Improving Activity Recognition by Segmental Pattern Mining

Umut Avci and Andrea Passerini

Abstract—Activity recognition is a key task for the development of advanced and effective ubiquitous applications in fields like ambient assisted living. A major problem in designing effective recognition algorithms is the difficulty of incorporating long-range dependencies between distant time instants without incurring substantial increase in computational complexity of inference. In this paper we present a novel approach for introducing long-range interactions based on sequential pattern mining. The algorithm searches for patterns characterizing time segments during which the same activity is performed. A probabilistic model is learned to represent the distribution of pattern matches along sequences, trying to maximize the coverage of an activity segment by a pattern match. The model is integrated in a segmental labeling algorithm and applied to novel sequences, tagged according to matches of the extracted patterns. The rationale of the approach is that restricting dependencies to span the same activity segment (i.e., sharing the same label), allows keeping inference tractable. An experimental evaluation shows that enriching sensor-based representations with the mined patterns allows improving results over sequential and segmental labeling algorithms in most of the cases. An analysis of the discovered patterns highlights non-trivial interactions spanning over a significant time horizon.

Index Terms—Activity recognition, pattern mining, segmental labeling

1 Introduction

The automatic recognition of activities from sensor data is crucial for developing advanced applications in areas like ambient assisted living and assisted cognition. Services like reminders and automatic reporting to clinicians and medical staff can help in improving healthcare and elders' independent living. From a machine learning viewpoint, activity recognition can be formalized as a sequence labeling task: given a sequence of sensor readings covering a timespan of interest (e.g., a day), predict the sequence of activities being performed. The timespan is typically divided into small time intervals, to be labeled with the activity or the activities taking place. We will refer to these time intervals as instants in this paper. A number of machine learning algorithms have been applied to this task, ranging from simple Naive Bayes [1] to sequential approaches like hidden Markov models (HMM) [2], conditional random fields [3] and their variants [4].

Local techniques like Naive Bayes or standard support vector machines label each time instant independently, possibly extending its input representation over neighboring instants. On the other hand, sequential approaches collectively assign labels to all instants within the period of interest. This allows exploiting the relationship between activities performed at different time and usually results in performance improvements, other things being equal [5]. In modeling temporal interactions, however, these models are limited to rather small spans. Sequential approaches rely on a Markovian assumption to limit the number of parameters to be learned and keep inference tractable.

There are a number of attempts in the literature in order to account for longer-range dependencies. Hierarchical approaches aim at representing activity relations in different levels of a hierarchy. Dependencies between short-range activities in the lower level of the hierarchy are fed into higher levels for creating longer-range ones [7], [8], [9]. However, creating hierarchies requires deep knowledge regarding the underlying structure of the problem. Adding shortcut links between arbitrary time instants along the sequence, e.g., in skip-chain CRF, is another alternative [14] but the complexity of the model and the cost of inference increase drastically depending on the number of shortcuts introduced.

An activity usually spans a certain amount of time, its average duration depending on the specific activity being performed (e.g., taking a shower or watching TV). We define an activity segment as a sequence of consecutive time instants in which the same activity is performed. Segmental labeling can be accomplished by semi-Markov models [15], which explicitly account for duration information. However, incorporating long-range dependencies between observations within each segment is again a bottleneck. In this paper we address the problem of introducing longer-range dependencies in sequential labeling algorithms relying on sequential pattern mining techniques [16], [17], [18], [19]. We show how to integrate sequential pattern mining into probabilistic segmental labeling algorithms, providing improved capacity to model longer-term dependencies. Our solution consists of mining segmental patterns, i.e., sequential patterns which cover segments corresponding to a certain activity. By allowing gaps between matches of individual pattern elements, distant observations can be related. We introduce a probabilistic duration model representing the distribution of pattern matches along sequences, and integrate it into a hidden semi-Markov model (HSMM).

• *The authors are with the Department of Information Engineering and Computer Science, University of Trento, Via Sommarive 5 POVO, Trento, Trentino 38123, Italy. E-mail: {avci, passerini}@disi.unitn.it.*

Manuscript received 30 July 2012; revised 24 Jan. 2013; accepted 20 July 2013; publised online xxxxx.
Recommended for acceptance by A. Gionis.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-2012-07-0543.
Digital Object Identifier no. 10.1109/TKDE.2013.127

A preliminary version of this work was presented in [6], where pattern matches were simply modeled as additional binary features for each time instant within the match.

We evaluate the proposed approach on two different freely available real-world data sets. The experimental evaluation shows that the pattern-based HSMM allows improving labeling results in almost all scenarios. A detailed analysis of mined patterns shows that non-trivial interactions along non-close time instants are modeled, e.g., by discovering trajectories or sequences of object interactions characterizing specific activities.

The paper is organized as follows. In the next section, related work is discussed. Section 3 introduces the data representation format and the notion of activity segment. Sequential and segmental activity recognition algorithms are briefly revised in Section 4. Section 5 describes our segmental pattern mining approach, while the pattern-based HSMM algorithm is detailed in Section 6. Computational complexity analysis for the algorithm is discussed in Section 7. An extensive experimental evaluation is reported in Section 8. Finally, conclusions are drawn in Section 9.

2 Related Work

The problem of dealing with long-range dependencies is well-known in the machine learning community. A number of attempts at addressing it rely on hierarchical models like hierarchical HMMs [7] and their many recent variants [8], [9]. These try to model higher level dependencies between short-range activities which should account for the long-term dependencies. Du et al. [10] assume that human activities can be decomposed into multiple interactive stochastic processes to represent distinct characteristics of activities, where each characteristic corresponds to a level in hierarchical DBNs. Activity modeling is then achieved by modeling the interactive processes. However, hierarchical approaches require a good amount of knowledge about the underlying structure of the problem in building the hierarchies. The SAMMPLE architecture [11] learns high-level activities as combinations of low-level locomotive micro-activities (e.g., sitting). These latter are learned with appropriate classifiers trained on supervised instances of micro-activities. Micro-activities are also used in [12] as building blocks, combined through topic models to predict daily routines like commuting or office work. Depending on the granularity and duration of the activities to be predicted, it is not always easy to develop effective hierarchical models. Indeed, preliminary experiments showed that a two-level hierarchical HMM, modeling activity segments at the lower lever and sequences of segments at the upper one, did not improve over plain HMM in any of our experimental scenarios. Another approach for modeling long-range dependencies consists of explicitly adding links between distant time instants which are deemed to be in direct relationship, like in skip-chain CRF [13]. Hu and Yang [14] propose using this approach in order to recognize concurrent and interleaved activities. Interleaving goals are modeled by leveraging the skip chains while concurrent are were identified by adjusting inferred probabilities via correlation graphs. However, the method requires a large amount of training data because of the many possible ways in which an ongoing activity can be interrupted and resumed. Furthermore, shortcut links considerably increase the complexity of the inference task and should thus be carefully selected. Skip-chain CRF have been successfully applied to named-entity recognition, where shortcut links are added between pairs of identical capitalized words.

The idea of mining patterns from the sensor data has been extensively studied in the activity recognition community. Hasan et al. [20] apply frequent set mining to create a low-dimensional feature representation from a large number of binary sensors. T-patterns are used in [21], where the authors propose two improvements over the base approach, namely testing independence between two temporal points and Gaussian mixture modeling of correlation times, to detect temporal patterns at a low computational cost and to make the model more robust to spurious patterns. Tao et al. develop a technique based on Emerging patterns to recognize sequential, interleaved and concurrent activities for single [22] and multiple [23] users. Contrary to previous work, Rashidi et al. [24] exploit patterns in order to discover activities in an unsupervised manner, using a modified edit distance to cluster together similar patterns. However, none of these studies addresses the long-term dependency problem.

3 Data Representation

A data set $\mathcal{D} = \{(\mathbf{x}, \mathbf{y})^{(1)}, \ldots, (\mathbf{x}, \mathbf{y})^{(d)}\}$ is a collection of input-ouput sequences for a number of days d. An input example $\mathbf{x} = \{\mathbf{x}_1, \ldots, \mathbf{x}_T\}$ consists of a consecutive sequence of observations, each covering a certain time instant t. An observation $\mathbf{x}_t$ is represented by the set of sensors which are active at that time instant (i.e., within its time interval). Different choices can be made in deciding when a sensor is considered active, as will be discussed in the experimental section. When feeding input sequences to labeling algorithms (see Section 4), observations will be represented as binary vectors rather than sets. Given N sensors, an observation $\mathbf{x}_t$ will thus be encoded as a binary feature vector $\mathbf{x}_t = (x_t^1, \ldots, x_t^N)$, each feature being 1 if the corresponding sensor is active and 0 otherwise.

The labeling task consists of predicting a sequence of activity labels $\mathbf{y} = \{y_1, \ldots, y_T\}$, one for each time instant. Each label $y_t \in [1, L]$ is one of L possible activities, with one indicating no activity. We assume here that activities are not simultaneous, i.e., only a single activity is performed at each time instant. The segmental pattern mining algorithm can however be generalized to deal with multiple simultaneous activities, as discussed in the conclusions of the paper. We define an activity segment as a sequence of consecutive time instants labeled with the same activity. A segment $s_u = (b_u, e_u, y_u)$ is represented by its starting and ending time instants $b_u, e_u \in [1, T]$, with $e_u \geq b_u$, and the segment label y_u. A label sequence $\mathbf{y}$ can be split into a sequence $\mathbf{s} = \{s_1, \ldots, s_U\}$ of activity segments such that $b_1 = 1$, $e_U = T$, $b_u = e_{u-1} + 1$ and $y_u \neq y_{u-1}$ for all u. We define as $\mathbf{x}_{b_u:e_u}$ the segment of $\mathbf{x}$ ranging from b_u to e_u included. A collection of $\mathbf{s}$ over all the days forms $\mathcal{S} = \{\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(d)}\}$ with d being the number of days. We define $\mathcal{S}_y$ as the set of segments for a particular activity y, i.e., $\mathcal{S}_y \subset \mathcal{S} : \forall\, s_u = (b_u, e_u, y_u) \in \mathcal{S}_y, y_u = y$. The corresponding set of input

TABLE 1
Notation Summary

Notation	Description
L	number of activities (states)
N	number of sensors (observations)
T	length of the input sequence
t	current time instant, $1 \leq t \leq T$
$\mathbf{y} = \{y_1, \ldots, y_T\}$	activity labels, $y_t \in [1, L]$
$\mathbf{x} = \{\mathbf{x}_1, \ldots, \mathbf{x}_T\}$	sequence of observations
$\mathbf{x}_t = (x_t^1, \ldots, x_t^N)$	binary feature vector at time t
$\mathcal{D} = \{(\mathbf{x}, \mathbf{y})^{(1)}, \ldots, (\mathbf{x}, \mathbf{y})^{(d)}\}$	dataset of input-output sequences
b_u, e_u, y_u	starting and ending time instants of an activity segment, and segment label. $b_u, e_u \in [1, T]$
$\mathbf{s} = \{s_1, \ldots, s_U\}$	sequence of activity segments, $b_1 = 1$, $e_U = T$, $s_u = (b_u, e_u, y_u)$, $1 \leq u \leq U$
$\mathbf{x}_{b_u:e_u}$	segment of observations ranging from b_u to e_u
$\mathcal{S} = \{\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(d)}\}$	set of $\mathbf{s}$ over all the days
$\mathcal{S}_y$	set of segments for activity y, $\mathcal{S}_y \subset \mathcal{S} : s_u = (b_u, e_u, y_u) \in \mathcal{S}_y, y_u = y$
$\mathcal{D}(\mathcal{S}_y) = \{\mathbf{x}_{b_u:e_u} : (b_u, e_u, y_u) \in \mathcal{S}_y\}$	set of input segments for $\mathcal{S}_y$

segments is $\mathcal{D}(\mathcal{S}_y) = \{\mathbf{x}_{b_u:e_u} : (b_u, e_u, y_u) \in \mathcal{S}_y\}$. A summary of notations is given in Table 1.

4 Activity Recognition Algorithms

An effective approach for performing activity recognition on temporal data should be able to model the relationships between time instants and between their respective labels. Hidden Markov models [25] and hidden semi-Markov models [15] are directed graphical models which have been successfully used to perform sequential and segmental labeling respectively. These algorithms have been recently compared [4], [5] on a benchmark consisting of wireless sensor network data. Both algorithms model the joint probability distribution of input and output $p(\mathbf{x}, \mathbf{y})$ and return the output maximizing this probability, $\mathbf{y}^* = \arg\max_{\mathbf{y}} p(\mathbf{x}, \mathbf{y})$.

A hidden Markov model is a sequential approach where: 1) the label at each time instant depends on the label at the previous time instant only, 2) the observation at each time instant depends on the label at that time instant only, 3) probabilities do not depend on the specific time instants but only on the values of labels/observations at those instants. The resulting joint probability is given by

$$p(\mathbf{x}, \mathbf{y}) = \prod_{t=1}^{T} p(\mathbf{x}_t \mid y_t) p(y_t \mid y_{t-1}),$$

where $p(y_1 \mid y_0)$ stands for the probability of having y_1 as the initial label. In modeling the conditional probability of observations given label, a common simple approach (also followed in [4], [5]) consists of making a Naive assumption of independence between observation features given the label. The resulting probability is

$$p(\mathbf{x}_t \mid y_t) = \prod_{i=1}^{N} p(x_t^i \mid y_t), \tag{1}$$

where for the binary case, probabilities for features $p(x_t^i | y_t)$ are represented as Bernoulli distributions.

HMMs imply an exponential distribution for state durations. The probability of seeing label l for d consecutive instants is d times the probability of a self transition $p(y_t = l \mid y_{t-1} = l)$. This assumption is often not appropriate when durations tend to have specific patterns, as happens in activity recognition tasks. Explicit duration distributions can be represented by hidden semi-Markov models, which consider probability of segmental labeling $(\mathbf{x}, \mathbf{s})$. Recall that $\mathbf{s}$ is a sequence of U consecutive segments $s_u = (b_u, e_u, y_u)$. The corresponding joint probability is represented as

$$p(\mathbf{x}, \mathbf{s}) = \prod_{u=1}^{U} p(y_u \mid y_{u-1}) p(d_u \mid y_u) p(\mathbf{x}_{b_u:e_u} \mid y_u),$$

where $d_u = e_u - b_u + 1$ is the duration of segment s_u, whose probability can be modeled by the desired distribution. Following [4], we employed a histogram distribution with five bins. Concerning the probability of a certain observation segment given its label, it is commonly computed (again, see e.g., [4]) as the product of the probabilities of its time instants

$$p(\mathbf{x}_{b_u:e_u} \mid y_u) = \prod_{t=b_u}^{e_u} p(\mathbf{x}_t \mid y_u) \tag{2}$$

where $p(\mathbf{x}_t \mid y_u)$ is further decomposed as in Eq. (1). However, additional knowledge on the dynamics of a certain activity could help devising a more complex probabilistic model for the sequence of observations measured while performing it. We will use mined activity patterns in order to enrich this representation with features capturing longer range dependencies.

Note that the approach that we introduce for directed graphical models can be straightforwardly applied to their undirected counterparts, conditional random fields [13] with their semi-Markov extension [26]. We did not include them in our comparison, as they require much longer training time and were shown to provide comparable and often worse results with respect to their directed counterparts (HMM and HSMM) on this benchmark [4].

5 Segmental Pattern Mining

Our aim is mining patterns characterizing timespans during which a certain activity is performed. Algorithm 1 shows the pseudocode of our segmental pattern miner. Training sequences are first split into activity segments, each labeled with the corresponding activity. These segments are fed to a

sequential pattern miner (procedure SEQUENTIALMINER). We employed PBOOST [27] which supports discriminative mining, i.e., mining of patterns distinguishing sequences of a certain class from the others. In the following we provide a brief description of the algorithm. Further details can be found in the original paper [27]. The algorithm takes as input sets of positive and negative examples, each example being a sequence of sets of integers (the sensor identifiers in our case). A pattern is itself a sequence of sets of integers. The algorithm mines for patterns matching positive and not negative examples. Let $\mathbf{p} = (\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_m)$ be a pattern of length m. Let $\mathbf{p}_i$ be a pattern element, corresponding to a non-empty set of active sensors. The pattern $\mathbf{p}$ matches sequence $\mathbf{x}$ if there is a match $(t_1, t_2, \ldots, t_m)$ such that: for all $i > j$, $t_i > t_j$; for all i, $\mathbf{x}_{t_i}$ contains active sensors $\mathbf{p}_i$, i.e., $\forall \mathbf{p}_i^k \in \mathbf{p}_i$, $\mathbf{x}_{t_i}^{\mathbf{p}_i^k} = 1$, where $\mathbf{x}_{t_i}^{\mathbf{p}_i^k}$ is the value of sensor $\mathbf{p}_i^k$ in the observation vector for time instant t_i. A *gap* is defined as a sequence of time instants separating two consecutive pattern element matches from each other. We define gap length g as the overall sum of time instants occurring between consecutive pairs of element matches along the pattern. More formally, pattern $\mathbf{p} = (\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_m)$ matches a sequence $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$ with gap length g if there exist time instants $1 \leq t_1 \leq t_2 \leq \cdots \leq t_m \leq n$ such that $\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_m$ are respectively contained in $\mathbf{x}_{t_1}, \mathbf{x}_{t_2}, \ldots, \mathbf{x}_{t_m}$ and $t_m - t_1 + 1 - m = g$. By defining a canonical ordering for sequences, the pattern space is searched in a tree-based fashion starting from the empty pattern [16]. Pruning of the search space is conducted combining the standard notion of *support* (i.e., number of matching sequences) with that of *gain*: PBOOST considers each pattern as a feature and learns a linear classifier (LPBoost [28]) on top of them, discriminating between positive and negative examples. The gain provided by a feature can be compared with an upper bound on the maximal gain achievable by further extending the corresponding pattern. If the current gain exceeds or equals the upper bound there is no need to proceed in this search direction.

Algorithm 1 Procedure for segmental pattern mining

Input:
$\mathcal{D}$: input-output sequences
ϕ: pattern selection threshold
Output:
$\mathcal{P}$: patterns for activities along with their gap sizes

procedure SEGMENTALMINER($\mathcal{D}$, ϕ)
  Initialize $\mathcal{P}$ to the empty set
  Split training sequences into activity segments $\mathcal{S}$
  for all activities y **do**
    $\mathcal{S}_y \leftarrow$ segments for y
    $\mathcal{S}_{\bar{y}} \leftarrow$ segments for $y' \neq y$
    $\mathcal{P}_y \leftarrow$ SEQUENTIALMINER($\mathcal{D}(\mathcal{S}_y), \mathcal{D}(\mathcal{S}_{\bar{y}})$)
    for all $\mathbf{p} \in \mathcal{P}_y$ **do**
      if SCORE($\mathbf{p}, \mathcal{D}(\mathcal{S}_y), \mathcal{D}(\mathcal{S}_{\bar{y}})$) $\geq \phi$ **then**
        $g \leftarrow$ MEDIANGAP($\mathbf{p}, \mathcal{D}(\mathcal{S}_y)$)
        $\mathcal{P} \leftarrow \mathcal{P} \cup \{(\mathbf{p}, g)\}$
  return $\mathcal{P}$
end procedure

For each of the possible activities, PBOOST is run providing segments of the target activity as positive examples and all other segments as negative ones. Each of the returned patterns $\mathbf{p}$ is evaluated according to its discriminative power, computed as its F1 score on the training segments (procedure SCORE). The F1 measure (see experiments) tradesoff precision, i.e., the fraction of segments covered by the pattern which do belong to the target activity, and recall, i.e., the fraction of segments of the target activity covered by the pattern. All patterns with a score lower than a certain threshold ϕ are discarded.

Sequential patterns extracted by PBOOST allow for arbitrary gaps between the pattern elements. Our aim is that these patterns cover the largest possible portion of an activity segment. However, in the test phase when used to label a novel time sequence, the activity segmentation will be unknown and the patterns will be applied to the whole sequence (i.e., a day). We thus need to estimate the expected number of gaps in pattern matches for activity segments. This is crucial for a correct use of patterns: during test, allowing for arbitrary gaps within a pattern could produce matches involving very distant time instants (e.g., in the morning and afternoon), which likely belong to different activity segments. The procedure MEDIANGAP described in Algorithm 2 estimates how many gaps should be expected on average for a match covering the largest possible portion of an activity segment. Given a pattern characterizing a certain activity, and a set of training segments for that activity, the procedure finds the longest possible match of the pattern on each segment and computes the corresponding gap length. Let $(\mathbf{p}, g)$ be a pattern $\mathbf{p} = (\mathbf{p}_1, \ldots, \mathbf{p}_m)$ of length m with gap length g. A g-gap-bounded match $(t_1, \ldots, t_m)$ for the pattern is a sequence of time instants such that: for all $i > j$, $t_i > t_j$; for all i, $\mathbf{p}_i$ is contained in active sensors of $\mathbf{x}_{t_i}$; the sequence has at most g gaps, i.e., $t_m - t_1 + 1 \leq m + g$. The procedure LONGESTMATCH($\mathbf{x}$,t_1,$\mathbf{p}$,g) finds the longest g-gap-bounded match of pattern $\mathbf{p}$ in sequence $\mathbf{x}$ starting at instant t_1. The procedure is used here for $g = T - m$, i.e., asking to cover the largest possible portion of the entire input sequence. As will be seen in the next section, the same sub-routine is also employed during inference to return the largest possible pattern match within the estimated gap length. The procedure is detailed in Algorithm 3. The function MATCH(x_c,p_u) checks whether a single position matches with a pattern element, i.e., $\mathbf{x}_c$ contains active sensors $\mathbf{p}_u$. The procedure verifies that the first $u \in [1, m-1]$ patterns are matched within their allowed lengths $u + g$, with the first pattern matching position t_1. Then it searches for the longest possible match by matching the last pattern $\mathbf{p}_m$ to a position as close as possible to the overall maximal allowed length $m + g$. The initial (M_1) and the final (M_2) positions of the match are returned as the border points. If the pattern does not match with the sequence, border points are assigned to $(0, -1)$. The median of the gap length computed over all activity segments is returned by the MEDIANGAP procedure as an estimate of the gap length which should be expected for a pattern match covering the longest possible portion of an activity segment. The final outcome of the segmental pattern mining algorithm is a set of patterns characterizing all activities, together with their estimated gap lengths. Fig. 1a and b illustrate the pattern mining process and the usage of gaps during pattern match respectively. Three input sequences, corresponding for instance to three days, are split into segments for activities a_1, a_2 and a_3. A time instant is either made up of sensor identifiers of interacted objects or empty in the absence of sensor interactions.

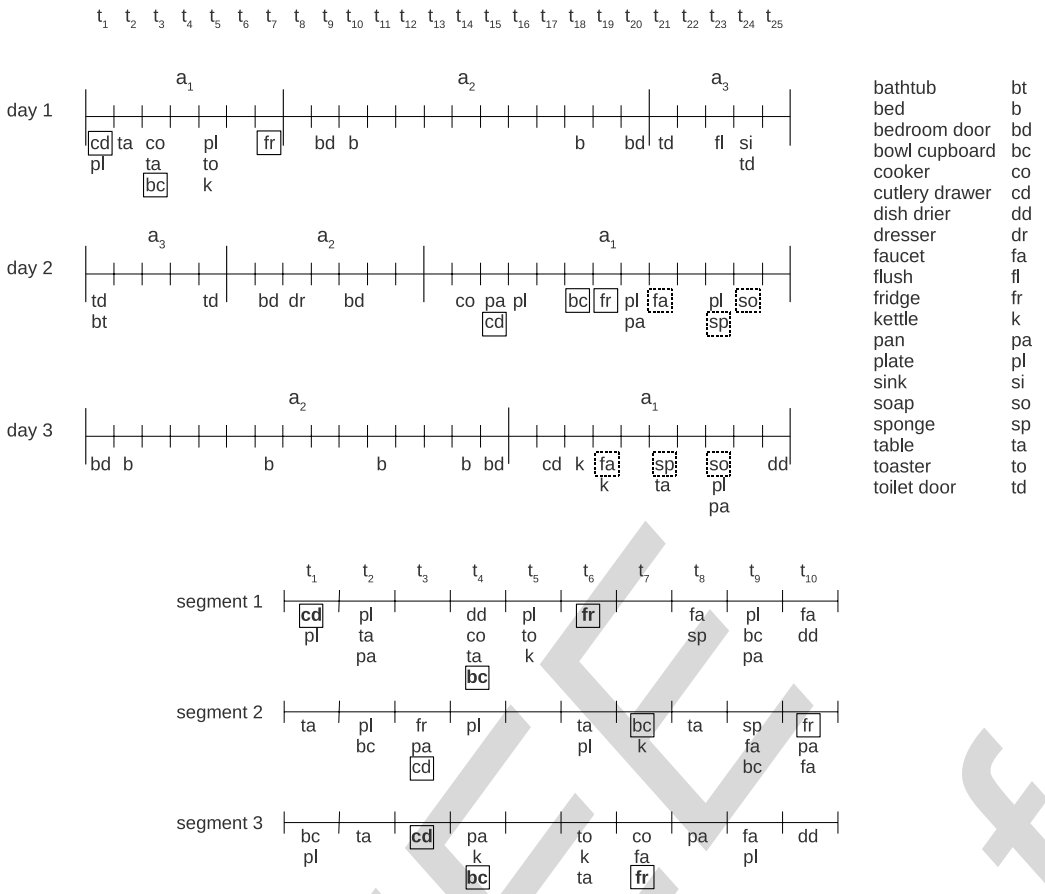


Fig. 1. Example of discriminative pattern mining output (top): the miner finds two sequential patterns discriminating segments for activity a_1 from segments for other activities. 3-gap-bounded match (bottom): segments one and three match pattern (cd,bc,fr) with three and two gaps respectively, while match for segment two contains five gaps and is thus not a 3-gap-bounded match.

Having provided the segments of activity a_1 as positive examples and the others as negative examples, the algorithm finds discriminative patterns (represented by solid and dashed squares) in the form of object interactions for a_1. Let us assume that MEDIANGAP procedure returns three as the gap length of the pattern (cd, bc, fr). 3-gap-bounded matches are searched over novel segments. The pattern (cd, bc, fr) matches the three segments with gap lengths 3, 5 and 2 respectively. Only segments one and three are thus 3-gap-bounded matches for the pattern (boldface) while match for segment two exceeds the limits.

Note that our algorithm is not bound to the specific mining technique, and can be fed with patterns obtained by any sequential pattern mining approach (see e.g., [17], [18], [19]).

6 PATTERN-BASED HSMM

Patterns extracted during the mining phase are used to enrich the observation models for segments, in order to provide additional and more expressive evidence for the occurrence of a certain activity.

We begin by showing how to identify pattern matches within a sequence (Algorithm 4). The algorithm takes as inputs a pattern with its estimated gap length and the sequence to be scanned for matches, e.g., a full day, and outputs a list of segments representing pattern matches. For all possible starting instants t_1, it uses the LONGESTMATCH $(\mathbf{x}, t_1, \mathbf{p}, g)$ procedure to compute the longest possible match with at most g gaps. The rationale is that the pattern match should try to cover the longest possible time span, within the estimated limits characterizing the cover of an activity segment by that pattern. The lower and upper borders of this match are added to the list of matching segments, unless the segment overlaps with the previously inserted one (recovered by TOP), in which case the two are merged into a single match spanning both segments.

Algorithm 2 Procedure computing median gap length

Input:
$\mathbf{p}$: a pattern for a specific activity
$\mathcal{D}(\mathcal{S}_y)$: set of training segments for that activity
Output:
MEDIAN($\mathcal{G}$): gap length of the pattern

```
1:  procedure MEDIANGAP(p, D(S_y))
2:      Initialize G to the empty set
3:      m ← LENGTH(p)
4:      for all x ∈ D(S_y) do
5:          g_max ← −1
6:          for all t_1 ∈ [1, T] do
7:              (M_1, M_2) ← LONGESTMATCH(x, t_1, p, T − m)
8:              g ← M_2 − M_1 + 1 − m
9:              if g > g_max then g_max ← g
10:         if g_max > −1 then
11:             G ← G ∪ {g_max}
12:     return MEDIAN(G)
13: end procedure
```

Algorithm 3 Longest pattern match computation

Input:
$\mathbf{x}$: a sequence to be matched
t_1: starting instant of the sequence
$\mathbf{p}$: a pattern for a specific activity
g: gap length of the pattern $\mathbf{p}$
Output:
(M_1, M_2): initial and final positions of the match

```
1:  procedure LONGESTMATCH(x,t1,p,g)
2:      m ← LENGTH(p)
3:      if not MATCH(x_{t1},p1) then
4:          return (0, −1)
5:      if m > 2 then
6:          c ← t1 + 1
7:          for all u ∈ [2, m − 1] do
8:              while c − t1 < u + g do
9:                  if MATCH(x_c,p_u) then
10:                     break
11:                 c ← c + 1
12:             if c − t1 ≥ u + g then
13:                 return (0, −1)
14:     else
15:         c ← t1
16:     v ← t1 + m + g − 1
17:     while v > c do
18:         if MATCH(x_v,p_m) then
19:             return (t1, v)
20:         v ← v − 1
21:     return (0, −1)
22: end procedure
```

Algorithm 4 Procedure for recovering pattern matches

Input:
$\mathbf{p}$: a pattern for a specific activity
g: gap length of the pattern $\mathbf{p}$
$\mathbf{x}$: sequence to be scanned for matches
Output:
$\mathcal{M}$: list of matching segments for the pattern $\mathbf{p}$

```
1:  procedure PATTERNMATCHES(p,g,x)
2:      Initialize M to the empty list
3:      for all t1 ∈ [1, T] do
4:          (M1, M2) ← LONGESTMATCH(x,t1,p,g)
5:          if M2 > −1 then
6:              if M is empty then
7:                  add (M1, M2) to M
8:              else
9:                  (M1', M2') ← TOP(M)
10:                 if M1 ≤ M2' then
11:                     replace M2' with MAX(M2', M2)
12:                 else
13:                     add (M1, M2) to M
14:     return M
15: end procedure
```

Pattern matches are used to compute the probability that a certain pattern covers a sequence segment given the segment label. Let $C^{(\mathbf{p},g)}_{\mathbf{x}_{b_u:e_u}}$ be a boolean random variable modeling approximate coverage of pattern $(\mathbf{p}, g)$ over segment $\mathbf{x}_{b_u:e_u}$, i.e.,

$$C^{(\mathbf{p},g)}_{\mathbf{x}_{b_u:e_u}} = \begin{cases} 1, & \text{if } \mathrm{COVER}(\mathbf{x}_{b_u:e_u}, \mathbf{p}, g, \tau), \\ 0, & \text{otherwise.} \end{cases}$$

Here $\mathrm{COVER}(\mathbf{x}_{b_u:e_u}, \mathbf{p}, g, \tau)$ is true if segment $\mathbf{x}_{b_u:e_u}$ is approximately *covered* by pattern $\mathbf{p}$. This happens if there is a g-gap-bounded match of the pattern spanning almost all the segment (with a threshold τ defining the desired approximation). Largest coverage of activity segments was indeed the driving principle when mining patterns on known activity segments. The (approximate) coverage is formally defined as

$$\mathrm{COVER}(\mathbf{x}_{b_u:e_u}, \mathbf{p}, g, \tau) \Leftrightarrow \frac{d_I}{d_u} \geq \tau,$$

where

$$d_I = \sum_{(b_v,e_v)\in\mathcal{M}^{(\mathbf{p},g)}} |(b_v, e_v) \cap (b_u, e_u)|$$

is the fraction of segment (b_u, e_u) covered by any match of pattern $\mathbf{p}$ and τ is the desired coverage approximation, which was set to 0.9 in our experiments (similar values generated similar results, while choosing substantially lower coverages produced performance worsening, see the discussion above). Note that the summation runs over disjoint matches as overlapping ones have already been merged by the PATTERNMATCHES procedure. The $\mathrm{COVER}(\mathbf{x}_{b_u:e_u}, \mathbf{p}, g, \tau)$ procedure returns true also if segment $\mathbf{x}_{b_u:e_u}$ is contained in a g-gap-bounded match as a proper subsequence, i.e., the match exceeds the borders of the segment. This is consistent with the fact that pattern matches are maximal within the g-gap-length limit. A probabilistic model of match duration, combined with the standard segment duration probability of HSMM, contributes to determine during inference the optimal segmentation according to the learned probabilities, as will be discussed in the following.

The pattern-based HSMM model is obtained modifying the conditional probability of seeing an observation segment given a segment label (Eq. 2) in order to include evidence concerning patterns

$$p(\mathbf{x}_{b_u:e_u} \mid y_u) = \prod_{t=b_u}^{e_u} p(\mathbf{x}_t \mid y_u) p\Big(C^{(\mathbf{p},g)^{(1)}}_{\mathbf{x}_{b_u:e_u}}, \ldots, C^{(\mathbf{p},g)^{(m)}}_{\mathbf{x}_{b_u:e_u}} | y_u\Big), \quad (3)$$

where $m = |\mathcal{P}|$ is the number of patterns and the joint probability ranges over all patterns $(\mathbf{p}, g)^{(i)} \in \mathcal{P}$. As for the term modeling sensor activations only, we make a Naive Bayes assumption of independency between patterns given the segment label. This substantially simplifies the probabilistic model, allowing for efficient inference as shown in Section 7. Note however that this assumption can be easily violated when e.g., patterns share common elements. We decided to tradeoff expressivity for tractability, but a more complex probabilistic model could be conceived. The simplified probability becomes

$$p(\mathbf{x}_{b_u:e_u} \mid y_u) = \prod_{t=b_u}^{e_u} p(\mathbf{x}_t \mid y_u) \prod_{(\mathbf{p},g)\in\mathcal{P}} p\Big(C^{(\mathbf{p},g)}_{\mathbf{x}_{b_u:e_u}} \mid y_u\Big). \quad (4)$$

The conditional probability of observing a pattern coverage is computed as the product of the probability of a match given the activity y_u under consideration, times the probability that the (approximately) covered segment has a certain duration d_u. Given a boolean random variable $M^{(\mathbf{p},g)}_{\mathbf{x}_{b_u:e_u}}$

indicating a pattern match and a random variable $D^{(\mathbf{p},g)}_{\mathbf{x}_{b_u:e_u}}$ modeling the duration of the covered segment, the probability can be written as

$$\begin{aligned} p\big(C^{(\mathbf{p},g)}_{\mathbf{x}_{b_u:e_u}} = 1 \mid y_u\big) &= P\big(M^{(\mathbf{p},g)}_{\mathbf{x}_{b_u:e_u}} = 1, D^{(\mathbf{p},g)}_{\mathbf{x}_{b_u:e_u}} = d_u \mid y_u\big) \\ &= P\big(M^{(\mathbf{p},g)}_{\mathbf{x}_{b_u:e_u}} = 1 \mid y_u\big) \\ &\quad P\big(D^{(\mathbf{p},g)}_{\mathbf{x}_{b_u:e_u}} = d_u \mid M^{(\mathbf{p},g)}_{\mathbf{x}_{b_u:e_u}} = 1, y_u\big). \end{aligned} \tag{5}$$

The former term is estimated during training as the fraction of y_u activity segments matching the pattern. The latter can be modeled with any appropriate duration distribution. For consistency in our experiments we choose the same distribution used for modeling segment duration, i.e., a multinomial distribution over n_b duration bins. Given the longest match duration d_{max} found in the training set, a uniform bin width bw is computed as

$$bw = \max\left(1, \frac{d_{max}}{n_b}\right).$$

Probabilities for each bin are then computed as normalized counts of the training activity segments whose duration fall into the bin. The duration probability for a novel segment then corresponds to the probability of the bin that d_u falls into. We run experiments for varying number of bins (from 3 to 15), achieving similar performance and consistent comparative behavior. All reported results are for $n_b = 5$ (which was the value used in [5]).

Given a test sequence, decoding consists of finding the most probable sequence of activity segments, i.e.,

$$\mathbf{s}^* = \arg\max_{\mathbf{s}} p(\mathbf{x}, \mathbf{s}),$$

which boils down to identify sequences of activity labels and segment durations. The problem can be addressed by the well-known Viterbi algorithm [15] appropriately modified to account for the novel pattern-based model.

A recursive procedure computes for each time instant t a value $\delta_t(y_j, d)$, representing the probability that activity y_j is performed in the time segment $(t-d+1, t)$, given the most probable segmentation and activity assignment for all past time instants. This is obtained by recursively maximizing over duration d' and activity label y_i of the previous segment

$$\begin{aligned} \delta_t(y_j, d) = \max_{1 \le y_i \le L} \Big(\max_{1 \le d' \le D} (\delta_{t-d}(y_i, d')) p(y_j \mid y_i) \Big) \\ \times p(d \mid y_j) p(\mathbf{x}_{t-d+1:t} \mid y_j), \end{aligned}$$

where segment observation probabilities $p(\mathbf{x}_{t-d+1:t} \mid y_j)$ are computed according to Equation (4). The maximum possible duration D is estimated as the maximum duration of all activity segments in the training set. The base step of the recursion is computed for segments starting at the beginning of the sequence (i.e., $t' \le d$) as

$$\delta_{t'}(y_j, d) = p(d \mid y_j) p(\mathbf{x}_{t'-d+1:t'} \mid y_j) \pi_0(j),$$

where $\pi_0(j)$ is the initial probability for activity y_j, computed as the fraction of training days starting with that activity. At the end of the recursion, the probability of the best activity assignment for the whole sequence is computed as

$$p^* = \max_{1 \le y_j \le L} \Big(\max_{1 \le d \le D} \delta_T(y_j, d) \Big).$$

In order to recover the activity assignment corresponding to p^*, an auxiliary variable $\psi_t(y_j, d)$ is used to keep information on the configuration originating $\delta_t(y_j, d)$, i.e., the most probable previous activity label and segment duration (y_i^*, d'^*) in case time segment $(t-d+1, t)$ is labeled with activity y_j

$$\begin{aligned} \psi_t(y_j, d) &= (y_i^*, d'^*) \\ &= \arg\max_{1 \le y_i \le L} \Big(\arg\max_{1 \le d' \le D} (\delta_{t-d}(y_i, d')) p(y_j \mid y_i) \Big). \end{aligned}$$

Note that terms outside the maximization were discarded here as they are irrelevant for deciding what the maximal configuration is. The best sequence of activity segments is recovered backtracking through these variables, i.e.,

$$\begin{aligned} (y_T^*, d_T^*) &= \arg\max_{1 \le y_T \le L} \Big(\arg\max_{1 \le d_T \le D} \delta_T(y_T, d_T) \Big), \\ (y_{T-d}^*, d_{T-d}^*) &= \psi_{t-d}(y_T^*, d_T^*), \end{aligned}$$

For a detailed description of inference for plain HSMM models, see [15]. Our version differs in the probability of observing a certain segment given its predicted label, i.e., Eq. (4). We can efficiently compute this probability by keeping for each pattern $\mathbf{p}$ some auxiliary structures throughout the inference process, for t ranging from 1 to T: $\mathcal{M}$ is the list of matches for pattern $\mathbf{p}$ (we dropped the superscript to avoid clumsy notation), pre-computed using the PATTERNMATCHES procedure; seg is the index of the currently active segment, initialized at the first segment (or at zero if there are no matches for pattern $\mathbf{p}$); $match$ is a flag indicating whether the current time instant t is after the beginning of the active segment seg (i.e., $\mathcal{M}^1_{seg} < t \le \mathcal{M}^2_{seg}$), or not (i.e., $t \le \mathcal{M}^1_{seg}$); **cov** is a zero-initialized vector of length D which is kept updated so that at time instant t, cov_d contains the coverage of the segment $[t-d, t]$ by pattern matches (for $d = [0, D-1]$). The procedure COVERAGE($\mathcal{M}$, **cov**, seg, $match$, t) updates these structures at each time instant t of the inference process. Algorithm 5 describes the update. First, the **cov** vector is shift of one position to the right, filling the first position with a zero, in order to account for the increased time instant. If the current segment is not matched, the algorithm checks whether t corresponds to its first position ($\mathcal{M}^1_{seg}$), otherwise it checks whether the current match is lost (i.e., $\mathcal{M}^2_{seg} < t$). In this latter case, it searches for the next active segment, updating $match$ in case t corresponds to its starting position. Finally, if $match$ is true (i.e., t is in a pattern segment match), the **cov** vector is updated by one. Note that

COVER $(\mathbf{x}_{t-d:t}, \mathbf{p}, g, \tau)$ can be easily computed as $(cov_d/d) \geq \tau$ for all d in $[0, D-1]$.

Algorithm 5 Procedure for updating coverage

Input:
$\mathcal{M}$: list of matching segments for the pattern $\mathbf{p}$
cov: vector representing coverage of a segment by pattern matches
seg: index of the current active segment
$match$: flag indicating the position of t wrt. seg
t: current time instant
Output:
cov: updated **cov**
seg: updated seg
$match$: updated $match$

```
1:  procedure COVERAGE(M, cov, seg, match, t)
2:      if seg = 0 then return null                          ▷ end of M reached
3:      Shift cov one pos to right adding zero
4:      if not match then
5:          if M^1_seg = t then                              ▷ t first pos in seg
6:              match ← True
7:      else if M^2_seg < t then                             ▷ t after last pos in seg
8:          match ← False
9:          seg ← 0
10:         for all nseg ∈ [seg + 1, |M|] do
11:             if M^1_nseg ≥ t then
12:                 seg ← nseg
13:                 if M^1_nseg = t then
14:                     match ← True
15:                 break
16:     if match then                                        ▷ update coverage
17:         increment all elements of cov by 1
18:     return (cov, seg, match)
19: end procedure
```

7 Computational Complexity

Complexity of the LONGESTMATCH is $O(\ell m N)$, where m is the pattern length, $\ell = m + g$ is the overall maximal length of a possible match and N is the number of sensors. We assume here that the matching procedure MATCH is linear in the number of sensors for a pattern element, as sensor activations are stored in lookup tables thanks to the limited size of N. Complexity of PATTERNMATCHES is thus $O(T\ell m N)$, and recovering all maximal matches for all patterns costs $O(PT\ell m N)$ with P the total number of mined patterns. This is a loose upper bound on the actual complexity, as for instance the number of pattern match evaluations is typically much smaller than ℓ. We experimentally verified that the cost of this procedure is negligible with respect to the overall cost of inference. Complexity of MEDIANGAP is $O(|\mathcal{S}|T^2 m N)$ where $|\mathcal{S}|$ is the number of activity segments, and the overall complexity of median gap computation for all patterns is $O(P|\mathcal{S}|T^2 m N)$. The procedure actually runs in time comparable to the PATTERNMATCHES one, given the average length difference between activity segments used in the former and full days used in the latter (difference ignored in the asymptotic analysis).

The inference step of plain HSMM has complexity $O(TL^2 D)$ where L is the number of activities (states) and D their maximal duration. The COVERAGE procedure has complexity $O(D)$. The pattern-related portion thus contributes with $O(TPD)$ to the overall inference complexity, which becomes $O(T(L^2 + P)D)$. Given that the number of patterns P is usually smaller the squared number of states, the complexity of inference is basically unaffected.

The computational bottleneck of the overall approach is the pattern mining step, i.e., the SEQUENTIALMINER procedure.

TABLE 2
Details of the Data Sets

		Duration	Sensors	Activities	Data points	Annotation
Van K.	House A	25 days	14	10	35486	Bluetooth
	House B	14 days	23	13	19968	Diary
	House C	19 days	21	16	26236	Bluetooth
CASAS	Resident 1	46 days	52	7	64795	Diary
	Resident 2	46 days	63	9	64785	Diary

Discriminative pattern mining is especially expensive as the anti-monotonic property typically used in standard mining does not hold. The PBOOST algorithm, for instance, needs to train a linear classifier in order to compute the discriminant power of the patterns during the mining procedure. The cost of the mining procedure widely varies on the different scenarios, strongly depending on the degree of sparsity of the sensor activations, but can be one or two orders of magnitude slower than the inference process in the worst cases. Sequential pattern mining is a popular research area, and a number of alternative approaches have been suggested in the literature (see [29] for a recent review). We did not investigate the performance of the different algorithms in our activity recognition scenario, e.g., discriminative versus non-discriminative approaches, as our contribution is focused on proposing a probabilistically sound framework to incorporate them. Further research on these efficiency issues is anyhow necessary in order to allow segmental pattern mining over large-scale data sets.

8 Experiments

In this section we first present the experimental setup used for evaluating the proposed approach and then provide results of the experiments.

8.1 Setting

We performed our experiments on a collection of freely available[1,2] benchmark data sets. Van Kasteren's data set include information regarding three different houses comprising several wireless sensor networks [4], [5]. Each node of the network is attached to ad-hoc sensors, e.g., reed switches, passive infrared. Annotation of the activities was achieved by recording the start and end time of the corresponding activity either via handwritten diary or bluetooth headset. CASAS data set differs from the previous one as two residents are simultaneously monitored in the apartment, with a sensor network mainly composed of motion and utility usage sensors [30]. Annotators labeled the data using a 3D visualization tool and residents' diaries.

Table 2 presents a summary of the characteristics of the data sets. Note that in CASAS data set there are distinct activities and sensors for the two residents.

Activities to be recognized were derived from the Katz ADL index which is a measure qualifying the ability of individuals to sustain their lives independently [31] for the first data set, and from the clinical questionnaires [32] for the second data set. The activities considered in these data sets

1. https://sites.google.com/site/tim0306/codeFramework.zip.
2. http://ailab.wsu.edu/casas/datasets/twor.2009.zip.

TABLE 3
Notations for Feature Representations

Feature (Notation)	Explanation
Raw (R)	Sensor is active as long as it fires
Changepoint (C)	Sensor is active only when it changes its state
Last-fired (L)	Sensor is active until another one changes its state
Dual C (DC)	Dual sensor feature: one for its activation and another for deactivation
C+L	Combination of C and L
C+LP	Combination of C and patterns mined on L
C+CP	Combination of C and patterns mined on C

are listed in tables from 5 to 9 which report the breakdown of results by single activity. 'idle' indicates that none of the annotated activities is being performed. Some of the activities, namely 'going to bed'/'sleeping', 'leaving house'/'going out to work' and 'idle' itself, take significantly more time than the others on average. We will refer to these as long-lasting activities as opposed to short-lived ones.

Data acquired from the sensors were processed in different ways to create feature representations (see Table 3 for a summary of notations). The *Raw* representation is the unprocessed one, where a sensor is active in all time instants in which it fires. The *Changepoint* (C) one considers a sensor active (value 1) only in the time instants in which it alters its state. The *last-fired* (L) representation keeps considering the last sensor which changed state active in all following time instants, until another sensor changes its state. Note that in case multiple sensors change their state in the same time instant, all of them are considered active for that time instant. The last one that changed its state is carried over to the following time instants. Fig. 2 depicts the working mechanism of these representations as compared to the original raw one. We also considered a dual changepoint (DC) representation, distinguishing between activation and de-activation events. While C and DC representations provide reasonable information for all types of sensors, L one is meaningful for object interaction sensors (e.g., toilet light switch activation remaining active indicates a 'using toilet' activity), mostly found in Van Kasteren data set, but is misleading for the motion sensors characterizing CASAS data set (within the one minute timeframe characterizing a time instant, only the last position would be recorded, losing the trajectory information). Indeed L representation is always harmful in CASAS data set, as will be seen in the experimental results.

A 'leave-one-day-out' (LOO) approach was used to split the data sets into training and test sets. Each day was in turn considered as a test set, while all other days made up the training set. Performance measures include precision ($\mathrm{Pr} = \mathrm{TP}/(\mathrm{TP}+\mathrm{FP})$), recall ($\mathrm{Re} = \mathrm{TP}/(\mathrm{TP}+\mathrm{FN})$), and

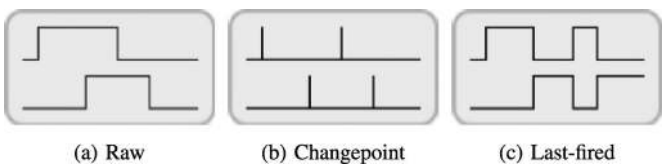


Fig. 2. Feature representations (taken from [5]).

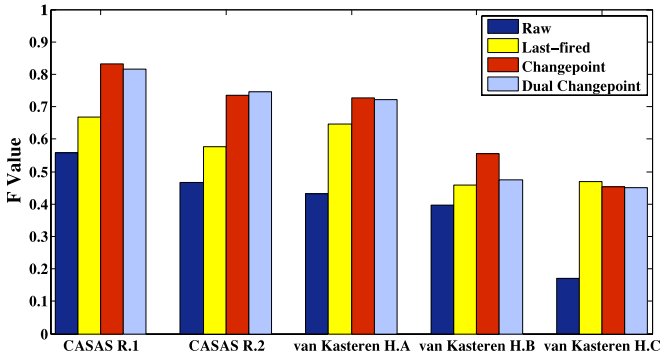


Fig. 3. Comparison of feature representations.

F1 ($\mathrm{F1} = (2 \cdot \mathrm{Pr} \cdot \mathrm{Re})/(\mathrm{Pr}+\mathrm{Re})$) for each class. Here TP and FP are the fraction of true and false positives respectively, while TN and FN are the fraction of true and false negatives respectively. F1 is the harmonic average of precision and recall trading off the two.

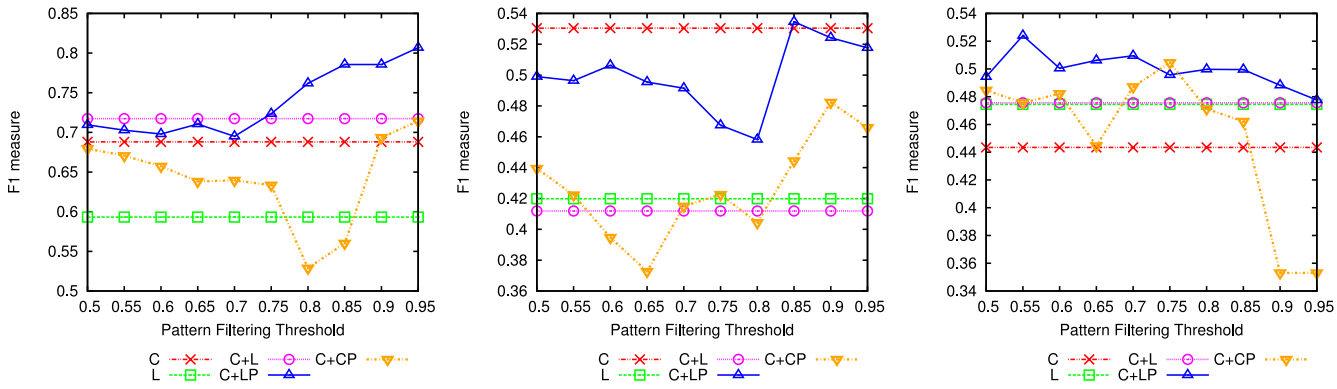
8.2 Results

Classification performances of predictive models differ greatly for varying feature representations. Fig. 3 reports F1 measures for a plain HSMM using different feature representations for all experimental data sets. Results with plain HMM have the same behavior, while being one or two points of F1 measure worse.

The Raw representation (R) is substantially worse than all others. In the Van Kasteren data set for instance, a sensor attached on a door that is left open after completion of an activity keeps firing continuously while other activities are being performed. In such situations, the R representation is incapable of capturing ongoing activities as it tends to have traces of already completed ones. Similarly, in the CASAS data set, motion sensors tend to keep firing for a long time after the position was left, possibly due to problems in the sensor measurements. This leads to multiple overlapping sensor activations, eventually deteriorating trajectory information. These results are consistent with those observed in [5]. We thus did not consider this representation in the rest of the experimental evaluation.

The L representation prevents these degenerate behaviors by focusing on the last sensor changing its state, forgetting about the state of previously recorded sensors. On the other hand, this can propagate a sensor activation too long if no other sensor is observed. This problem is especially relevant when an activity is followed by "idle", where no sensor fires. This is especially relevant in the CASAS data set, where a number of activities (e.g., sleeping, watching TV) do not produce movement sensor firings. The C representation tends to provide the best results on average, again consistent with results in [5]. The DC representation has very similar results to the C one while being slightly more computationally expensive. We thus used the C representation as a baseline in all following experiments.

Figs. 4 and 5 report results of pattern-based HSMM for increasing values of the threshold over pattern discriminative power, as compared to a set of baseline alternatives. Note that results for the alternative methods are slightly different from the ones reported in [5] and [6], as we learned



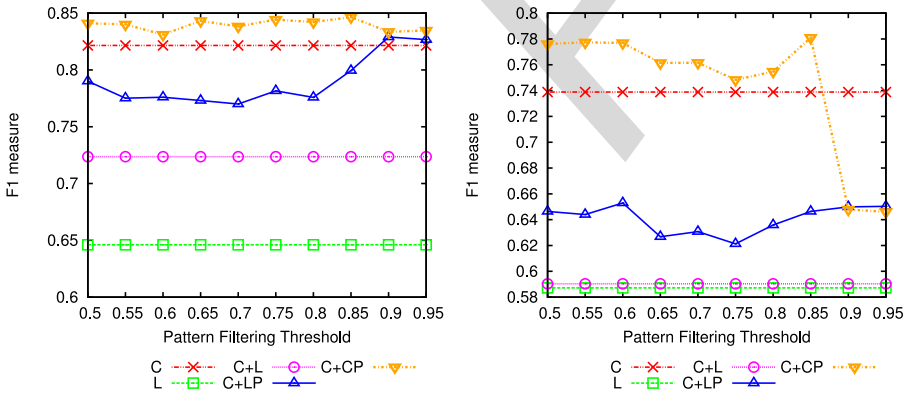
(a) House A (b) House B (c) House C

Fig. 4. Experimental results for van Kasteren data set.

duration models for all activities, while previous settings limited this to short-lived ones. Results show that simply combining the two representations (rows C+L) without using patterns does not allow improving over the best of the two in general. Results obtained by including pattern-based features differ depending on informativeness of the representation on which patterns are mined. As explained when introducing sensor encodings, the L representation is informative when applied to object interaction sensors. Indeed C+LP models, combining C representation with patterns mined on the L one (LP), are usually better than any model lacking patterns, i.e., C, L or C+L, and are almost always better than C+CP models, combining C representation with patterns mined on the C one (CP). The improvement is more evident when the L representation is clearly informative and complementary to the C one (e.g., Van Kasteren, House A), and fades away when it tends to be harmful, as in Van Kasteren, House B. When dealing with motion sensors, i.e., the CASAS data set, the L representation is always misleading, as previously explained. Hence, C+LP models mostly fail to provide improvement over baseline models. However, patterns mined on the C representation (CP) succeed in improving performance when combined to plain C features, as shown in Fig. 5.

We also evaluated the relevance of the segmental probabilistic model in exploiting the discriminative power of patterns. To this aim we labeled each time instant within a pattern match with the activity represented by the pattern (allowing for multiple labels for the same instant). The performance in this case is worse than that of all other methods for all values of the pattern selection threshold ϕ (results not shown).

Concerning the number of patterns to be used, there is a tradeoff between having enough patterns to model all activities, and focusing on the most discriminative ones. While improvements (apart from the single case in which patterns are useless, i.e., Van Kasteren, House B) over all alternative representations are observed for most values of the threshold on average, the best tradeoff differs in the different scenarios. In Van Kasteren House A, for instance, the best results are obtained when only the most discriminative patterns are used. Less discriminative ones suffer from the tendency of the L representation to extend the signal of an activity to the following "idle" period, as previously discussed. On the other hand, in Van Kasteren, House C (see Fig. 4c) a too high threshold leads to a decrease in performance. This is due to a lack of patterns discriminating similar activities. 'brushing teeth', 'shaving', and 'taking medication' are held in front of the faucet by interacting with very similar objects. Applying a high threshold yields patterns only for the 'shaving' among the three and leads to confusion in the prediction. The problem of wrongly missing "idle" periods for lower thresholds is less severe here, as in this case there are



(a) Resident 1 (b) Resident 2

Fig. 5. Experimental results for CASAS data set.

TABLE 4
Thresholds Obtained from Internal CV Procedure

	Van Kasteren			CASAS	
	H. A	H. B	H. C	R. 1	R. 2
Thresholds	0.95	0.85	0.55	0.75	0.5

patterns specifically characterizing the "idle" period (usually repetitions of the couch pressure sensor). A similar behavior can be observed in CASAS, Resident 2 (see Fig. 5b), where focusing on a small set of patterns discards these "idle" characterizing patterns. The large drop in performance observed for thresholds larger than 0.85 is mostly due to an overprediction of the 'sleeping' activity. This is characterized by patterns with no pattern activations at the borders, movement related sensors in between and quite large gaps. While these patterns correctly discriminate 'sleeping' activity segments from the others during training, when applied to full days they tend to span time periods consisting of pairs of "Idle" segments with some other motion-related activity in between. The presence of patterns for "idle" as well as other activities helps to disambiguate these cases for lower pattern selection thresholds.

In order to select a single threshold for a more detailed experimental evaluation, we run an internal cross validation procedure within each training fold of the outer LOO. The final ϕ was chosen as the best performing when averaging across internal cross validations, and kept fixed for the outer cross validation procedure. Thresholds for the different data sets are shown in Table 4.

Tables 5, 6 and 7 show the breakdown of the results by activity for the Van Kasteren data sets at the thresholds determined by the inner cross validation. Results indicate that the contribution of the L representation is rather unstable across activities, preventing an overall improvement for the combined C+L representation. Conversely, the C+LP representation is much more robust, managing to combine the advantages of the two representations. Consider, for instance, the activities 'taking shower', 'idle' and 'leaving house'. These are closely related as they are typically performed in a row by the resident. A simple pattern commonly found in House A consists of a long repetition of the 'front door' sensor. This clearly indicates a 'leaving house' activity is taking place. It also helps in disambiguating temporally close activities like the just mentioned 'idle' and 'taking shower' ones. Conversely, including L representation also introduces noisy features: a sensor activated while taking shower will continue to be considered active when the resident is actually idle, leading to a drop in precision for the 'taking shower' prediction. Concerning House B, overall improvements are limited as patterns fail to improve recognition of the 'idle' "activity", which is by far the most common one. The main improvement is observed for the 'going to bed' (aka 'sleeping') activity. This is due to a common pattern made of a sequence of bed pressure mat sensor

TABLE 5
Breakdown of the Results by Activity for Van Kasteren House A

	C			C+L			C+LP		
Activity	**Precision**	**Recall**	**F-1**	**Precision**	**Recall**	**F-1**	**Precision**	**Recall**	**F-1**
Idle	86.1	57.5	68.9	96.6	58.4	72.8	94.9	76.0	84.4
Leaving house	92.0	99.5	95.6	98.4	99.7	99.0	97.9	99.9	98.9
Using toilet	74.1	82.5	78.1	72.9	78.9	75.8	76.9	78.4	77.6
Taking shower	97.8	34.7	51.2	35.9	92.4	51.7	94.7	78.9	86.1
Brushing teeth	13.3	34.4	19.1	11.0	31.3	16.3	25.0	43.8	31.8
Going to bed	90.2	89.2	89.7	98.1	99.5	98.8	95.7	99.4	97.5
Preparing breakfast	61.6	70.1	65.6	67.1	56.3	61.3	64.1	75.9	69.5
Preparing dinner	60.8	57.8	59.3	25.7	85.4	39.5	64.2	70.0	67.0
Getting snack	38.3	54.8	45.1	18.2	61.9	28.1	54.8	54.8	54.8
Getting drink	73.2	61.2	66.7	50.9	59.2	54.7	76.9	61.2	68.2

TABLE 6
Breakdown of the Results by Activity for Van Kasteren House B

	C			C+L			C+LP		
Activity	**Precision**	**Recall**	**F-1**	**Precision**	**Recall**	**F-1**	**Precision**	**Recall**	**F-1**
Idle	71.3	57.6	63.7	56.0	56.8	56.4	60.6	61.7	61.1
Leaving house	85.1	91.1	88.0	95.7	32.5	48.6	88.1	91.3	89.7
Using toilet	41.7	71.4	52.6	40.0	72.7	51.6	44.4	67.5	53.6
Taking shower	92.8	92.8	92.8	13.8	91.9	23.9	83.3	90.1	86.6
Brushing teeth	13.3	22.2	16.7	11.9	13.9	12.8	8.4	22.2	12.2
Going to bed	95.4	69.9	80.6	93.4	80.4	86.4	99.1	75.0	85.4
Dressing	20.0	65.2	30.6	13.3	69.6	22.4	19.9	60.9	30.0
Preparing brunch	43.7	61.9	51.2	35.3	35.7	35.5	47.9	53.6	50.6
Preparing dinner	40.0	53.5	45.8	25.2	38.0	30.3	42.7	53.5	47.5
Getting drink	17.7	42.9	25.0	11.4	28.6	16.3	15.4	42.9	22.6
Washing dishes	15.6	47.6	23.5	5.6	9.5	7.0	0	0	0
Eating dinner	0.7	14.3	1.4	0.4	42.9	0.8	0.7	14.3	1.4
Eating brunch	11.5	21.2	14.9	0.9	26.7	1.8	13.3	21.2	16.3

TABLE 7
Breakdown of the Results by Activity for Van Kasteren House C

	C			C+L			C+LP		
Activity	**Precision**	**Recall**	**F-1**	**Precision**	**Recall**	**F-1**	**Precision**	**Recall**	**F-1**
Idle	58.5	41.3	48.4	65.1	69.8	67.4	67.9	75.6	71.5
Leaving house	77.6	94.5	85.2	98.3	86.0	91.7	98.1	86.2	91.8
Eating	38.5	43.6	40.9	40.1	32.2	35.7	35.0	51.9	41.8
Using toilet down	38.5	57.0	45.9	26.1	68.4	37.8	31.4	63.9	42.1
Taking shower	61.6	32.1	42.2	34.5	58.4	43.4	52.3	72.1	60.6
Brushing teeth	16.8	27.7	20.9	21.8	31.7	25.8	16.3	15.8	16.1
Using toilet up	19.0	36.3	24.9	20.1	45.0	27.8	37.6	43.8	40.5
Shaving	0	0	0	13.1	11.6	12.3	27.8	21.7	24.4
Going to bed	99.7	82.6	90.3	99.7	98.2	98.9	99.7	98.3	99.0
Dressing	52.5	65.2	58.2	62.1	68.8	65.3	64.5	71.4	67.8
Taking medication	10.8	26.7	15.4	11.8	26.7	16.3	11.2	66.7	19.2
Preparing breakfast	4.2	2.8	3.4	4.4	5.6	5.0	6.8	25.4	10.8
Preparing lunch	12.9	25.0	17.1	5.1	11.7	7.1	9.9	11.7	10.7
Preparing dinner	64.2	47.6	54.7	23.1	75.2	35.3	28.4	56.6	37.8
Getting snack	18.2	41.7	25.3	15.7	45.8	23.4	31.8	29.2	30.4
Getting drink	0	0	0	0	0	0	0	0	0

TABLE 8
Breakdown of the Results by Activity for CASAS Resident 1

	C			C+L			C+CP		
Activity	**Precision**	**Recall**	**F-1**	**Precision**	**Recall**	**F-1**	**Precision**	**Recall**	**F-1**
Idle	87.9	99.9	93.5	96.4	62.7	76.0	95.8	98.1	97.0
Bed to toilet	43.5	17.0	24.4	37.5	67.5	48.2	45.1	34.8	39.2
Preparing breakfast	96.1	75.1	84.3	95.0	75.5	84.1	95.7	87.3	91.3
Grooming	73.4	75.3	74.3	80.0	66.2	72.4	76.8	77.2	77.0
Sleeping	99.6	77.9	87.4	68.0	99.4	80.8	97.0	96.0	96.5
Working at computer	99.3	83.8	90.9	51.1	85.2	63.9	96.5	81.5	88.4
Working at dining room	70.5	86.0	77.5	41.4	87.2	56.1	41.1	65.8	50.6

TABLE 9
Breakdown of the Results by Activity for CASAS Resident 2

	C			C+L			C+CP		
Activity	**Precision**	**Recall**	**F-1**	**Precision**	**Recall**	**F-1**	**Precision**	**Recall**	**F-1**
Idle	88.4	99.7	93.7	92.9	55.8	69.7	94.1	97.8	95.9
Bed to toilet	47.3	83.6	60.4	50.0	82.1	62.2	66.3	88.1	75.6
Preparing breakfast	44.7	52.8	48.4	42.2	46.8	44.3	56.0	60.2	58.0
Grooming	95.5	62.2	75.3	43.6	87.8	58.2	86.3	94.2	90.1
Preparing dinner	47.3	32.5	38.5	44.4	32.3	37.4	37.8	24.7	29.9
Preparing lunch	26.8	10.1	14.7	25.1	10.2	14.5	19.8	22.3	21.0
Sleeping	99.4	86.1	92.3	64.0	98.6	77.6	96.3	91.6	93.9
Watching TV	84.5	71.7	77.6	62.4	85.1	72.0	85.5	89.8	87.6
Working at computer	99.4	84.3	91.2	39.1	88.4	54.2	99.1	84.7	91.3

activations, which mainly disambiguates it with respect to leaving house'. Note that simple C+L representation (without patterns) leads to substantial performance worsening with respect to C representation alone in this setting. Spurious activations of kitchen sensors are wrongly taken as indication of kitchen activities when the resident is actually outside, while the pattern-based representation is robust to these noisy observations. Concerning House C, a commonly found pattern consists of a sequence of sensor activations for the fridge, the herbs cupboard and the fridge again, characterizing the 'preparing dinner' activity. Let us consider a possible scenario for this situation. A resident starts cooking his dinner by taking ingredients from the fridge. After a while, flavoring herbs and spices taken from the herbs cupboard are added into the blend. As soon as the meal is ready, the resident puts the remaining ingredients back to the fridge. Introducing such patterns allows relating observations which are not sufficient to discriminate among similar activities if taken alone. For instance, a similar scenario involving usage of fridge and other kitchen appliances can be observed for preparing breakfast. The patterns found for this last activity include activations for the cutlery drawer, the bowl cupboard, and the fridge. Both activities are actually better recognized using the C+LP representation.

Activity by activity analysis on the CASAS data set using the cross validated thresholds is presented in Tables 8 and 9. CP patterns manage to provide improvements over almost all activities. Relatively lower cross-validated thresholds with respect to Van Kasteren data set enable more patterns to be retained. All activities

(but 'bed to toilet' for Resident 1) possess corresponding patterns for both residents. Found patterns mostly characterize activities in terms of trajectories, sometimes combined with the object usage like burner or faucet. Taking the first resident into consideration (see Table 8), one can anticipate that each activity is performed in a specific location, e.g., 'grooming' in the bathroom, 'sleeping' in one corner of the bedroom, 'working at computer' in another corner of the bedroom, yielding distinct patterns. 'Grooming' patterns, for instance, show a trajectory going from the bedroom to the faucet (and possibly the mirror) in the bathroom, where some time is spent (indicated by multiple activations of the motion sensor facing the faucet). The pattern for 'sleeping' includes activations of two adjacent motion sensors (representing the locations on the bed) combined with the 'no sensor activation' in one corner of the bedroom. 'Working at computer' contains repetitive activations of a single motion sensor (representing the location of the computer) in another corner.

Given the sparsity of C representation (with respect to the L one), patterns sometimes contain 'no sensor activation', especially towards the end of the activity, which can occasionally lead to confusion with the 'idle' activity. This explains the few observed performance drops, occurring for 'working at computer' and 'working at dining room'. On the other hand, a long sequence of 'no sensor activation' characterizes the 'idle' activity and helps disambiguating it from the 'bed to toilet' one, characterized by a much shorted duration, which is in turn better predicted without having patterns in itself. The second resident highlights how patterns help disambiguating among similar activities, like the ones performed in the kitchen: 'preparing breakfast', 'preparing dinner', and 'preparing lunch'. 'Preparing breakfast' can be easily distinguished from others thanks to its rich unique patterns modeling trajectories, e.g., from the kitchen towards the upper floor to the room of the resident or from the kitchen to the cellar to the upper floor. It is obvious from the patterns that the resident prefers having his breakfast in his room, which is a distinct property.

9 Conclusions

We presented a segmental pattern mining approach for enriching sequential representations with patterns characterizing interactions within activity segments. Experimental results show the usefulness of the mined patterns in improving predictive power of learning algorithms. However, the amount of improvement varies greatly from one feature representation to another. The choice of the appropriate representation can be made according to its robustness in correlating sensor activation patterns and activities being performed. Sensors signalling object interactions, like in van Kasteren data set, are suitable to the L representation, since the latest sensor activation tends to indicate the activity that is to be performed, e.g., bedroom door for sleeping, front door for leaving. The C representation, on the other hand, is useful in identifying activities represented by trajectories, like in CASAS data set, where each motion sensor activation constitutes a part of a trajectory. We indeed achieved the best performance using C+LP and C+CP for the houses with contact switch sensors and with motion sensors respectively.

In our experiments, we observed that no single pattern selection threshold provided the best performance in all scenarios. A common problem causing a drop in performance for certain threshold choices stems from mishandling of idle class, e.g., lack of idle patterns for higher thresholds (resident 2, CASAS) and co-occurrence of distinct activity patterns suppressing the effect of those representing the idle one (house A, van Kasteren). A possible solution consists of treating the idle class as a separate case, for which e.g., patterns with a lower threshold could be retained.

Our approach is currently limited to non-concurrent activities. However, many real world scenarios involve overlapping and concurrent activities, especially when modeling multiple interacting agents. The underlying idea can be generalized to such scenarios by combining mined patterns with more expressive sequential models like factorial HSMMs. The case of multiple agents is especially challenging as it requires to disambiguate those sensors which do not provide information on the agent involved (e.g., infrared sensors). Finally, our approach assumes a batch setting, in which a whole (temporal) sequence has to be jointly labeled after being fully observed. Online activity recognition, where the aim is predicting the currently performed activity, requires to adapt the segmental pattern mining algorithm to search for incremental patterns modeling increasingly long portions of an activity segment and deal with the increased complexity of the mining, matching, modelling and inference steps.

Our approach is not limited to activity recognition tasks and is readily applicable to sequential labeling problems characterized by segments of consecutive positions sharing the same label (e.g., intron-exon identification in DNA sequences). The segmental mining strategy can also be used for suggesting promising topologies for graphical models trying to directly incorporate long-range dependencies. Our segmental pattern miner extracts patterns which should approximately span activity segments. Their matches are thus natural candidates to add shortcuts as in skip-chain CRF, possibly connecting distant segments representing the same or closely related activities. We are pursuing this direction in our ongoing research.

Acknowledgments

This research was partially supported by grant PRIN 2009LNP494 (Statistical Relational Learning: Algorithms and Applications) from Italian Ministry of University and Research and by the Autonomous Province of Trento, Call for proposal Major Projects 2006 (project ACube). The authors would like to thank Tim van Kasteren for providing the framework for replicating his experiments.

References

[1] L. Bao and S. Intille, "Activity Recognition in the Home Setting Using Simple and Ubiquitous Sensors," *Proc. Int'l Conf. Pervasive Computing*, pp. 273-297, 2004.

[2] M. Philipose, K. Fishkin, M. Perkowitz, D. Patterson, D. Fox, H. Kautz, and D. Hähnel, "Inferring Activities from Interactions with Objects," *IEEE Pervasive Computing*, vol. 3, no. 4, pp. 50-57, Oct.-Dec. 2004.

[3] D. Vail, M. Veloso, and J. Lafferty, "Conditional Random Fields for Activity Recognition," *Proc. Sixth Int'l Joint Conf. Autonomous Agents and Multiagent Systems (AAMAS '07)*, pp. 1331-1338, 2007.
[4] T. van Kasteren, G. Englebienne, and B. Kröse, "Activity Recognition Using Semi-Markov Models on Real World Smart Home Datasets," *J. Ambient Intelligence and Smart Environments*, vol. 2, no. 3, pp. 311-325, Aug. 2010.
[5] T. van Kasteren, G. Englebienne, and B. Kröse, "Human Activity Recognition from Wireless Sensor Network Data: Benchmark and Software," *Activity Recognition in Pervasive Intelligent Environment*, pp. 165-186, Atlantis Press, 2010.
[6] U. Avci and A. Passerini, "Improving Activity Recognition by Segmental Pattern Mining," *Proc. IEEE Int'l Conf. Pervasive Computing and Comm. Workshops (PerCom Workshops)*, pp. 709-714, 2012.
[7] S. Fine, Y. Singer, and N. Tishby, "The Hierarchical Hidden Markov Model: Analysis and Applications," *Machine Learning*, vol. 32, pp. 41-62, July 1998.
[8] T. Duong, D. Phung, H. Bui, and S. Venkatesh, "Efficient Duration and Hierarchical Modeling for Human Activity Recognition," *Artificial Intelligence*, vol. 173, pp. 830-856, May 2009.
[9] P. Natarajan and R. Nevatia, "Hierarchical Multi-Channel Hidden Semi Markov Models," *Proc. 20th Int'l Joint Conf. Artifical Intelligence (IJCAI '07)*, pp. 2562-2567, 2007.
[10] Y. Du, F. Chen, W. Xu, and W. Zhang, "Activity Recognition through Multi-Scale Motion Detail Analysis," *Neurocomputing*, vol. 71, pp. 3561-3574, 2008.
[11] Z. Yan, D. Chakraborty, A. Misra, H.Y. Jeung, and K. Aberer, "SAMMPLE: Detecting Semantic Indoor Activities in Practical Settings Using Locomotive Signatures," *Proc. 16th Int'l Symp. Wearable Computers (ISWC)*, pp. 37-40, 2012.
[12] T. Huynh, M. Fritz, and B. Schiele, "Discovery of Activity Patterns Using Topic Models," *Proc. 10th Int'l Conf. Ubiquitous Computing, (UbiComp '08)*, pp. 10-19, http://doi.acm.org/10.1145/1409635.1409638, 2008.
[13] C. Sutton and A. McCallum, "An Introduction to Conditional Random Fields for Relational Learning," *Introduction to Statistical Relational Learning*, MIT Press, 2007.
[14] D.H. Hu and Q. Yang, "Cigar: Concurrent and Interleaving Goal and Activity Recognition," *Proc. 23rd Nat'l Conf. Artificial Intelligence (AAAI '08)*, pp. 1363-1368, 2008.
[15] S.-Z. Yu, "Hidden Semi-Markov Models," *Artificial Intelligence*, vol. 174, no. 2, pp. 215-243, 2010.
[16] J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu, "Mining Sequential Patterns by Pattern-Growth: The Prefixspan Approach," *IEEE Trans. Knowledge and Data Eng.*, vol. 16, pp. 1424-1440, Nov. 2004.
[17] Y. Hirate and H. Yamana, "Generalized Sequential Pattern Mining with Item Intervals," *J. Computers*, vol. 1, no. 3, pp. 51-60, 2006.
[18] X. Zhu and X. Wu, "Mining Complex Patterns Across Sequences with Gap Requirements," *Proc. 20th Int'l Joint Conf. Artificial Intelligence (IJCAI '07)*, pp. 2934-2940, 2007.
[19] C. Li, Q. Yang, J. Wang, and M. Li, "Efficient Mining of Gap-Constrained Subsequences and Its Various Applications," *ACM Trans. Knowledge Discovery from Data*, vol. 6, no. 1, article 2, 2012.
[20] M.K. Hasan, S. Lee, and Y.-K. Lee, "Using Sensor Sequences for Activity Recognition by Mining and Multi-Class Adaboost," *Proc. Int'l Conf. Artificial Intelligence (IC-AI)*, pp. 735-740, 2010.
[21] A.A. Salah, E. Pauwels, R. Tavenard, and T. Gevers, "T-Patterns Revisited: Mining for Temporal Patterns in Sensor Data," *Sensors*, vol. 10, no. 8, pp. 7496-7513, 2010.
[22] T. Gu, Z. Wu, X. Tao, H.K. Pung, and J. Lu, "epSICAR: An Emerging Patterns Based Approach to Sequential, Interleaved and Concurrent Activity Recognition," *Proc. IEEE Int'l Conf. Pervasive Computing and Comm. (PerCom)*, pp. 1-9, 2009.
[23] T. Gu, Z. Wu, L. Wang, X. Tao, and J. Lu, "Mining Emerging Patterns for Recognizing Activities of Multiple Users in Pervasive Computing," *Sixth Ann. Int'l Mobile and Ubiquitous Systems: Networking Services (MobiQuitous '09)*, pp. 1-10, 2009.
[24] P. Rashidi, D.J. Cook, L.B. Holder, and M. Schmitter-Edgecombe, "Discovering Activities to Recognize and Track in a Smart Environment," *IEEE Trans. Knowledge and Data Eng.*, vol. 23, no. 4, pp. 527-539, Apr. 2011.
[25] L. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proc. IEEE*, vol. 77, no. 2, pp. 257-286, Feb. 1989.
[26] S. Sarawagi and W.W. Cohen, "Semi-Markov Conditional Random Fields for Information Extraction," *Advances in Neural Information Processing Systems 17*, pp. 1185-1192, 2004.
[27] S. Nowozin, G. Bakir, and K. Tsuda, "Discriminative Subsequence Mining for Action Classification," *Proc. IEEE 11th Int'l Conf. Computer Vision (ICCV '07)*, pp. 1-8, 2007.
[28] A. Demiriz, K.P. Bennett, and J. Shawe-Taylor, "Linear Programming Boosting via Column Generation," *Machine Learning*, vol. 46, pp. 225-254, 2002.
[29] N.R. Mabroukeh and C.I. Ezeife, "A Taxonomy of Sequential Pattern Mining Algorithms," *ACM Computing Surveys*, vol. 43, no. 1, article 3, http://doi.acm.org/10.1145/1824795.1824798, Dec. 2010.
[30] D.J. Cook and M. Schmitter-Edgecombe, "Assessing the Quality of Activities in a Smart Environment," *Methods of Information in Medicine*, vol. 48, pp. 480-485, 2009.
[31] S. Katz, T. Downs, H. Cash, and R. Grotz, "Progress in the Development of the Index of ADL," *Gerontologist*, vol. 10, pp. 20-30, 1970.
[32] B. Reisberg, S. Finkel, J. Overall, N. Schmidt-Gollas, S. Kanowski, H. Lehfeld, F. Hulla, S.G. Sclan, H.-U. Wilms, K. Heininger, I. Hindmarch, M. Stemmler, L. Poon, A. Kluger, C. Cooler, M. Bergener, L. Hugonot-Diener, P.H. Robert, and H. Erzigkeit, "The Alzheimer's Disease Activities of Daily Living Int'l Scale (adl-is)," *Int'l Psychogeriatrics*, vol. 13, pp. 163-181, 2001.

Umut Avci received the master's degree in statistics from Ege University in 2006 and in smart systems from the University of Abertay Dundee in 2009. He is currently working toward the PhD degree in the Department of Information Engineering and Computer Science, University of Trento. His main research interest is in the area of statistical relational learning.

Andrea Passerini received the graduate degree in computer science in 2000 and the PhD degree in 2004 from the University of Florence. He is currently an assistant professor in the Department of Information Engineering and Computer Science of the University of Trento. His main research interests include the area of machine learning and knowledge discovery. In recent years, he developed techniques aimed at combining statistical and symbolic approaches to learning, via the integration of inductive logic programming and kernel machines. He is also pursuing a deeper integration of machine learning approaches and complex optimization techniques. He coauthored more than 50 scientific publications.