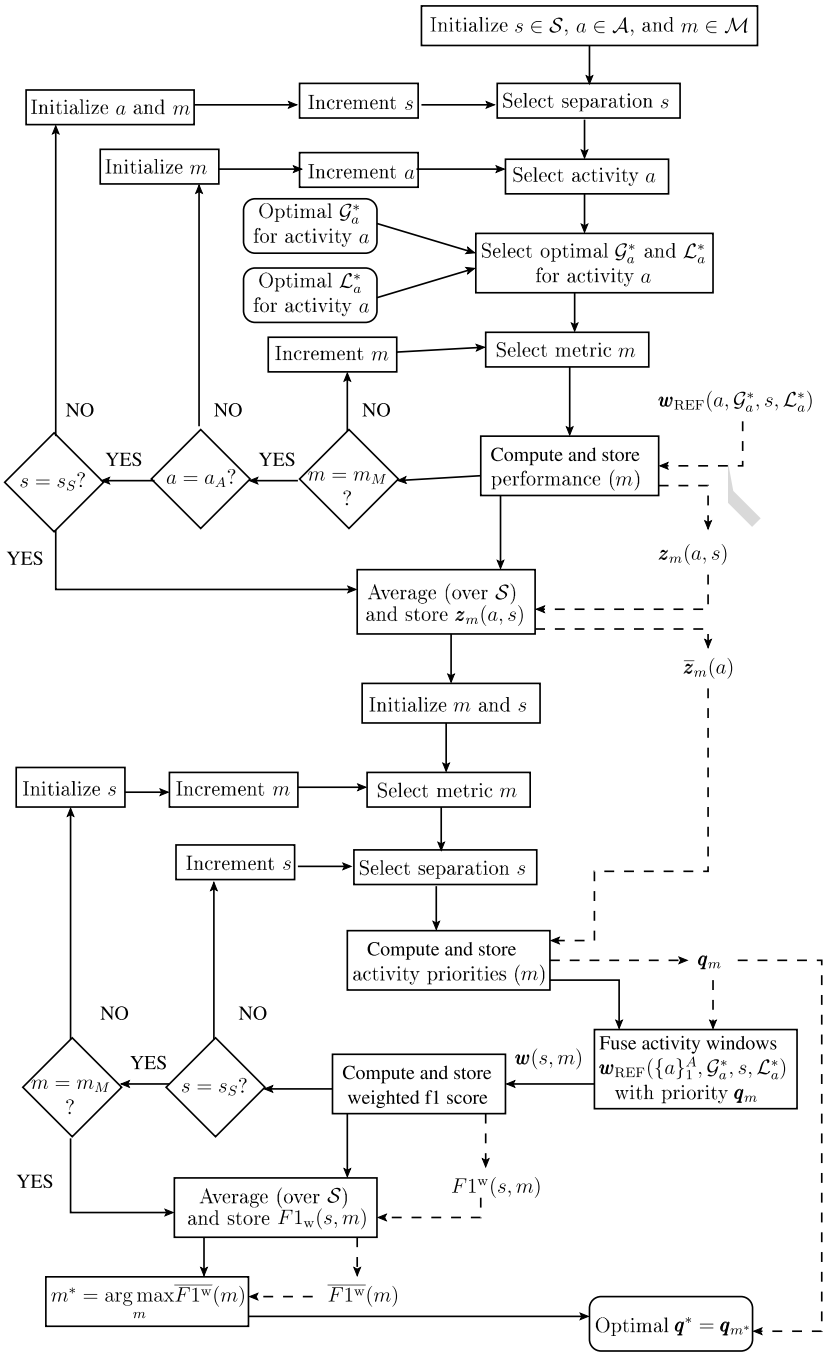


Fig. 4. Detailed flow diagram of the implementation steps of the proposed algorithm's training phase: the final *priority-based activity combination* step.

plained in Section 2.2.3, the activity windows are derived, for a given activity a, by applying to each feature $f \in \mathcal{F}$ (considering the dataset $\mathcal{S}_{\mathrm{O}}$ associated with the selected s) the (properly estimated) thresholds $t_1(a, f, r, s)$ and $t_2(a, f, r, s)$ – unlike in the online phase, t_1 and t_2 depend not only on a and f, but also on s (as, obviously, the thresholds depend on the used data-set) and r (as the "width" between the thresh-

olds depend on the desired confidence – the larger, the higher the required confidence).

In order to clarify the process which leads to the identification of the thresholds for activity classification, we consider an example relative to the acceleration signals produced at the thigh node (i.e., node 1 in Fig. 1) and used to detect the "sit" activity. An illustrative description of the thresholds estimation step is shown in Fig. 5: the p-feature (i.e., $f = 1\text{p}$) and the dev-feature (i.e., $f = 1\text{d}$), shown in Fig. 5(b), are first extracted from the acceleration signals (shown in Fig. 5(a), in the x, y, and z axes) and evaluated discretely in the "sit" intervals (assuming that $a =$ sit and for a given $s \in \mathcal{S}$) in order to obtain their PMFs in such intervals (the PMFs of the p-feature and the dev-feature are shown in Figs 5(c) and 5(d), respectively). Given a, f, s, for each value of $r \in \mathcal{R}$, the thresholds $t_1(a, f, r, s)$ and $t_2(a, f, r, s)$ are then chosen as the extremes of the shortest interval (t_1, t_2) which comprises at least $r\%$ of the probability mass.

The second part of the coarse classification step, still performed independently for each $a \in \mathcal{A}$, is based on the fusion of different combinations of the previously stored activity windows $\{\boldsymbol{w}(a, f, r, s)\}_{f\in\mathcal{F}, r\in\mathcal{R}, s\in\mathcal{S}}$. The goal is to find, for each activity, the optimal combination $\mathcal{G}_a^*$ (among the G possible combinations), where the generic combination $\mathcal{G}_a$ is associated with a possible subset of features $\mathcal{C}_a \subset \mathcal{F}$ and two values of r, i.e., r_p and r_dev, from which the thresholds $t_1(a, f, r, s)$ and $t_2(a, f, r, s)$ can be estimated when the considered f is, respectively, a p-feature or a dev-feature. An example, a possible instance of $\mathcal{G}_a$ could be ($\mathcal{C}_a = \{1\text{d}, 1\text{p}, 4\text{p}\}$, $r_\text{p} = 93\%$, $r_\text{dev} = 97\%$).

Given a combination $\mathcal{G}_a$ and a separation $s \in \mathcal{S}$, the set $\mathcal{W}_a(\mathcal{G}_a, s)$ is defined as the subset of $\{\boldsymbol{w}(a, f, r, s)\}_{f\in\mathcal{F}, r\in\mathcal{R}}$ with $(f, r) \in \mathcal{G}_a$. The sequences of activity windows in $\mathcal{W}_a(\mathcal{G}_a, s)$ are then fused together (and stored for future use) into a single sequence of activity windows $\boldsymbol{w}(a, \mathcal{G}_a, s)$, obtained from the "logical AND" of all the considered sequences of activity window. The f1 score of $\boldsymbol{w}(a, \mathcal{G}_a, s)$, denoted as $F1_a(\mathcal{G}_a, s)$, is then computed and stored. By averaging $\{F1_a(\mathcal{G}_a, s)\}_{s\in\mathcal{S}}$ over all possible separations in $\mathcal{S}$ and denoting this average as $\overline{F1}_a(\mathcal{G}_a) = \sum_{s\in\mathcal{S}} F1_a(\mathcal{G}_a, s)/S$, the optimal $\mathcal{G}_a^*$ is selected as follows:

$$\mathcal{G}_a^* = \arg\max_{\mathcal{G}_a} \overline{F1}_a(\mathcal{G}_a). \tag{4}$$

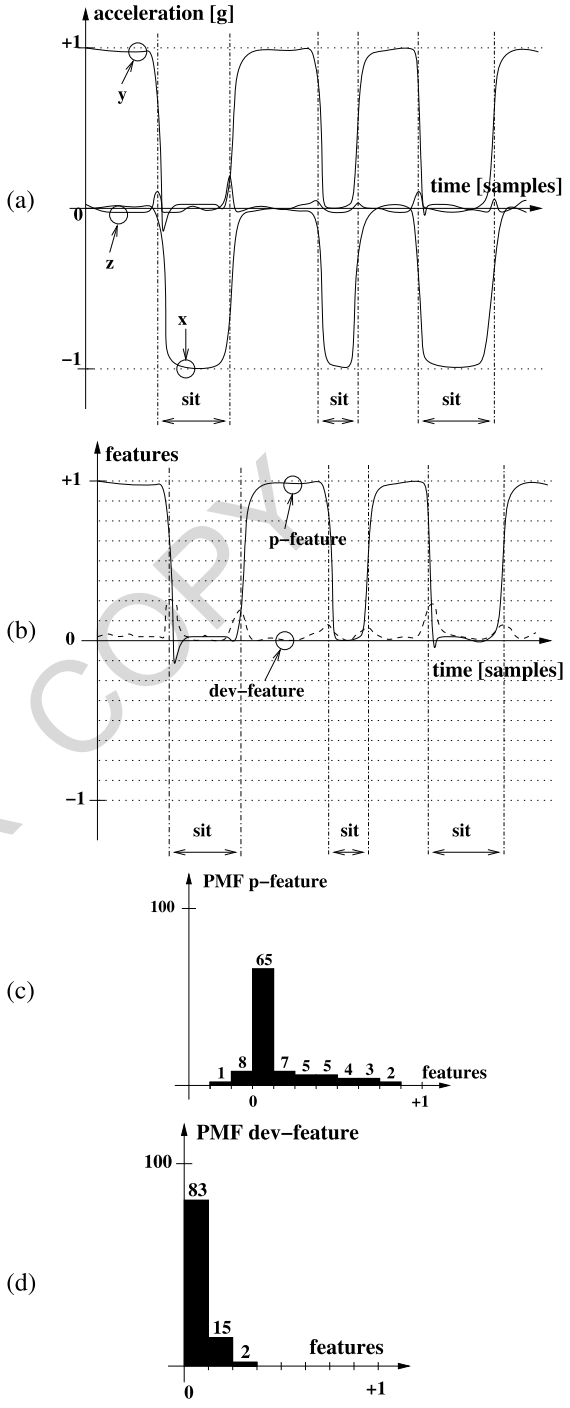


Fig. 5. Description of the thresholds estimation step, which appears in Fig. 2, for an illustrative acceleration signal produced at the thigh node: (a) acceleration signal (the x, y, and z components are highlighted); (b) p-feature (straight line) and dev-feature (dashed line) extracted from the previous acceleration signal; PMFs of (c) p-feature and (d) dev-feature evaluated in the "sit" intervals.

Refinement The next refinement step, shown in Fig. 3 and executed for each activity a, retraces the operations of the previous step in order to estimate the optimal thresholds $\mathcal{L}_a^* = (\ell_1^*, \ell_2^*, \ell_3^*)$ used to refine the previously estimated activity windows, but considering now just the optimal combination $\mathcal{G}_a^*$. To this end, three sets $\mathcal{L}_1 \subset \mathbb{N}$, $\mathcal{L}_2 \subset \mathbb{N}$, $\mathcal{L}_3 \subset \mathbb{N}$ are defined.[6] Given a separation $s \in \mathcal{S}$, all possible combinations of $\ell_1 \in \mathcal{L}_1$, $\ell_2 \in \mathcal{L}_2$, and $\ell_3 \in \mathcal{L}_3$ are used to refine the previously estimated sequence of activity windows $\boldsymbol{w}(a, \mathcal{G}_a^*, s)$ as already described in Section 2.2.3. For every considered combination of $\ell_1 \in \mathcal{L}_1$, $\ell_2 \in \mathcal{L}_2$, and $\ell_3 \in \mathcal{L}_3$, the refined sequences of activity windows $\{\boldsymbol{w}_{\text{REF}}(a, \mathcal{G}_a^*, s, \ell_1, \ell_2, \ell_3)\}_{s\in\mathcal{S}}$ are then stored (for future use) and their f1 scores $\{F1_a(\mathcal{G}_a^*, s, \ell_1, \ell_2, \ell_3)\}_{s\in\mathcal{S}}$ are computed and stored. By averaging $\{F1_a(\mathcal{G}_a^*, s, \ell_1, \ell_2, \ell_3)\}_{s\in\mathcal{S}}$ over all possible separations in $\mathcal{S}$ and denoting this average as $\overline{F1}_a(\mathcal{G}_a^*, \ell_1, \ell_2, \ell_3) = \sum_{s\in\mathcal{S}} F1_a(\mathcal{G}_a^*, s, \ell_1, \ell_2, \ell_3)/S$, the optimal thresholds $\mathcal{L}_a^*$ are selected as follows:

$$\mathcal{L}_a^* = (\ell_1^*, \ell_2^*, \ell_3^*) = \underset{(\ell_1,\ell_2,\ell_3)}{\arg\max}\, \overline{F1}_a(\mathcal{G}_a^*, \ell_1, \ell_2, \ell_3). \tag{5}$$

Priority-based activity combination During the final step, shown in Fig. 4, the optimal list of activities' priorities $\boldsymbol{q}^*$ is estimated. To this end, a set of performance metrics $\mathcal{M} = \{m_1, m_2, \ldots, m_M\}$ (where $|\mathcal{M}| = M$) is considered. Specifically, in this work we consider the following performance metrics (defined as in Section 2.1): (i) f1 score, (ii) precision, (iii) recall, (iv) specificity, and (v) accuracy. First, given an activity a and a separation s, every performance metric $m \in \mathcal{M}$ is used to evaluate the performance associated with the sequence of activity windows $\boldsymbol{w}_{\text{REF}}(a, \mathcal{G}_a^*, s, \mathcal{L}_a^*)$ and the corresponding value $z_m(a, s)$ is stored. For given activity a and metric m, the obtained values $\{z_m(a, s)\}_{s\in\mathcal{S}}$ are then properly averaged over all possible separations $s \in \mathcal{S}$ and the resulting $\overline{z}_m(a) = \sum_{s\in\mathcal{S}} z_m(a, s)/S$ is used to compose (and store) a priority-based list of activities $\boldsymbol{q}_m = (q_m(a_1), q_m(a_2), \ldots, q_m(a_A)) \triangleq (\overline{z}_m(a_1), \ldots, \overline{z}_m(a_A))$. Specifically, the priorities' lists $\{\boldsymbol{q}_m\}_{m\in\mathcal{M}}$ defines the priorities assigned to every activity according to the specific performance metric m.

[6] These values should be properly chosen in the order of at most a few seconds in order to filter out just the windows of samples which correspond to little pauses or random movements.

At this point, for given $s \in \mathcal{S}$ and $m \in \mathcal{M}$, the A sequences of activity windows $\{\boldsymbol{w}_{\text{REF}}(a, \mathcal{G}_a^*, s, \mathcal{L}_a^*)\}_{a\in\mathcal{A}}$ are combined together, on the basis of the priorities' list $\boldsymbol{q}_m$, resulting in a single sequence of activity windows $\boldsymbol{w}(s, m)$ and the corresponding weighted f1 score, denoted as $F1^{\text{w}}(s, m)$, is computed and stored. By averaging $\{F1^{\text{w}}(s, m)\}_{s\in\mathcal{S}}$ over all possible separations in $\mathcal{S}$ and denoting this average as $\overline{F1^{\text{w}}}(m) = \sum_{s\in\mathcal{S}} F1^{\text{w}}(s, m)/S$, the optimal $\boldsymbol{q}^* = (q^*(a_1), q^*(a_2), \ldots, q^*(a_A))$ is then estimated as the list $\boldsymbol{q}_{m^*}^*$ where the metric m^* is selected as follows:

$$m^* = \underset{m}{\arg\max}\, \overline{F1^{\text{w}}}(m). \tag{6}$$

At the end of the training phase, for each activity a, the optimal G_a^* is used to further estimate the optimal thresholds $\{t_1^*(a, f)\}_{f\in C_a^*}$ and $\{t_2^*(a, f)\}_{f\in C_a^*}$, which will be actually used in the online phase, with the only difference that the entire training dataset is now considered to derive them. Therefore, for each activity a, the following optimal parameters have to be stored for future use in the online phase: C_a^*; $\{t_1^*(a, f)\}_{f\in C_a^*}$ and $\{t_2^*(a, f)\}_{f\in C_a^*}$; $\mathcal{L}_a^* = (\ell_1^*, \ell_2^*, \ell_3^*)$. Finally, the optimal list of priorities $\boldsymbol{q}^* = (q^*(a_1), q^*(a_2), \ldots, q^*(a_A))$, over all possible activities, is also stored.

3. Results and discussion

The performance of the proposed algorithm has been evaluated for the classification of activities related to modes of user locomotion. In particular, $A = 4$ activities are considered: *stand*, *walk*, *sit*, and *lie* (i.e., $\mathcal{A} = \{\text{stand}, \text{walk}, \text{sit}, \text{lie}\}$). In Section 3.1, we outline the considered configurations of nodes and features for the proposed algorithm and the following existing classification algorithms (as anticipated in Section 1): the k-NN (with $k = 1$ and $k = 3$), the NCC, the LDA, and the QDA – more details about their practical implementation are given in [4,5,12]. All the algorithms (for all possible configurations) are tested on the same dataset, which, for the purpose of repeatability, takes into account four different subjects [14]. In all cases, $\mathcal{R} = \{90, 91, 92, \ldots, 100\}$, $\mathcal{L}_1 = \{0, 1, 2, \ldots, 60\}$, $\mathcal{L}_2 = \{0, 5, 10, \ldots, 200\}$, $\mathcal{L}_3 = \{0, 5, 10, \ldots, 200\}$, $\mathcal{M} = \{\text{f1 score, precision, recall, specificity, accuracy}\}$, and $S = 4$ different separations of the training dataset are considered. The obtained classification performance results are presented in Section 3.2. In Section 3.3, the

time complexity of all considered algorithms is summarized. Finally, in Section 3.4 the robustness of the proposed algorithm against rotational noise is investigated, as this is particularly meaningful for BSN-based activity classification applications.

3.1. Configurations of nodes and features

Overall, we consider 38 different configurations of nodes (and feature types): configurations 1–31 (with at most 7 nodes) apply to the proposed algorithms and the four considered existing classification algorithms (k-NN, NCC, LDA, and QDA) and rely on the use of accelerometric data; configurations 32–38 (with more than 7 nodes) apply only to the four considered existing classification algorithms and rely on the use of other (besides accelerometers) inertial sensors (e.g., magnetometers). Each configuration involves a specific nodes' configuration, explicitly shown in the x axis of Fig. 6 (which will be described in the next subsection) with reference to the node number in Fig. 1.

Note that, given a set of activities that need to be classified, the number and placement of the devices could be preliminary devised and used by determining which devices may provide the most different signal patterns in correspondence to different activities. However, in this work we take advantage of the fact that an exhaustive dataset was provided in the Opportunity Challenge [4,14,18], since the BSN used to acquire the given dataset was composed by a large number of sensor devices placed all over the users' body. The automatic selection of the best devices and features is then left to the "artificial intelligence" of the algorithm (specifically, during the training phase). In this way, some sensor devices, which could have appeared to be intuitively useless at classifying some activities, may be instead selected as good candidates. Furthermore, once the best set of devices is determined, the proposed approach also allows to automatically determine the best subset of devices and features for each activity that has to be classified and such subset may likely change from activity to activity.

The choice of the nodes' feature types for each classification algorithm can be summarized as follows.

– **Configurations 1–31; proposed algorithm.** Only p-features and dev-features are considered. Specifically: for nodes 7, 1, 2, and 6 (namely, the nodes of the main vertical segment of the human body) both types of features are extracted; for nodes 13, 19, and 21 (feet nodes and a second node for the back) only the d-feature is extracted. In particular, at most $F = 11$ features are considered (for instance, the set of features of Configuration 1 is $\mathcal{F} = \{1p, 2p, 6p, 7p, 1d, 2d, 6d, 7d, 13d, 19d, 21d\}$).

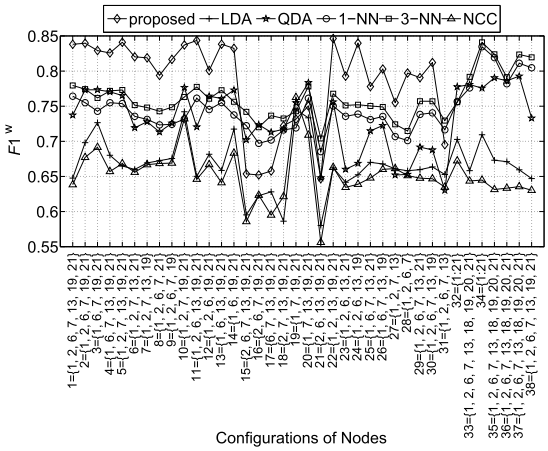


Fig. 6. Average (over 4 subjects) classification performance (i.e., weighted f1 score) as a function of the considered configurations of nodes. The performance of the proposed algorithm is compared with that of some existing algorithms, averaging the performance of the four considered subjects. For every configuration, the considered subsets of BSN nodes (numbered as in Fig. 1) are highlighted. The features per configuration and per algorithm are properly selected as summarized in Section 3.1.

– **Configurations 1–31; k-NN, NCC, LDA, and QDA algorithms.** For all the nodes, the mean of every acceleration component within a sliding window is considered, leading to 3 features extracted at each node. Therefore, at most $F = 21$ features are considered.
– **Configurations 32–38; k-NN, NCC, LDA, and QDA algorithms.** For all the nodes, the mean of every acceleration component within a sliding window is still considered (as in the previous case). In addition, for configurations from 34 to 38, the mean within a sliding window is also computed for the 3 components of the gyroscope and magnetometer signals, or some combinations of them, as summarized below:
  * for configuration 34: every node equipped with gyroscope and magnetometer (i.e., nodes from 13 to 21) produces 6 additional features (3 from gyroscope and 3 from magnetometer);
  * for configuration 35: 6 additional features (3 from gyroscope and 3 from magnetometer) are extracted at node 13;

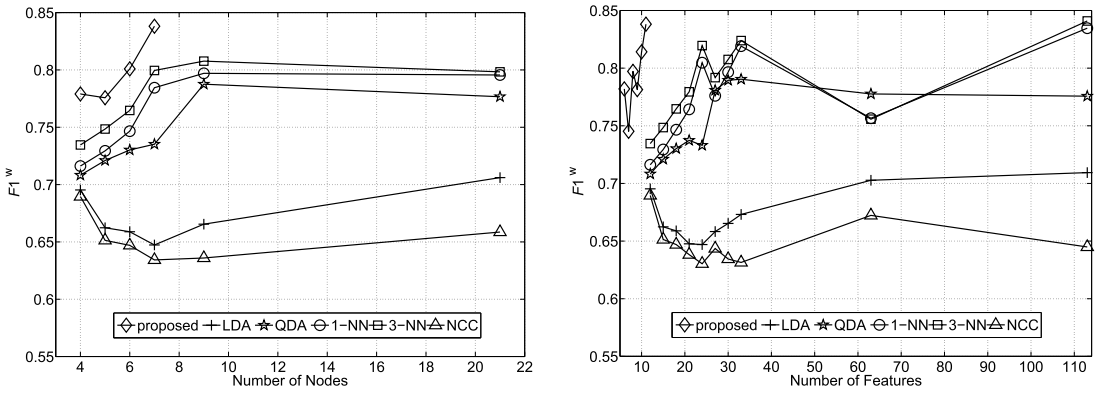


Fig. 7. Average (over 4 subjects) classification performance (i.e., weighted f1 score) as a function of the considered (a) number of nodes and (b) number of features. For each considered algorithm, the configurations which use the same number of (a) nodes or (b) features have been averaged together.

- for configuration 36: 3 additional gyroscopic features are extracted at node 13;
- for configurations 37 and 38: 3 additional magnetometric features are extracted at node 13.

3.2. *Classification performance*

The performance of the proposed algorithm has been evaluated for 31 different configurations of nodes in order to determine the optimal subset of BSN nodes (and, thus, of features) useful for the activity classification. In particular, in Fig. 6, the performance of our algorithm (in terms of $F1^w$) is shown as a function of the configurations of nodes, averaging over the considered four subjects. A comparison with the performance obtained with the other considered algorithms, run with the same configurations of nodes, is also shown. Note that the performance of these algorithms is also evaluated for more complex configurations of nodes (for a total of 38 configurations), which consider the use of a larger number of nodes and features. It is easy to observe that, in most of the considered configurations of nodes, our algorithm outperforms the other ones. It can also be seen that some of the reference algorithms (in particular, k-NN and QDA) have similar performance to ours, but only when considering, in their cases, more complex configurations with larger numbers of nodes and features. Note also that the central "hole" in Fig. 1 is associated with those configurations of nodes (namely, 15, 16, 17, 18, and 21 in Fig. 6) which do not use node 1 (the thigh node) in Fig. 1, i.e., a key node in discriminating between sit and stand activities. A similar observation can be made for configuration 31 (as denoted in Fig. 6), which hardly discriminates between stand and walk due to the absence of both feet nodes (namely, nodes 19 and 21 in Fig. 1).

In order to better investigate the impact of the number of nodes on the performance of the considered algorithms, in Fig. 7(a) the performance of the considered algorithms is properly averaged over all configurations with the same number of nodes. It can be observed that, for a given number of nodes in the BSN, our algorithm, making use of configurations with at most 7 nodes, outperforms the others. Moreover, with only 7 nodes, our algorithm outperforms all existing algorithms, including the k-NN and QDA, run with a much larger number of nodes (e.g., 21).

Another aspect to take into account is the number of features used in the algorithms. Unlike the existing algorithms, where each node generates at least three features,[7] our algorithm is such that a maximum of two features (i.e., the p-feature and the dev-feature) can be extracted at each node. In Fig. 7(b), it is then shown how the algorithms' performance changes with respect to the considered overall number of features (over all nodes). It can be concluded that our algorithm provides the same, or even better, performance, with respect to the other algorithms, using a significantly smaller number of features. The number of used features has a significant impact on the time complexity

[7] The typical features computed at each node are the mean of every acceleration component within a sliding window. In addition, other six features are considered for nodes provided with gyroscopes and magnetometers. A standard deviation-related feature has been also used giving however poor performance.

of a classification algorithm, as it will be explained in Section 3.3, allowing a better real-time applicability of our algorithm with respect to the others. As previously observed, k-NN and QDA are the only algorithms which can achieve, for a very large number of features (over 110), a performance similar to that of our algorithm (using only 11 features).

For the sake of completeness, we want to highlight that, if the previously cited configurations 15, 16, 17, 18, 21 in Fig. 6 (which do not use node 1 in Fig. 1) are not taken into account for the evaluation of the curves in Fig. 7, the performance of our algorithm improves significantly more (in relative terms) than those of the other algorithms.

Finally, on the basis of the previous results, configuration 22 (as denoted in Fig. 6) can be identified as the best configuration of nodes for our algorithm. In particular, it needs 5 nodes (namely, nodes {1, 2, 13, 19, 21} in Fig. 1) and generates $F = 7$ features (5 dev-features, one per node, and 2 p-features, associated with nodes 1 and 2 in Fig. 1). Using this configuration, our algorithm obtains a value of $F1^{\mathrm{w}}$ around 85%. The second best algorithm, i.e., the k-NN (with $k = 3$), reaches a value of $F1^{\mathrm{w}}$ around 77% using 15 features (more than twice the number in our algorithm) and more sensors (indeed, gyroscopes and magnetometers are available in nodes {13, 19, 21} in Fig. 1). Furthermore, the best among the other algorithms (again, the k-NN with $k = 3$) obtains its highest $F1^{\mathrm{w}}$ score (namely, $F1^{\mathrm{w}} = 84\%$) using a significantly more complex configuration (namely, configuration 34 in Fig. 6), which comprises a total of 21 nodes and $F = 113$ features.

3.3. Time complexity

The time complexity of the proposed algorithm has been evaluated and compared with those of the classification algorithms previously considered for performance comparison.[8] In particular, it is possible to prove that the time complexity of the online phase of our algorithm is a linear function of the number of features F and the number of activities A. In Table 1, the time complexity of our algorithm is reported along with those of the other considered algorithms [5,12].

[8] Recall that, we here consider the time complexity of the online phase due to its impact on real-time applicability of the algorithm and due to the fact that the training phase should be performed just once (offline), provided that the BSN configuration does not change over time.

Table 1
Time complexity of the online phase for the considered algorithms

Algorithm	Time Complexity
proposed, NCC, LDA, and QDA	$O(F \cdot A)$
k-NN	$O(F \cdot D)$

It can be observed that the time complexity of all algorithms is a linear function of the number of features F. In addition, our algorithm, the NCC, LDA, and QDA algorithms have a complexity linearly dependent on the number of activities A, but independent of the training dataset size D, whereas the time complexity of the k-NN algorithm depends linearly on D and is independent of A. Two considerations can now be made about A, D, and F:

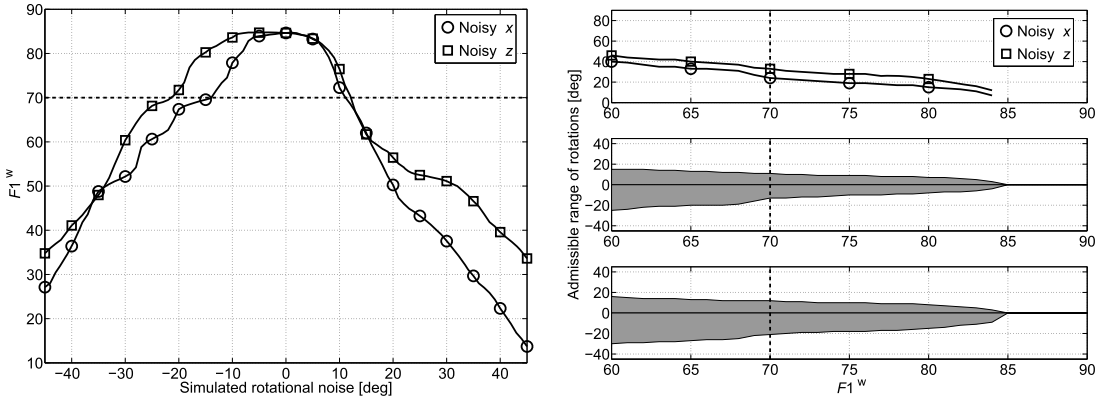
- the number of activities is typically quite larger than the size of the training dataset (i.e., $A \ll D$);
- the classification performance of the proposed algorithm, when considering a few features (i.e., for small values of F), is far better than those of the NCC, the LDA, and the QDA algorithms (and also slightly better than that of the k-NN algorithm).

For these reasons, it can be concluded that our algorithm (i) guarantees the best compromise between classification performance and time complexity and (ii) outperforms, for a given number of features, the other algorithms.

3.4. Robustness to noise of the proposed algorithm

A typical problem of a real BSN scenario consists of unwanted rotations in the displacement of the BSN nodes, that can differ between the training and the online phases. It is then of interest to investigate the robustness of BSN-based activity classification algorithm to rotational noise. To this end, artificial rotational noise has been added to the accelerometric data, simulating possible rotations of a device around its three reference axes. We remark that modeling rotations around the accelerometer axes, rather than around the body axes, is just an assumption, which is made to avoid taking devices position shifts also into account. It is, however, a reasonable assumption, since the position shifts would be rather limited.

Two considerations can be preliminary made: the dev-feature is actually insensitive to rotational noise, due to its definition; the p-feature is invariant to rotational noise around the axis about which the feature is measured (i.e., the one parallel to the related body



(a) (b)

Fig. 8. Average (over 4 subjects) classification performance (i.e., weighted f1 score) of the proposed algorithm in the presence of simulated rotational noise: (a) the weighted f1 score as a function of the intensity of the simulated rotational noise; (b) the admissible range of rotations as a function of the weighted f1 score. The previously estimated optimal configuration of nodes (i.e., configuration 22, as denoted in Fig. 6) is considered. Indicative thresholds (dashed lines), corresponding to an admissible minimum performance of $F1^{\mathrm{W}} = 70\%$, are also shown in two subfigures.

segment, which we assume to be the y axis for every device). For such reasons, the rotational noise is then being only added to p-features and the device rotations have been simulated only around its other two axes, namely (due to the previous assumption) the x and z axes. More specifically, from now on, we assume that the x axis is directed from the front to the back of the user, whereas the z axis is directed from his/her right side to his/her left side. For ease of simplicity, we only simulate rotations around one axis at a time.

In Fig. 8(a), the average performance of our algorithm (averaged over the 4 considered subjects) is shown as a function of the intensity of the simulated rotational noise (in terms of degrees of rotation with respect to the initial orientation), for the previously estimated (at the end of Section 3.2) optimal configuration of nodes (i.e., configuration 22, as denoted in Fig. 6). In Fig. 8(b), for the same configuration 22 (as denoted in Fig. 6) and averaging over the same 4 subjects, the curves show the range of rotations that can be applied to the nodes in order to keep a user-defined admissible minimum performance (in terms of weighted f1 score). The results in Fig. 8 show that our algorithm, possibly due to the nature of the activities that we want to classify, suffers less from rotations around the z axis than those around the x axis. As an example, if one accepts as a minimum acceptable performance a $F1^{\mathrm{W}}$ equal to 70%, a system using the proposed algorithm can tolerate rotations in the range of $[-10°, 10°]$.

We also remark that, due to the realistic data collection (often operated in different times with respect to the training acquisition), the testing data (used in the online phase) implicitly presents real rotational noise. Therefore, the results previously presented implicitly assume that the proposed algorithm has to combat some rotational noise.

4. Conclusion

In this work, a simple, yet effective, activity classification algorithm has been presented. Its performance has been evaluated and compared with that of existing algorithms. The data used to test the algorithms are publicly available and, thus, represent a valid and unbiased benchmark for the evaluation of the performance of different algorithms. The proposed algorithm is based on simple comparisons of properly selected features with thresholds that are automatically optimized during a preliminary training phase performed once (offline). In order to simplify the operations of the online phase of the algorithm, the training phase is also used to automatically select the optimal subset of nodes and features to be used.

The time complexity of the proposed algorithm has also been evaluated. Our results show that its complexity is on the order of that of existing algorithms, but its performance is better. On the other hand, some of the existing algorithms show a performance similar to that of ours at the cost of higher complexity. In particular, our algorithm significantly outperforms the others when using a small numbers of nodes and features.

Taking also into account its robustness against rotational noise, it can be concluded that the proposed algorithm can be used effectively for real-time activity classification, especially when some constraints on the number of BSN nodes are introduced.

References

[1] R. Aylward and J. Paradiso, A compact, high-speed, wearable sensor network for biomotion capture and interactive media, in: *Proc. of the 6th International Conference on Information Processing in Sensor Networks (IPSN)*, Cambridge, MA, USA, April 2007, pp. 380–389. doi:10.1145/1236360.1236408.

[2] L. Bao and S. Intille, Activity recognition from user annotated acceleration data, in: *Pervasive Computing*, April 2004, pp. 1–17. doi:10.1007/978-3-540-24646-6_1.

[3] R. Chavarriaga, H. Bayati and J.d.R. Millán, Unsupervised adaptation for acceleration-based activity recognition: Robustness to sensor displacement and rotation, *Personal and Ubiquitous Computing* **17**(3) (March 2013), 479–490. doi:10.1007/s00779-011-0493-y.

[4] R. Chavarriaga, H. Sagha, A. Calatroni, S.T. Digumarti, G. Troster, J.d.R. Millan and D. Roggen, The opportunity challenge: A benchmark database for on-body sensor-based activity recognition, *Pattern Recognition Letters* **34**(15) (November 2013), 2033–2042. doi:10.1016/j.patrec.2012.12.014.

[5] R.O. Duda, P.E. Hart and D.G. Stork, *Pattern Classification and Scene Analysis*, 2nd edn, Wiley-Interscience, New York, NY, USA, 2000.

[6] A. Fleury, M. Vacher and N. Noury, SVM-based multimodal classification of activities of daily living in health smart homes: Sensors, algorithms, and first experimental results, *IEEE Trans. on Information Technology in Biomedicine* **14**(2) (March 2010), 274–283. doi:10.1109/TITB.2009.2037317.

[7] M. Giuberti and G. Ferrari, Simple and robust BSN-based activity classification: Winning the first BSN contest, in: *Proc. 4th International Symposium on Applied Sciences in Biomedical and Communication Technologies (ISABEL)*, Barcelona, Spain, October 2011.

[8] M. Giuberti and G. Ferrari, BSN-based activity classification: A low complexity windowing-&-classification approach, *Advances in Science and Technology* **85** (2013), 53–58. doi:10.4028/www.scientific.net/AST.85.53.

[9] E. Guenterberg, S. Ostadabbas, H. Ghasemzadeh and R. Jafari, An automatic segmentation technique in body sensor networks based on signal energy, in: *Fourth International Conference on Body Area Networks (BodyNets)*, Los Angeles, CA, USA, April 2009, pp. 1–7.

[10] T. Huynh and B. Schiele, Analyzing features for activity recognition, in: *Proc. of the 2005 Joint Conference on Smart Objects and Ambient Intelligence: Innovative Context-Aware Services: Usages and Technologies*, Grenoble, France, 2005, pp. 159–163.

[11] R. Jafari, W. Li, R. Bajcsy, S. Glaser and S. Sastry, Physical activity monitoring for assisted living at home, in: *Proc. of International Workshop on Wearable and Implantable Body Sensor Networks (BSN)*, Aachen, Germany, March 2007, pp. 213–219. doi:10.1007/978-3-540-70994-7_37.

[12] A.K. Jain, R.P.W. Duin and J. Mao, Statistical pattern recognition: A review, *IEEE Trans. on Pattern Analysis and Machine Intelligence* **22**(1) (January 2000), 4–37. doi:10.1109/34.824819.

[13] A. Mannini and A.M. Sabatini, Machine learning methods for classifying human physical activity from on-body accelerometers, *Sensors* **10**(2) (February 2010), 1154–1175. doi:10.3390/s100201154.

[14] Opportunity Dataset, http://archive.ics.uci.edu/ml/datasets/OPPORTUNITY+Activity+Recognition.

[15] S. Patel, H. Park, P. Bonato, L. Chan and M. Rodgers, A review of wearable sensors and systems with application in rehabilitation, *Journal of NeuroEngineering and Rehabilitation* **9**(21) (April 2012), 1–17.

[16] S. Pirttikangas, K. Fujinami and T. Nakajima, Feature selection and activity recognition from wearable sensors, in: *International Symposium on Ubiquitous Computing Systems*, Seoul, Korea, October 2006, pp. 516–527. doi:10.1007/11890348_39.

[17] S.J. Preece, J.Y. Goulermas, L.P.J. Kenney, D. Howard, K. Meijer and R. Crompton, Activity identification using body-mounted sensors – A review of classification techniques, *Physiological Measurement* **30**(4) (April 2009), R1. doi:10.1088/0967-3334/30/4/R01.

[18] D. Roggen, A. Calatroni, M. Rossi, T. Holleczek, K. Forster, G. Troster, P. Lukowicz, D. Bannach, G. Pirkl, A. Ferscha, J. Doppler, C. Holzmann, M. Kurz, G. Holl, R. Chavarriaga, H. Sagha, H. Bayati, M. Creatura and J.d.R. Millan, Collecting complex activity datasets in highly rich networked sensor environments, in: *2010 7th International Conference on Networked Sensing Systems (INSS)*, Kassel, Germany, June 2010, pp. 233–240. doi:10.1109/INSS.2010.5573462.

[19] H. Sagha, J.d.R. Millán and R. Chavarriaga, Detecting anomalies to improve classification performance in opportunistic sensor networks, in: *Proc. of 2011 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, Seattle, WA, USA, March 2011, pp. 154–159. doi:10.1109/PERCOMW.2011.5766860.

[20] D.M. Sherrill, M.L. Moy, J.J. Reilly and P. Bonato, Using hierarchical clustering methods to classify motor activities of copd patients from wearable sensor data, *Journal of NeuroEngineering and Rehabilitation* **2**(1) (June 2005), 1–14. doi:10.1186/1743-0003-2-16.

[21] J.A. Ward, P. Lukowicz, G. Tröster and T.E. Starner, Activity recognition of assembly tasks using body-worn microphones and accelerometers, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **28**(10) (October 2006), 1553–1567. doi:10.1109/TPAMI.2006.197.

[22] A. Yang, S. Iyengar, S.S. Sastry, R. Bajcsy, P. Kuryloski and R. Jafari, Distributed segmentation and classification of human actions using a wearable motion sensor network, in: *Proc. of the Computer Vision and Pattern Recognition Workshops on Human Communicative Behavior Analysis (CVPRW)*, Anchorage, AK, USA, June 2008, pp. 1–8.

[23] P. Zappi, T. Stiefmeier, E. Farella, D. Roggen, L. Benini and G. Tröster, Activity recognition from on-body sensors by classifier fusion: Sensor scalability and robustness, in: *Proc. of 3rd International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, Melbourne, Australia, December 2007, pp. 281–286.