Activity discovery using Dirichlet multinomial mixture models from discrete sensor data in smart homes

Ken Sadohara

Received: 10 October 2021 / Accepted: 30 May 2022 / Published online: 19 July 2022
© The Author(s) 2022

Abstract
To develop smart home technology designed to analyze the activity of residents based on the logs of installed sensors, an activity model tailored to individuals must be constructed from less privacy-invasive sensors to avoid interference in daily life. Unsupervised machine learning techniques are desirable to automatically construct such models without costly data annotation, but their application has not yet been sufficiently successful. In this study, we show that an activity model can be effectively estimated without activity labels via the Dirichlet multinomial mixture (DMM) model. The DMM model assumes that sensor signals are generated according to a Dirichlet multinomial distribution conditioned on a single unobservable activity and can capture the burstiness of sensors, in which even sensors that rarely fire may fire repeatedly after being triggered. We demonstrate the burstiness phenomenon in real data using passive infrared ray motion sensors. For such data, the assumptions of the DMM model are more suitable than the assumptions employed in models used in previous studies. Moreover, we extend the DMM model so that each activity depends on the preceding activity to capture the Markov dependency of activities, and a Gibbs sampler used in the model estimation algorithm is also presented. An empirical study using publicly available data collected in real-life settings shows that the DMM models can discover activities more correctly than the other models and expected to be used as a primitive activity extraction tool in activity analysis.

Keywords Activity discovery · Unsupervised learning · Topic models · Smart home · Power law · Burstiness

1 Introduction
Data mining to identify behavioral patterns of residents of smart homes has attracted considerable attention [1–3]. For the elderly in particular, such an approach is considered to be effective for assisting in their health care and contributing to their independence [2, 4, 5]. In this study, we propose a modeling method for behavioral patterns of aged persons living alone to monitor their activities of daily living (ADLs).

To develop the behavior model, ADL classifiers based on hand-crafted rules [6] can be used. While specification-based methods can be used to easily describe high-level activities, Describing low-level activities based on sensor signals is difficult. In contrast, learning-based methods have been used to build models from sensor data [1–3].

Most learning-based methods employed to date use sensor data annotated with activity labels to construct a classifier of activities through supervised learning, which is known as activity recognition. Because the cost of an annotation is generally high, preparing annotated data for each resident is cost-prohibitive, and the cost is warranted only if a model trained for a particular resident is applicable to other residents. However, sharing activity models between residents is difficult, particularly when only unobtrusive binary sensors, e.g., passive infrared ray (PIR) motion sensors, are allowed [6, 7]. More informative sensors such as video, audio, or wearable sensors are unacceptable in monitoring activities conducted over extended periods owing to intrusiveness, and potential to interfere with daily life [2, 5]. The identification of activities with motion sensors depends not only on detecting motion but also on the layout of the sensors. The location where a sensor is installed can be a strong indicator for certain activities (e.g., using the bathroom), or a sequence of reactions of adjacent sensors can identify other activities (e.g., moving toward an entrance). Because the living environment and the sensor layout of
each resident is unique, sharing an activity model is difficult, and the model should therefore be individually tailored to each resident. To keep the modeling cost within reasonable bounds, unsupervised learning techniques have been identified as a promising approach [3], which is known as activity discovery [4, 8].

Among unsupervised learning techniques for activity discovery, this study focuses on generative models [9] of sensor signals. These are probabilistic models describing sensor signals generated from unobservable activities. After fitting a model to a sequence of sensor signals, trained model can be used to estimate an activity that is most likely to have generated a new set of sensor signals. In previous research, topic models such as latent Dirichlet allocation (LDA) models [10] and hierarchical Dirichlet processes mixture (DPM) models [11] have been applied to activity discovery [12–15]. In this study, we investigate the use of Dirichlet multinomial mixture (DMM) models [16, 17].

The DMM model, which is a uni-topic version of the LDA model, assumes that sensor signals are generated from a single activity that occurs among unobservable activities. Because DMM models have been successfully applied to the modeling for short documents [17], which tend to concern a single topic, they can also be effective for modeling sensor signals within a short time period, which are reasonably hypothesized as being generated from a single activity rather than from multiple activities.

Another advantage of the DMM model is that it can capture burstiness in sensor data: once a sensor fires, there is a high probability that it will continue to fire, even if it fires only rarely. The burstiness stems from the fact that the distribution of sensor signals follows the power law, which is also observed in a wide variety of natural and social phenomena [18]. As discussed in a subsequent section, burstiness is indeed observed in motion sensor data, and the DMM model can fit a model to such data more accurately and robustly than alternative methods.

However, applying the DMM model to activity discovery involves the drawback that it cannot model the Markov dependency of the activities. Because the sensor data are time-series data, a particular activity naturally depends on previous activities. To cope with this dependency, hidden Markov models (HMMs) have been extensively used to perform activity recognition [19, 20]. As with HMMs, the DMM models can be extended to take into account the Markov dependency on activities by applying their bigram statistics. To fit the extended DMM model to the data, a Gibbs sampler used in the Markov chain Monte Carlo (MCMC) method [9] is also presented.

We evaluate the applicability of the DMM model to activity discovery by an empirical study using publicly available datasets collected by experiments in real-life settings, one of which used reed switch sensors and does not exhibit burstiness, whereas the other used PIR motion sensors and does exhibit burstiness. Activity discovery is typically used in human-in-the-loop analysis with the help of visualization tools [21]. Activity patterns to be discovered have various levels of granularity, ranging from gestures, ambulation, and ADL to social interaction [1]. An evaluation that takes all of these into account is beyond the scope of this study. Rather, we focus on evaluating the performance of clustering primitive patterns in sequences of binary sensor signals under as few assumptions as possible, because the basic clustering performance is an important first step in a complex activity analysis. Also, clustering can be directly used to evaluate the quantity and quality of primitive activities, such as the restlessness behavior of patients with Alzheimer’s disease [22]. To this end, we evaluated the performance of clustering activities corresponding to short time periods using various generative models for sensor signals as well as the k-means method, a typical clustering method, and frequent sequence mining [8] previously used in activity discovery other than generative models.

Reminder of this work is organized as follows. After briefly reviewing previously studied activity discovery techniques in Section 2 and particularly the use of probabilistic generative models in Section 3, the DMM model and its extension are presented together with Gibbs samplers in Section 4. Then, in Section 5, we describe our empirical investigation of the DMM model and the DMM model with activity bigrams together with other clustering methods for activity discovery. Finally, the last sections conclude with a summary of the results and areas of future study.

2 Related works

Activity recognition, which identifies known activities, has been studied extensively. In contrast activity discovery, which discovers unknown activities latent in data, has not been studied to the same extent [1–3]. Research on activity discovery conducted to date has included the use of background knowledge regarding activities and sensors [23] and the clustering of sensor data based on heuristics of daily behaviors [5]. In [24], frequent itemset mining [25] together with heuristics of daily behavior is applied for activity discovery.

Because the sensor data comprise time-series, subsequences of sensor signals are extracted from data and are used to form patterns of activities in [4, 8, 21, 24]. In [8], frequent patterns of activities can be extracted by using frequent sequence mining. These are then used to partially
define “other activities” that do not correspond to the predefined activities and thus can contribute to improving the accuracy of activity recognition. In [21], subsequences of sensor signals are processed further by using business process mining techniques, and the frequent patterns of activities are extracted as graphs with subsequences as nodes. Though the process is not fully unsupervised, users of the visualization tool can analyze the behavior of residents using activity patterns extracted semi-automatically from a massive number of subsequences. However, the susceptibility of such sequence mining to noise is a clear weakness. Noise occurs frequently in data from PIR motion sensors; for example, it occurs due to false detection of nonhuman heat sources or by incorrectly triggering adjacent sensors due to movement in slightly anomalous locations. If noise is treated as a gap in sequences of data, the computational complexity of sequence mining becomes intractable in general [26].

Methods that use probabilistic models are robust to noise. In [27], the Gaussian mixture model for time spent in each room is learned based on which mixture components specify the distribution of time required for certain activities. Other studies assume models in which unobservable activities generate raw sensor signals or low-level discriminative features. For example, the LDA model, which is a topic model that defines a process to generate words depending on topics in documents, is used to model the process of generating low-level discriminative features depending on daily routines [12]. In [13], an LDA model is also used to discover human actions in video from spatio-temporal codebooks extracted from a video sequence. To relax the need to specify the number of activities, hierarchical DPM models are used in [14, 15]. In these methods, after the parameters of the models are estimated through unsupervised learning, an activity that generates new observations is estimated using the posterior probability of the model.

In this study, we focus on activity discovery using generative models of binary sensor signals generated from latent activities. In addition to the LDA and the DPM models, we also consider the use of the HMM for activity discovery. Although the HMM has been extensively used to perform activity recognition under supervised learning settings, it can be trained without activity labels under an unsupervised learning setting. Moreover, in this study, we consider the DMM model, which has favorable properties for modeling motion sensor signals. When no supervised signal is available, the joint probability distribution of the observations is used as a clue to fit the model to the data. We consider that the key to improving activity models is to ensure that the model’s assumptions about the generation of the observations capture the essence of the real data generation process.

3 Generative models for activity discovery

Generative models [9] are probabilistic models describing the generation of observations depending on states. After fitting a model to data, the trained model can be used to estimate a state that would have generated new observations. For activity discovery, the observations are sensor signals and the states are unobservable activities that remain to be discovered. In this section, we review the generative models that have been used in activity discovery in more detail, along with the related key concepts such as the burstiness of sensor signals, modeling latent activities, and Markov dependency on activities.

3.1 Modeling burstiness of sensor signals

Perhaps the simplest generative model of binary sensors would include the naive Bayes assumption: given an activity, each sensor signal is generated independently of other signals within that activity. When each signal \( w_n \in \{1, \ldots, V\} \) (1 ≤ \( n \) ≤ \( N \)) is assumed to be distributed according to a categorical distribution with parameter \( \phi = \phi_1, \ldots, \phi_V \) \((\sum \phi_v = 1)\), the probability of generating an observation \( w = w_1, \ldots, w_N \) vector can be written as follows:

\[
P(w \mid \phi) = \prod_{n=1}^{N} \prod_{v=1}^{V} \phi_{w_n=v} = \prod_{v=1}^{V} \phi_v^{x_v},
\]

where \( x_v = \sum_{n=1}^{N} [w_n=v] \) and \([\cdot]\) is Iverson bracket, i.e., [\( P \)] = 1 if \( P \) is true and \([ P ] = 0 \) otherwise.

For any count vector \( x = (x_1, \ldots, x_V) \), \( x \) is then distributed according to a multinomial distribution with parameters \( \phi \):

\[
\text{Mul}(x \mid \phi) = N! \prod_{v=1}^{V} \frac{\phi_v^{x_v}}{x_v!},
\]

where \( N = \sum_{v=1}^{V} x_v \).

Fig. 1 Power law of count probabilities of motion sensors in a dataset. The probabilities of firing exactly \( x \) times within every activity were averaged for 3 groups. The 6 most frequently fired sensors, 11 sensors with an average frequency, and 15 rarely fired sensors.
However, modeling sensor signals with a multinomial distribution overlooks an important property, i.e., their burstiness. Figure 1 shows count probabilities of PIR motion sensors in a dataset described in a subsequent section. All 32 sensors are divided into three groups according to their frequency of occurrence: the 6 most frequently occurred sensors, 11 sensors with average frequency of occurrence, and 15 rarely occurred sensors. For each sensor, the probability of firing exactly \( x \) times within an activity is calculated and averaged for each group. If \( x \) is distributed according to a multinomial distribution, the probability of firing exactly \( x \) times decays exponentially and exhibited a straight descending line in the semi-logarithmic graph. In reality, the count probabilities of all three groups of sensors follow the power law of the count probability, that is, \( p(x) = O(x^{-\beta}) \), may be observed. In Eq. 10 presented in Section 4, the burstiness is explained by placing greater emphasis on sensors that fire multiple times. This can be viewed as a Pólya urn model [29]: after a sensor \( v \) is drawn from the urn, it is then returned to the urn and an additional sensor \( v \) is added. This rich-get-richer process causes burstiness in sensor signals.

For any observation vector \( w \), the DM distribution is written as follows.

\[
DM(w | \beta) = \frac{\Gamma(\beta)}{\Gamma(\beta + N)} \prod_v \frac{\Gamma(x_v + \beta_v)}{\Gamma(\beta_v)} = \frac{B(x + \beta)}{B(\beta)},
\]

where \( B(\cdot) \) is the multi-valuate beta function.

### 3.2 Modeling latent activities

The generative models of sensor signals introduced above can be naturally extended to a mixture model with latent activities.

\[
P(w | \alpha, \beta) = \int_{\Theta} P(z | \Theta) \sum_{k=1}^{K} P(z = k | \theta) DM(w | z = k, \beta_k) \, d\theta.
\]

This mixture model is called the DMM model [17].

The LDA model [10] is perhaps a more well-known mixture model with latent activities. It assumes a more complex generation process: an activity is assigned to each sensor output rather than to all outputs, as follows.

\[
P(w | \alpha, \beta) = \int_{\Theta} P(z | \Theta) \left( \prod_{m=1}^{M} P(z = k | \theta) DM(w_m | z = k, \beta_k) \right) \, d\theta.
\]

Figure 2 shows the DMM and LDA models for \( M \) segments, i.e., for \( \mathbf{w}_1, \ldots, \mathbf{w}_M \). Again, the activity \( z \) to which is assigned is a notable difference between DMM and LDA models. In the DMM model, \( z \) is assigned to each \( w_m \), whereas in the LDA model, \( z \) is assigned to each \( w_{mn} \in w_m \).

To express the multi-topic nature of a document, the LDA model assigns multiple topics to a document. The hierarchical DPM model has the same generation process of
words though the number of topics is determined by data. In contrast, the DMM model is used to model short documents [17], given that short documents tend to discuss a single topic. The current research hypothesizes that considering that a single activity generates sensor signals within a short period is more effective than considering a mixture of multiple activities. As discussed in [1], there are many types of home activities with different levels of complexity. An activity with a higher complexity may have sub-activities, and the LDA model can be a suitable approach to describing a complex activity. However, in activity discovery, a stream of sensor signals is often structured in a bottom-up manner, being divided into fixed-length short segments, and a sequence of activities generating the segments is estimated during the first step. The DMM model is expected to be effective in modeling each short segment generated by a single activity.

Burstiness is another advantage of the DMM model over the LDA model. Although both models use a DM distribution, an LDA model cannot capture burstiness as DMM models do. Note that, with an LDA model, each sensor signal in a segment is obtained using a single draw from a DM distribution conditioned on an activity. Although a single drawing of a sensor $v$ from an urn increases the number of sensors $v$ in the urn, a burst of such sensors is suppressed because the next $v$ may not necessarily be drawn from the same urn. In contrast, in the DMM model, because all sensor signals in a segment are drawn from the same urn, the more sensors $v$ that are chosen from the urn, the more likely a specific $v$ will be chosen.

### 3.3 Modeling Markov dependency on activities

As a drawback of the LDA and DMM models, neither can capture the dependencies on the activities. Because any activity naturally depends on previous activities, a generative model that can capture these dependencies is expected to be effective. For this reason, the HMM is used to perform activity recognition. The HMM assumes that a transition into another state depends on the current state which is defined by the transition probability of such states. Then, each sensor signal is assumed to be emitted from a state according to the emission probability. In activity recognition, because the transition of states, i.e., activities, can be observed along with the sensor signals, the transition probability can be estimated using the maximum likelihood method directly from the observations. In activity discovery, the transition of activities cannot be observed. In this case, using the Baum-Welch algorithm [30], the parameters of the HMM can be estimated solely from the sensor signals.

In the HMM, sensor signals in a segment are generated from multiple activities, similar to the LDA model, because an activity generates a single sensor signal. Although, in this respect, the DMM model is expected to be advantageous for modeling the burst signal generation within a short time period, it requires the ability to describe dependencies between the activities. In the following section, we extend the DMM model to capture the dependency on activities using activity bigrams.

### 4 DMM models

In [17], the DMM model is applied to the clustering of short texts, and a Gibbs sampler used to estimate the parameters of the model is presented. In the present work, we present an extension of the DMM model with activity bigrams and another Gibbs sampler for the extended model. To introduce the extension, we revisit the derivation of the original Gibbs sampler.

#### 4.1 Gibbs sampler for DMM model

Although activities $Z = z_1, \ldots, z_M$ are unobservable in a real setting, we start by showing the joint distribution of activities $Z$ and sensors data $W = w_1, \ldots, w_M$, where...


\[ w_m = w_{m,1}, \ldots, w_{m,N_m} (1 \leq m \leq M) \] are sensor outputs in the \( m \)-th segment with length \( N_m \). The joint probability is composed of two factors, which comprise the following.

\[ P(W, Z | \beta, \alpha) = P(Z | \alpha)P(W | Z, \beta, \alpha). \]

The first factor is

\[
P(Z | \alpha) = \int_\theta D \text{ir}(\theta | \alpha) \prod_k \left( \frac{1}{B(\alpha)} \right)^{\sum_{\beta_k} (\theta_k)^{n_{k,v}}} d\theta
\]

\[
= \int_\theta \left( \frac{1}{B(\alpha)} \right)^{\sum_{\beta_k} (\theta_k)^{n_{k,v}}} d\theta = B(\tau + \alpha),
\]

where \( \tau = (\tau_1, \ldots, \tau_K) \) are counters for each activity, i.e., \( \tau_k = \sum_m [z_m = k] \) for each \( k \) (\( 1 \leq k \leq K \)). The second factor is

\[
P(W | Z, \beta, \alpha) = \prod_k \left( \frac{1}{B(\beta_k)} \right)^{\sum_{\phi_{k,v}} (\phi_{k,v})^{n_{k,v}}} d\phi_k
\]

\[
= \prod_k \left( \frac{1}{B(\beta_k)} \right)^{\sum_{\phi_{k,v}} (\phi_{k,v})^{n_{k,v}}} d\phi_k = \prod_k B(\omega_k + \beta_k),
\]

where \( \omega_k = (\omega_{k,1}, \ldots, \omega_{k,V}) \) are counters for each sensor, that is, \( \omega_{k,v} = \sum_m \sum_n [z_m = k \text{ and } w_{m,n} = v] \) for each \( k \) (\( 1 \leq k \leq K \)) and \( v \) (\( 1 \leq v \leq V \)).

To estimate unobservable \( Z \) using the MCMC method, a Gibbs sampler randomly samples each activity \( z_m \) according to the following conditional probability.

\[
P(z_m = k | Z\text{'}^m, W, \alpha, \beta) = \frac{P(Z, W | \alpha, \beta)}{P(Z\text{'}^m, W | \alpha, \beta)} \]

\[
= \frac{P(Z | \alpha)P(W | Z, \beta)}{P(Z\text{'}^m | \alpha)P(W\text{'}^m | Z, \beta, \alpha)} \]

\[
= \frac{P(Z | \alpha)P(W | Z, \beta)}{P(Z | \alpha)P(W | Z, \beta)}
\]

Thus, Eq. 8 is obtained as follows.

\[
P(z_m = k | Z\text{'}^m, W, \alpha, \beta) \propto \frac{r_{k,v} + \omega_k + \beta_k}{M - 1 + \alpha} \prod_{\phi_{k,v}} (\phi_{k,v})^{n_{k,v}}. \]

From Eq. 3, the factor for activities in Eq. 8, which is equivalent to

\[
P(Z_m = k | Z\text{'}^m, \alpha), \]

is obtained as follows.

\[
P(Z | \alpha) = \frac{B(\tau + \alpha)}{B(\tau^m + \alpha)} = \frac{\Gamma(\tau_{k,v} + \alpha)}{\Gamma(\tau_{k,v} + \alpha)} \frac{\Gamma(M - 1 + \alpha)}{\Gamma(M + \alpha)}
\]

\[
= \frac{\tau_{k,v}^m + \alpha}{M - 1 + \alpha},
\]

where \( \alpha = \sum_k \omega_k \), \( \tau_{k,v}^m = \tau_k - [z_m = k] \), and \( \tau^m = (\tau_1^m, \ldots, \tau_K^m) \). The last equation is obtained by not-

\[
\text{From Eq. 4, the factor for sensors in Eq. 8, which is equivalent to}
\]

\[
P(w_m | Z, W\text{'}^m, \beta), \]

is obtained as follows.

\[
P(w_m | Z, \beta) = \frac{B(\omega_k + \beta_k)}{B(\omega_k + \beta_k)} = \frac{\Gamma(\omega_{k,v} + \beta_{k,v})}{\Gamma(\omega_{k,v} + \beta_{k,v})} \frac{\Gamma(M - 1 + \alpha)}{\Gamma(M + \alpha)}
\]

\[
= \frac{\omega_{k,v}^m + \beta_k}{M - 1 + \alpha}.
\]
4.2 DMM model with activity bigrams

To capture the Markov dependency on the activities, we assume that \( z_m \) depends on \( z_{m-1} \). Because \( z_m \) is drawn from a categorical distribution \( \theta, \theta \) should also be conditioned on \( z_{m-1} \). Let us assume that a distribution \( \theta_q \) depends on the previous activity \( q \), and is drawn from a Dirichlet distribution with hyper parameters \( \alpha_q = (\alpha_{q,1}, \ldots, \alpha_{q,K}) \). Figure 3 shows an extended DMM model with activity bigrams.

For the DMM models, although the dynamical change in topic distributions is introduced in [31], we consider a static correlation between topics of adjacent documents employing topic bigrams.

For the extended model, whereas the factor for the sensor outputs in a joint distribution is the same as Eq. 4, the factor for the activities in a joint distribution differs from Eq. 3 as follows.

\[
P(Z | \alpha) = \prod_q \int_{\theta_q} \text{Dir}(\theta_q | \alpha_q) \prod_r (\theta_{q,r})^{z_{m,r}} d\theta_q
\]

\[
= \prod_q \int_{\theta_q} \left( \frac{1}{B(\alpha_q)} \prod_r (\theta_{q,r})^{z_{m,r}-1} \right) \prod_r (\theta_{q,r})^{z_{m,r}} d\theta_q
\]

\[
= \prod_q B(\tau_q + \alpha_q)
\]

(12)

where \( \alpha = \alpha_1, \ldots, \alpha_K \), \( \tau_{q,r} = \sum_{m=1}^{M} [z_{m-1} = q \text{ and } z_m = r] \), \( \tau_q = (\tau_{q,1}, \ldots, \tau_{q,K}) \).

To obtain a Gibbs sampler for the extended DMM model, we can again use Eq. 8 because \( w_m \) and \( w_{m'} (m' \neq m) \) are independent given \( z_m \) even when dependency on \( Z \) exists, and thus Eq. 7 remains equals to Eq. 6. However, the factor for the activities differs from Eq. 9 and is obtained from Eq. 12 as follows.

\[
P(z_m = k | Z|^{m}, \alpha) = \frac{P(Z | \alpha)}{P(Z|^{m}) | \alpha)}
\]

\[
= \prod_q \frac{B(\tau_q + \alpha_q)}{B(\tau_q^{m} + \alpha_q)}
\]

\[
= \prod_q \Gamma(\tau_{q,1}^{m} + \alpha_{q,1}) \ldots \prod_q \Gamma(\tau_{q,K}^{m} + \alpha_{q,K})
\]

Here, recall that \( z_m \) depends not only on \( z_{m-1} \) but also on \( z_{m+1} \), and thus \( z_m \) affects \( r_{m-1} \) and \( r_m \). Let us assume that \( z_m = j \) and \( z_{m+1} = \ell \), and consider the following three cases.

For \( j \neq k \), we can see that \( \tau_{j,r} = \tau_{j,r}^{m} + 1 \) and \( \tau_{k,r} = \tau_{k,r}^{m} + 1 \), and for the other cases \( q \neq j, k, \tau_{q,r} = \tau_{q,r}^{m} \) hold. It may be observed that \( \tau_{j,k} = \tau_{j,k}^{m} + 1 \) and \( \tau_{k,\ell} = \tau_{k,\ell}^{m} + 1 \) hold, as does
\( \tau_{q,r} = \gamma_{q,r}^m \) for the other cases. We thus obtain the following.

\[
P(z_{m} = k \mid Z^m, \alpha) = \prod_{q \neq k} \frac{\Gamma(\gamma_{q,k}^m + \alpha_{k})}{\Gamma(\gamma_{q,k})} \prod_{r \neq k} \frac{\Gamma(\gamma_{k,r}^m + \alpha_{r})}{\Gamma(\gamma_{k,r})}
\]

\[
P(z_{m} = k \mid Z^m, \alpha) = \frac{\Gamma(\gamma_{k,k}^m + \alpha_{k})}{\Gamma(\gamma_{k,k} + 1)} \frac{\Gamma(\gamma_{k,k}^m + \alpha_{k,k} + 1)}{\Gamma(\gamma_{k,k}^m + \alpha_{k,k} + 1)} \frac{\Gamma(\gamma_{k,k}^m + \alpha_{k,k})}{\Gamma(\gamma_{k,k}^m + \alpha_{k,k})}
\]

where the last equation is obtained by using \( \Gamma(x + 1) = x \Gamma(x) \).

For \( j = k \) and \( j' \neq k \), we can see that \( \tau_{k,k} = \tau_{k,k}^m + 2 \) and for the other case \( q \neq k \), \( \tau_{q,r} = \tau_{q,r}^m \) hold. We can also note that \( \tau_{k,k} = \tau_{k,k}^m + 1 \) and \( \tau_{k,k'} = \tau_{k,k'}^m + 1 \) hold, as does \( \tau_{q,r} = \tau_{q,r}^m \) for the other cases. We thus have the following.

\[
P(z_{m} = k \mid Z^m, \alpha) = \frac{\tau_{k,k}^m + \alpha_{k}}{\tau_{k,k} + \alpha_{k}} \frac{\tau_{k,k}^m + \alpha_{k,k} + 1}{\tau_{k,k}^m + \alpha_{k,k} + 1} \frac{\tau_{k,k}^m + \alpha_{k}}{\tau_{k,k}^m + \alpha_{k}}
\]

For \( j = k \) and \( j' = k \), we can see that \( \tau_{k,k} = \tau_{k,k}^m + 2 \) and for the other case \( q \neq k \), \( \tau_{q,r} = \tau_{q,r}^m \) hold. Moreover we can also note that \( \tau_{k,k} = \tau_{k,k}^m + 2 \) hold, as does \( \tau_{q,r} = \tau_{q,r}^m \) for the other cases. We thus have the following.

\[
P(z_{m} = k \mid Z^m, \alpha) = \frac{\tau_{k,k}^m + \alpha_{k}}{\tau_{k,k} + \alpha_{k}} \frac{\tau_{k,k}^m + \alpha_{k,k} + 2}{\tau_{k,k}^m + \alpha_{k,k} + 2} \frac{\tau_{k,k}^m + \alpha_{k,k}}{\tau_{k,k}^m + \alpha_{k,k}}
\]

To summarize, for \( Z_{m-1} = j \) and \( Z_{m+1} = \ell \),

\[
P(z_{m} = k \mid Z^m, \alpha, \beta) = \alpha \frac{c_{k,j} + \alpha_{j}}{c_{k} + \alpha_{j} + \beta_{j}} \frac{c_{k,k} + \alpha_{k,k} + 1}{c_{k,k} + \alpha_{k,k} + 1} \frac{c_{k,k} + \alpha_{k,k} + 1}{c_{k,k} + \alpha_{k,k} + 1}
\]

Because the factor for the sensor outputs is the same as Eq. 10, we obtain the following conditional probability for a Gibbs sampling to estimate the parameters of the DMM model with activity bigrams.

\[
P(z_{m} = k \mid Z^m, W, \alpha, \beta) = \alpha \frac{c_{k,j} + \alpha_{j}}{c_{k} + \alpha_{j} + \beta_{j}} \frac{c_{k,k} + \alpha_{k,k} + 1}{c_{k,k} + \alpha_{k,k} + 1} \frac{c_{k,k} + \alpha_{k,k} + 1}{c_{k,k} + \alpha_{k,k} + 1}
\]

By using Gibbs samplers described above, we obtain the learning algorithm for the DMM models, which is listed in the Appendix 1.

## 5 Experiments

In this study, we examine the effectiveness of DMM models in activity discovery using two publicly available datasets obtained from real-life environments. In particular, we aim to gain insight in the following research questions.

1. Are DMM models, which are known to be effective in clustering short texts, also effective in activity discovery? We compared the DMM models with other generative models, HMMs, LDA models and hierarchical DPM models as well as the k-means clustering method and the sequence mining method to answer this question.
2. Are the bigrams of activities in DMM models effective?
3. How does the number of clusters pre-fed to a model affect the performance? In particular, we would like to know how much overfitting to the data occurs if we apply a larger number of clusters than is strictly necessary.
4. How much do the hyper parameters \( \alpha \) and \( \beta \) affect the results? In particular, we would like to know whether it is appropriate to give initial values \( \alpha_k = 1 \) \((1 \leq k \leq K)\) and \( \beta_v = 1 \) \((1 \leq v \leq V)\), which seems reasonable in the absence of prior knowledge.
5. How does the length of the analysis window affect the result? Though the use of the DMM model for short time interval seems to be effective, we would like to know the extent to which its performance degrades as the length increases.
The target sensor stream is divided into segments with a constant time width, and their clusters are predicted as possible activities using the generative models trained from training data without actual activity labels. Because the actual labels also give clusters of segments, that is, clusters of segments with the same activity label, the agreement of these two sets of clusters is evaluated by commonly used clustering metrics.

However, in the actual use of activity discovery algorithms, discovered clusters are examined by the users and assigned appropriate activity labels. A cluster can be more specific than an activity assumed by the users. To examine the correspondence between discovered clusters and activities given by the annotators, we calculate a mapping from the clusters to the activity labels based on the activity labels given to the training data. The mapping can also be used to compute activity labels for any segments in the evaluation data, and more straightforward evaluation metrics such as the accuracy, and F1 score are calculated. These metrics allow us to compare the results of activity discovery algorithms with those of previously studied activity recognition algorithms.

### 5.1 Datasets

We use two publicly available datasets for which activity discovery has been studied.

The first dataset is the house A testbed used in [19]. The data were collected from a 26-year-old man living in a three-room apartment for 25 days. A total of 14 sensors were installed in three rooms, including reed switch sensors to detect the opening and closing of doors, cupboards, microwave ovens, refrigerators, and washing machines, and a flush sensor to detect toilets flushing. In addition, 10 activities, including leaving the residence, using the toilet, showering, brushing one’s teeth, sleeping, preparing breakfast and dinner, snacking, drinking water, and others, were annotated. In the literature [24], frequent itemset mining was applied together with various assumptions regarding activities to obtain a result comparable to supervised classification. We refer to this dataset as the Kasteren dataset.

The Kasteren dataset is a benchmark that has been cited in previous studies. However, because reed switch sensors require significant time and effort to install, PIR motion sensors, which are battery-powered and easy to install, are often used to monitor and analyze the daily activities of the elderly [6, 7]. For this study, we used the Aruba testbed data from the WSU CASAS dataset [20]. This dataset contains data from motion sensors, door switch sensors, and temperature sensors, which were placed in the home of an elderly female volunteer. Although it contains data for a period of approximately 7 months, we only use the data from the first 30 days. This is because living environment and daily activity patterns are highly likely to change during a lengthy period, and we want to remove such data drift in the evaluation of the ability of the generative model to fit the data. Furthermore, we ignore the data from the temperature sensors and the wide-range motion sensors covering an entire room. A total of 26 motion sensors and 3 door open-sensors are used. For the motion sensors, we ignore “off” signals because they are not directly related to activities, and reach an “off” state when a certain time has passed since the sensor stopped detecting people. The dataset is annotated with 11 activities: meal preparation, relaxing, eating, working, sleeping, washing dishes, using the toilet, entering the home, leaving the home, housekeeping, and using a medical device. During the experiments described below, we merge the periods between leaving and entering the residence into a single period and label them as “going out” because we are interested only in the fact that the resident is not present in the house. In addition, we assigned the activity label “others” to a time interval in which no activity label is given. As a result, a total of 11 activity labels were assigned. We refer to such data as the Aruba dataset. The literature [21] shows that, for the Aruba dataset, activity patterns corresponding to the predefined activities can be extracted semi-automatically.

Both datasets were divided into day-by-day data for the evaluation using leave-one-day-out cross-validation. As in a previous study [19], we divided the data at 3:00 a.m., which is the probable end time of the day’s activities. As a result, the Kasteren dataset was divided into 25 days, and the Aruba dataset was divided into 30 days.

### 5.2 Data representation

Because the time intervals of the activities were not known in advance, the time series of the sensor data were divided into analysis windows for which the activities were estimated. Previous studies have used analysis windows defined by a fixed time width or a fixed number of sensor events. The larger the analysis window used, the more error there will be based on the boundary discrepancies between activities and windows. In contrast, if the analysis window is reduced, less information is used for analysis and the prediction error increases. Therefore, the size of the analysis window is directly related to the performance of the activity recognition and the activity discovery. In [19], various fixed time lengths were examined, and finally analysis windows with a 1-min length were adopted. In this paper, we use the same 1-min analysis windows for the Kasteren dataset. For the Aruba dataset, we examined the different lengths of windows: from 30 seconds to 10 minutes.
In [19], each analysis window is represented with a multi-hot vector whose dimensions are the same as the number of sensors and in which three different representations were considered. The first representation was referred to as “raw representation,” where any component corresponding to the sensor turned on in the analysis window is set to a value of 1. The second is “change representation,” which sets 1 to those components corresponding to sensors whose values have changed by turning on and off in the analysis window. The third is “last representation,” which sets a value of 1 to the component corresponding to the last sensor that changed its value before the end of the analysis window. According to the results of the experiment described in [19], the raw representation exhibited a poor activity classification performance. This is partly because a switch sensor is not necessarily turned off when the resident stops using the corresponding object. For example, doors are often left open by mistake, and the switch sensor attached to the door always remains on even if the resident is out of the house. The change representation and the last representation are not affected by such noise.

However, as a drawback of the change representation, the values of all components in a feature vector will be 0 when the state of all sensors is not changed. For example, for 10 minutes while the dishwasher is running, no sensor signal is obtained for most of the segments within 10 min except for the first and last segments. Models that do not consider a context including the first and the last segment, e.g., the LDA model and the DMM model without activity bigrams, have no information to predict activities for the segments with no sensor signal. In this situation, the last representation is helpful, that is, the power on signal of the dishwasher is given to the segments that have no signal. The last representation assumes that the resident stays in the same situation when no sensor information is available because if a situation of the resident changes, it is highly likely that another sensor will be activated in the new situation. This assumption has an unexpectedly positive side effect for the going-out activity, which occupies a large amount of time in the data. During time periods in which the resident is out, for segments that have no sensor signal, the bits corresponding to the front door are set to 1. Although this is simply a heuristic, in fact, the resident is not in the house during the activity, it seems to have been effective, at least for the Kasteren dataset. As shown later, it was also effective for the Aruba dataset, especially for models that cannot describe the context of activities. However, for models capable of taking into account the context of the activities, it is not necessarily needed, which is discussed in Section 5.6.3.

As a drawback of the last representation, when the values of multiple sensors change within a certain analysis window, all information other than the last changed sensor is ignored. Therefore, in this study, we combine the change representation with the last representation. That is, sensors whose values have changed within a certain window were added to the corresponding observation vector, and if no such sensor existed, the last sensor outputting a changed value is then added to the observation vector.

As mentioned above, if the sensor does not change for a long period of time, for example, when the resident is out of the house, a series of analysis windows that include only the last changed sensor are considered. To ensure that such windows are included in a consistent cluster, and to improve the computational efficiency by reducing the number of analysis windows, we merge the consecutive windows in which no sensor changes. We refer to an analysis window or merged windows as a segment.

### 5.3 Generative models and hyper parameters

We investigated five different generative models with latent activities, i.e., the HHM, the LDA model, the hierarchical DPM model, the DMM model, and the DMM model using activity bigrams as well as the $k$-means clustering method and an activity discovery algorithm using frequent sequence mining.

- For the HMM, we used the class `hmmlearn.HMM` and `MultinomialHMM` in the `hmmlearn` toolbox. The maximum number of iterations was set to 3000, and different values of the possible number $K$ of hidden states were attempted. Default values were used for all other parameters. Because multiple states were estimated for a segment, the state with the highest frequency was considered as the state of the segment.

- For the LDA models, we used the class `models.ldamodel` in the Gensim toolbox [36]. For the number $K$ of topics, namely, latent activities, different values were attempted. We used the option in which the hyperparameters $\alpha$ and $\beta$ were learned as asymmetric values from the data. All other parameters were set to the default values. After obtaining the proportion of topics for each segment, the topic with the maximum weight is assigned to the segment.

- For the hierarchical DPM models, we used the class `models.hdpmodel` in the Gensim toolbox [36].

- For the DMM model, we used Algorithms 1 and 2 with $B = 0$, and Eq. 11 as a Gibbs sampler. For the DMM model using activity bigrams, we use Algorithms 1 and 2 with $B = 1$, and Eq. 14 as a Gibbs sampler. The numbers of

---

2 The `hmmlearn` toolbox can be found here: https://github.com/hmmlearn/hmmlearn
iterations $I_1$ and $I_2$ were set to 3000 and 1000, respectively. All hyperparameters $\alpha_k (1 \leq k \leq K)$ and $\beta_v (1 \leq v \leq V)$ are set to 1. These symmetric and flat parameters are reasonable when no assumption about the prior probabilities is known because they give the same probability mass on the $(K - 1)$-simplex of the activities and the $(V - 1)$-simplex of the sensors. Different values of the number $K$ of allowable clusters were attempted.

Other than these generative models, a typical clustering algorithm, the $k$-means method, was applied. We used the class sklearn.cluster.KMeans in the Scikit-learn [35] toolbox. Each segment was represented by a vector whose elements were the probabilities of occurrence of each sensor. Different values of the number $K$ of clusters were attempted.

A frequent sequence mining algorithm, called the AD algorithm, presented in [8] was also applied. We used the code downloadable from CASAS repository [20]. This algorithm loops a specified number of times to extract sequential patterns from sensor signals labeled as the activity “others” and then performs clustering to extract frequent sequence patterns that can compress the data well. In the following experiment, all labels for predefined activities were converted into “others,” and then $K$ loops were run to extract the frequent sequence patterns. While the code can extract frequent patterns in a training dataset, it cannot extract the same patterns in a test dataset. Therefore, we applied the code for the entire dataset and extracted frequent patterns and evaluated the extracted patterns with leave-one-day-out cross-validation. This means the evaluation was not completely fair and is advantageous to this method: patterns were extracted not only from training splits but also from a test split.

### 5.4 Evaluation metrics

To compare the performance of activity discovery algorithms that predict clusters of segments, we use commonly used evaluation metrics of clustering that measures how much discovered clusters agree with annotated activities. In addition, to analyze the correspondence between the clusters and the activities, mappings from cluster labels to activity labels are computed from the activity labels assigned to the training data. With this mapping, we can assign activity labels to the clusters of the evaluation data, and evaluate discovered clusters using more straightforward metrics, i.e., the accuracy, precision, recall, and F1 score. These metrics were computed for all analysis windows in the evaluation data. When consecutive analysis windows were merged into a single segment, the cluster label assigned to the segment was copied to the merged analysis windows. After the evaluation metrics for all analysis windows in an evaluation day were calculated, the computed metrics were then averaged across all evaluation days through leave-one-day-out cross-validation.

To compare the results of experiments using different lengths of the analysis window, we equalized the length of windows prior to the above evaluation. When the length of the analysis window was longer than 60 seconds, then each window was divided into windows of length 60 seconds, and the cluster label of the original window was assigned to the divided windows. If the length of the analysis window was shorter than 60 seconds, shorter windows were merged into a window of length 60 seconds and the cluster label that appeared most frequently in the shorter windows was assigned to the merged window.

The first clustering metric is the adjusted mutual information (AMI) [32], which is an information theoretic measure and quantifies the amount of common information between two partitions of data. The upper bound of AMI is 1, which indicates that the two partitions are equal up to the permutation. For a random partition, the AMI score reaches close to 0.

Another clustering metric, the adjusted Rand index (ARI) [33], is also used. The Rand index [34] counts the number of pairs with matched label assignments, that is, both are assigned to the same cluster label and the same activity label, or both are assigned to different cluster labels and different activity labels. The ARI score is the normalized version of the Rand index; and thus, it is adjusted to close to 0 for random label assignments. For a cluster label assignment equal with the class labels up to the permutation, the ARI score is 1.

To compute both metrics, we used the code in the Scikit-learn [35] toolbox.

When applying an activity discovery to real-world problems, the discovered clusters are examined by humans and are assigned appropriate activity labels. To simulate this process, we used the activity labels assigned to the training data to calculate the mapping $act(\cdot)$ from the cluster labels to the activity labels, which is simply defined as a majority vote. More precisely,

$$act(c) = \arg \max_a \left\{ \left| \left\{ t \in T \mid C(t) = c \text{ and } A(t) = a \right\} \right| \right\},$$

where $T$ is the set of analysis windows in the training data, $C(t)$ is the cluster label of $t$, and $A(t)$ is the activity label of $t$. If $act(\cdot)$ is not undefined for $c$, which is the case that $c$ is discovered only in the evaluation data, then we assume $c$ is mapped to the activity “others”. Assuming the mapping $act(\cdot)$, the accuracy, precision, and recall for the evaluation data are computed as usual. In the literature [19], the
Table 1 The numbers of discovered clusters for (a) Kasteren dataset and (b) Aruba dataset: the numbers are listed for different models with different $K$ starting from the numbers of annotated activities, 10 for Kasteren dataset and 11 for Aruba dataset. For each model, the first row shows the arithmetic means (AMs) of the number of clusters obtained through leave-one-day-out cross-validation. The second row shows their standard deviations (SDs).

|          | (a) Kasteren |          | (b) Aruba |
|----------|--------------|----------|-----------|
|          | $K$ | 10 | 15 | 30 | 50 | 100 |          | $K$ | 11 | 15 | 30 | 50 | 100 |
| HMM      | AM | 10.0 | 15.0 | 29.7 | 47.5 | 88.3 | 11.0 | 15.0 | 30.0 | 49.77 | 95.6 |
|          | SD | 0.0 | 0.2 | 0.5 | 1.8 | 3.1 | 0.0 | 0.0 | 0.0 | 0.5 | 2.2 |
| DMM2     | AM | 10.0 | 14.9 | 27.9 | 42.5 | 80.0 | 11.0 | 15.0 | 30.0 | 47.3 | 72.6 |
|          | SD | 0.0 | 0.3 | 1.4 | 2.5 | 5.2 | 0.0 | 0.0 | 0.0 | 1.4 | 2.9 |
| DMM      | AM | 9.9 | 13.1 | 17.8 | 21.3 | 25.7 | 11.0 | 15.0 | 26.7 | 32.9 | 39.5 |
|          | SD | 0.3 | 1.0 | 1.7 | 2.4 | 1.7 | 0.0 | 0.0 | 1.5 | 1.8 | 2.7 |
| LDA      | AM | 10.0 | 14.3 | 25.1 | 36.5 | 54.2 | 11.0 | 14.9 | 27.3 | 41.1 | 70.4 |
|          | SD | 0.2 | 0.9 | 1.9 | 3.0 | 3.9 | 0.0 | 0.3 | 1.2 | 2.4 | 3.3 |
| DPM*     | AM | 38.4 |          |      |      |      | 111.3 |          |      |      |      |
|          | SD | 4.0 |          |      |      |      | 5.3 |          |      |      |      |
| SeqM     | AM | 5.0 | 8.0 | 15.0 | 25.0 | 49.0 | 6.0 | 8.0 | 15.0 | 25.0 | 50.0 |
|          | SD | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| KM       | AM | 10.0 | 15.0 | 30.0 | 50.0 | 100.0 | 11.0 | 15.0 | 30.0 | 50.0 | 100.0 |
|          | SD | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |

* The values for DPM do not in fact depend on $K$ but rather are placed at the column for the closest $K$.

precision and recall are macro-averaged across the activity labels, and the F1 score is computed from the macro-averaged precision and recall. More precisely, they are computed as follows.

\[
\text{Accuracy}(E) = \frac{\sum_{\mathcal{E}} |TP_{a}(E)|}{|E|},
\]

\[
\text{pre}(E) = \frac{1}{|A(E) \cup AC(E)|} \sum_{a \in AC(E)} |TP_{a}(E)|,
\]

\[
\text{rec}(E) = \frac{1}{|A(E)|} \sum_{a \in A(E)} |TP_{a}(E)|,
\]

\[
F1(E) = \begin{cases} 
\frac{2 \text{pre}(E) \text{rec}(E)}{\text{pre}(E) + \text{rec}(E)} & \text{pre}(E) + \text{rec}(E) > 0 \\
0 & \text{pre}(E) + \text{rec}(E) = 0,
\end{cases}
\]

where $E$ is the set of analysis windows in the evaluation data, $TP_{a}(E) = \{e \in E \mid act(C(e)) = a(e) \}$, $A(E) = \{a(e) \mid e \in E\}$, and $AC(E) = \{act(C(e)) \mid e \in E\}$. To compare the results in [19], we use the same definition above.

As described above, these metrics are calculated for all analysis windows with the same length. Therefore, the accuracy is evaluated by placing more weight on the activities that occupy a greater amount of time. In contrast, the precision, recall, and F1 score are obtained by averaging the scores calculated for each activity; and thus, activities occupying a small amount of time are evaluated with equal weight.

It is also important to note that the accuracy, precision, recall, and F1 score tend to become higher as the number of discovered clusters increases. As an extreme case, if we consider the case in which each cluster consists of only a single element, these evaluation metrics take a value of 1 under a mapping of each cluster label into the activity label given to the single element. Therefore, although such metrics can be used to compare the results obtained with a similar number of clusters, they should not be used to compare the results with significantly different numbers of clusters. In this regard, the AMI and ARI score can be used to compare the results obtained with different numbers of clusters because the effect of the number of clusters is adjusted for both scores.

Table 1 shows the numbers of discovered clusters for the Kasteren dataset and the Aruba dataset, where $HMM$, $DMM2$, $DMM$, $DPM$, $SeqM$ and $KM$ denote the HMM, the DMM model with activity bigrams, the DMM model, the hierarchical DPM model, the frequent sequence mining algorithm and the k-means clustering method respectively. It may be observed that the DMM model maintains a small number of discovered clusters even when $K$ is large. In terms of dataset differences, it is also worth noting that more clusters have been found in the Aruba dataset than in the Kasteren dataset.

5.5 Results for the Kasteren dataset

Table 2 shows the AMI and ARI scores of different clustering methods applied to the Kasteren dataset. We also show the results of an activity classifier, denoted by $supHMM$, using the HMM trained with supervised learning with activity labels. Although the classifier may be considered impractical owing to the limited availability of labeled data, we expected it to exhibit better performance than unsupervised clustering methods. It also shows the results of the DMM model with $a = 100$, denoted by $DMMa$, and the DMM model using activity bigrams with $a = 100$, denoted by $DMM2a$. These extreme settings were motivated by the
unexpected highest performance of the k-means clustering, which only uses the similarity between distributions of sensor signals. A large c approximates the factor of the activities, Eq. 9 or 13, as constants and is equivalent to assuming that activities are determined almost solely by sensor signals, without the help of the distribution of activities. Under the assumption, DMM performed better than DMM, and there exists no significant difference between DMM and DMM.

Moreover, DMM performed comparably with KM and DPM, and it performed slightly, though not significantly, worse than supHMM. The sequence mining, SeqM, performed significantly worse than DMM as K increased and exhibited overfitting as the capacity of the model increased. The LDA model performed comparably or worse than DMM at K = 30, 50.

Table 3 shows the accuracy and F1 scores of of different models. As explained in Section 5.4, both scores were computed after discovered clusters were transformed into activity labels by a mapping defined by the activity labels given to the training data. For both scores, similar results were observed as well as the AMI and ARI scores. Again the k-means clustering method, KM, performed best, and the sequence mining, SeqM, performed worst especially at a small K. The DMM model using activity bigrams with the same K.

---

Table 2 Kasteren dataset: clustering scores (a) AMI and (b) ARI for different models with different numbers K of clusters starting from 10, which is the number of annotated activities. For each model, the first row shows the arithmetic means (AMs) of scores obtained through leave-one-day-out cross-validation. The second row shows their standard deviations (SDs). Each p-value in the third row was obtained from paired t-test against DMM with the same K.

| Model   | K | (a) AM | (b) ARI |
|---------|---|--------|---------|
|         | 10 | 15     | 30      | 50      | 100     | 10     | 15     | 30      | 50      | 100     |
| supHMM  | AM | .858   | .913    |
|         | SD | .085   | .105    |
|         | p-val | .080  | .066   |
| HMM     | AM | .835   | .851    |
|         | SD | .117   | .085   |
|         | p-val | .441  | .342   |
| DMM2a   | AM | .851   | .909    |
|         | SD | .085   | .120   |
|         | p-val | .006  | .010   |
| DMM     | AM | .817   | .887    |
|         | SD | .099   | .119   |
|         | p-val | .007  | .075   |
| DMMa    | AM | .848   | .905    |
|         | SD | .091   | .120   |
|         | p-val | .436  | .012   |
| LDA     | AM | .819   | .866    |
|         | SD | .121   | .173   |
|         | p-val | .081  | .149   |
| DPM     | AM | .850   | .910    |
|         | SD | .084   | .106   |
| SeqM    | AM | .721   | .743    |
|         | SD | .192   | .241   |
|         | p-val | .000  | .000   |
| KM      | AM | .853   | .911    |
|         | SD | .081   | .103   |
|         | p-val | .255  | .131   |

© The score was worse than that of DMM2a with the same K at the 1% significance level
* An activity classifier using the HMM trained through supervised learning
b The score was obtained independently of the parameter K. The p-values were computed against each score of DMM2a with different K.

The highest score f or each K among clustering methods is written in bold font.
\( \alpha = 100 \), \( DMM2a \), performed slightly worse than \( KM \), and comparable with \( DMMa \) and \( DPM \), and performed better than \( HMM \) and \( LDA \).

Of note, the accuracy of the classifier \( supHMM \) was worse than other clustering methods; in particular it was worse than \( HMM \) employing the same type of the HMM model. A more detailed analysis shows that \( supHMM \) exhibited a poor recall rate for the activity labeled “others.” This is because the activity “others” includes a variety of different activities, and a single multinomial distribution of sensor signals cannot represent the class of multiple activities. The HMM model trained through unsupervised learning was able to assign different clusters for activities corresponding to the activity “others” and thus obtained better accuracy. In contrast, for the F1 score, \( supHMM \) performed better than clustering methods because clustering methods failed to assign clusters for activities occupying a small amount of time and the F1 score evaluates such minor activities with equal weight.

In general, activities in the Kasteren dataset seems to be strongly correlated with specific sensor signals; thus, predicting activities based on the sensor signals is relatively easy.

### 5.6 Results for the Aruba dataset

Table 4 shows the AMI and ARI scores of different models applied to the Aruba dataset.

We can see that the DMM model and the DMM models with activity bigrams performed better than the other methods. In particular, the DMM model with activity bigrams scored significantly higher than the HMM, the LDA model,
the hierarchical DPM model, the sequence mining algorithm, and \( k \)-means clustering method at all \( K \)s, and also scored significantly higher than the DMM model at \( K = 11, 15 \). As another observation, both the DMM models showed consistently high scores even when \( K \) increased, whereas the LDA model and the HMM degraded as \( K \) increased.

Table 5 shows the accuracy and F1 scores of different models applied to the Aruba dataset. It may be observed that the DMM model and the DMM model with activity bigrams performed better at a small \( K \). Even in comparison with the classifier, \( supHMM \), trained through supervised learning, the DMM model with activity bigrams showed a comparable performance. As we stated above for the Kasteren dataset, the lower performance of \( supHMM \) stems from a lower recall rate for the activity “others”.

As \( K \) increases, the difference against other methods becomes small. However, overly specific clusters make subsequent analysis difficult. In this regard, for \( HMM, DPM \), and \( KM \), the large number discovered clusters shown in Table 1 might be undesirable.

### 5.6.1 Influence of the initial hyper parameters

Table 6 shows the effects of the initial parameters \( \alpha \) and \( \beta \) on the AMI scores at \( K = 11 \) for the Aruba dataset. Here, we assume the parameters of Dirichlet distributions are symmetric, that is, \( \alpha_k = \alpha \) \( (1 \leq k \leq K) \) and \( \beta_v = \beta \) \( (1 \leq v \leq V) \). We can see that for both the DMM model and the DMM model with activity bigrams, \( \alpha \) did not seem to have any significant effect on the AMI score. For parameter \( \beta \), it made a significant difference in the results, but for the DMM model, \( \beta = 1 \) yielded the best score, whereas for the DMM model with activity bigrams, \( \beta = 4 \) provided the highest AMI score. In any case, there was no large difference from the case of \( \beta = 1 \). Therefore, for this dataset, learning with the initial values of \( \alpha = 1 \) and \( \beta = 1 \), which is reasonable in the absence of prior knowledge, is not an unsuitable option.

### 5.6.2 Influence of the length of the analysis window

Tables 7, 8, and 9 show the AMI and ARI scores of different models applied to the Aruba dataset represented with the

### Table 4 Aruba dataset: clustering scores (a) AMI and (b) ARI for different models with different numbers \( K \) of clusters

| \( K \)  | \( \supHMM \) AM | SD | \( p \)-val | \( HMM \) AM | SD | \( p \)-val | \( DMM2 \) AM | SD | \( p \)-val | \( DMM \) AM | SD | \( p \)-val | \( LDA \) AM | SD | \( p \)-val | \( DPM \) AM | SD | \( p \)-val | \( SeqM \) AM | SD | \( p \)-val | \( KM \) AM | SD | \( p \)-val |
|-------|--------------|-----|------------|--------------|-----|------------|---------------|-----|------------|---------------|-----|------------|--------------|-----|------------|----------------|-----|------------|----------------|-----|------------|----------------|-----|------------|
| 11    | .795         | .058|            | .719         | .070|            | .741           | .024|            | .669           | .073|            | .577           | .097|            | .795           | .017|            | .799           | .011|            |
| 15    | .795         | .058|            | .719         | .070|            | .741           | .024|            | .669           | .073|            | .577           | .097|            | .795           | .017|            | .799           | .011|            |
| 30    | .795         | .058|            | .719         | .070|            | .741           | .024|            | .669           | .073|            | .577           | .097|            | .795           | .017|            | .799           | .011|            |
| 50    | .795         | .058|            | .719         | .070|            | .741           | .024|            | .669           | .073|            | .577           | .097|            | .795           | .017|            | .799           | .011|            |
| 100   | .795         | .058|            | .719         | .070|            | .741           | .024|            | .669           | .073|            | .577           | .097|            | .795           | .017|            | .799           | .011|            |

\( \supHMM \): supHMM, trained through supervised learning, the DMM model with activity bigrams yielded the best score, whereas for the DMM model, \( \beta = 1 \) yielded the best score, whereas for the DMM model with activity bigrams, \( \beta = 4 \) provided the highest AMI score. In any case, there was no large difference from the case of \( \beta = 1 \). Therefore, for this dataset, learning with the initial values of \( \alpha = 1 \) and \( \beta = 1 \), which is reasonable in the absence of prior knowledge, is not an unsuitable option.
Figure 4 shows the change in both scores versus the length of the analysis window at $K = 11$. As the length of the analysis window increases, the performance of the LDA model and the hierarchical DPM model, which generates sensor signals according to a mixture distribution of multiple activities, are expected to improve. However, for the lengths studied here, the performance of the DMM models was consistently better.

### Table 5
Aruba dataset: classification scores (a) accuracy and (b) F1 score

| $K$ | (a) Accuracy | (b) F1 |
|-----|--------------|-------|
|     | supHMM       |       |
| 11  | AM .878      | .584  |
|     | SD .044      | .076  |
|     | p-val .193   | .000  |
| 15  | AM .805      |   .649 |
|     | SD .134      | .494  |
|     | p-val .007   | .508  |
| 30  | AM .869      | .162  |
|     | SD .058      | .508  |
|     | p-val .000   | .518  |
| 50  | AM .869      | .162  |
|     | SD .058      | .508  |
|     | p-val .000   | .518  |
| 100 | AM .869      | .162  |
|     | SD .058      | .508  |
|     | p-val .000   | .518  |

### Table 6
Aruba dataset: comparing the AMI score of (a) the DMM model and (b) the DMM model with activity bigrams at $K = 11$ for different initial parameters of Dirichlet distributions. Each $p$-value in the third row was obtained from paired $t$-test against the setting $\alpha_1 = 1$ and $\beta_1 = 1$.

| $\alpha$ | (a) DMM | (b) DMM2 |
|----------|---------|----------|
| 0.1      | AM .731 | .738     |
|          | SD .058 | .063     |
|          | p-val .055 | .048 | .000 |
| 1.0      | AM .731 | .740     |
|          | SD .063 | .061     |
|          | p-val .047 | .041 | .000 |
| 4.0      | AM .722 | .747     |
|          | SD .062 | .062     |
|          | p-val .001 | .307 | .080 |

5.6.3 Influence of the use of the last representation

Table 10 shows the AMI and ARI scores of different models applied to the Aruba dataset represented without the last representation. For any segment where there was no sensor signal, a special symbol, referred to as the null symbol, was inserted into the null segment.

Models other than $HMM$ and $DMM2$ showed significant degradation in performance. This is because they cannot
distinguish activities such as “sleeping,” “going out,” and “relaxing,” in which the null segments frequently appear. Since there is no information to distinguish the null segments other than context information, they cannot identify activities to which the null segments belong. On the other hand, for a model capable of taking into account the Markov property on activities, it is possible to estimate which activity generated the null segment depending on the previous and next activity.

Another interesting observation is that the performance of supHMM is worse than that of HMM and DMM2. For activities “sleeping” and “going out,” the probability of the occurrence of the null signal is relatively high since the frequency of sensor signals other than the null signal is low in these activities. In contrast, for the activity “relaxing,” there exist many sensor signals other than the null signal occurring in this activity; and thus, the probability of the occurrence of the null signal is relatively low in this activity. Therefore, it is difficult to predict the activity “relaxing” for the null segment even if contextual information of previous and next activities is available. For the HMM and the DMM model with activity bigrams trained through unsupervised learning, a specific cluster can be assigned to the null segments during the activity “relaxing”; and thus, their performance does not degrade. If the data is described by the last representation, sensor signals that frequently appear in the activity “relaxing” will be inserted in the null segment, so supHMM can predict the null segment correctly; and thus, its performance degradation can be avoided.

As shown above, the last representation was a useful heuristics for the Aruba dataset as well as for the Kasteren dataset though it is not necessarily useful for other datasets. Without the use of the last representation, the HMM model and the DMM model with activity bigrams trained through unsupervised learning showed good performance. Especially, the DMM model with activity bigrams showed consistently better performance even when its model capacity became large in contrast with the HMM model, which overfitted as $K$ increased.

### 5.7 Discussion

As may be seen above, in the Kasteren dataset, there is no significant performance difference between different activity discovery methods investigated in this study. However, the HMM degrades at a large number $K$ of activities, and $K$

| $K$ | (a) AMI | (b) ARI |
|-----|--------|--------|
| 11  | 0.797  | 0.828  |
| 15  | 0.807  | 0.828  |
| 30  | 0.807  | 0.828  |
| 50  | 0.807  | 0.828  |
| 100 | 0.807  | 0.828  |

### Table 7 Aruba dataset: clustering scores (a) AMI and (b) ARI for data represented with analysis windows of length 30 seconds

| Method | AMI     | ARI     |
|--------|---------|---------|
| supHMM | AM     | ARI     |
| SD     |        |         |
| $p$-val|        |         |
| HMM    | AM     | ARI     |
| SD     |        |         |
| $p$-val|        |         |
| DMM2   | AM     | ARI     |
| SD     |        |         |
| $p$-val|        |         |
| DMM    | AM     | ARI     |
| SD     |        |         |
| $p$-val|        |         |
| LDA    | AM     | ARI     |
| SD     |        |         |
| $p$-val|        |         |
| DPM    | AM     | ARI     |
| SD     |        |         |
| $p$-val|        |         |
| SeqM   | AM     | ARI     |
| SD     |        |         |
| $p$-val|        |         |
| KM     | AM     | ARI     |
| SD     |        |         |
| $p$-val|        |         |
should be carefully chosen for the sequence mining method. Also for the DMM models, by setting the hyperparameter \( \alpha \) to an extremely large value, the factor of the distribution of activities should be canceled out. Then activities were estimated only from the distribution of sensor signals; and thus, good performance is obtained. Together with the highest performance of the \( k \)-means clustering methods that use only the similarity between the distribution of sensor signals, activities in this dataset can be easily predicted directly from sensor signals. It is also consistent with the result shown in [24] that the frequent itemset mining is applied to the dataset and yields a comparable performance to the classifier using the HMM trained through supervised learning. We could not find much merit in applying the DMM models to the dataset although the models achieve comparable performance by adapting to the extreme condition.

In contrast, for the Aruba dataset, the DMM models perform better than the other activity discovery methods above. One of the differences between the two datasets is the burstiness. In the Kasteren dataset, after a sensor is turned on, it is unlikely that the sensor will continue to be turned on, for example, it is unlikely that a toilet flushing sensor will be turned on multiple times. In contrast, in the case of the Aruba dataset, when a sensor detects a motion, it likely fires more than once owing to the same motion, and it makes sense that we would observe the burstiness shown in Fig. 1. The results of the experiment in which the DMM models perform well in the Aruba dataset support the hypothesis that the DMM models are advantageous for data that show burstiness. One might think that the burstiness can be artificially removed by filtering out sensor signals that fire repeatedly. However, it is not obvious that burstiness can be removed because the firing of a signal is not necessarily repeated consecutively, but is interrupted by the firing of other signals and noise. Moreover, artificially manipulating the sensor signal has a high risk of distorting the true distribution. Rather, the burstiness should be used as a hint for estimating the true distribution. For example, if some sensor signals are not inherently associated with activities and do not exhibit burstiness, then we can reduce the effect of the noisy signals by capturing burstiness of sensors generated

| Table 8 Aruba dataset: clustering scores (a) AMI and (b) ARI for data represented with analysis windows of length 300 seconds |
| --- |
| **K** | **AMI** | **(a) AMI** | **(b) ARI** |
| 11 | .745 | .800 |
| 15 | .618 | .777 |
| 30 | .691 | .778 |
| 50 | .671 | .765 |
| 100 | .720 | .757 |

| supHMM | AM | .745 | .800 |
| SD | .066 | .079 |
| p-val | .000 | .000 | .000 | .002 | .000 | .000 | .000 | .000 | .000 |
| HMM | AM | .638 | .691 | .677 | .625 | .594 |
| SD | .142 | .174 | .196 | .209 | .199 | .241 | .237 |
| p-val | .001 | .004 | .007 | .015 | .016 | .003 | .000 |
| DMM2 | AM | .725 | .720 | .697 | .691 | .777 | .778 | .765 | .757 | .765 |
| SD | .064 | .064 | .070 | .070 | .068 | .070 | .070 | .070 | .070 | .070 |
| p-val | .000 | .000 | .000 | .000 | .000 | .000 | .000 | .000 | .000 | .000 |
| DMM | AM | .691 | .687 | .672 | .671 | .741 | .743 | .739 | .738 | .737 |
| SD | .062 | .064 | .070 | .070 | .069 | .070 | .070 | .070 | .070 | .070 |
| p-val | .000 | .000 | .000 | .000 | .000 | .000 | .000 | .000 | .000 | .000 |
| LDA | AM | .662 | .674 | .656 | .635 | .614 | .684 | .707 | .676 | .620 | .573 |
| SD | .087 | .074 | .073 | .067 | .069 | .146 | .112 | .121 | .128 | .122 |
| p-val | .000 | .000 | .000 | .000 | .000 | .000 | .000 | .000 | .000 | .000 |
| DPM | AM | .647 | .647 |
| SD | .070 | .070 |
| p-val | .000 | .000 | .000 | .000 | .000 | .000 | .000 | .000 | .000 | .000 |
| SeqM | AM | .474 | .495 | .563 | .574 | .584 | .374 | .403 | .549 | .583 | .613 |
| SD | .087 | .078 | .078 | .075 | .072 | .147 | .137 | .149 | .144 | .141 |
| p-val | .000 | .000 | .000 | .000 | .000 | .000 | .000 | .000 | .000 | .000 |
| KM | AM | .683 | .674 | .642 | .626 | .612 | .657 | .647 | .625 | .612 | .603 |
| SD | .072 | .070 | .070 | .070 | .071 | .116 | .120 | .126 | .126 | .127 |
| p-val | .000 | .000 | .000 | .000 | .000 | .000 | .000 | .000 | .000 | .000 |
from the activities. Thus, a more noise-robust and accurate activity model is expected to be estimated from the data.

As can be seen in Table 4, the number \( K \) of possible clusters does not affect the AMI and ARI scores of the DMM model. In contrast, the HMM and the LDA model degrade significantly when a large \( K \) is given, which means that both models require a method for determining the appropriate \( K \) depending on the data. Hierarchical DPM model is a topic model similar to the LDA model, but it determines the number of clusters from the data. The DMM model performs consistently better than the DPM model, regardless of the value of \( K \). Especially, the DMM model with activity bigrams performs significantly better than the original DMM model at \( K = 11, 15 \), and slightly superior at \( K = 100 \). Information on the context of activities seems to be effective even when data are described by using the last representation, but it is more vital when the last representation is not used as shown in Table 10.

**Fig. 4** Aruba dataset: clustering scores (a) AMI and (b) ARI for data represented with the analysis window with lengths of 30, 60, 300, and 600 seconds at \( K = 11 \)
Compared with the activity classifier supHMM using the HMM trained through supervised learning, the DMM model with activity bigrams performs significantly worse for the AMI and ARI scores. In terms of accuracy, it performs comparable or slightly better than the classifier. As stated before, the performance gain is because different clusters can be assigned to the various activities included in the activity labeled as “others”. The improvement is consistent with a result in the literature [8], in which a newly discovered activity that does not correspond to the predefined activities can contribute to improving the accuracy of activity recognition. Data collected in a real-life setting contains many activities other than predetermined activities, and accurate modeling is possible by finding the latent activities through activity discovery. This is another advantage of activity discovery other than lower annotation cost.

For the F1 score, the DMM model with activity bigrams performs comparable or slightly worse than supHMM. This is due to the poor prediction of some activities, such as “using the toilet,” “washing dishes,” “housekeeping,” and “using medical device.” This is partly because of the lack of data to estimate clusters corresponding to the activities. However, another reason stems from the mapping of the activity label for discovered clusters based on the majority voting. For a cluster of segments, the activity label for the cluster is determined as the most frequent label among labels assigned to the segments. Now, if we consider a case in which one discovered cluster is a common sub-activity of multiple activities, the label of this cluster will be mapped to the label of the most frequent activity. As a result, the precision, recall, and F1 scores for less frequent activities will deteriorate significantly, whereas the accuracy, which depends heavily on the frequent activities, does not deteriorate much. Even if a common sub-activity is correctly identified, it will not be utilized if it is converted into an activity label defined by the majority vote. It is thus necessary to have a mechanism to assign a new and different label as a common sub-activity or to assign different sub-activity labels depending on which activity it belongs to.

Table 10: Aruba dataset: clustering scores (a) AMI and (b) ARI for data represented without use of the last representation

|       | (a) AMI |    |    |    |    | (b) ARI |    |    |    |    |
|-------|--------|----|----|----|----|--------|----|----|----|----|
|       | K      | 11 | 15 | 30 | 50 | 100    | 11 | 15 | 30 | 50 |
| supHMM| AM     | .599|   |   |    | .516   |    |    |    |    |
|       | SD     | .010|   |   |    | .057   |    |    |    |    |
|       | p-val  | .000 |   |   |    | .000 | .000 | .000 | .000 | .000 |
| HMM   | AM     | .709| .725| .685| .643| .647   | .740| .770| .705| .666| .668 |
|       | SD     | .078| .070| .067| .134| .066   | .117| .108| .115| .162| .109 |
|       | p-val  | .005 |   |   |    | .000 | .000 | .000 | .004 | .011 | .000 |
| DMM2  | AM     | .683| .694| .697| .692| .705   | .722| .744| .768| .755| .774 |
|       | SD     | .086| .066| .075| .064| .057   | .140| .104| .118| .110| .091 |
|       | p-val  | .000 |   |   |    | .000 | .000 | .000 | .000 | .000 | .000 |
| LDA   | AM     | .326| .328| .318| .270| .259   | .185| .185| .179| .148| .140 |
|       | SD     | .040 |   |   |    | .042 | .043 | .046 | .043 | .055 | .056 |
|       | p-val  | .000 |   |   |    | .000 | .000 | .000 | .000 | .000 | .000 |
| DPM   | AM     | .301|   |    |    | .180   |    |    |    |    |    |
|       | SD     | .058|   |    |    | .055   |    |    |    |    |    |
|       | p-val  | .000 |   |    |    | .000 | .000 | .000 | .000 | .000 | .000 |
| SeqM  | AM     | .300| .316| .427| .465| .504   | .189| .192| .336| .332| .436 |
|       | SD     | .058 |   |   |    | .058 | .069 | .065 | .076 | .068 | .069 |
|       | p-val  | .000 |   |    |    | .000 | .000 | .000 | .000 | .000 | .000 |
| KM    | AM     | .358| .333| .320| .307| .288   | .187| .184| .180| .178| .177 |
|       | SD     | .042 |   |   |    | .041 | .040 | .038 | .038 | .057 | .056 |
|       | p-val  | .000 |   |    |    | .000 | .000 | .000 | .000 | .000 | .000 |

To be continued...
A clear drawback of the DMM model is the simplistic assumption that sensor signals are generated from a single activity. This contrasts with the LDA and hierarchical DPM models, which assume that the sensor signals are generated from a mixture of multiple activities. It seems reasonable to assume that sensor signals within short time intervals are generated by one specific activity, not multiple activities. However, the more the length of the analysis window, the more likely multiple activities are involved in the process of the generation of sensor signals. To see the influence of the length of the analysis window, we conducted experiments using the analysis window with different lengths: 30, 60, 300, and 600 seconds. The experiments show that the DMM model consistently performed well in all cases. Therefore, the DMM model seems to be effective in discovering activities that correspond to the time granularity considered here.

6 Limitations and future work

In the experiments discussed above, to automatically calculate the accuracy and F1 scores, the mapping from clusters to activity labels is calculated using the activity labels assigned to the training data. However, the actual use of an activity discovery algorithm requires interacting with users to determine the activity labels of the discovered clusters. To this end, it is important to have a means to efficiently determine the correspondence between clusters and activities with as few questions for the users as possible.

For example, a cluster may need to be identified as different activities depending on the day of the week or the time of the day. In other cases, when a common sub-activity of different activities is extracted as a single cluster, it may be desirable to distinguish the cluster depending on which activity it belongs to. Alternatively, it may be necessary to hierarchize different clusters to form a common activity. Though we cannot discuss the problem of transforming discovered clusters into activity labels understandable for users, it is an important future research topic.

The proposed method can also be improved by incorporating various assumptions used in the activity analysis. For example, if the possible activities differ depending on the day of the week or the time of the day, it would be effective to divide up data by the day of the week or the time of the day and then to train separate models. In addition, the proposed method does not model the duration of activities. Although it clusters segments with a fixed time interval and identifies successive identical clusters as a single activity a posteriori, it may be possible to improve the clustering performance by explicitly modeling the duration.

Finally, data drift needs to be addressed in an actual deployment of the proposed method. The manner in which activities are conducted can change over even a short period of time owing to changes in the physical or psychological capabilities of the residents, or owing to changes in the living environment such as the layout of furniture or equipment. Since the proposed method uses unsupervised learning, it has the advantage that a new model can be built without data annotation cost if it is retrained with the most recent data. However, there are many unresolved issues, such as when to relearn and how to maintain the consistency of the discovered activities before and after relearning.

7 Conclusion

We have applied the DMM model to the activity discovery problem. The DMM model assumes that a latent activity generates sensor outputs, and employs the DM distribution as a probability distribution for the generation of sensor outputs. We empirically confirmed the effectiveness of the DMM model over other generative models such as the HMM, the LDA model, and the hierarchical DPM model, as well as the k-means method and frequent sequence mining. In particular, based on the DM distribution, the DMM model can more accurately model the data generation process in a dataset using PIR motion sensors, which exhibit burstiness, where the distribution of sensors follows the power law and even sensors that rarely fire may fire repeatedly after being triggered. To cope with the Markov dependency on activities, the DMM model was extended so that each activity depends on the preceding activity. To estimate the parameters of the extended model, we have presented a Gibbs sampler used in the MCMC algorithm. It was also shown empirically that the extended DMM model can discover a latent structure of activities in the dataset more accurately than the original DMM model. The activity discovery method discussed in this paper can be used as a primitive activity extraction tool in a more advanced human-in-the-loop activity analysis system. It is also expected that the DMM model can be useful to model other data with burstiness generated from latent states with the Markov property.
Appendix 1: Activity discovery algorithm using the DMM models

Algorithm 1 Learning from $W_{\ell}$ ($1 \leq \ell \leq L$) with length $M_{\ell}$

Require: $K$, $I_{\ell}$ and $B$  \text{\textdeg$B = 1$ enables the use of activity bigrams}
Output: cluster labels $Z_{\ell}$ ($1 \leq \ell \leq L$) for $W_{\ell}$

1: InitCounters($\tau$, $\omega$)
2: for $\ell = 1$ to $L$ do
3: \hspace{1em} InitTopics($Z$, $W_{\ell}$, $M_{\ell}$, $\tau$, $\omega$)
4: \hspace{1em} for $i = 1$ to $I_{\ell}$ do
5: \hspace{2em} for $m = 1$ to $M_{\ell}$ do
6: \hspace{3em} $k \leftarrow z_{m}$
7: \hspace{2.3em} \hspace{1em} $q, r \leftarrow \text{Adjacents}(Z, m)$
8: \hspace{2.3em} \hspace{1em} ReduceCtrs($q$, $k$, $r$, $w_{\ell,m}$, $\tau$, $\omega$)
9: \hspace{2.3em} \hspace{1em} $k \leftarrow \text{Sampling}(q, k, r, w_{\ell,m}, \tau, \omega)$
10: \hspace{2.3em} \hspace{1em} $z_{m} \leftarrow k$
11: \hspace{2.3em} \hspace{1em} IncreaseCtrs($q$, $k$, $r$, $w_{\ell,m}$, $\tau$, $\omega$)
12: \hspace{2em} end for
13: end for
14: end for
15: for $\ell = 1$ to $L$ do
16: \hspace{1em} for $m = 1$ to $M_{\ell}$ do
17: \hspace{2em} $z_{\ell,m} \leftarrow \text{GetMax}(q, r, w_{\ell,m}, \tau, \omega)$
18: \hspace{2em} end for
19: end for

Algorithm 2 Estimate cluster labels for $W'$ with length $M'$

Require: $Z_{\ell}$ ($1 \leq \ell \leq L$) for $W_{\ell}$, $K$, $I_{\ell}$ and $B$

Output: cluster labels $Z'$ for $W'$

1: InitCounters($\tau$, $\omega$)
2: for $\ell = 1$ to $L$ do
3: \hspace{1em} for $m = 1$ to $M_{\ell}$ do
4: \hspace{2em} $k \leftarrow z_{\ell,m}$
5: \hspace{2em} \hspace{1em} $q, r \leftarrow \text{Adjacents}(Z_{\ell}, m)$
6: \hspace{2em} \hspace{1em} IncreaseCtrs($q$, $k$, $r$, $w_{\ell,m}$, $\tau$, $\omega$)
7: \hspace{2em} end for
8: end for
9: InitCtrs($\tau'$, $\omega'$)
10: InitTopics($Z'$, $W'$, $M'$, $\tau'$, $\omega'$)
11: for $i = 1$ to $I_{\ell}$ do
12: \hspace{1em} for $m = 1$ to $M'$ do
13: \hspace{2em} $k \leftarrow z'_{m}$
14: \hspace{2em} \hspace{1em} $q, r \leftarrow \text{Adjacents}(Z', m)$
15: \hspace{2em} \hspace{1em} ReduceCtrs($q$, $k$, $r$, $w'_{m}$, $\tau'$, $\omega'$)
16: \hspace{2em} \hspace{1em} $k \leftarrow \text{Sampling}(q, k, r, w'_{m}, \tau' + \tau', \omega + \omega')$
17: \hspace{2em} \hspace{1em} $z'_{m} \leftarrow k$
18: \hspace{2em} \hspace{1em} IncreaseCtrs($q$, $k$, $r$, $w'_{m}$, $\tau'$, $\omega'$)
19: \hspace{2em} end for
20: end for
21: for $m = 1$ to $M_{n}$ do
22: \hspace{1em} $z'_{m} \leftarrow \text{GetMax}(q, r, w'_{m}, \tau' + \tau', \omega + \omega')$
23: end for
Algorithm 3 Auxiliary functions

Sampling\( (q, r, w_m, \tau, \omega) \)
1: \( p_0 \leftarrow 0 \)
2: \textbf{for} \( k = 1 \) \textbf{to} \( K \) \textbf{do}
3: \hspace{1em} \textbf{if} \( B = 1 \) \textbf{then} \hspace{2em} \text{▷ use activity bigrams}
4: \hspace{2em} \hspace{1em} \( p \leftarrow (14) \)
5: \hspace{1em} \textbf{else} \hspace{2em} \text{▷ use activity unigram}
6: \hspace{2em} \hspace{1em} \( p \leftarrow (11) \) \hspace{2em} \text{▷ use} \( \tau_{K+1,k} \) \text{ in place of} \( \tau_k \)
7: \hspace{2em} \textbf{end if}
8: \hspace{1em} \( p_k \leftarrow p_{k-1} + p \)
9: \textbf{end for}
10: \( p \leftarrow \text{generate random number in} \ (0, p_K) \)
11: \( k \leftarrow 1 \)
12: \textbf{while} \( p \geq p_k \) \textbf{do}
13: \hspace{1em} \( k \leftarrow k + 1 \)
14: \textbf{end while}
15: \textbf{return} \( k \)

GetMax\( (q, r, w_m, \tau, \omega) \)
16: \textbf{for} \( k = 1 \) \textbf{to} \( K \) \textbf{do}
17: \hspace{1em} \textbf{if} \( B = 1 \) \textbf{then} \hspace{2em} \text{▷ use activity bigrams}
18: \hspace{2em} \hspace{1em} \( p_k \leftarrow (14) \)
19: \hspace{1em} \textbf{else} \hspace{2em} \text{▷ use activity unigrams}
20: \hspace{2em} \hspace{1em} \( p_k \leftarrow (11) \) \hspace{2em} \text{▷ use} \( \tau_{K+1,k} \) \text{ in place of} \( \tau_k \)
21: \hspace{2em} \textbf{end if}
22: \textbf{end for}
23: \textbf{return} \( \arg\max_k (p_k) \)
Algorithm 4 auxiliary functions 2

\begin{algorithmic}
\State $z_{m} \leftarrow \text{generate random integer in } [1, K]$
\For {$m = 1$ to $M$}
\State $r_{m} \leftarrow \text{Adjacent}(\mathcal{Z}, m)$
\EndFor
\EndFor
\end{algorithmic}

\begin{algorithmic}
\State $z_{m} \leftarrow \text{generate random integer in } [1, K]$
\For {$m = 1$ to $M$}
\State $r_{m} \leftarrow \text{Adjacent}(\mathcal{Z}, m)$
\EndFor
\EndFor

Acknowledgements This work was supported by Japan Science and Technology Agency (JST) ‘Strategic Promotion of Innovation Research and Development’ Grant Number JPMJSV11011, and was partially based on results obtained from a project, JPNP20006, commissioned by the New Energy and Industrial Technology Development Organization (NEDO).

Declarations

Conflict of interest The authors declare no competing interests.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article’s Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article’s Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. Cook DJ, Krishnan N (2014) Mining the home environment. J Intell Inf Syst 43(3):503–519. https://doi.org/10.1007/s10844-014-0341-4
2. Amiribesheli M, Benmansour A, Bouchachia A (2015) A review of smart homes in healthcare. J Ambient Intell Hum Comput 6(4):495–517. https://doi.org/10.1007/s12652-015-0270-2
3. Leotta F, Mecella M, Sora D, Catacrì T (2019) Surveying human habit modeling and mining techniques in smart spaces. Future Internet 11(1):23. https://doi.org/10.3390/fin11010023
4. Saives J, Pianon C, Faraut G (2015) Activity discovery and detection of behavioral deviations of an inhabitant from binary sensors. IEEE Trans Autom Sci Eng 12(4):1211–1224. https://doi.org/10.1109/tase.2015.2471842
5. Shang C, Chang C-Y, Chen G, Zhao S, Chen H (2020) BIA: Behavior identification algorithm using unsupervised learning based on sensor data for home elderly. IEEE J Biomed Health Inform 24(6):1589–1600. https://doi.org/10.1109/jbhi.2019.2943391
6. Stucki RA, Urwyler P, Rampa L, Müri R, Mosimann UP, Nef T (2014) A web-based non-intrusive ambient system to measure and classify activities of daily living. J Med Internet Res 16(7):175. https://doi.org/10.2196/jmir.3465
7. Mizuno J, Saito D, Sadohara K, Nihei M, Ohnaka S, Suzurikawa J, Inoue T (2021) Effect of the information support robot on the daily activity of older people living alone in actual living environment. Int J Environ Res Public Health 18(5):2498. https://doi.org/10.3390/ijerp19052498
8. Cook DJ, Krishnan NC, Rashidi P (2013) Activity discovery and activity recognition: A new partnership. IEEE Transactions on Cybernetics 43(3):820–828. https://doi.org/10.1109/tsmcb.2012.2216873
9. Bishop CM (2006) Pattern Recognition and Machine Learning. Springer
10. Blei DM, Ng AY, Jordan MI (2003) Latent Dirichlet allocation. J Mach Learn Res 3:993–1033
11. Teh YW, Jordan MI, Beal MJ, Blei DM (2006) Hierarchical Dirichlet processes. J Am Stat Assoc 101(476):1566–1581
12. Huyt T, Fritz M, Schiele B (2008) Discovery of activity patterns using topic models. In: Proc of Internat Conf on Ubiquitous Computing, pp 10–19
13. Niebles JC, Wang H, Fei-Fei L (2008) Unsupervised learning of human action categories using spatial-temporal words. Int J Comput Vis 79(3):299–318
14. Sun FT, Yeh YT, Cheng HT, Kuo C, Griss M (2014) Nonparametric discovery of human routines from sensor data. In: Proc of Internat Conf on Pervasive Computing and Communications, pp 11–19
15. Tskai M-J, Wu, C-L, Pradhan SK, Xie Y, Li T-Y, Fu L-C, Zeng, Y-C (2016) Context-aware activity prediction using human behavior pattern in real smart home environments. In: Proc of Internat Conf on Automation Science and Engineering, pp 168–173. IEEE
16. Nigam K, McCallum AK, Thrun S, Mitchell T (2000) Text classification from labeled and unlabeled documents using EM. Mach Learn 39(2):103–134
17. Jianhua Y, Wang J (2014) A Dirichlet multinomial mixture model-based approach for short text clustering. In: Proc of Internat Conf on Knowledge Discovery and Data Mining, pp 233–242
18. Andriani P, Mckelvey B (2007) Beyond Gaussian averages: redirecting international business and management research toward extreme events and power laws. J Int Bus Stud 38(7):1212–1230. https://doi.org/10.1057/palgrave.jibs.8400324
van Kasteren TL, Englebienne G, Kröse BJ (2011) Human activity recognition from wireless sensor network data: benchmark and software. pp 165–186. Activity Recognition in Pervasive Intelligent Environments, Atlantis Press

Cook D (2012) Learning setting-generalized activity models for smart spaces. IEEE Intell Syst 27(1):32–38. https://doi.org/10.1109/mis.2010.112

Leotta F, Mecella M, Sora D (2020) Visual process maps: A visualization tool for discovering habits in smart homes. J Amb Intell Hum Comput 11(5):1997–2025. https://doi.org/10.1007/s12652-019-01211-7

Mizuno J, Sadohara K, Nihei M, Onaka S, Nishiura Y, Inoue T (2021) The application of an information support robot to reduce agitation in an older adult with Alzheimer’s disease living alone in a community dwelling: a case study. Hong Kong J Occup Ther 34(1):50–59. https://doi.org/10.1177/15691861211005059

Dimitrov T, Pauli J, Naroska E (2010) Unsupervised recognition of ADLs. In: Proc of Hellenic Conf on Artificial Intelligence. Springer, pp 71–80

Hoque E, Stankovic J (2012) AALO: Activity recognition in smart homes using active learning in the presence of overlapped activities. In: Proc of Internat Conf on Pervasive Computing Technologies for Healthcare, pp 139–146

Agrawal R, Srikant R et al (1994) Fast algorithms for mining association rules. In: Proc of Conf on Very Large Data Bases, vol 1215, pp 487–499

Han J-W, Pei J, Yan X-F (2004) From sequential pattern mining to structured pattern mining: a pattern-growth approach. J Comp Sci Technol 19(3):257–279. https://doi.org/10.1007/BF02944897

Barger TS, Brown DE, Alwan M (2004) Health-status monitoring through analysis of behavioral patterns. IEEE Transactions on Systems, Man, and Cybernetics part A: Systems and Humans 35(1):22–27

Madsen RE, Kauchak D, Elkan C (2005) Modeling word burstiness using the Dirichlet distribution. In: Proc of Internat Conf on Machine Learning, pp 545–552

Johnson NL, Kotz S, Balakrishnan N (1997) Discrete Multivariate Distributions. John Wiley & Sons, Inc.

Rabiner LR (1989) A tutorial on hidden Markov models and selected application in speech recognition. Proceedings of the IEEE 77(2):257–286

Liang S, Yilmaz E, Kanoulas E (2016) Dynamic clustering of streaming short documents. In: Proc of Internat Conf on Knowledge Discovery and Data Mining, pp 995–1004

Vinh NX, Epps J, Bailey J (2010) Information theoretic measures for clusterings comparison: variants, properties, normalization and correction for chance. J Mach Learn Res 11:2837–2854

Hubert L, Arabie P (1985) Comparing partitions. J Classif 2(1):193–218

Rand WM (1971) Objective criteria for the evaluation of clustering methods. J Am Stat Assoc 66(336):846–850

Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E (2011) Scikit-learn: Machine learning in Python. J Mach Learn Res 12:2825–2830

Řehůřek R, Sojka P (2010) Software framework for topic modeling with large corpora. In: Proc of workshop on new challenges for NLP frameworks, pp 45–50