An Incremental Learning Method Based on Formal Concept Analysis for Pattern Recognition in Nonstationary Sensor-Based Smart Environments

Jianguo Hao, Abdenour Bouzouane, Sébastien Gaboury*

LIARA Laboratory, Université du Québec à Chicoutimi
555 Boulevard de l'Université, Chicoutimi, G7H 2B1, Canada

Abstract

Smart homes are typical nonstationary environments that keep generating new data. Ideally, predictive models learned from historical data should automatically adapt to new data without retraining. However, for many data mining algorithms, adding new data with new features to an existing model means we need to retrain the entire model. Compared with static ones, the models having incremental learning mechanisms are more suitable for handling streaming data in the aspect of self-adaptation. Therefore, we propose an incremental algorithm based on the formal concept analysis (FCA) for mining sequential patterns and apply it to recognize various human behavioral patterns in nonstationary sensor-based smart homes. It can automatically adapt to new training data with new classes or features to enhance the currently built model and have competitive recognition results compared to other classical graphical models.

*Corresponding author
Email addresses: jianguo.hao1@uqac.ca (Jianguo Hao), abdenour.bouzouane@uqac.ca (Abdenour Bouzouane), sebastien.gaboury@uqac.ca (Sébastien Gaboury*)

Keywords: Pattern Recognition, Smart Homes, Formal Concept Analysis, Incremental Learning

1. Introduction

Currently, machine learning or data mining methods have been widely applied to our lives. Most recent real-world applications built by these methods usually assume that there is no change in probability distribution between training data and unseen out-of-sample data. However, many practical problems cannot satisfy this assumption. Their predictions are also biased. An environment that generates data with unstable probability distributions over time is a nonstationary environment. Learning in such an environment (or learning concept drift [1, 2]) has been investigated by various approaches, such as transfer learning [3], adaptive learning [4, 5], and incremental learning [5, 6]. Their common idea is to adapt trained models to new unseen data with imbalanced classes or with dynamic changes in distribution.

Activity recognition in sensor-based smart environments is a challenging study in the field of ambient intelligence [7]. By using a dynamic wireless sensor network, massive data describing environmental changes caused by human activities can be captured in real-time. In such nonstationary environments, newly collected data containing new target activities and sensor events can be used to improve the performance of current systems. Thus, there are higher requirements and expectations on how to update a built model with less training costs.

While most activity recognition systems train their models from histori-

cal data, the collected behavioral patterns cannot cover all possibilities. This is because a resident may perform the same activities in different ways. The systems should learn additional information from the new training data to improve accuracy and robustness. Sometimes, the sensor layout of a smart environment can be extended with new sensors or new target activities. We hope that the systems can automatically adapt to these changes and update the trained model only with new data having new features to avoid retraining.

In terms of the integration of new data, the scalability of an activity recognition model is one of the most important requirements of the sensor-based smart environments. This is because smart environments are usually nonstationary, which constantly consider and introduce new situations. The performance of a model often becomes less accurate as time passes, because the statistical properties or probability distributions may change over time in unforeseen ways [5, 8]. Therefore, the recognition model needs to constantly update itself to accommodate these new changes. Furthermore, when the current layout of the smart environment is not sufficient to identify all the target activities, new sensors can be deployed to enhance the ability to distinguish between misclassified activities.

However, non-incremental learning methods are usually static, which means that they first import all available data for training, and then use the immutable constructed models for prediction, classification or pattern recognition. When non-incremental learning models want to improve their performance with new training data, they must be rebuilt in most cases to accommodate new training data or introduce new features. However, as

the training data increases, the time spent on reconstruction increases at the same time. Without an effective solution, frequent and time-consuming model training is intolerable for most smart environment applications.

In this paper, we introduce an incremental learning method to improve current activity recognition models based on the formal concept analysis (FCA) [9, 10, 11, 12]. Unlike most FCA incremental lattice construction algorithms that only update FCA-based models by data with fixed features, the newly proposed one enables the system to integrate new data with new features into current FCA-based models and satisfies the requirement of scalability.

The remainder of the paper is organized as follows. Section 2 outlines related work using incremental learning to solve activity recognition problems in sensor-based smart environments. Section 3 describes the preliminaries about formal concept analysis and our proposed incremental lattice construction algorithm. Section 4 compares the performance of the proposed algorithm with the others under the same benchmark dataset. The conclusion is reported in Section 5.

2. Related Work

Considering the complexity, flexibility, and variability of the situations when recognizing human behavioral patterns in smart environments, different methods and architectures are proposed by scientific communities. Their common practice is to make appropriate changes based on classic algorithms such as decision tree, random forests, naive Bayes and neural networks, etc.

For instance, Lu et al. [13] proposed a Hybrid User-assisted Incremental Model Adaptation (HUIMA) to reconfigure previously learned activity models within a dynamic environment. HUIMA consists of an automatic mechanism for simplifying the unseen data annotation task, and an enhanced Dynamic Bayesian Network model for incrementally updating the models by new annotated data. However, the correctness of data annotation cannot be always guaranteed. Another data-annotation wizard with human interventions was used in case of ambiguity. However, it will decrease the capability of self-adaptation of the model.

Wang et al. [14] combined probabilistic neural networks (PNN) and an adjustable fuzzy clustering algorithm (AFC) to build an incremental learning method for sensor-based human activity recognition. Their process of adding or removing an activity is almost independent of the pattern neurons of other activities. They tested their method with their own dataset. However, the generalization capability of the proposed method was limited by their subject-independent training.

Hu et al. [15] proposed an incremental growing mechanism of the decision tree and a novel splitting strategy to construct Class Incremental Random Forests (CIRF). Their solution can tackle the dynamic changes in activity recognition. However, the CIRF algorithm required maintaining large-scale training samples all the time.

ID5R [16] is an incremental algorithm for inducing decision trees equivalent to Quinlan's non-incremental ID3 algorithm [17]. It allows the induction process to learning tasks in which training data are presented serially. Because the ID5R algorithm does not support to handle numeric variables,

multi-class classification tasks, or missing values, an extension of ID5R which incrementally augments leaf nodes and allows them to be multi-labeled is proposed in the work of Prossegger and Bouchachia [18]. A loosely-coupled Hierarchical Dynamic Bayesian Network (HDBN) is proposed by Alam et al. [19] to exploit the spatio-temporal relationships across the activities. Their method was evaluated using their own dataset. However, a state space pruning should be performed before employing the model for complex activity recognition.

Considering heterogeneous concept drift, Losing et al. [20] proposed a self-adjusting memory model for the k-nearest neighbors (KNN) algorithm. They realized a non-parametric KNN constructing two different memories: the short-term memory (STM) containing data of the current concept and the long-term memory (LTM) maintaining knowledge of past concepts. The conflicting information between the two memories is removed and the rest is compressed.

As a thriving field with many successful applications, deep learning has been introduced to classify human activities. For example, Deepika et al. [21] proposed a recurrent neural network model based on the Long Short-Term Memory (LSTM) classifier for activity recognition. However, adding a new ability to neural networks often results in so-called "catastrophic forgetting" that neural networks completely and abruptly forget previously learned information upon learning new information. At most of the time, retraining the entire model is necessary to preserve performance. Alternative solutions may involve supervised fine-tuning of networks or transfer learning [22].

At last, the survey conducted by Ditzler et al. [23] roughly divided algorithms learning in nonstationary environments to active and passive approaches. Once a change in the data distribution is detected, the active approaches discard the obsolete knowledge and adapt to the new environments, while the passive ones continuously update the model over time. From this point of view, our proposed algorithm can be classified as the active one. The adaptation mechanism is activated if and only if new data will change the structure of the predictive model, which means that the new patterns are unseen for training before.

In brief, the incremental designs of most of the previous studies are limited by their algorithms, without considering the complicated situations and frequent layout updates in smart environments. Some of them only take into account for incremental training and focus only on learning new data with stable classes and features. It is unrealistic in the nonstationary smart environments. It is also difficult to apply them directly to solve complex activity recognition, such as composite and multi-resident scenarios [10, 12]. Incremental algorithms should separately identify the parts of the original model that need to be modified and the ones that do not need to be modified. In addition, to ensure correctness, an updated model built by an incremental algorithm should have the same or similar performance as other non-incremental ones.

3. FCA-based Incremental Activity Recognition

Formal concept analysis (FCA) is an efficient knowledge-driven solution to represent and discover knowledge from heterogeneous data. It is widely

used in various domains like knowledge discovery [24], ontology learning [25] and information retrieval [26]. Similar to association rule learning, its typical utilization is to discover interesting relations (such as frequent itemsets) between variables in large databases.

FCA-based model is a kind of inductive learning methods. Without providing any preamble about how the inferences work, rules are summarized from specific examples or cases. Its concept learning process is more transparent than black box models.

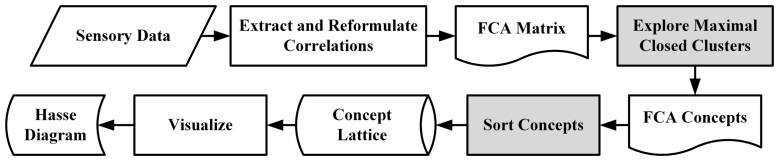

Figure 1: Overview procedure of FCA learning

Like the other data mining methods, the process of obtaining formal lattice from raw data is called "learning" or "training". Figure. 1 depicts the overview procedure of FCA learning. It represents how to construct the Hasse diagram, a visual knowledge base, by using sequential sensor data. First of all, binary relations between activities and sensor data are extracted from captured sensor data. The extracted relations should be represented in an FCA binary matrix. It is usually implemented by an ad-hoc script on the basis of the original format of captured sequences. Then, specific algorithms of lattice construction [27, 28] will explore all the maximal closed clusters through FCA matrix, and sort them by their partial orders. They are the key processes in the FCA modeling and emphasized in gray in the figure.

3.1. Formal Concept Analysis

Formal concept analysis (FCA) is a mathematical theory based on the conceptual hierarchy [29]. The correlations between defined target classes and selected features can be unified as homogeneous binary relations. By using a specific lattice construction algorithm, FCA firstly clusters similar classes (i.e. activities to be recognized) sharing the same ontological features (i.e. sensor data), then encapsulates the discovered itemsets as inferences, and finally orders them for fast retrieval. Therefore, based on such a key-value tuple structure of itemsets, the FCA-based model can infer the ongoing activities of residents according to partially observed sensor data.

To analyze temporal and sequential patterns, first of all, correlations between target classes and features have to be extracted from data and reformulated into a specific data structure named formal context.

Formal context $\mathbb{K}(G, M, I)$ is triplet structure consisting of two disjoint sets G and M, and their Cartesian product set $I = G \times M$. It can be represented and visualized by a $|G| \times |M|$ matrix. The elements in G are formally called *objects*, which represent coarse-grained target classes (i.e. activities to be recognized). The ones in M are called *attributes*, which represent fine-grained observable features (i.e. captured sensor data). If $g \in G$ is correlated with $m \in M$, the correlation could be written as gIm [29].

To extract and reformulate correlations from sensor data, if a sensor event m_j appears in the sequence describing an activity g_i, it means $g_i I m_j$, then a cross will be filled in the row g_i and column m_j in the binary matrix. Fig. 2 shows a concrete example. It is generated from a simplified version

Simplified CASAS Activities [30]		m_1: D07	m_2: D11	m_3: D12	m_4: D13	m_5: D14	m_6: I04	m_7: M04	m_8: M06	m_9: M07	m_{10}: M09	m_{11}: M13	m_{12}: M17	m_{13}: M23
Fill medication dispenser	g_1	×					×						×	
Hang up clothes	g_2			×						×	×			×
Move furniture	g_3							×						×
Read magazine	g_4								×	×	×			
Water plants	g_5								×	×				
Sweep floor	g_6			×					×	×	×	×	×	×
Play checkers	g_7		×										×	
Prepare dinner	g_8										×	×		
Set table	g_9			×							×			×
Read magazine	g_{10}	×											×	
Pay bills	g_{11}					×							×	
Pack picnic food	g_{12}								×	×				
Retrieve dishes	g_{13}				×				×	×	×	×		
Retrieve dishes	$g_{13'}$								×	×	×	×		
Pack picnic supplies	g_{14}	×									×		×	
Pack and bring supplies	g_{15}			×		×				×	×		×	×

Figure 2: Binary matrix representing the correlations between activities g_i and sensor data m_j.

of the CASAS benchmark dataset [30]. There are fifteen activities described by thirteen non-intrusive sensors passively capturing human behaviors in a smart apartment. It is worth mentioning that g_{13} and $g_{13'}$ are two different patterns to implement the same activity "Retrieve dishes".

3.1.1. Similarity Maximization by Concept-Forming Operations

In the data clustering technique of data mining, similarity metrics are essential to generate clusters [31]. Similarly, FCA also has its own metrics to maximize similarity. Items in the same itemset have high similarity because they share the same ontological features.

A pair of FCA similarity metrics, so-called the *concept-forming operators*

is introduced. For a subset $G_1 \subseteq G$, we define

$$G_1' := \{m \in M \mid \text{for all } g \in G_1,\ gIm\} \tag{1}$$

as a closure operation to find out all the sensor data $G_1' \subseteq M$ that are jointly shared by all the activities in G_1. Likewise, for $M_1 \subseteq M$, we define

$$M_1' := \{g \in G \mid \text{for all } m \in M_1,\ gIm\} \tag{2}$$

as another one to find out all the activities $M_1' \subseteq G$ that jointly share the same sensor data M_1 [29].

For instance, as shown in Fig. 2, $\{m_3 m_{10}\}' = \{g_2 g_6 g_9 g_{15}\}$ and $\{g_2 g_6 g_9 g_{15}\}' = \{m_3 m_{10} m_{13}\}$. Using the two operators at the same time, FCA can generate stable key-value itemsets named formal concepts from a binary matrix, which cluster similar activities sharing the same sensor data.

3.1.2. Cluster Representation by Formal Concept

In order to infer ongoing activities from given observable sensor data, FCA clusters similar activities according to different centroids, and encapsulate these inferences in the itemsets. Moreover, to ensure the reliability of inferences, FCA only uses the itemsets that simultaneously satisfy the two concept-forming operations. The satisfied itemsets are so-called formal concepts.

Formal concept $c := (G_1, M_1)$ is a closure under the limitation of the concept-forming operations where $(G_1')' = (M_1)' = G_1$. G_1 is called the *extent* of c, written as $\text{ext}(c)$. Likewise, M_1 is called the *intent* of c, written

as int(c) [29], which is also treated as the centroid of a cluster [31]. The space of all the formal concepts is denoted by $\mathfrak{B}(G, M, I)$.

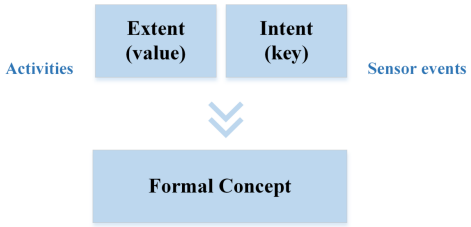

Figure 3: Key-value structure of formal concept

As shown in Fig. 3, each formal concept consists of two parts. An extent is the value of the key-value structure, it indicates the inferred ongoing activities given the observed key from an intent. A concept c clusters similar activities ext(c) based on their shared sensor data in the int(c). Furthermore, if an observed sequence $\alpha \subset \text{int}(c)$, the elements in the ext(c) indicate the recognized ongoing activities given α.

$$\{\ \underbrace{g_2 g_6 g_9 g_{15}}_{\text{possible ongoing activities}}\ ,\ \overbrace{m_3 m_{10} m_{13}}^{\text{current observed data}}\ \}$$

In the example above, $\text{ext}(c) = \{g_2 g_6 g_9 g_{15}\}$ and $\text{int}(c) = \{m_3 m_{10} m_{13}\}$. As described in Section 3.1.1, the sensor data in int(c) exist in all the patterns of activities in ext(c). Therefore, if current observed data are m_3, m_{10} and m_{13}, the scope of possible ongoing activities should be g_2, g_6, g_9 or g_{15}.

3.1.3. Cluster Indexation by Formal Concept Lattice

After the generation of concepts clustering similar objects by different centroids (i.e. feature variables), FCA provides an automatic solution to

index all the discovered concepts according to a mathematical order. The objective is to efficiently manage and construct a graph-based knowledge base to fast retrieve inferences.

Formal concept lattice $\underline{\mathfrak{B}}$ is an ordered version of $\mathfrak{B}(G, M, I)$. All the concepts in $\mathfrak{B}(G, M, I)$ are ordered by a predefined partial order $\preceq$ indicating hierarchical relations between two concepts [29].

Suppose (G_1, M_1) and (G_2, M_2) are two concepts, (G_1, M_1) is called the *subconcept* of (G_2, M_2) if either $G_1 \subseteq G_2$ or $M_2 \subseteq M_1$, written as $(G_1, M_1) \preceq (G_2, M_2)$. The symbol $\preceq$ is named as the *hierarchical order*. Meanwhile, (G_2, M_2) is the *superconcept* of (G_1, M_1). It is worth pointing out that the subconcept and the superconcept of a concept are not unique in $\mathfrak{B}(G, M, I)$ due to the existing transitive relation.

Lattice construction is a process that first discovers all the concepts of context $\mathbb{K}$ by the concept-forming operations, and then orders them simultaneously. In recent decades, great efforts have been devoted to efficiently construct an FCA lattice [27, 32, 33, 34, 35, 36].

For instance, three concepts, $\{g_6 g_8 g_{13} g_{13'}, m_{10} m_{11}\}$, $\{g_6 g_{13} g_{13'}, m_8 m_9 m_{10} m_{11}\}$ and $\{g_{13}, m_4 m_8 m_9 m_{10} m_{11}\}$, are discovered from the matrix in Fig. 2. As shown in Equation (3), the last two concepts are the superconcepts of the first one.

$$\begin{aligned} \{g_{13}, m_4 m_8 m_9 m_{10} m_{11}\} &\preceq \{g_6 g_{13} g_{13'}, m_8 m_9 m_{10} m_{11}\} \\ &\preceq \{g_6 g_8 g_{13} g_{13'}, m_{10} m_{11}\} \end{aligned} \tag{3}$$

The relations among concepts having different centroids are established

and linked by the hierarchical order. Thus, a lattice $\underline{\mathfrak{B}}$ could be visualized as a graphical model.

3.1.4. Knowledge Visualization by Hasse Diagram

In mathematics, a finite partially ordered set could be depicted by a Hasse diagram. In our case, a lattice $\underline{\mathfrak{B}}$ could also be visualized as an undirected graph, such as the one shown in Fig. 4. Each node refers to a discovered concept, and partial orders are represented by edges, which are also named Galois connections [29].

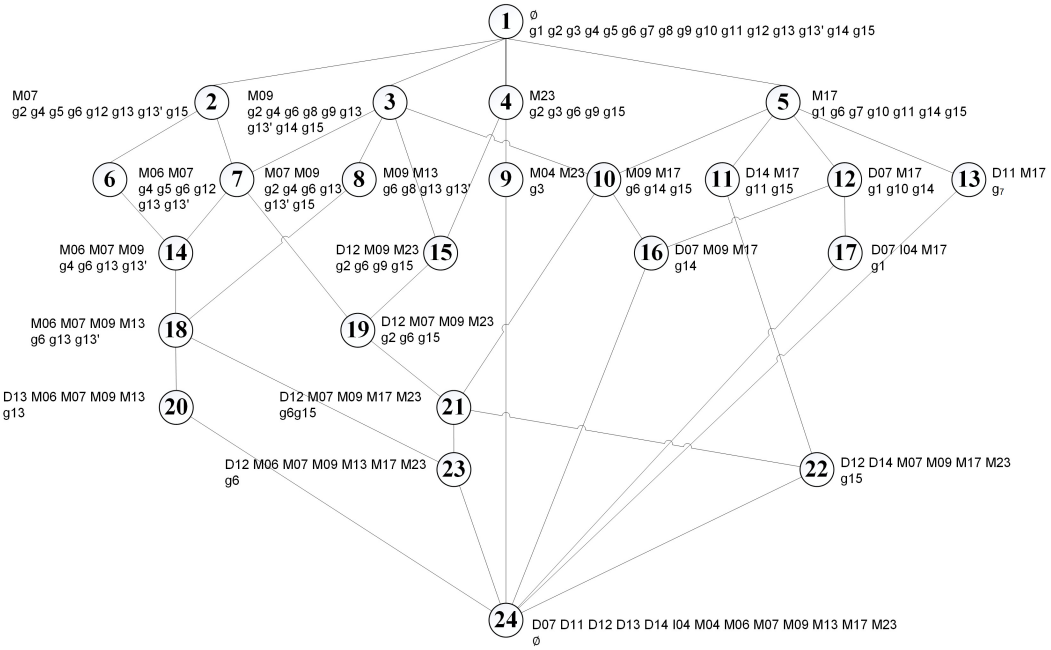

Figure 4: Hasse diagram of Matrix in Fig. 2

There are two special nodes in a Hasse diagram: the topmost one $\{G, \varnothing\}$ named *Supremum* and the lowermost one $\{\varnothing, M\}$ named *Infimum*. Once the Hasse diagram is built, the next step is to use efficient algorithms to retrieve

knowledge encapsulated inside concepts from the graphical structure.

For example, as can be seen in Fig. 4, concepts are organized by different levels. From the top to down in a Hasse diagram, the scope of possible activities shrinks when more sensor data are observed.

3.1.5. Applications of FCA-based Models

For the principle of FCA inference, the scope of possible activities shrinks while more and more sensor data are observed. To locate the most relevant concept based on the observed data, we need an efficient inference retrieval algorithm. For this reason, we proposed a concept retrieval algorithm called *Half-Duplex Search (HDS) algorithm* [9]. It can be treated as an algorithm using an observed key to search for the key-value formal concept with the best corresponding value. It is the basic algorithm used in our previous research [10, 11, 12]. It consists of two parts: the top-down search of HDS algorithm can fast retrieve a concept with a value satisfying the key, and the bottom up search of HDS algorithm can further find and ensure the optimal one.

An overview of the FCA-based activity recognition framework is given in Fig. 5. The framework is divided into two individual modules. One module focuses on incremental learning, and the other focuses on recognizing activities in smart environments. In the recognition module, there are several ad-hoc inference retrieval strategies for different scenarios. The number of residents directly affects the choice of these strategies. For one resident performing simple activities in smart homes, we proposed the basic searching strategy to recognize activities without complex patterns [9]. For more

complex situations such as the sequential activities performed in a continuous and sequential mode without interweaving execution, or the interleaved ones performed with pauses, or the ones concurrent performed together in a period of time, the search strategy pays more attention to the analysis of the characteristics of each mode, and then recognize each case [10]. In [11], we proposed another solution for detecting cognitive errors. We analyzed the behavioral patterns of people suffering from cognitive impairment and defined six common cognitive errors and their typical abnormal behavioral patterns.

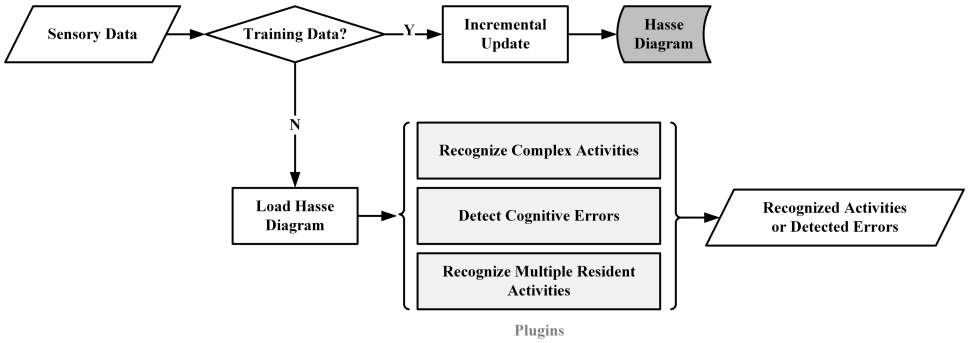

Figure 5: Recognizing activities in smart environments

3.2. Incremental Algorithm for Constructing Concept Lattice

As mentioned before, different lattice construction algorithms have quite different performances in terms of memory consumption and processing time. Our incremental method is based on an algorithm proposed by Valtchev and Missaoui [28]. This algorithm is an efficient lattice building approach which is more effective than many classic incremental algorithms such as Godin et al. [32, 35], Norris [36] and other batch ones, such as Bordat [33],

Ganter [34] and Fast [27]. It investigates the incremental updating of the constructed lattice by a set of previously unseen labeled sequences of sensor data. Its basic idea is to recognize the lattice parts requiring restructuring and to carry out the reconstructing at a minimal cost. Thus, two categories of formal concepts must be identified: those which changed their extent and those which remain the same. The latter is further qualified as if it gives rise to a new concept. However, its implementation [32] does not consider updating new data with new features. The implementation assumes that the feature space is stable and all the new training items only involve those existing features. For our new extension, we incrementally update the feature space when a new labeled training item is available. Therefore, our new approach can analyze the subsumption relations between concepts in considering new features on the fly. It can handle the new training items that have never seen before with unseen features.

Thus, in Algorithm 1, we illustrate the extension of incrementally updating a constructed lattice. The input data D is a collection of labeled sequences of sensor data that arrive gradually. To achieve an incremental manner, as an extension, the space of features is incrementally updated (lines 2-3). The algorithm initializes a lattice if it does not exist before (Lines 4-8). For each item in the new training dataset, an iteration of the lattice verifies whether the iterated concept should be updated, created or ignored (lines 9-26). We optimize and simplify the logic of an internal function called *minAdjacentParent*, described in Algorithm 2. The updated lattice $\underline{\mathfrak{B}}^+$ normally exists in the memory and can be serialized in a database or in a disk file. Compared with batch algorithms, our proposed algorithm can

Algorithm 1: Extended Valtchev Algorithm

Data: A constructed lattice $\underline{\mathfrak{B}}$, a new training dataset $\mathcal{D} = (\mathcal{X}, \mathcal{Y}) = \{(x^{(0)}, y^{(0)}), (x^{(1)}, y^{(1)}), ..., (x^{(m)}, y^{(m)})\}$, the space of new features $\mathcal{M}$.

Result: Updated lattice $\underline{\mathfrak{B}}^{+}$.

```
1  begin
2      if M ∩ X ≠ X then
3          M = M ∪ X
4      if 𝔅 = ∅ then
5          supremum = newConcept(y^(0), x^(0))
6          infimum = newConcept(∅, M)
7          createLink(supremum, infimum)
8      modified= ∅
9      foreach (x^(i), y^(i)) ∈ D do
10         foreach c ∈ 𝔅 do
11             if int(c) ⊆ y^(i) then
12                 ext(c) = ext(c) ∪ y^(i), mark it as modified
13             else
14                 n = newConcept(ext(c) ∪ y^(i), int(c) ∩ x^(i))
15                 m = minAdjacentParent(n,c)
16                 createLink(m, n)
17                 if ext(m) has been modified then
18                     dropLink(m, c)
19                 end
20                 createLink(n, c)
21                 if c==supremum then
22                     supremum=n
23                 end
24             end
25         end
26     end
27 end
```

Algorithm 2: Discover Adjacent Super Concept

Function *minAdjacentParent(m,c)*

Data: Concept m to compare, current concept c

Result: Adjacent parent of c having minimal superset of ext(m)

```
1      parents = sorted(parents(c))
2      foreach p ∈ parents do
3          if ext(m) == ext(m) ∩ ext(p) then
4              return p
5          end
6      end
```

update current model by new training data and reduce the time of lattice construction by avoiding retraining. It is quite suitable for data mining in nonstationary environments. However, because it is a knowledge-based approach analyzing ontological relations among entities, it is difficult to quantify uncertainty by itself.

4. Experiments and Results

The experiment is carried out on a desktop with an Intel Core i7-6700 CPU and 8GB of RAM running Windows 10. The benchmark dataset used in the experiment is the Kyoto-4 dataset [1]. It contains sensor data collected by the smart apartment testbed of the CASAS laboratory of Washington State University. The data represents two residents performing fifteen activities in different ways in the apartment at the same time.

Although different activity recognition datasets may have different formats, there are several essential data fields: timestamps, sensor id, sensor value, and a label indicating the ground truth. The process about how to convert raw sequential data to FCA matrix is given in Fig. 6. Once the fields representing activity labels, sensor ids, and their values are determined, they can be extracted and temporarily reserved for the following operations. It is worth point out that a sensor event is the combination of sensor id and its binary or nominal value. For continuous numeric values, they first should be transformed into limited nominal ones.

Based on the space of indexed sensor data, we map the sequence into

[1] http://ailab.wsu.edu/casas/datasets/

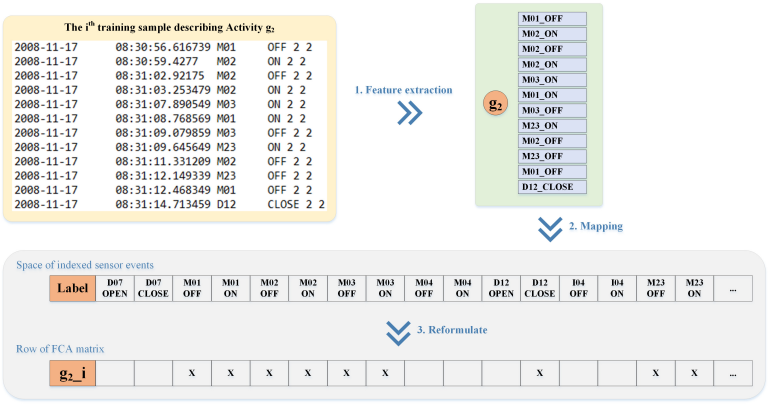

Figure 6: An example of the feature extraction and reformulation from a sequence of sensor data to generate a row of FCA matrix

a row of FCA matrix. If a sensor event exists in the sequence describing an activity, there is a cross in the corresponding element in the matrix. After that, we explore and construct an FCA lattice by using FCA matrix. There are 270 target classes with 73 sensor data in one of the leave-one-out cross-validation training data of CASAS dataset. Thus, the binary matrix consists of 270 rows and 73 columns, and it constructs a lattice with 29,118 formal concepts. It is worth mentioning that both incremental and non-incremental lattice construction algorithms using the same training dataset will produce the totally same lattice without any difference. Thus, their recognition results are also the same, because the lattice construction and pattern recognition are two independent modules.

4.1. Comparisons about Lattice Construction

We compare our results with both non-incremental and incremental algorithms published in [27, 32, 33, 34], However, Godin and Norris algorithms

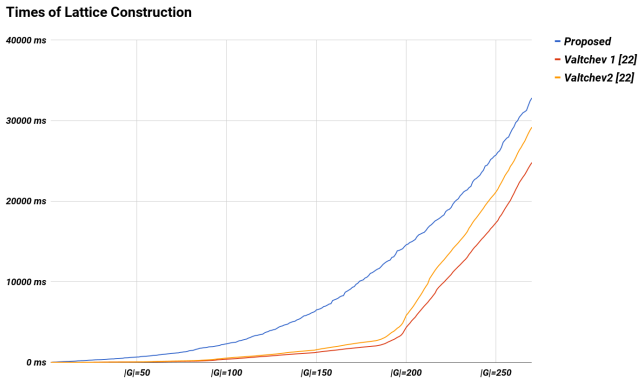

Figure 7: Time of lattice construction

[32] cannot handle the training data having multi-level inheritance [2] [10]. Thus, Fig. 7 presents the time of lattice construction of three incremental algorithms at different stages. The time consumption of lattice construction increases while the amount of target classes ($|G|$) grows. However, almost all the non-incremental algorithms load and generate the lattice by learning on the entire training dataset at once. Once a lattice is constructed, it cannot be modified by any new training data. Thus, these algorithms do not update the lattice one by one. Compared with the other two incremental algorithms, ours sacrifices the efficiency in speed in exchange for the functional expansion to incrementally update new data with new features.

The time intervals of all the iterations are shown in Fig. 8. As shown in this figure, in the beginning, the time of each update tends to be stable, and later, the time intervals begin to fluctuate. This is because when the lattice construction has reached a certain dimension, the complexity of updating

[2]For two activities g_1 and g_2, their features having $g_1' \subseteq g_2'$ or $g_2' \subseteq g_1'$

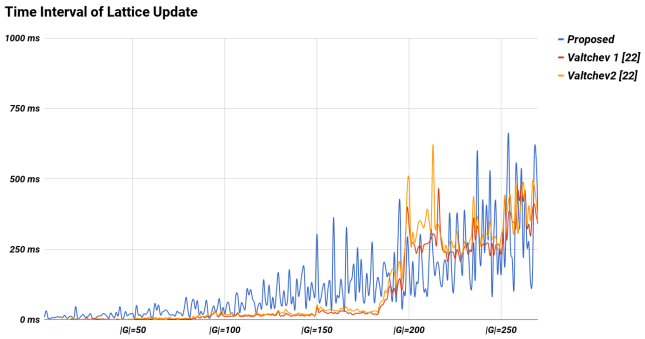

Figure 8: Time interval for each incremental update

Algorithm	Type	Time for Lattice Construction
Bordat [33]	Batch	49.625s
Ganter [34]	Batch	180.331s
Fast [27]	Batch	9216.659s
Valtchev 1 [32]	Incremental	25.449s
Valtchev 2 [32]	Incremental	29.598s
Proposed	Incremental	33.664s

Table 1: Comparison of results of lattice construction by different algorithms

becomes uncertain, largely depending on the relationship between the new data and the old one.

In Table 1, a comparison of different lattice construction algorithms including non-incremental and incremental ones is given. As shown, the incremental algorithms construct faster than the non-incremental ones. This provides us with a powerful practical basis for using incremental algorithms.

Our extension has paid an extra cost in speed. However, instead of using all the data to retrain the entire model, new features like new sensor events and new activities are allowed to incrementally update constructed lattice.

Activity ID	Activity	CACE [19]	FCA
1	Fill medication dispenser	0.932	1.0
2	Hang up clothes	0.965	1.0
3	Move furniture	0.973	1.0
4	Read magazine	0.607	1.0
5	Water plants	0.593	0.672
6	Sweep floor	0.955	1.0
7	Play checkers	0.945	1.0
8	Prepare dinner	0.976	0.958
9	Set table	0.943	1.0
10	Read magazine	0.923	1.0
11	Pay bills	0.98	1.0
12	Pack picnic food	0.955	0.724
13	Retrieve dishes	0.979	0.978
14	Pack picnic supplies	0.558	0.978
15	Pack and bring supplies	0.615	0.978
	Overall Precision	0.965	0.989
	Overall Recall	0.945	0.948
	Overall F1-score	0.936	0.954

Table 2: Comparison of F1-score between two models

4.2. Comparisons about Activity Recognition

First of all, we compare our model with another one called CACE [19], and show their results in Table. 2. CACE is a constraint and correlation mining engine based on a loosely-coupled hierarchical dynamic Bayesian network. It learns behaviorally-driven context correlations in the form of association rules from data. Then, we also compare our model with derived HMM methods described in [37] and the results are shown in Table. 3. In this comparison, we use the same leave-one-out method to evaluate the performance.

Based on the results shown in Table. 2, we can see that most of the recognition results are excellent except for two activities: water plants (activity g_5) and picnic food (activity g_{12}). The reason has been indicated in

Approach	Accuracy	Precision	Recall	F-measure
CL-HMM [37]	91.47±7.5	91.68±6.1	92.12±6.42	91.89±6.17
LHMM [37]	93.27±6.21	93.58±5.41	92.68±6.18	93.1±5.62
FCA	95.98±1.27	97.97±0.93	97.94±0.41	97.75±0.68

Table 3: Performance of recognizing multi-resident activities using both non-incremental and incremental methods

[30] that the activities with insufficient sensor data are difficult to be distinguished from other activities and lead to lower recognition results. In the view of FCA models, the distinguishable ability of a sensor is negatively correlated with the number of shared activity.

```
M08_ON 5_new
M07_OFF 5_new
M07_ON 5_new
M08_OFF 5_new
M07_OFF 5_new
iCAN_ON 5_new
M07_OFF 5_new
M07_ON 5_new
iCAN_OFF 5_new
```

(a) new training data describing activity g_5 with new features iCAN_OFF and iCAN_ON

```
M06_ON 12_new
M07_ON 12_new
M08_ON 12_new
M06_OFF 12_new
M07_OFF 12_new
M08_OFF 12_new
M06_ON 12_new
iCupbord_ON 12_new
iCupbord_OFF 12_new
```

(b) new training data describing activity g_{12} with new features iCupbord_OFF and iCupbord_ON

Figure 9: Constructed lattice enhanced by new data with new features

To solve this problem, we use new training data with new features to help to distinguish the misclassified activities g_5 and g_{12}. We find that activity g_5 must interact with the watering can that is located in the hallway closet, but activity g_{12} does not. Thus, we can add an RFID tag or other sensors to monitor the moving states (e.g. *iCAN_ON* and *iCAN_OFF*) of the watering can. Likewise, for activity g_{12}, food has to be gathered from the kitchen cupboard. Thus, we can monitor the open/close states (e.g. *iCupbord_ON*, *iCupbord_OFF*) of the kitchen cupboard. In the experiment, simulative sequences with four new sensor events, *iCupbord_ON*, *iCup-*

bord_OFF, *iCAN_ON* and *iCAN_OFF*, are incrementally introduced into the constructed lattice for the enhancement of knowledge base (see Fig. 9).

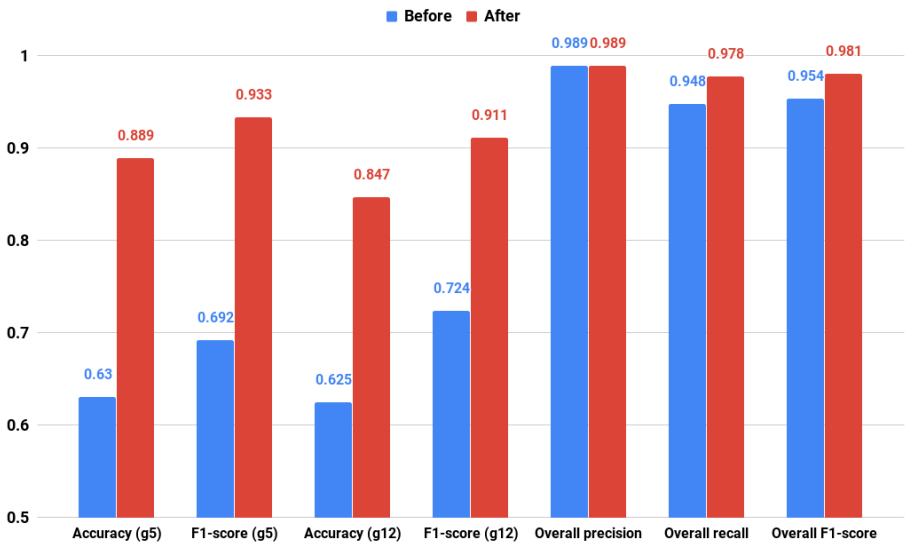

Figure 10: Recognition results before and after incremental updates with new features

As can be seen from Fig. 10, the ability distinguishing activities g_5 and g_{12} is greatly improved by new training data with new sensor data. Moreover, the enhancement of introducing new features into the existing lattice does not reduce the overall recognition rates.

5. Conclusions

Based on the excellent performance of formal concept analysis (FCA) in pattern recognition, we proposed a new incremental learning method for the FCA-based activity inference engine to dynamically integrate new data with new features to existing models in nonstationary smart environments. Most incremental lattice construction algorithms only update the

FCA-based models by data with fixed features. However, the proposed incremental algorithm overcomes such a limitation and its performance outperforms most non-incremental algorithms by avoiding retraining the entire model. Moreover, the incremental mechanism for updating the constructed FCA knowledge base is suitable for processing streaming data. Instead of storing the entire training dataset, the update does not need to use previous training data and directly modify the constructed lattice by new data. At the same time, the independence of updating and recognition modules can fast updating the model without interruption. The worldwide trends in privacy, data protection, and security [38] make transparent and explainable FCA-based models easier to use in practice. All these features will ease the maintenance burden of the inference engine and knowledge base.

According to the recognition results, new sensors or training data can be dynamically added to the current model in order to improve the local accuracy rates of misclassified activities. In the further, we may combine the active learning [39] to reuse those well-recognized patterns to enhance the knowledge base. The newly recognized patterns with high accuracy are added to class-labeled data for incrementally updating the current model. The updated model will continue to recognize unlabeled data. This process will always repeat. In addition, we can combine other graphical models, such as Bayesian classifiers, to evaluate the confidence of each inference in a probabilistic manner. Because of similar layouts or adopted sensors in smart environments, it would be interesting to reuse previously learned knowledge in different scenarios. Therefore, methods combining transfer learning and ensemble learning [39] will become active research areas to

address this problem.

References

[1] Gerhard Widmer and Miroslav Kubat. Learning in the presence of concept drift and hidden contexts. *Machine learning*, 23(1):69–101, 1996.

[2] Alexey Tsymbal. The problem of concept drift: definitions and related work. *Computer Science Department, Trinity College Dublin*, 106(2):58, 2004.

[3] Leandro L. Minku. Transfer learning in non-stationary environments. In *Learning from Data Streams in Evolving Environments*, pages 13–37. Springer, 2018.

[4] Albert Bifet. Adaptive learning and mining for data streams and frequent patterns. *ACM SIGKDD Explorations Newsletter*, 11(1):55–56, 2009.

[5] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, 46(4):44, 2014.

[6] Gregory Ditzler and Robi Polikar. Incremental learning of concept drift from streaming imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 25(10):2283–2301, 2013.

[7] Diane J Cook, Juan C Augusto, and Vikramaditya R Jakkula. Ambient intelligence: Technologies, applications, and opportunities. *Pervasive and Mobile Computing*, 5(4):277–298, 2009.

[8] Indrė Žliobaitė, Mykola Pechenizkiy, and Joao Gama. An overview of concept drift applications. In *Big data analysis: new algorithms for a new society*, pages 91–114. Springer, 2016.

[9] Jianguo Hao, Bruno Bouchard, Abdenour Bouzouane, and Sébastien Gaboury. Real-time activity prediction and recognition in smart homes by formal concept analysis. In *International Conference on Intelligent Environments*, pages 103–110. IEEE, 2016.

[10] Jianguo Hao, Abdenour Bouzouane, and Sébastien Gaboury. Complex behavioral pattern mining in non-intrusive sensor-based smart homes using an intelligent activity inference engine. *Journal of Reliable Intelligent Environments*, 3(2):99–116, 2017.

[11] Jianguo Hao, Sébastien Gaboury, and Bruno Bouchard. Cognitive errors detection: Mining behavioral data stream of people with cognitive impairment. In *Proceedings of the 9th ACM International Conference on PErvasive Technologies Related to Assistive Environments*, pages 15:1–15:8. ACM, 2016.

[12] Jianguo Hao, Abdenour Bouzouane, and Sébastien Gaboury. Recognizing multi-resident activities in non-intrusive sensor-based smart homes by formal concept analysis. *Neurocomputing*, 318:75–89, 2018.

[13] Ching-Hu Lu, Yu-Chen Ho, Yi-Han Chen, and Li-Chen Fu. Hybrid user-assisted incremental model adaptation for activity recognition in a dynamic smart-home environment. *IEEE Transactions on Human-Machine Systems*, 43(5):421–436, 2013.

[14] Zhelong Wang, Ming Jiang, Yaohua Hu, and Hongyi Li. An incremental learning method based on probabilistic neural networks and adjustable fuzzy clustering for human activity recognition by using wearable sensors. *IEEE Transactions on Information Technology in Biomedicine*, 16(4):691–699, 2012.

[15] Chunyu Hu, Yiqiang Chen, Lisha Hu, and Xiaohui Peng. A novel random forests based class incremental learning method for activity recognition. *Pattern Recognition*, 78:277–290, 2018.

[16] Paul E Utgoff. Incremental induction of decision trees. *Machine learning*, 4(2):161–186, 1989.

[17] J. Ross Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[18] Markus Prossegger and Abdelhamid Bouchachia. Multi-resident activity recognition using incremental decision trees. In *Adaptive and Intelligent Systems*, pages 182–191. Springer, 2014.

[19] Mohammad Arif Ul Alam, Nirmalya Roy, Archan Misra, and Joseph Taylor. CACE: Exploiting behavioral interactions for improved activity recognition in multi-inhabitant smart homes. In *International Conference on Distributed Computing Systems*, pages 539–548. IEEE, 2016.

[20] Viktor Losing, Barbara Hammer, and Heiko Wersing. KNN classifier with self adjusting memory for heterogeneous concept drift. In *IEEE International Conference on Data Mining*, pages 291–300. IEEE, 2016.

[21] Deepika Singh, Erinç Merdivan, Ismini Psychoula, Johannes Kropf, Sten Hanke, Matthieu Geist, and Andreas Holzinger. Human activity recognition using recurrent neural networks. In *Machine Learning and Knowledge Extraction*, pages 267–274. Springer, 2017.

[22] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[23] Gregory Ditzler, Manuel Roveri, Cesare Alippi, and Robi Polikar. Learning in nonstationary environments: A survey. *IEEE Computational Intelligence Magazine*, 10(4):12–25, 2015.

[24] Petko Valtchev, Rokia Missaoui, and Robert Godin. Formal concept analysis for knowledge discovery and data mining: The new challenges. In *International Conference on Formal Concept Analysis*, pages 352–371. Springer, 2004.

[25] Rokia Bendaoud, Amedeo Napoli, and Yannick Toussaint. Formal concept analysis: A unified framework for building and refining ontologies. In *International Conference on Knowledge Engineering and Knowledge Management*, pages 156–171. Springer, 2008.

[26] Carmen De Maio, Giuseppe Fenza, Vincenzo Loia, and Sabrina Senatore. Hierarchical web resources retrieval by exploiting fuzzy formal

concept analysis. *Information Processing & Management*, 48(3):399–418, 2012.

[27] Christian Lindig. Fast concept analysis. In *International Conference on Conceptual Structures*, pages 152–161, 2000.

[28] Petko Valtchev and Rokia Missaoui. Building concept (Galois) lattices from parts: generalizing the incremental methods. In *International Conference on Conceptual Structures*, pages 290–303. Springer, 2001.

[29] Bernhard Ganter and Rudolf Wille. *Formal concept analysis: mathematical foundations*. Springer Science & Business Media, 1999.

[30] Geetika Singla, Diane J Cook, and Maureen Schmitter-Edgecombe. Recognizing independent and joint activities among multiple residents in smart environments. *Journal of ambient intelligence and humanized computing*, 1(1):57–63, 2010.

[31] Charu C Aggarwal and Chandan K Reddy. *Data clustering: algorithms and applications*. CRC press, 2013.

[32] Petko Valtchev, David Grosser, Cyril Roume, and M Rouane Hacene. Galicia: an open platform for lattices. In *International Conference on Conceptual Structures*, pages 241–254. Shaker Verlag, 2003.

[33] Jean-Paul Bordat. Calcul pratique du treillis de galois d'une correspondance. *Math. Sci. Hum*, 96:31–47, 1986.

[34] Bernhard Ganter. Two basic algorithms in concept analysis. In *Proceed-*

ings of the 8th International Conference on Formal Concept Analysis, pages 312–340. Springer-Verlag, 2010.

[35] Robert Godin, Rokia Missaoui, and Hassan Alaoui. Incremental concept formation algorithms based on galois (concept) lattices. *Computational intelligence*, 11(2):246–267, 1995.

[36] Eugene M Norris. An algorithm for computing the maximal rectangles in a binary relation. *Revue Roumaine de Mathématiques Pures et Appliquées*, 23(2):243–250, 1978.

[37] Asma Benmansour, Abdelhamid Bouchachia, and Mohammed Feham. Modeling interaction in multi-resident activities. *Neurocomputing*, 230:133–142, 2016.

[38] Andreas Holzinger, Peter Kieseberg, Edgar Weippl, and A Min Tjoa. Current advances, trends and challenges of machine learning and knowledge extraction: From machine learning to explainable ai. In *Machine Learning and Knowledge Extraction*, pages 1–8. Springer International, 2018.

[39] Jiawei Han, Jian Pei, and Micheline Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.