# Accurate Activity Recognition in a Home Setting

**Tim van Kasteren, Athanasios Noulas, Gwenn Englebienne and Ben Kröse**
Intelligent Systems Lab Amsterdam
University of Amsterdam
Kruislaan 403, 1098 SJ, Amsterdam , The Netherlands
T.L.M.vanKasteren@uva.nl

**ABSTRACT**
A sensor system capable of automatically recognizing activities would allow many potential ubiquitous applications. In this paper, we present an easy to install sensor network and an accurate but inexpensive annotation method. A recorded dataset consisting of 28 days of sensor data and its annotation is described and made available to the community. Through a number of experiments we show how the hidden Markov model and conditional random fields perform in recognizing activities. We achieve a timeslice accuracy of 95.6% and a class accuracy of 79.4%.

**Author Keywords**
Activity Recognition, Probabilistic Models, Dataset, Annotation, Sensor Networks

**ACM Classification Keywords**
I.2.1 - Applications and Expert Systems, I.2.6 - Learning

## INTRODUCTION

Activities are a very important piece of information for ubiquitous applications [3]. A sensor system capable of automatically recognizing activities would allow many potential applications in areas such as health care, comfort and security. For example, in elderly care, activities of daily living (ADLs) are used to assess the cognitive and physical capabilities of an elderly person [10]. An activity recognition system allows us to automatically monitor their decline over time and detect anomalies [21].

For activity recognition research, we need a large variety of datasets. Many datasets for activity recognition were recorded in houses especially built for research purposes [2, 7, 13]. Sensors are installed during construction and people live there only for the duration of an experiment. Since people are unfamiliar with the house the resulting data is not very representative. Also, recording datasets in the same house over and over again makes research sensitive to the architecture of that particular house.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
*UbiComp'08*, September 21-24, 2008, Seoul, Korea.
Copyright 2008 ACM 978-1-60558-136-1/08/09...$5.00.

Furthermore, accurate annotation of activities is also important. It is required for evaluating the performance of recognition models and allows the use of supervised models from machine learning. Current annotation methods are either very expensive [13] or have limited accuracy [18].

Our paper contributes on the following topics. First, we present a sensor network setup that can be easily installed and used in different houses. Second, we present an inexpensive and accurate method for annotation and provide the software for doing so. Third, we offer our sensor data and its annotation online, so that it can be shared by the research community. Fourth, we run a number of experiments on our sensor data showing how to effectively use a probabilistic model for recognizing activities.

The remainder of this paper is organized as follows. In the next section we discuss related work. After that, we give a description of our sensor and annotation system, continued by a description of our recorded dataset. We then present the probabilistic models we used for activity recognition and discuss the experiments and results. Finally, we conclude by summing up our findings.

## RELATED WORK

The sensors we can use range from various wall-mounted sensors (e.g. reed switches [18, 20], motion detectors [1], cameras [5]) to all sorts of wearables (e.g. accelerometers [12], wrist worn RFID reader [14]). The various technologies differ from each other in terms of price, intrusiveness, ease to install and the type of data they output [16].

Annotation has been performed in many different ways. The least interfering method while inhabitants are performing their activities is using cameras [13]. The downside of this method is that processing video data is very costly. Less costly alternatives typically come in the form of self-reporting. The downside here is that the inhabitants are continuously aware of the annotation process, which may lead to biased or unrealistic data. Examples of self-reporting methods are keeping an activity diary on paper or using a PDA which triggers self-report entries either based on the sensor output or at a constant time interval [8].

Models used for recognizing activities can be probabilistic based [5, 14, 20], logic based [11] or hand-crafted [6]. Probabilistic models are popular because sensor readings are noisy and activities are typically performed in a non-deterministic fashion. Different models have been used in different set-

tings, dynamic Bayesian networks were used to recognize the activities of multiple people in a house [20]. Hidden Markov models were used to perform activity recognition at an object usage level using a wrist worn RFID reader [14]. Finally, hierarchical hidden Markov models were used to perform activity recognition from video [5].

In previous work we showed how the use of temporal probabilistic models increases the performance in activity recognition. However, results strongly suffered from the limited accuracy of the annotation in the dataset used [19]. Since the annotation is used for training the models, it is very important to get as accurate training examples as possible. In this work we will show how accurate annotation results in very good performance results using temporal probabilistic models for activity recognition.

## SYSTEM

Our system was built upon two main lines: ease of installation and minimal intrusion. Furthermore, we coupled it with an efficient, accurate and inexpensive annotation method.

### Sensors

Our sensor network consists of wireless network nodes to which simple off-the-shelf sensors can be attached. After considering different commercially available wireless network kits, we selected the RFM DM 1810 (fig. 1). The RFM kit comes with a very rich and well documented API and the standard firmware includes an energy efficient network protocol. The kit comes with a base station which is attached to a PC using USB. A new sensor node can be easily added to the network by a simple pairing procedure, which involves pressing a button on both the base station and the new node.

**Figure 1. Wireless sensor node used: RFM DM 1810.**

The RFM wireless network node has an analog and digital input. It sends an event when the state of the digital input changes or when some threshold of the analog input is violated. This allows the nodes to be used with a large variety of sensors, ranging from simple contact switches to temperature or humidity sensors. Special low-energy consuming radio technology, together with an energy saving sleeping mode result in a long battery life. The node can reach a data transmission rate of 4.8 kb/s, which is enough for the binary sensor data that we need to collect.

Nodes are easily installed using some tape and do not require any drilling. Because they run on batteries they can be left running for several months and there is no need to connect it to the powernet. Furthermore, because sensors are mounted on walls or inside cupboards, as much out of sight as possible, the intrusion of the sensor system is minimal.

### Annotation

Annotation was performed using a bluetooth headset combined with speech recognition software. The starting and end point of an activity were annotated out of a predefined set of commands.

We used the Jabra BT250v bluetooth headset (fig. 2) for annotation. It has a range up to 10 meters and battery power for 300 hours standby or 10 hours active talking. This is more than enough for a full day of annotation, the headset was recharged during sleep. The headset contains a button which we used to trigger the software to add a new annotation entry.

**Figure 2. Jabra BT250v bluetooth headset.**

The software for storing the annotation was custom made by our research team. It was written in C and combines elements of the bluetooth API with the Microsoft Speech API[1]. The bluetooth API was needed to catch the event of the headset button being pressed and should work with any bluetooth dongle and headset that uses the Widcomm[2] bluetooth stack.

The Microsoft Speech API provided an easy way to use both speech recognition and text to speech. When the headset button is pressed the speech recognition engine starts listening for commands it can recognize. We created our own speech grammar, which contains possible combinations of commands the recognition engine could expect. By using very distinctive commands such as 'begin use toilet' and 'begin take shower', the recognition engine had multiple words by which it could distinguish different commands. This resulted in near perfect recognition results during annotation. The recognized sentence is outputted using the text-to-speech engine. Any errors that do occur can be immediately corrected using a 'correct last' command.

[1]For details about the Microsoft Speech API see: http://www.microsoft.com/speech/

[2]For details about the Widcomm stack see: http://www.broadcom.com/products/bluetooth_sdk.php

Because annotation is provided by the user on the spot this method is very efficient and accurate. The use of a headset together with speech recognition results in very little interference while performing activities. This results in annotation data which requires very little post processing, making this a very inexpensive method.

Our custom made annotation software is available for download from: `http://www.science.uva.nl/~tlmkaste`

## ANNOTATED DATA SET

Using our system, we recorded a dataset in the house of a 26-year-old man. He lives alone in a three-room apartment where 14 state-change sensors were installed. We only used digital sensors for this dataset, although the use of analog sensors is possible. Locations of sensors include doors, cupboards, refrigerator and a toilet flush sensor (see fig. 3). Sensors were left unattended, collecting data for 28 days in the apartment. This resulted in 2120 sensor events and 245 activity instances.

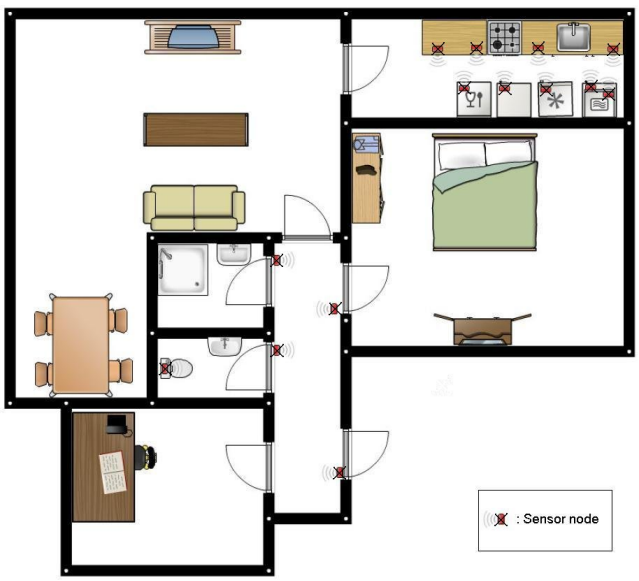


**Figure 3. Floorplan of the house, red rectangle boxes indicate sensor nodes. (created using: http://www.floorplanner.com/)**

Activities were annotated by the subject himself using a bluetooth headset as described above. Seven different activities were annotated, namely: 'Leave house', 'Toileting', 'Showering', 'Sleeping', 'Preparing breakfast','Preparing dinner' and 'Preparing a beverage'. These activities were chosen based on the Katz ADL index, a commonly used tool in healthcare to assess cognitive and physical capabilities of an elderly person [10]. Times at which no activity is annotated is referred to as 'Idle'. Table 1 shows the number of separate instances of activities and the percentage of time each activity takes up in the data set. This table clearly shows how some activities occur very frequently (e.g. toileting), while others that occur less frequently have a longer duration and therefore take up more time (e.g. leaving and sleeping).

The dataset together with its annotations is available for download from: `http://www.science.uva.nl/~tlmkaste`

| | Number of instances | Percentage of time |
|---|---|---|
| Idle | - | 11.5% |
| Leaving | 34 | 56.4% |
| Toileting | 114 | 1.0% |
| Showering | 23 | 0.7% |
| Sleeping | 24 | 29.0% |
| Breakfast | 20 | 0.3% |
| Dinner | 10 | 0.9% |
| Drink | 20 | 0.2% |

**Table 1. Number of instances and percentage of time activities occur in the dataset.**

## TEMPORAL PROBABILISTIC MODEL

Our objective is to recognize activities from sensor readings in a house. This is a formidable task, since the label in question (activity performed) cannot be estimated directly from our data (sensor readings). Furthermore, the temporal patterns of our data contain valuable information regarding the performed activity. A suitable framework for this task is temporal probabilistic models.

In order to work with these models, we divide our time series data in time slices of constant length and label the activity for each slice. We denote the duration of a time slice with $\Delta t$. We denote a sensor reading for time $t$ as $x_t^i$, indicating whether sensor $i$ fired at least once between time $t$ and time $t + \Delta t$, with $x_t^i \in \{0, 1\}$. In a house with $N$ sensors installed, we define a binary observation vector $\vec{x}_t = (x_t^1, x_t^2, \ldots, x_t^N)^T$. The activity at time slice $t$ is denoted with $y_t$ and so formally our task is to find a mapping between a sequence of observations $\mathbf{x} = \{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_T\}$ and a sequence of labels $\mathbf{y} = \{y_1, y_2, \ldots, y_T\}$ for a total of $T$ time steps (fig. 4).

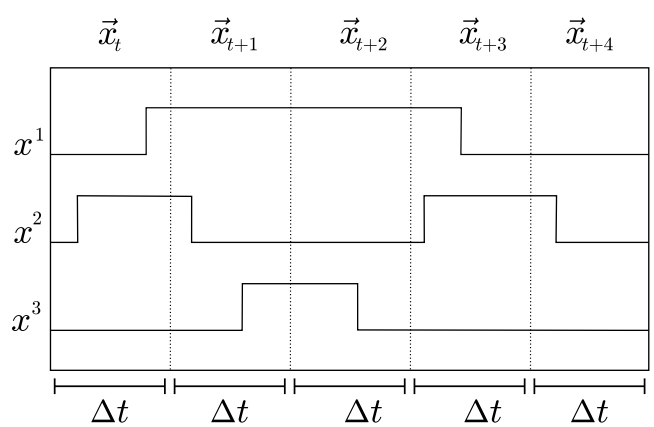


**Figure 4. Showing the relation between sensor readings $x^i$ and time intervals $\Delta t$.**

We utilize the framework of probabilistic models to capture this mapping, which requires us to do two things: First, we have to learn the parameters of our probabilistic model from training data. Second, we have to infer the labels for

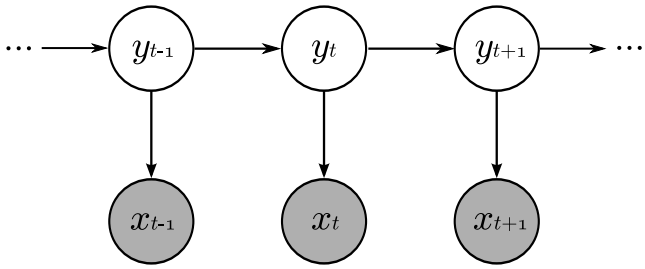


**Figure 5. The graphical representation of a HMM. The shaded nodes represent observable variables, while the white nodes represent hidden ones.**

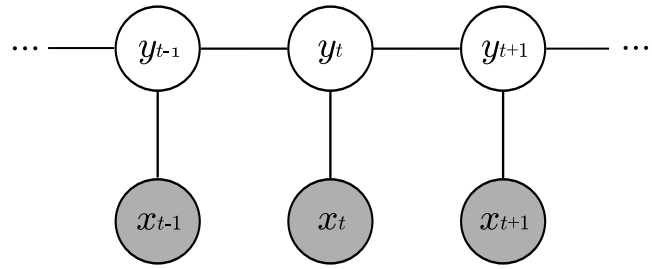


**Figure 6. The graphical representation of a linear-chain CRF. The shaded nodes represent observable variables, while the white nodes represent hidden ones.**

novel observation sequences. In this section we will present how these two steps work for both hidden Markov models (HMMs) and conditional random fields (CRFs).

### Hidden Markov Model

The Hidden Markov Model (HMM) is a generative probabilistic model consisting of a hidden variable and an observable variable at each time step (fig. 5). In our case the hidden variable is the activity performed, and the observable variable is the vector of sensor readings. There are two dependency assumptions that define this model, represented with the directed arrows in the figure.

- The hidden variable at time $t$, namely $y_t$, depends only on the previous hidden variable $y_{t-1}$ (*Markov assumption* [15]).
- The observable variable at time $t$, namely $x_t$, depends only on the hidden variable $y_t$ at that time slice.

With these assumptions we can specify an HMM using three probability distributions: the distribution over initial states $p(y_1)$; the transition distribution $p\left(y_t|y_{t-1}\right)$ representing the probability of going from one state to the next; and the observation distribution $p\left(x_t|y_t\right)$ indicating the probability that the state $y_t$ would generate observation $x_t$.

Learning the parameters of these distributions corresponds to maximizing the joint probability $p(\mathbf{x}, \mathbf{y})$ of the paired observation and label sequences in the training data. We can factorize the joint distribution in terms of the three distributions described above as follows [17]:

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

in which we write the distribution over initial states $p(y_1)$ as $p(y_1 \mid y_0)$, to simplify notation.

The parameters that maximize this joint probability are found by frequency counting. Because in our case we are dealing with discrete data, we can simply count the number of occurrences of transitions, observations and states [15].

Inferring which sequence of labels best explains a new sequence of observations can be performed efficiently using the Viterbi algorithm. This dynamic programming technique is commonly used in combination with Hidden Markov models [15].

### Conditional Random Fields

A Conditional Random Field (CRF) is a discriminative probabilistic model that can come in many different forms. The form that most closely resembles the HMM is known as a linear-chain CRF and is the model we use in this paper (fig. 6). As the figure shows, the model still consists of a hidden variable and an observable variable at each time step. However, the arrowheads of the edges between the various nodes have disappeared, making this an undirected graphical model. This means that two connected nodes no longer represent a conditional distribution (e.g. a given b), but instead we speak of the potential between two connected nodes. Unlike a probability, potentials are not restricted to a value between 0 and 1 [4].

The potential functions that specify the linear-chain CRF are $\psi(y_t, y_{t-1})$ and $\psi(y_t, x_t)$. In this work, we adopt the notation [17, 4] which allows different forms of CRFs to be expressed using a common formula. Therefore we define: $\psi(y_t = i, y_{t-1} = j) = \lambda_{ijk} f_{ijk}(y_t, y_{t-1}, x_t)$ in which the $\lambda_{ijk}$ is the parameter value (the actual potential) and $f_{ijk}(y_t, y_{t-1}, x_t)$ is a feature function that in the simplest case returns 1 when $y_t = i$ and $y_{t-1} = j$, and 0 otherwise. The second potential function is defined similarly: $\psi(y_t = i, x_t = k) = \lambda_{ijk} f_{ijk}(y_t, y_{t-1}, x_t)$, where $\lambda_{ijk}$ is the parameter value and the feature function now returns 1 when $y_t = i$ and $x_t = k$, and 0 otherwise. The index $ijk$ is typically replaced by a one-dimensional index, so we can easily represent the summation over all the different potential functions [17].

In comparison with HMM, the conditional probabilities $p\left(x_t|y_t\right)$ and $p\left(y_t|y_{t-1}\right)$ have been replaced by the corresponding potentials. The essential difference lies in the way we learn the model parameters. In the case of HMMs the parameters are learned by maximizing the *joint* probability distribution $p(\mathbf{x}, \mathbf{y})$. The parameters of a CRF are learned by maximizing the *conditional* probability distribution $p(\mathbf{y} \mid \mathbf{x})$ which belongs to the family of exponential distributions as [17]:

$$p\left(\mathbf{y}|\mathbf{x}\right) = \frac{1}{Z(x)} \exp\left\{\sum_{k=1}^{K} \lambda_k f_k(y_t, y_{t-1}, x_t)\right\} \tag{2}$$

where $Z(x)$ is the normalization function

$$Z(x) = \sum_{\mathbf{y}} \exp\left\{\sum_{k=1}^{K} \lambda_k f_k(y_t, y_{t-1}, x_t)\right\} \quad (3)$$

notice that the formula sums over all the different potential functions. The feature function $f_k(y_t, y_{t-1}, x_t)$ will return a 0 or 1 depending on the values of the input variables and therefore determines whether a potential should be included in the calculation. Finally, the sum of potentials is divided by the normalization term $Z(x)$ which guarantees that the outcome is a probability. One of the main consequences of this choice, is that while learning the parameters of a CRF we avoid modeling the distribution of the observations, $p(x)$. As a result, we can only use CRFs to perform inference (and not to generate data), which is a characteristic of the discriminative models. In activity recognition, the only thing we are interested in is classification and therefore CRFs fit our purpose perfectly. Model parameters can be learned using an iterative gradient method. Particularly successful have been quasi-Newton methods such as BFGS, because they take into account the curvature of the likelihood function. Inference can be performed using a slightly altered version of the viterbi algorithm [17].

## EXPERIMENTS

In this section we present the experimental results acquired in this work. We start describing our experimental setup, and then describe the objective of our experiments. This section concludes with a presentation of the acquired results.

### Setup

In our experiments the sensor readings are divided in data segments of length $\Delta t = 60$ seconds. This time slice duration is long enough to be discriminative and short enough to provide high accuracy labeling results. Unless stated otherwise, we separate our data into a test and training set using a 'leave one day out' approach. In this approach, one full day of sensor readings is used for testing and the remaining days are used for training. In this way, we get inferred labels for the whole dataset by concatenating the results acquired for each day.

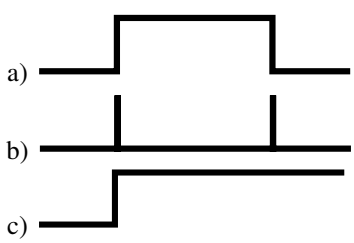


**Figure 7. Example of sensor firing showing the a) *raw*, b) *change point* and c) *last* observation representation.**

Our *raw* sensor representation gives a 1 when the sensor is firing and a 0 otherwise (fig. 7a). Next to this raw data as observations we experiment with a *change point* representation. In this representation, the sensor gives a 1 to timeslices where the sensor reading changes (fig. 7b). Finally, we experiment with a *last* sensor fired representation, in which the last sensor that changed state continues to give 1 and changes to 0 when a different sensor changes state (fig. 7c).

Because our dataset contains some classes that appear much more frequent than other classes, classes are considered to be imbalanced [9]. We therefore evaluate the performance of our models by two measures, the time slice accuracy and the class accuracy. The timeslice accuracy represents the percentage of correctly classified timeslices, while the class accuracy represents the average percentage of correctly classified timeslices per class. The measures are calculated as follows:

**Time slice:** $\frac{\sum_{n=1}^{N}[inferred(n)=true(n)]}{N}$

**Class:** $\frac{1}{C}\sum_{c=1}^{C}\left\{\frac{\sum_{n=1}^{N_c}[inferred_c(n)=true_c(n)]}{N_c}\right\}$

in which $[a = b]$ is a binary indicator giving 1 when true and 0 when false. $N$ is the total number of time slices, $C$ is the number of classes and $N_c$ the total number of time slices for class $c$.

Measuring the time-slice accuracy is a typical way of evaluating time-series analysis. However, we also report the class average accuracy, which is a common technique in datasets with a dominant class. In these cases classifying all the test data as the dominant class yields good time-slice accuracy, but no useful output. The class average though would remain low, and therefore be representative of the actual model performance.

### Objective

Using the models and the recorded dataset discussed in the previous sections, we ran three experiments.

The first experiment compares the accuracy of all the possible model-representation combinations. We test the HMM or CRF model coupled with one of the *raw*, *change point*, *last* and a concatenation of *change point and last* data representation.

The second experiment explores the relationship of the size of the available training data with the performance of the acquired parameters. This is a good indication of the size of training data our framework would need, when installed on a novel real-world situation.

The third experiment shows the difference in performance in using offline inference and online inference. Offline inference means that all the data is available a priori for our calculations. In processing each time slice, this allows us to use sensor data from both before and after the point of inference. Online inference, on the other hand, implies we can only incorporate data collected up to the point of inference. Online inference is significantly harder, however it is necessary for specific applications.

### Experiment 1: Model comparison

In this experiment we compare which sensor representation combined with which model performs best on our recorded

| | Idle | Leaving | Toileting | Showering | Sleeping | Breakfast | Dinner | Drink |
|---|---|---|---|---|---|---|---|---|
| Idle | 60.9 | 6.4 | 0.7 | 9.8 | 4.6 | 0.4 | 16.3 | 0.9 |
| Leaving | 0.8 | 98.4 | 0.2 | 0.3 | 0.0 | 0.1 | 0.2 | 0.0 |
| Toileting | 6.3 | 2.9 | 83.9 | 3.2 | 2.6 | 0.3 | 0.3 | 0.5 |
| Showering | 7.2 | 0.0 | 3.8 | 89.1 | 0.0 | 0.0 | 0.0 | 0.0 |
| Sleeping | 0.1 | 0.0 | 0.4 | 0.1 | 99.4 | 0.0 | 0.0 | 0.0 |
| Breakfast | 22.0 | 0.0 | 0.9 | 0.0 | 0.9 | 52.3 | 16.5 | 7.3 |
| Dinner | 12.4 | 0.3 | 0.3 | 0.0 | 0.0 | 6.0 | 78.4 | 2.6 |
| Drink | 11.9 | 1.7 | 1.7 | 0.0 | 0.0 | 3.4 | 13.6 | 67.8 |

**Table 2. Confusion Matrix for HMM using changepoint+last representation and *offline* inference. The values are percentages.**

dataset. Experiments were performed using offline inference.

Table 3 shows the accuracies for all model-representation combinations. We see that for both models the changepoint+last representation gives the best results. Furthermore, we see that the CRF achieves a higher timeslice accuracy for all the sensor representations. The HMM achieves the overall highest class accuracy using the changepoint+last representation.

| | | Timeslice | Class |
|---|---|---|---|
| HMM | raw | 51.5% | 49.2% |
| | changepoint | 80.0% | 67.2% |
| | last | 91.8% | 71.2% |
| | changepoint+last | **94.5%** | **79.4%** |
| CRF | raw | 69.4% | 44.6% |
| | changepoint | 89.4% | 61.7% |
| | last | 95.1% | 63.2% |
| | changepoint+last | **95.6%** | **70.8%** |

**Table 3. Timeslice and class accuracies for HMM and CRF using various sensor representations.**

The confusion matrix for HMM using the changepoint + last representation can be found in table 2. The activities 'Leaving', 'Toileting', 'Showering' and 'Sleeping' give the highest accuracy. Activities 'Idle' and 'Breakfast' perform worst. Most confusion takes place in the 'Idle' activity and the three kitchen activities 'Breakfast', 'Dinner' and 'Drink.

The confusion matrix for CRF using the changepoint + last representation can be found in table 4. The activities 'Idle', 'Leaving' and 'Sleeping' give the highest accuracy. Activities 'Breakfast', 'Dinner' and 'Drink' perform worst. Most confusion takes place in the 'Idle' activity and the three kitchen activities 'Breakfast', 'Dinner' and 'Drink.

### Experiment 2: Minimum amount of training data

In this experiment we show how the number of training days affects the accuracy of the classifier. We use one day for testing, the previous $n$ days for training and cycle over all the days in the dataset. If the previous $n$ days are not available, we continue from the back of the dataset.

| | Idle | Leaving | Toileting | Showering | Sleeping | Breakfast | Dinner | Drink |
|---|---|---|---|---|---|---|---|---|
| Idle | 80.4 | 12.3 | 0.4 | 0.4 | 4.3 | 0.0 | 2.0 | 0.2 |
| Leaving | 1.1 | 98.4 | 0.2 | 0.1 | 0.1 | 0.0 | 0.1 | 0.0 |
| Toileting | 7.6 | 12.4 | 69.5 | 1.8 | 8.7 | 0.0 | 0.0 | 0.0 |
| Showering | 23.0 | 8.7 | 2.3 | 66.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| Sleeping | 0.1 | 0.0 | 0.2 | 0.0 | 99.7 | 0.0 | 0.0 | 0.0 |
| Breakfast | 32.1 | 0.0 | 0.9 | 0.0 | 0.9 | 55.0 | 9.2 | 1.8 |
| Dinner | 33.0 | 7.2 | 0.6 | 0.0 | 0.6 | 4.3 | 53.4 | 0.9 |
| Drink | 32.2 | 5.1 | 3.4 | 0.0 | 0.0 | 5.1 | 10.2 | 44.1 |

**Table 4. Confusion Matrix for CRF using changepoint+last representation and *offline* inference. The values are percentages.**

Figure 8 shows the timeslice and class accuracy for both HMM and CRF as function of the number of days used for training. The figure shows that CRFs continuously outperform HMMs in terms of timeslice accuracy, while HMMs continuously outperform CRFs in terms of class accuracy. The timeslice accuracy shows a horizontal trend for 3 training days or higher, the class accuracy shows a horizontal trend for 12 training days or higher.

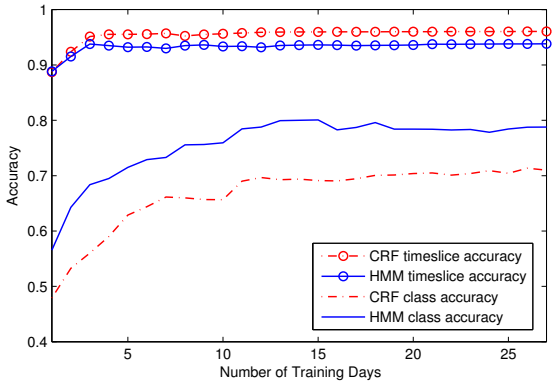


**Figure 8. Timeslice and class accuracy as a function of the number of days used for training.**

### Experiment 3: Offline vs online inference

In this experiment we compare offline and online inference, we used the changepoint + last representation for both models and inference methods.

Table 6 shows the accuracies for both HMMs and CRFs using offline and online inference. We see that for both models offline inference performance better, the difference between offline and online inference is bigger for CRFs than for HMMs.

The confusion matrix for HMM using offline inference can be found in table 2 and online inference in table 5. Comparing the online and offline confusion matrices we see the accuracy of activities 'Toileting', 'Showering', 'Dinner' and 'Drink' are lower for online, while the activity 'Breakfast' is slightly higher.

| | Idle | Leaving | Toileting | Showering | Sleeping | Breakfast | Dinner | Drink |
|---|---|---|---|---|---|---|---|---|
| Idle | 61.5 | 6.4 | 0.8 | 9.8 | 4.6 | 0.6 | 15.4 | 0.8 |
| Leaving | 0.9 | 98.3 | 0.2 | 0.4 | 0.0 | 0.1 | 0.2 | 0.0 |
| Toileting | 12.1 | 4.5 | 72.6 | 5.3 | 3.9 | 0.5 | 0.8 | 0.3 |
| Showering | 9.8 | 0.8 | 5.3 | 84.2 | 0.0 | 0.0 | 0.0 | 0.0 |
| Sleeping | 0.1 | 0.0 | 0.3 | 0.2 | 99.4 | 0.0 | 0.0 | 0.0 |
| Breakfast | 25.7 | 0.0 | 0.9 | 0.0 | 0.9 | 56.0 | 11.9 | 4.6 |
| Dinner | 19.8 | 1.4 | 0.3 | 0.0 | 0.0 | 9.2 | 67.8 | 1.4 |
| Drink | 15.3 | 5.1 | 3.4 | 1.7 | 0.0 | 6.8 | 10.2 | 57.6 |

**Table 5. Confusion Matrix for HMM using changepoint+last representation and *online* inference. The values are percentages.**

| | Idle | Leaving | Toileting | Showering | Sleeping | Breakfast | Dinner | Drink |
|---|---|---|---|---|---|---|---|---|
| Idle | 68.2 | 12.1 | 1.4 | 2.4 | 4.5 | 1.2 | 9.6 | 0.7 |
| Leaving | 8.6 | 90.7 | 0.2 | 0.2 | 0.1 | 0.0 | 0.1 | 0.0 |
| Toileting | 11.8 | 13.2 | 54.5 | 4.7 | 15.5 | 0.3 | 0.0 | 0.0 |
| Showering | 48.7 | 5.3 | 7.9 | 37.7 | 0.0 | 0.0 | 0.4 | 0.0 |
| Sleeping | 3.7 | 0.3 | 0.5 | 0.1 | 95.4 | 0.0 | 0.0 | 0.0 |
| Breakfast | 15.6 | 0.0 | 6.4 | 0.0 | 6.4 | 56.9 | 12.8 | 1.8 |
| Dinner | 25.0 | 25.0 | 1.4 | 0.0 | 1.1 | 2.9 | 44.0 | 0.6 |
| Drink | 11.9 | 28.8 | 5.1 | 0.0 | 0.0 | 0.0 | 13.6 | 40.7 |

**Table 7. Confusion Matrix for CRF using changepoint+last representation and *online* inference. The values are percentages.**

The confusion matrix for CRF using offline inference can be found in table 4 and online inference in table 7. Comparing the online and offline confusion matrices we see the accuracy of all activities except for 'Breakfast' are lower, the accuracy for 'Breakfast' is slightly higher.

Comparing the two confusion matrices we see that the 'Idle' activity has significantly higher accuracy with CRFs. While 'Toileting', 'Showering', 'Dinner' and 'Drink' have significantly higher accuracies with HMMs.

| | | Timeslice | Class |
|---|---|---|---|
| HMM | online | 94.4% | 73.3% |
| | offline | 94.5% | 79.4% |
| CRF | online | 88.2% | 61.0% |
| | offline | 95.6% | 70.8% |

**Table 6. Timeslice and class accuracies for HMM and CRF using offline and online inference.**

## Discussion

The first experiment shows that our proposed framework gives state of the art results in recognizing activities in a home setting. Out of the different sensor data representations the 'raw' representation gave the worst results. This was expected, because in the dataset many sensors continue to fire because a door is left open, while the activity involved already ended. The 'changepoint' representation solves this issue by only representing the point in time at which a sensor was used. Such a representation allows for a much better discrimination and therefore leads to much better results. The 'last' representation, in which the last used sensor continues to fire until another sensor is used, gave even better results. This representation is especially useful for door sensors. When somebody is inside a room and closes the door, the only sensors that can fire are within that room. Since people tend to typically close doors behind them for a number of activities (e.g. sleeping, showering and leaving) this sensor representation works surprisingly well. By using both the 'changepoint' and 'last' representation in a concatenated feature matrix, we were able to use best of both worlds and achieved the highest accuracy for both HMMs and CRFs.

With respect to comparing the two probabilisitic models we see that CRFs outperform HMMs in all the cases with respect to timeslice accuracy, but HMMs achieve the overall highest class accuracy. This is a result of the way both models maximize their parameters. HMMs make use of a Bayesian framework in which a separate model $p(x \mid y)$ is learned for each class. Using Bayes rule we can calculate the posterior probability $p(y \mid x)$ for a novel point. In the case of CRFs we use a single model for all classes, by calculating $p(y \mid x)$ directly. Parameters are learned by maximizing the conditional likelihood $p(\mathbf{y} \mid \mathbf{x})$. Because CRFs use a single model for all classes, classes compete during maximization. In a dataset with a dominant class, modelling everything as the dominant class might yield a higher likelihood than including the minor classes and misclassifying parts of the dominant class. This view is confirmed by the two confusion matrices for HMM (table 2) and CRF (table 4). The accuracy of the 'Idle' activity is much higher in the case of CRFs than HMMs, but the accuracy of 'Toileting', 'Showering', 'Dinner' and 'Drink' is much higher in the case of HMMs. The 'Idle' activity takes up $11.5\%$ of the dataset, while the other activities each take up $1\%$ at most.

The second experiment shows that for this dataset and these models a minimum of 12 days is needed for accurate parameter estimation. The timeslice accuracy seems to indicate that after 3 training days the accuracy already converged. However, this is because some classes appear very rarely in the dataset. Using too few training days, leads to bad parameter estimations for these classes. The increase in timeslice accuracy is so little, because they only take up a few timeslices.

The third experiment shows that HMMs still give good results when using online inference. This is important, because when using offline inference, activities could only be inferred when a full day has passed. Using online inference, we can infer activities the minute after they happen. Some applications, such as sounding an alarm after an anomaly detection need to be able to respond this quickly.

We end this discussion with a note that the dataset described in this paper only uses digital state change sensors, however, our framework also allows the use of analog sensors. First, the network nodes we use also have an analog input. Second,

by setting a threshold value, the nodes only send out an event when the threshold is violated. Since most activity related things we wish to sense have an on and off state (e.g. shower, stove), we could easily measure their status using an analog sensor (e.g. humidity, temperature). This would result in binary data, just like with the use of digital sensors. Third, even though the sensors might need some time to pick up the change in state (e.g. room slowly becomes more humid), the probabilistic nature of our models is ideal for modelling such correlations.

## CONCLUSIONS

This paper introduces a sensor and annotation system for performing activity recognition in a house setting. We experimented with probabilistic models that can utilize the data collected from this system to learn the parameters of activities in order to detect them in future sensor readings. A number of sensor reading representations were proposed and their effectiveness in activity recognition was compared. We strongly believe that other researchers can use these representations and select the optimal one on their field. Furthermore, our recorded dataset is available on the author's website to motivate the comparison of other models with the ones we proposed. Additionally, we proposed two measures for model accuracy, which we believe capture the potential value of a model in the field of activity recognition. As long as it concerns our choice of models, we performed a series of experiments that reveal the potential of discriminative and generative modeling in activity recognition.

### Acknowledgements

This work is part of the Context Awareness in Residence for Elders (CARE) project. The CARE project is partly funded by the Centre for Intelligent Observation Systems (CIOS) which is a collaboration between UvA and TNO, and partly by the EU Integrated Project COGNIRON (The Cognitive Robot Companion). Athanasios Noulas was funded by MultimediaN.

## REFERENCES

1. Living independently - quietcare system. www.livingindependently.com.
2. G. Abowd, A. Bobick, I. Essa, E. Mynatt, and W. Rogers. The aware home: Developing technologies for successful aging. In *Proceedings of AAAI Workshop and Automation as a Care Giver*, 2002.
3. Gregory D. Abowd, Anind K. Dey, Peter J. Brown, Nigel Davies, Mark Smith, and Pete Steggles. Towards a better understanding of context and context-awareness. In *HUC '99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, pages 304–307, London, UK, 1999. Springer-Verlag.
4. Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, August 2006.
5. Thi V. Duong, Hung H. Bui, Dinh Q. Phung, and Svetha Venkatesh. Activity recognition and abnormality detection with the switching hidden semi-markov model. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1*, pages 838–845, Washington, DC, USA, 2005. IEEE Computer Society.
6. James Fogarty, Carolyn Au, and Scott E. Hudson. Sensing from the basement: a feasibility study of unobtrusive and low-cost home activity recognition. In *UIST '06: Proceedings of the 19th annual ACM symposium on User interface software and technology*, pages 91–100, New York, NY, USA, 2006. ACM Press.
7. Sumi Helal, William Mann, Hicham El-Zabadani, Jeffrey King, Youssef Kaddoura, and Erwin Jansen. The gator tech smart house: A programmable pervasive space. *Computer*, 38(3):50–60, 2005.
8. Stephen S. Intille, Emmanuel Munguia Tapia, John Rondoni, Jennifer Beaudin, Chuck Kukla, Sitij Agarwal, Ling Bao, and Kent Larson. Tools for studying behavior and technology in natural settings. In *Ubicomp*, pages 157–174, 2003.
9. Nathalie Japkowicz and Shaju Stephen. The class imbalance problem: A systematic study. *Intell. Data Anal*, 6(5):429–449, 2002.
10. S. Katz, T.D. Down, H.R. Cash, and et al. Progress in the development of the index of adl. *Gerontologist*, 10:20–30, 1970.
11. Niels Landwehr, Bernd Gutmann, Ingo Thon, Matthai Philipose, and Luc De Raedt. Relational transformation-based tagging for human activity recognition. In Donato Malerba, Annalisa Appice, and Michelangelo Ceci, editors, *Proceedings of the 6th International Workshop on Multi-relational Data Mining (MRDM07)*, pages 81–92, Warsaw, Poland, September 2007.
12. Jonathan Lester, Tanzeem Choudhury, Nicky Kern, Gaetano Borriello, and Blake Hannaford. A hybrid discriminative/generative approach for modeling human activities. In *IJCAI*, pages 766–772, 2005.
13. Beth Logan, Jennifer Healey, Matthai Philipose, Emmanuel Munguia Tapia, and Stephen S. Intille. A long-term evaluation of sensing modalities for activity recognition. In *Ubicomp*, pages 483–500, 2007.
14. Donald J. Patterson, Dieter Fox, Henry A. Kautz, and Matthai Philipose. Fine-grained activity recognition by aggregating abstract object usage. In *ISWC*, pages 44–51. IEEE Computer Society, 2005.
15. L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.
16. Albrecht Schmidt. *Ubiquitous Computing - Computing in Context*. PhD thesis, Lancaster University, 2002.

17. Charles Sutton and Andrew McCallum. *Introduction to Statistical Relational Learning*, chapter 1: An introduction to Conditional Random Fields for Relational Learning, page (Available online). MIT Press, 2006.

18. Emmanuel Munguia Tapia, Stephen S. Intille, and Kent Larson. Activity recognition in the home using simple and ubiquitous sensors. In *Pervasive Computing, Second International Conference, PERVASIVE 2004*, pages 158–175, Vienna, Austria, April 2004.

19. T. L. M. van Kasteren and B. J. A. Kröse. Bayesian activity recognition in residence for elders. In *Intelligent Environments, 2007. IE 07. 3rd IET International Conference*, pages 209–212, Ulm, Germany, 2007.

20. Daniel Wilson and Chris Atkeson. Simultaneous tracking and activity recognition (star) using many anonymous binary sensors. In *Pervasive Computing, Third International Conference, PERVASIVE 2005*, pages 62–79, Munich , Germany, 2005.

21. Daniel H. Wilson. *Assistive Intelligent Environments for Automatic Health Monitoring*. PhD thesis, Carnegie Mellon University, 2005.