Applying Data Mining in Smart Home

Chapter · January 2017

DOI: 10.1201/9781315145686-7

3 authors:

Kévin Bouchard
Université du Québec à Chicoutimi
158 PUBLICATIONS **1,602** CITATIONS
SEE PROFILE

Frédéric Bergeron
Université de Sherbrooke
10 PUBLICATIONS **69** CITATIONS
SEE PROFILE

Sylvain Giroux
Université de Sherbrooke
255 PUBLICATIONS **2,347** CITATIONS
SEE PROFILE

Applying Data Mining in Smart Home

Kevin Bouchard, Frederic Bergeron and Sylvain Giroux

[2]DOMUS, Universite de Sherbrooke
Sherbrooke (Quebec), Canada J1H 1H9
{Kevin.Bouchard, Frederic.Bergeron2, Sylvain.Giroux} @usherbrooke.ca

1 Introduction

Research on smart environments covers a broad range of topics and applications which require expertise from many disciplines [1]. Smart environments are used to improve energy efficiency and increase quality of life while being home [2]. They can also be exploited to provide more complex and more specialized services such as the recognition of faces in the workplace to assess the mood of employees (emotion recognition) or even to provide continuous healthcare support. Healthcare support is generally accompanied of security services and monitoring of ongoing Activity of Daily Living (ADL) [3]. ADLs compose a set of common activities that a normal person is supposed to be able to realize to be qualified as autonomous [4, 5]. The performance in the realization of ADLs is a good indicator of the state of the resident and can help an artificial intelligence to provide punctual adequate support services automatically [6]. In order to exploit the information related to the ADLs, the smart home must possess an intelligent software capable of deducing the action plan and the intended goal of the resident. That task is a well-known problem in research called the activity recognition challenge [7, 8].

Activity recognition is defined as the process of identifying a sequence of actions from sensors data and matching them to the corresponding model of ADL. In a smart home context, it is generally qualified as *keyhole* since the observation is performed unbeknown to the resident [9]. The difficulty of activity recognition in the smart home context lies in the fact that the basic actions cannot be directly observed. The sensors data are generally sparse and noisy, the source is multi-modal and preprocessing is necessary in order to use the data. Despite this, the classical approaches to this problem were mostly logic based. While there is no consensus on the classification of the activity recognition methods, they are usually grouped under the knowledge driven and the data driven approaches. Knowledge driven approaches were first investigated and generally make assumption that a complete plans library exists to recognize the activities. That library can be encoded with first-order logic [10], description logic [11], ontologies [12] or any other formalisms. It can also be described by probabilistic models such as Hidden Markov Model (HMM) [13] or Bayesian network [14] which represents well the uncertainty level in the decision process. The

adfa, p. 1, 2011.

main problem with knowledge driven approaches is the hard assumption they make on the plans' library that pose fundamental limits to the implementation of such algorithms in real deployed smart home [15]. First, the construction of the library is performed by a human expert and it can grows very complex within a small set of few ADLs. Second, the library is static; that is, it cannot evolve over time and adapt to the profile of the residents. Finally, it is generally assumed that the library is complete (i.e.: contains all possible ADLs). However, there are endless way to perform an ADL and there is a large number of possible ADLs a human can perform in his daily life. These problems often hamper the ability of researchers to deploy real-world activity recognition algorithms.

One avenue of solution to these limitations lies within the second family of methods: the data-driven approaches [16]. They consist to exploit data mining algorithms to learn the models corresponding to the set of ADLs instead of presupposing their existence [17]. Using data mining for that purpose is challenging, but the advantages would be numerous. First, the plans' library could be built automatically. Second, this library could evolve with upcoming data from the sensors and adapt to the particular profile of the resident (e.g.: If he is suffering from Alzheimer, his state will slowly worsen over time). Third, the deployment of an assistive smart home would require less intervention from human experts. Indeed, it is near impossible to clearly define how the sensors are bounded to basic actions and thus a data mining solution could adapt automatically. We have to keep in mind that the configuration of a new smart home is time-consuming and costly. Finally, the same solutions could be exploited to create tools that could enables the healthcare professionals to perform a closer and better monitoring of the state of the residents [18]. In this chapter, we introduce the reader to the main data mining approaches and the works of researchers that applied them in the smart home context. Our goal is to provide understandable material to exploit data mining for activity recognition in a smart home. This chapter can be seen as an introductory tutorial on the subject and be used as a basis for further development and research.

The remainder of this chapter is divided as follow. Section 2 discusses the general knowledge related to data mining. It defines what it is and describe the main challenges related to research and application of data mining. Section 3 introduces the reader to the decision trees. The principles are reviewed along with the main limitations for their application in smart home research. Section 4 covers the association rules mining algorithms. Again the basic principles are reviewed and the main works related to smart home research are reviewed. Section 5 investigates clustering for activity recognition. The main algorithms are explained and examples of their applications are provided. An opening on spatial data mining algorithms concludes the section. Finally, section 6

discusses the main challenges for smart home research in the next decades and concludes the chapter.

2 Data Mining Primers

Data mining is the set of methods and algorithms allowing the exploration and analysis of database [19]. It exploits tools from statistics, artificial intelligence and SGBD. Data mining is used to find patterns, association, rules or trends in datasets and usually to infer knowledge on the essential part of the information [20]. It is often seen as a subtopic of machine learning. However, machine learning is typically supervised, since the goal is to simulate the learning of known properties from *experience* (training set) in an intelligent system. Therefore, a human expert usually guides the machine in the learning phase [21]. Within realistic situations, it is often not the case. While the two are similar in many ways, generally, in data mining, the goal is to discover previously *unknown* knowledge [22] that can then be exploited in intelligent systems and business intelligence to take better decisions.

The complete process of data mining is illustrated on Figure 1. Before beginning the cycle, it is important to understand the context and the data related to our situation. For example, what is the goal of the data mining? What are the consequences of errors? Are they insignificant (e.g.: marketing decision for a new product) or critical (e.g.: healthcare decision support service)? Considering the nature of the data available is also important but usually for the design of the data mining strategy. First of all, what types of attributes are interesting? Is there any strong association between two attributes? Those are examples of questions one should try to answer before even beginning the data mining cycle. Once this preliminary phase is accomplished, the data mining can begin. The first step is to collect and clean the data from potentially more than one source, which can be devices, sensors, software or even websites. The goal of this step is to create the data warehouse that will be exploited for the data mining. The cleaning is often not necessary. However, sometime the data might be composed of noisy elements easy to remove or of attributes/objects that are known to be uninteresting for our current research. The second step consists in the preparation of the data in the format required by the data mining algorithm. Sometime in this step, the numerical values are bounded or discretized; other time, two or more attributes can be merged together. It is also at this step that high level knowledge (temporal or spatial relationships, etc.) can be inferred for suitable algorithms. For example, the team of Jakkula & Cook [23] exploited the temporal relationship of Allen [24] for association rules mining. These relations were extracted during a preparation step prior to the data mining phase. The next step is the data mining itself. It is important to choose or design an algorithm for the context and the data. There are many

algorithms to be used that we will discuss in the next sections. These algorithms are generally grouped under three main families: decision trees, association rules and clustering.

Finally, the data mining step should results in a set of models that need to be evaluated. In a supervised context, it is usually easily done with statistical methods such as the F-Measure, K-Statistic or the ROC curve [19]. However, in an unsupervised context it is often required to design more complex validation methods. If the evaluation is not conclusive enough, the cycle can be repeated until we are satisfied. Indeed, data mining is a method that often does not give expected results the first time. Note that the collection and cleaning step is generally done only once regardless of the results.

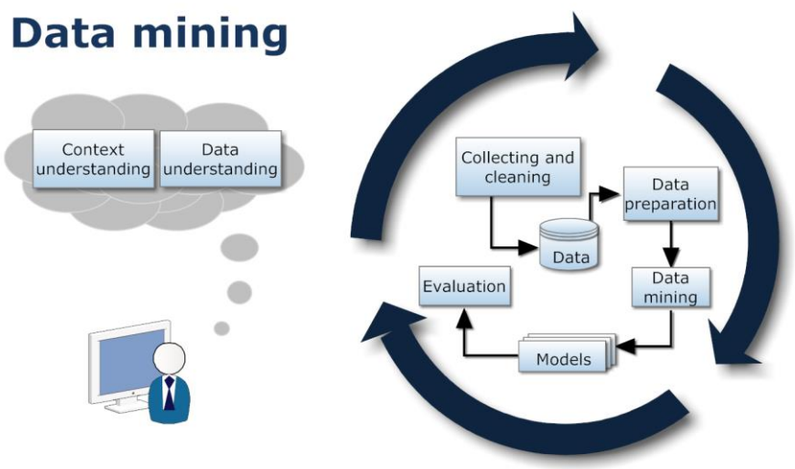

Figure 1: The overall data mining process

2.1 Supervised and unsupervised learning

Whether we talk about data mining method or machine learning in general, the process is usually classified under different categories [25]. The first one is supervised learning. The method is said supervised since it is based on training dataset with labeled examples or classes. The signification is that the algorithm can create a model that describes each class by using the known answers in the training set. In that situation, the idea is to generalize a function that maps the input to the output, and that can be used to generate output for previously unseen situations. The main implication is that somehow a human expert on the subject must label the dataset. On the opposite, unsupervised learning [21] works by using unlabeled examples. The idea is then to find hidden structure or association within the dataset and generalize a model from it. The results are sometime disappointing whether or not hidden knowledge exists in the dataset, but also sometime very surprising as the users do not know necessarily what they look for. The main implication is that there is no reward signal to evaluate the potential solutions. Unsupervised learning is often much harder to implement. Some researchers also use the name semi-supervised learning to describe their models. In that case, it usually means that the training set is partially labeled. However, it is also

used to mean that unsupervised learning was applied on a training set divided into several classes by a human or an algorithm [26].

3 Decision Trees

In the field of data mining, we generally classify all the algorithms under three main categories: decision trees, association rules and clustering. The general idea behind decision trees (DTs) is to take a large set of data and find the most discriminative properties to take classifying decisions from. In order to do that, the training set must be labeled (i.e. each entry must have the corresponding class it belongs to). In that sense, decision trees are supervised algorithms as we defined it in the introduction. From that data set, the algorithm will generally go through each attribute and choose, using a heuristic, the one that best divides the instances. It will then divide the data entries using that attribute and repeat the operation for the newly created nodes. However, it is necessary to prevent overtraining. If the DT fits to closely the data, it might be impossible to classify new instances (unknown). The Figure 2 shows an over fitting versus a representative model.

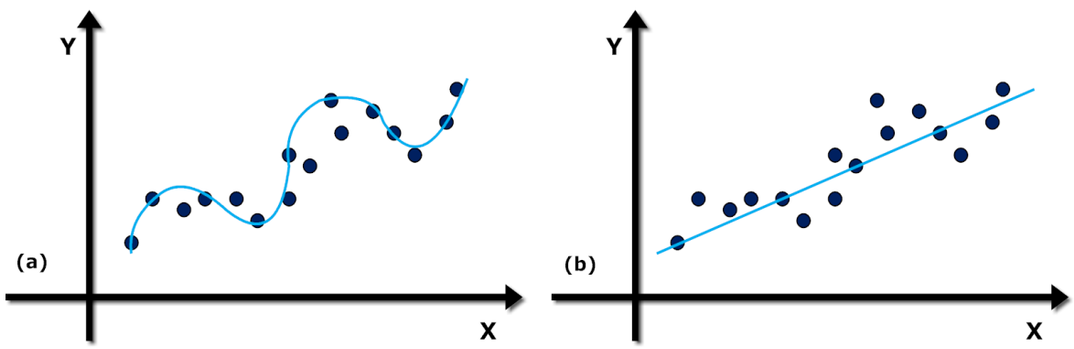

Figure 2: (a) Over fitting the data points. (b) A more interesting and simpler model.

To prevent over training, a decision tree classifier needs to have a stop condition. That condition can be: a maximum branching factor, all attributes are used, number of instances per node, etc. The classification of new instances is then simply performed by following the tree until reaching a leaf. In the next subsection, we will review two of the most important algorithms that are exploited to construct a decision tree.

3.1 ID3

To illustrate the basic ideas behind decision trees, let us have a closer look over a well-known algorithm. The algorithm that we wanted to present is called Iterative Dichotomiser 3 or, more commonly, ID3 [20]. This precursor of the well-known C4.5 is an algorithm used to generate a decision tree from the top to down without backtracking. To select the most useful attribute for classification, a criterion named the information gain based on the information theory is exploited.

The information gain of a given attribute X with respect to the class attribute Y is the reduction in uncertainty about the value of Y when we know the value of X. In order to calculate the information gain we need to know the information entropy. If $E(S)$ is the information entropy of the set S and n is the number of different values of the attribute in S, and $f_s(i)$ is the frequency of the value i in the set S, then the information entropy is calculated according to the following formula (1):

(1) $$E(S) = -\sum_{i=1}^{i=n} f_s(i) log_2\big(f_s(i)\big)$$

The entropy is always a number comprised between 0 and 1 inclusively. If all the examples are in the same class, the entropy of the population is nil. If there is the same number of positives and negative examples in binary classification, the entropy is maxed. The best attribute is selected based on the information gain factor that is given by the following formula (2):

(2) $$G(S,A) = E(S) - \sum_{i=1}^{m} f_s\,(A_i) E\big(S_{A_i}\big)$$

Where $G(S,A)$ is the gain of the set S after a split over the A attribute, m refers to the number of different values of the attribute A in S, $f_s(A_i)$ is the frequency of the items possessing A_i as i^{th} value of A in S and $S_{A_i}$is a subset of S. There are three requirements for the training data of ID3 algorithm. The first one is that all of the training data objects must have common attributes, and these attributes should be previously defined. The second requirement is that the attributes' values should be clearly indicated and a value indicating a special attribute should indicate no more than one state. The third requirement is that there must be enough test cases to distinguish valid patterns from chance occurrences. The Algorithm 1 details the ID3 process:

Algorithm 1: ID3.

Input: S learning data set; the set of attributes $A = \{a_j \epsilon \{1, \dots, p\}\}$ where p is the number of attributes remaining

If all elements in S are positive **Then**
    Add $root = positive$
    Return $root$
End
If all elements in S are negative **Then**
    Add $root = negative$
    Return $root$
End
If $A = \emptyset$ **Then**
    Add $root = negative$
    Return $root$
End

Set $a^* = arg \max_{a \epsilon A} gain(S, a)$
Set $root = a^*$
For all values v_i of a^*
    Add a branch to $root$ corresponding to v_i

Create $S_{a^*=v_i} \subset S$
If $S_{a^*=v_i} = \emptyset$ **Then**
Put a leaf with the most common value of the class among S at the extremity of this branch
Else
Put ID3($S_{a^*=v_i}$,$A - \{a^*\}$) at the extremity of this branch
End
End

ID3 possesses the advantage that it is fast, and it builds short trees. Nevertheless, if a small sample is tested, only one attribute at a time is tested for making a decision, and classifying continuous data may be computationally expensive. As any other DT algorithm, data may be over-fitted or over-classified by ID3. The classes created by ID3 are inductive, meaning that, given a small set of training instances, the specific classes created by ID3 are expected to work for all future instances. A limitation of ID3 is that the distribution of the unknown conditions must be the same as the test cases, and the induced classes cannot be proven to work in every case since they may classify an infinite number of instances.

3.2.1 Example of construction of a DT

To show the main characteristics of the construction of a decision tree with ID3, we will exploit the example dataset found on Table 1.

Table 1: Example dataset

Meal	*Filling*	*Size*	*Pattern*	*Class*
Poutine	Ketchup	Small	Filled	BBQ
Hot-Dog	Mustard	Small	Filled	BBQ
Hot-Dog	Ketchup	Small	Striped	Oven
Pizza	Mustard	Big	Striped	BBQ
Poutine	Mustard	Big	Striped	BBQ
Hot-Dog	Ketchup	Medium	Filled	BBQ
Pizza	Mayonnaise	Big	Striped	Oven
Pizza	Ketchup	Medium	Striped	Oven

From this dataset S, the overall entropy would be:

$$E(S) = \frac{5}{8} log_2 \left(\frac{5}{8}\right) + \frac{3}{8} log_2 \left(\frac{3}{8}\right) \approx 0.9544$$

To construct the tree, we would then need to calculate the information gain for each attribute. For example, the calculation for the attribute *Filling* would be:

$$G(S, Filling) = E(S) - \left(\frac{4}{8}\mathrm{E}(2,2) + \frac{3}{8}\mathrm{E}(3,0) + \frac{1}{8}\mathrm{E}(0,1)\right)$$

$$G(S, Filling) = E(S) - \left(\frac{4}{8} * 1 + \frac{3}{8} * 0 + \frac{1}{8} * 0\right)$$

$$G(S, Filling) = E(S) - 0.5 \approx 0.4544$$

Note that there are three entropy calculations made for each possible value of the attribute. For instance, the $\frac{4}{8}\mathrm{E}(2,2)$ is the part for *Ketchup* and 4 out of 8 are octagonal. The 2,2 means that

two of the *Ketchup* data entries are for BBQ and two are for the *Oven*. The gain of the three others attributes would be:

$$G(S, Meal) = E(S) - \left(\frac{2}{8}\mathrm{E}(2,0) + \frac{3}{8}\mathrm{E}(2,1) + \frac{3}{8}\mathrm{E}(1,2)\right) \approx 0.2657$$

$$G(S, Size) = E(S) - \left(\frac{3}{8}\mathrm{E}(2,1) + \frac{2}{8}\mathrm{E}(1,1) + \frac{3}{8}\mathrm{E}(2,1)\right) \approx 0.0157$$

$$G(S, Pattern) = E(S) - \left(\frac{3}{8}\mathrm{E}(3,0) + \frac{5}{8}\mathrm{E}(2,3)\right) \approx 0.3476$$

As it can be seen, the *Filling* would give the highest information gain, thus it is chosen as the root of our DT. The tree would have three branches after this first iteration as shown on Figure 3.

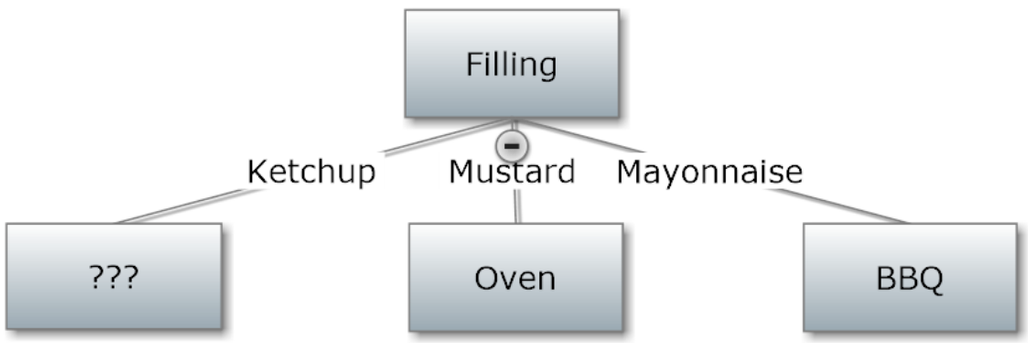

Figure 3: The DT after one iteration.

Now, only the *Ketchup* branch does not enable to clearly classify the population of the training set. The entropy of the ketchup subset (S_{ket}) must be calculated and then the information gain for the remaining attributes. In that case, the calculation would be:

$$G(S_{ket}, Meal) = E(S_{ket}) - \left(\frac{1}{4}\mathrm{E}(1,0) + \frac{2}{4}\mathrm{E}(1,1) + \frac{1}{4}\mathrm{E}(0,1)\right) = 0.5$$

$$G(S_{ket}, Size) = E(S_{ket}) - \left(\frac{2}{4}\mathrm{E}(1,1) + \frac{2}{4}\mathrm{E}(1,1)\right) = 0$$

$$G(S_{ket}, Pattern) = E(S_{ket}) - \left(\frac{2}{4}\mathrm{E}(2,0) + \frac{2}{4}\mathrm{E}(0,2)\right) = 1$$

As we can see, the *Pattern* value gives a maximal information gain for the subset S_{ket} and thus it is chosen to construct the DT. The final decision tree is illustrated by the Figure 4.

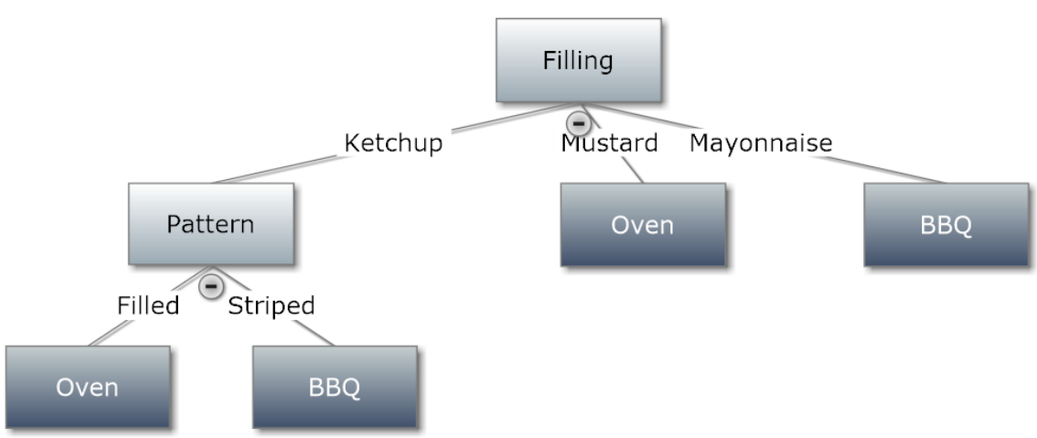

Figure 4: The resulting decision tree for the example dataset.

3.2 Variant of the decision tree

Over the time, many variant of the basic decision tree have been developed. However, the basic idea is always similar to what we have shown until now. In this sub-section, we describe one of them that is interesting for the advanced usage that was made of it. Section 3.1 presented the ID3 algorithm as a tree that was prompt to over fit the data. One way to avoid over fitting is by providing more training data. But, acquiring enough data is not always possible due to time or money constraint. Another way to prevent over training is to only consider a subset of the available information when training the model. Of course, this subset must be chosen wisely, in a certain way that will not miss important things.

This idea of considering only part of the data led to the tree called Random Tree (RT). The RT works exactly like ID3, with the simple difference that, at each node, only of subset of the attributes is considered for the information gain. The subset is chosen randomly at each node and we only compute the entropy for attributes in this subset. This means the split of a given node might not be optimal, thus reducing the chances of over fitting. Indeed, ID3 suppose that the future data will always have the same distribution over its attributes to classify. By not using the best split attribute, we can weaken the importance of this supposition in the final accuracy.

As we said, the key point of the random tree is to choose the attributes that will be used by this line of the previous pseudo-code of ID3:

(3) $$\textbf{Set } a^* = arg\max_{a \in A} gain(S, a)$$

There are many ways to do so. Let's call k the number of attributes we chose at a given node. A is again the set of all attributes and B is the subset. One common way is to randomly pick k times an attribute a from A without repetition. At each step, each remaining a as the probability $\frac{1}{|A|-|B|}$ of being chosen. The last remaining point is to choose the value for k. Again, one common way is the have a varying k that will adjust its value to number of remaining attributes as in:

(4) $$k = \text{int}(\log_2(|A| + 1))$$

We now have a simple tree that is not optimal but that should still be doing well with a good value of *k* and true randomness. With true randomness, there is no way of predicting what tree will be constructed. Thus, there are many possible trees for the same data. Is one really better than the other? If so, could we still consider the other when classifying? If they could work together, might the accuracy improve? These questions, among others, led to the creation of the Random Forest (RF) algorithm. The idea is simple: train many RT, let's say N, and make them vote for the class they think best represents a given instance. The number of RT is then another hyper-parameter that must be optimised, along with k and the ones present in ID3. A large N might over fit the data, while a small N might not improve over a single RT.

3.3 The Work of Stankovski

Stankovski is one of the researchers that has applied the decision trees algorithm in a smart home context [27]. As for any DT based system, a first step consisted of building a supervised dataset. In that case, the dataset contained the whereabouts of a person; interactions with appliances, duration, etc. The DT was created so the usual rules describing the normal setting leading to a particular event in the smart home could be known. The events occurring outside the normal setting were considered as abnormal behaviors, and in that case assistance could be triggered (alarm, message, etc.). To create the training dataset, heavy human expert intervention was required. After that the observations are gathered, the expert needs to specify two more data fields. For each record of observation, he needs to assign an activity and mark which records are normal (usual). The construction of the decision tree is done with ID3. The Figure 4 below shows a part of the decision tree built by Stankovski from a dataset of 35 examples.

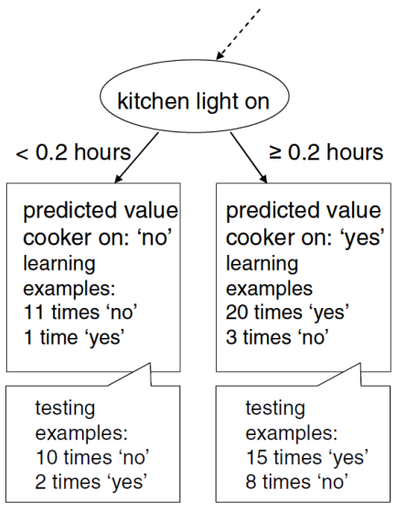

Figure 4: A part of a decision tree induced in [27].

3.4 The work of Bao & Intille

Bao & Intille [28] also worked on supervised learning methods for activity recognition. For this purpose, they collected datasets from subjects wearing five biaxial accelerometers. These subjects performed many activities and twenty were studied for this work. The authors extracted four features from the acceleration data: the mean, the energy, the entropy and the correlation. They tried various algorithms on these datasets including a C4.5 decision tree (direct successor of ID3). They trained a decision table, a nearest neighbor, a naïve Bayes and a C4.5 classifiers using two protocols. Under the first one, each classifier was trained on each subject's activity sequence data and tested on a second dataset were they performed a list of task on a sheet. Under the second protocol, they trained the classifiers on all dataset for all subjects except one. The classifiers were tested on the left out subject's datasets. This protocol was repeated for all twenty subjects. As you

can see on the Table 2, they have found that the decision tree outperformed all the other classifiers. In fact, only the nearest neighbor gave close results.

Table 2: Results of the classifiers from Bao & Intille [28]

Classifier	*User-specific training*	*Leave one out*
Decision Table	36.32 ± 14.501	46.75 ± 9.296
Nearest Neighbor	69.21 ± 6.822	82.70 ± 6.416
Naïve Bayes	34.94 ± 5.818	52.35 ± 1.690
C4.5	71.58 ± 7.438	84.26 ± 5.178

3.3 The work of Ravi et al.

Ravi et al. [29] worked on the same challenge than Bao & Intille [28]: activity recognition using accelerometer data. However, they wanted to use only one triaxial accelerometer worn near the pelvic region. They also formulate their problem as one of classification. Similarly, they tested well-known base classifiers. However, their focus was on meta classifiers. The principles behind these meta classifiers is usually to exploit knowledge on the problem (properties) or combine multiple classifiers to improve the results. They are divided among two families. The first one is voting. It includes methods such as Boosting [30] and Bagging [31] that both can be used in combination to classic classifiers to improve their performance. Boosting basically applies a single algorithm repeatedly and combine the hypothesis learned each time by voting. The voting is done by weighting each training examples depending on either they were correctly or incorrectly classified during a learning iteration. Bagging is similar but work by training each classifier on a random redistribution of the dataset (it also uses only one algorithm). The second family is named stacking and includes Meta Decision Trees (MDT) and Ordinary Decision Trees (ODT) [32]. MDT learns a meta-level decision tree whose leaves consist of each of the base classifiers. ODT instead specify the class of the given test instance at the leaves level. Therefore, MDT is an improvement over ODT which specifies the classifier to use to optimally classify an instance.

In their work, Ravi et al. [29] have shown that the performance of decision trees in activity recognition context can be improved with meta parameters. Nevertheless, the difference in performance is often not significant such as in the case of simple decision tree classification results (97.29-98.53-77.95-57.00) versus Boosted decision tree (98.15-98.35-77.95-57.00) and Bagged decision tree (97.29-95.22-78.82-63.33). However, they have found that MDT and ODT both produce higher classification accuracy for all four settings they tested.

3.4 The Bottom Line

There are many advantages to use decision trees. First, they create models that are easy to understand and use from a human perspective. They are also very robust to missing data and noise (which there are a lot in smart homes). Furthermore, the classification (not the learning phase) is

very fast and therefore makes them well suited for online activity recognition in smart homes. There are many models of decision trees that have been exploited in activity recognition researches such as the famous ID3 [28] or Meta Decision Tree (MDT) [29]. There are two types of application of DTs in the literature. They are often used to perform low granularity AR from a very specific type of information. These works focus on the technological platform rather than on the algorithm and mostly want to demonstrate the feasibility of their idea. For example, Ravi et al. [29] wanted to recognize ADLs from only one simple accelerometer worn by a subject at the belt level. The other type use decision trees in combination with another approach of AR (usually clustering, but it can also be a classical artificial intelligence approach). The DT then acts as a post filtering classifier [33].

The main problem with DTs is that they require a large set of labeled data to perform well. If there is not enough training data, the selected attributes might be misleading and the resulting classification performance poor. Figure 5 shows a simple yet stunning example of what can happen if the training set is too small. DTs also do not really support data evolution; that is learning must be redone if the data change too much (new attributes, new type of values, new number range, etc.). Finally, the last but probably the most important limitation for AR is their weakness to distinguish a large number of classes within a dataset.

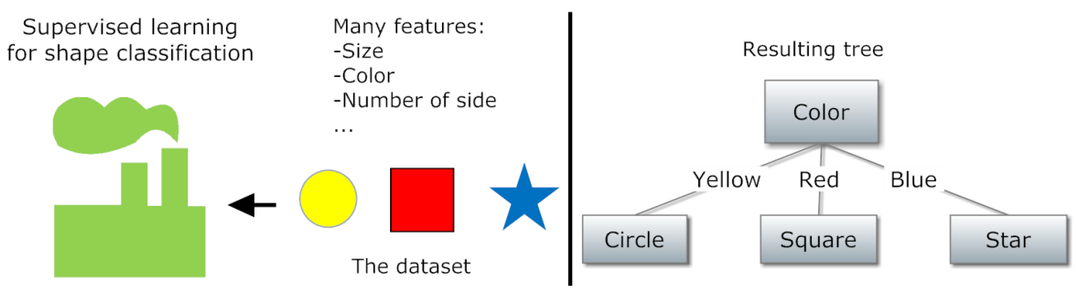

Figure 5: A three examples dataset for shape classification resulting in a strange DT based on the color.

4 Association Rules

Association rules mining is often confused with decision trees since the latest can always be represented by a set of rules. However, in most situations, rules are different than trees. For example, a large tree can often be represented by a smaller equivalent set of rules instead of the exhaustive decision nodes of the tree. Additionally, DTs try to split all classes while association rules mining considers one class at the time. Finally, association rules do not require labeled dataset. They are generally considered as fully unsupervised data mining algorithms. Nevertheless, we will see that in the smart home context they are often exploited as semi-supervised algorithms.

An association rule is a rule of the form *condition* => *consequence* that aims to find underlying relations between the data. For example, let's say that we have a dataset comprised of

transactions made at a store by customers. We could discover a rule such as *if Saturday and Beers => Potato chips*. That rule would mean that *very often*, when it is Saturday and someone buy beers that person will also buy potato chips. Association rules mining algorithms define the terms *very often* with two attributes named the support and the confidence. The first one defines the minimum frequency of both the left and right part of the rule. For example, supposes we have the item set {{A}, {B}, {AB}, {BA}, {B}, {AB}, {AB}}, the support of AB would be $Support(\{AB\}) = \frac{3}{7} \approx 43\%$. The second, the confidence, is the probability threshold of the right part being true if the left part is validated:

(5) $$Confidence(X => Y) = p(Y|X) = \frac{p(X \cup Y)}{P(X)} = \frac{Support(\{AB\})}{Support(\{A\})}$$

4.1 Apriori

The main principles behind the various association rules mining algorithms are very similar. To understand those principles, we will take a closer look to the most renowned and perhaps the most important algorithm of the family. It is named Apriori and was introduced by Agrawal & al. [34]. It relies upon two principles. The first one is the research for frequent k-itemsets whose support is higher than a fixed minimum support. The second consists to build the association rules from the found frequent k-itemsets. A rule is retained only if its confidence is higher than a fixed minimum confidence.

The Algorithm 2 shows the phase one of the Apriori algorithm.

Algorithm 2: Apriori, first phase.

Input: S learning data set; minimum support (σ) and confidence thresholds
Output: Set of frequent itemsets

Fetch the item sets that whose $> \sigma \rightarrow L_1$
Set $k = 1$
Repeat
  Increase k
  From L_{k-1} finds C_k the set of frequent itemsets candidates comprising k items
  Set $C_k = L_{k-1} \times L_{k-1}$
  Set $L_k = 0$
  For all $e \in C_k$ **do**
    If $Support(e) > \sigma$ **Then**
      Add e to L_k
    End
  End
Until $L_k \neq \emptyset$

4.2 Generalized Sequential Pattern

Another interesting algorithm that was also introduced by the team of Agrawal [35] is Generalized Sequential Pattern (GSP). This algorithm relies on the same foundation than Apriori

but was developed to work precisely on data sets of sequence of transactions instead of simple transactional data. The meaning is that the algorithm does not only take into account the presence of items together, but also the sequential ordering. Another particularity of GSP is its capability to exploit a taxonomy by encoding it in the data set. Let's look at an example from the original paper of Agrawal. Suppose we have the sequence <(Foundation, Ringworld) (Second Foundation)> and the taxonomy shown in Figure 6. To exploit the said taxonomy, all is required to do is to integrate it directly in the data set: <(Foundation, Ringworld, Asimov, Nirven, Science Fiction) (Second Foundation, Asimov, Science Fiction)>. It is also possible to optimize the encoding in order to avoid the explosion of data [35].

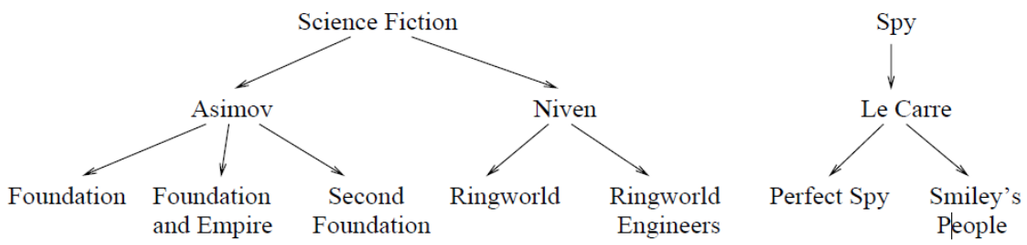

Figure 6: Example of taxonomy from Agrawal et al. [35].

Another interesting part of the GSP algorithm is the pruning which is done directly on the candidate itemsets by introducing the concept of contiguous subsequence. The idea is to suppress the candidates who possess a (k-1)-sequence contiguous with a support smaller than the fixed minimum support. A subsequence contiguous c of s is a sequence for which one of those three criterions is true:

1. c derivates from s by rejecting either s_1 or s_k
2. c derivates from s by rejecting an item from a s_i which possess at least two items
3. c is a contiguous subsequence of c' which is a contiguous subsequence of s

For example, considers the set s=<(1, 2) (3, 4) (5) (6)>. The subsequence <(2) (3, 4) (5)>, <(1, 2) (3) (5) (6)> and <(3) (5)> are all contiguous subsequence of s. However, <(1, 2) (3, 4) (6)> and <(1) (5) (6)> are not. Now let's look at an example dataset to demonstrate how the pruning work within GSP algorithm. Considers the seed set consisting of those six frequent 3-sequences:

1. <(1, 2) (3)>
2. <(1, 2) (4)>
3. <(1) (3, 4)>
4. <(1, 3) (5)>
5. <(2) (3, 4)>
6. <(2) (3) (5)>

The junction step of the algorithm would lead to obtain these two frequent 4-sequence considering a support of 100%: <(1,2)(3,4)> and <(1,2)(3)(5)>. The second sequence, <(1,2)(3)(5)>, would be abandoned during the pruning because subsequence <(1)(3)(5)> is not part of L_3 (for GSP, the fourth sequence is not equivalent to <(1)(3)(5)>). In fact, this sequence is

contiguous since the criterions number two is true for it. The next subsection will describe a complete smart home solution exploiting rules mining for activity recognition and activity prediction.

4.3 The Work of Jakkula & Cook

Jakkula & Cook developed a renowned approach of activity discovery for smart home which is based on association rules mining. Their system was built in a multi-agents [36] architecture where the agents perceive directly the state of the environment from sensor's output raw data.

They collected temporal information constructed from Allen's intervals calculus presented in [24]. Their goal was to process raw data to discover frequent sequential patterns. In that case, it enables the discovery of temporal links existing between frequent events. For example, if recorded data tends to demonstrate that every time *Take Tea* happens the kettle is activated soon after, the recognition system will infer a temporal rule from Allen's thirteen relations (*Boil Water* after *Take Tea*). Supposing that a lot of training data are available, Jakkula & Cook's model works as follows. First, the temporal intervals are found using the timestamp of events and the on/off state of binary sensors. The algorithm that associates these intervals to one of Allens' relations is illustrated below (Algorithm 3):

Algorithm 3: Temporal Interval Analyzer [26].

Input: $E = \{\text{set of events}\}$

Output: Set of Allen's relation

Repeat
  While (E_i && E_{i+1})
    Find pair ON/OFF events in data to determine temporal range
    Read next event and find temporal range
    Associate Allen's relation between events
    Increment Event pointer
  End
Until end of input

The algorithm loops until all the pairs of events are compared. Between each pair, it establishes the Allen's relationship from the beginning and end markers of both events.

The second step in their model is to identify frequent activities or events that occur during a day to establish a reduced set of activities. This step is mandatory because there are too much data from smart home sensors, and many potential anomalies are just noise that should be ignored. They accomplish this task using the Apriori algorithm [37] that was described previously. In their work, Jakkula & Cook not only demonstrated that temporal relationships provide insights on patterns of resident behaviors, but also that it enhances the construction of other smart home assistance algorithms. To do so, they calculated the probability that a certain hypothetic event occurs or not,

given the observed occurrence of other events temporally related. It is done from the frequency of the nine relationships out of thirteen they determined that could affect anomaly detection: *before*, *contains*, *overlaps*, *meets*, *starts*, *started-by*, *finishes*, *finished-by* and *equals*. The formula to calculate the evidence of the occurrence of an event X is given by the observation of other events (such as Y) that are temporally related (from previous learning phase). The equation 6 below allows such calculus:

(6) $$P(X|Y) = |After(Y,X)| + |During(Y,X)| + |OverlappedBy(Y,X)| + |MetBy(Y,X)| + |Starts(Y,X)| + |StartedBy(Y,X)| + |Equals(Y,X)| \,/\, |Y|$$

That equation gives the likelihood of X considering Y. To combine evidence of X from multiple events that are in temporal relationship with X, we have to improve the equation. Consider the events Z, Y that had been observed in this order, the prediction of X is given by the formula $Prediction_x = P(X)$ that is calculated as follows (7):

(7) $$P(X|Z \cup Y) = \frac{P\big(X \cap (Z \cup Y)\big)}{P(Z \cup Y)} = P(X \cap Z) \cup \frac{P(X \cap Y)}{P(Z)} + P(Y) - P(Z \cap Y)$$
$$= P(X|Z).P(Z) + P(X|Y).\frac{P(Y)}{P(Z)} + P(Y) - P(Z \cap Y)$$

From the formula, anomalies can be detected and predictions can be made. If an event X as a probability approaching 1, then it is considered as most likely to occur. On the other hand, if its probability is close to 0, it will be considered as an unusual event and will be ignored from further predictions. The final step is to use an enhanced version of the *Active LeZi* (ALZ) [38] algorithm for the prediction by adding these discovered temporal rules as input data. This predictor is sequential and employs incremental parsing and uses Markov models. It should be noted that *ALZ* improved could be used for anomaly detection. This could be done by using the prediction as input in an anomalies detection algorithm and by comparing prediction sequence with observations. Thus, if the new observation does not correspond to the expected event, an assisting sequence could be triggered. The add-on to the *Active LeZi* is shown below (algorithm 4):

Algorithm 4: Temporal Rules Enhanced prediction.

Input: Output of ALZ a, Best rules r, Temporal dataset

While $(a! = null)$
  Repeat
    Set r_1 to the first event in the first best rule
    If $(r_1 == a)$ Then
      If $(Relation! = "After")$ Then

Calculate evidence
If ($Evidence > (Mean + 2\ Std.Dev.)$) Then
Make event in the best rule as next predictor output
Else
*Get next predicted event and look for their temporal relation in the temporal relations database based on the frequency.
If the relation is after again **Then**
Go to * Until no more after relations found then calculate evidence
If high **Then** predict;
Else Calculate evidence and if high then predict this event based on the relation; Continue.
End
End
End
Until end of rules
End While

Following the creation of this algorithm, they have conducted experiments that can be seen in Table 2 below. It shows the accuracy of the observed prediction performance on real data sets and synthetic. There is a performance improvement in the prediction of activities of the resident of the intelligent environment. The main reason for the important error rate is the small amount of data in the datasets used for learning. The search for knowledge-based temporal rules is a new area of research in smart homes and should be further explored in the future. Note that the use of temporal relationships provided a unique new approach for prediction.

Table 2: Comparison of ALZ prediction with and without temporal rules

Datasets	*Percentage accuracy*	*Percentage error*
Real (without rules)	55	45
Real (with rules)	56	44
Synthetic (without rules)	64	36
Synthetic (with rules)	69	31

4.4 The work of Bouchard & al.

In the same line of idea of Jakkula & Cook [23], Bouchard & al. [39] worked with association rules mining for activity recognition in a smart home. To do so, they exploited the topological relationships that exist between entities present in the smart environment. It was done by using the framework of Egenhofer & Franzosa [40] which defines the relation between two entities e_1 et e_2 with the formal intersection structure between their interior (°) and boundary (∂) points: $< \partial e_1 \cap \partial e_2,\ e_1{}^\circ \cap e_2{}^\circ, \partial e_1 \cap e_2{}^\circ, e_1{}^\circ \cap \partial e_2 >$. By using the simple invariant empty property of sets, there are sixteen possible relation types. However, only eight exist for physical regions without holes.

In their model, activities are defined by a set of constraints *K* such that:

(8) $$K = \{T(e_1, e_2) | e_1, e_2 \subseteq O \times O \cup R \times A\}$$

where T is a topological relation, O is a physical object, R is the resident and A is a logical area of the smart home. The recognition process consists then to evaluate the plausibility of each ADLs in the knowledge base from the constraints observations made in the environment. The plausibility is calculated by using the neighborhood graph of the topological relationships. Considering that the $Similarity$ function returns the percentage of similarity from 0 to 100% between observed topological relationships and those known to define an activity, the scoring of an activity ($a_{\delta,i}$) for a single iteration i is:

(9) $$a_{\delta,i} = \sum_{n=a_{t,0}}^{a_{t,i} \in a_T} \sum_{m=l_0}^{l_j \in L} \varphi * Similarity(n, m)$$

where a_T is the set of topological relationships defining the activity a. It is the same calculation for the topological relationships implying the resident and a smart home zone. The next step of the algorithm is to choose the most plausible activity that is ongoing. In other words, it has to choose which ADL best explains the observations made up until the current iteration. The plausibility of an activity after i_c iterations is calculated by the function below (10):

(10) $$plausibility(a) = \sum_{i=0}^{i_c} a_{\delta,i} * \phi^{i_c - i}$$

That is, the plausibility of a is the sum of all the points gained modulated by an inverted exponential function. The constant parameter $\phi \in (0, 1)$ modulates the speed at which the function tends to 0. Bigger it is, the longer iteration's score has an impact. The last step is to normalize the points gained by the activities with equation 11:

(11) $$NormalizedADL = \bigcup_{i=0}^{a_i \in ADL} \frac{score(a_i)}{\sum_{j=0}^{x_j \in ADL} score(x_j)}$$

The ADL with the highest score is the one selected as currently being realized. They exploited GSP and Aprori to automatically build an activity library for their recognition algorithm. To collect their datasets, they used a real subject that performed four different activities three time each. They collected more than 350 000 lines of data which they used with both GSP and Apriori. From the learned spatial rules, they were able to recognize 100% of the activities in real time in the smart home.

4.5 The Bottom Line

As you can see, association rules mining approaches are very interesting and more general than DT. Due to their inherent working, they are perfectly adapted to learn logical rules about activity of daily living and be exploited for AR. Despite this, association rules mining usually results in an important number of trivial and non interesting rules. That is, a human usually needs to check all the extracted rules in order to find the few that could be exploited. This supplementary step often require a lot of efforts and time and constitutes one of the major limitations. Additionally, the collected dataset is not always adapted to this kind of algorithms. They work well on logical information such as spatial or temporal relationship, but are less suited to deal with raw data from sensors. Thus, it is often necessary to conceive an ad hoc method to transform the data into an appropriate form. Finally, the method is not working well for rare items. Due to the high dimensionality of our data, frequent patterns might not be that frequent in real contexts.

5 Clustering

To address the issues that exist with DTs and association rule mining, many researchers aim to exploit completely unsupervised learning. Clustering could be a good solution since it can extract similar data automatically from unlabeled data. The idea behind this type of algorithm is simple. The goal is to find *clusters* in the dataset that could separate the records into a number of similar classes. A cluster is, in that context, a set of similar objects, where similarity is defined by some distance measure. The goal of the distance measure is to obtain clusters with a high intraclass similarity and a low interclass similarity. The distance measure should respect these four properties:

1. $d(x, y) \geq 0$
2. $d(x, y) = 0 \; iff \; x = y$
3. $d(x, y) = d(y, x)$
4. $d(x, z) \leq d(x, y) + d(y, z)$

Among the popular known distances, here are respectively the Euclidian distance, the Manhattan distance and the Minkowski distance:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \qquad d(x, y) = \sum_{i=1}^{n} |x_i - y_i| \qquad d(x, y) = \sqrt[q]{\sum_{i=1}^{n} |x_i - y_i|^q}$$

Euclidian *Manhattan* *Minkowski*

The clustering problem is a difficult challenge because the attributes (or features) and their values that differentiate one cluster from another are not known. There is no data examples to tell what features differentiate objects that belong to different clusters, and as the size of the dataset

increases, the number of clusters, as well as the number and type of differentiating factors might change. Moreover, there is no guide to indicate what constitutes a cluster and the success of the clustering algorithms is influenced by the presence of noise in the data, missing data, and outliers.

5.1 K-Means

The most important clustering algorithm is without a doubt the *K-Means* [41]. The goal of this algorithm is to split a dataset into *k* clusters where the value of *k* is selected beforehand by the user. The first step of the algorithm is to select *k* random data points as the center of each cluster from the data space D which might comprise records that are not part of the training set S. Then, the other data points (or records) are assigned to the nearest center. The third step is to compute the gravity center of each cluster. These k gravity centers are the new centers for the clusters. The algorithm then repeats until it reaches stability. The stability means that none of the data points in S change of cluster from an iteration to another or that the intraclass inertia is now smaller than a certain threshold. The Algorithm 5 details the K-Means process:

Algorithm 5: K-Means algorithm

Input: S the dataset, k the number of clusters to create

Output: Set of k clusters

Set the intraclass inertia $I_w = \infty$
Select k center points $c_j \epsilon D$
Repeat
  For ($j \epsilon \{1, \dots, k\}$)
    Set cluster $G_j = \emptyset$
  End
  For ($i = 1\ to\ |S|$)
    Set $j^* = \underset{j \epsilon \{1,\dots,k\}}{\operatorname{argmin}}\, d(s_i, c_j)$
    Set $G_{j^*} = G_{j^*} \cup s_i$
  End
  For ($j \epsilon \{1, \dots, k\}$)
    Set $c_j = gravity\ center\ of\ G_j$
  End
  Calculate I_w
Until $I_w < threshold$

To really understand the algorithm, we have to specify two concepts: the gravity center and the intraclass inertia. The center of gravity of a dataset X described by p features (attributes) is a synthetic data equal to the average a of each attributes in X:

(12) $$center\ of\ gravity = (a_1, a_2, \dots, a_p)$$

The inertia of a dataset X of $|X|$ records is defined by equation 13:

(13) $$I_X = \sum_{i=1}^{|X|} d^2(x_i, g)$$

where g is the gravity center of X and x_i the i^{th} record of the dataset. The function d^2 represents the Euclidian distance. Finally, the intraclass inertia I_w is given by the following calculation (14):

(14) $$I_w = \sum_{i=1}^{k} w_i I_i$$

where w_i is the weight of the i^{th} cluster and I_i its inertia. If the data have all the same weight, this weight is calculated by using the number of elements member of the cluster G_i and using the formula 15:

(15) $$w_i = |G_i|/|X|$$

We will now look through a visual example of how K-Means works. Suppose that the dataset is visually represented in a Cartesian plane as shown on Figure 7(a). In this example, the goal is to find three clusters, therefore the parameter k is set to three. The Figure 7(b) shows a possible initialization for the center points of these three clusters. The records of the dataset are assigned to the nearest center.

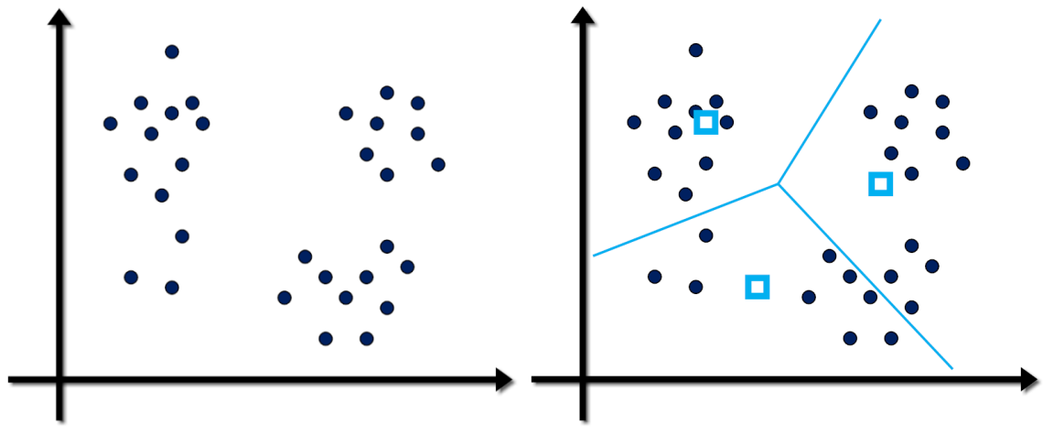

Figure 7 : (a) The dataset before the beginning. (b) Example of initialization with three clusters.

Then, as explained, the centers of gravity of each cluster are computed from the instances they contain. The data records are reassigned accordingly from their distance to the new centers (see Figure 8(a)). Finally, the process is repeated until stability is reached. The Figure 8(b) shows the final clusters in our example.

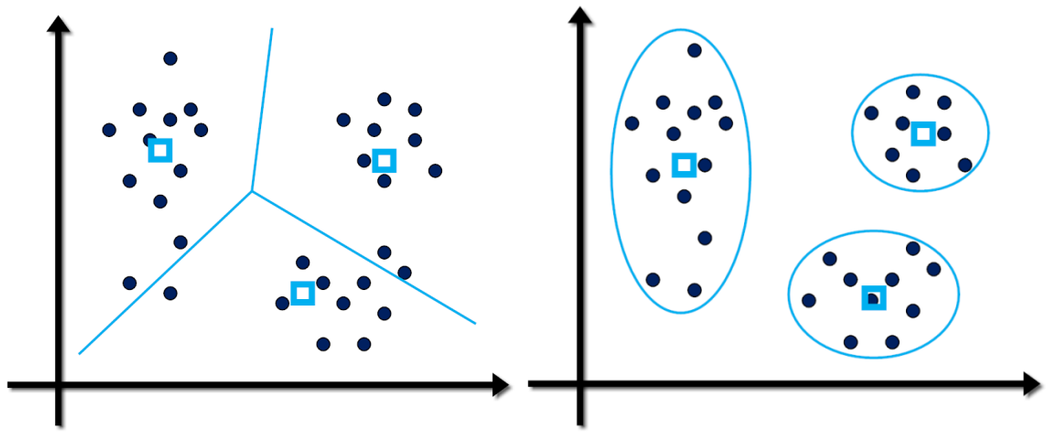

Figure 8: (a) New clusters after calculation of the new centers. (b) Final clusters.

The K-Means algorithm is a fast algorithm: it is considered as an algorithm of linear complexity. In fact, it is considered as one of the fastest clustering algorithms, and it usually requires a small number of iterations to find the final clusters [19]. However, there are many drawbacks to the exploitation of this algorithm. First of all, the final clusters are highly dependent on the initial centers that were selected semi randomly. Second, the algorithm converges to local minima. That is, the centers of each cluster move toward a reduction in the distance from their data but there is no guaranty that the global distance will be minimal.

An improved version named K-Means++ was introduced [19] and has for goal to select the center of the first cluster such that it has a uniform probability distribution. Then, the subsequent centers are determined such that their position is proportional to a square of a certain distance value from the first one. That enhancement improves the execution speed and also the precision of the results. However, there is a last problem that remains. To work, the user of K-Means must know the number of clusters beforehand. In the smart home context, it is usually not possible to do since we do not know in advance the number of ADLs that have been realized in the training dataset. Therefore, a clustering algorithm that does not require to specify k the number of clusters is required.

5.2 Density based clustering

The first spatial data mining algorithm we present is named Density-Based Spatial Clustering of Applications with Noise or simply DBSCAN [42]. It is a clustering algorithm that supports noise in the dataset. The goal of this algorithm is to address two of the problems of K-Means based algorithms. The first one is the weirdly shaped clusters that cannot be recognized with K-Means. The second is the noise that is necessarily assigned to one of the clusters with K-Means algorithm. The Figure 9 shows three sample dataset taken directly from an example of Ester original paper. A human can easily find the clusters just by looking at each dataset, but K-Means will give poor results on the latest two.

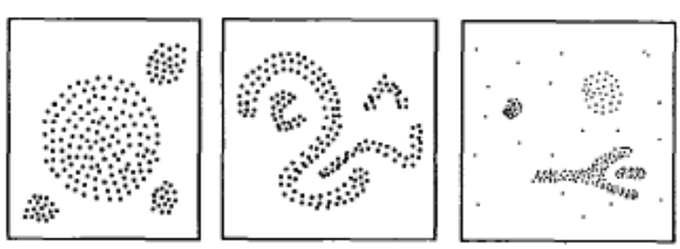

Figure 9: Three samples dataset from the original paper of Ester.

DBSCAN is based on four important definitions to establish the notion of dense clusters of points. The first definition is the $\epsilon - neighborhood$ of a point which come from mathematical topology:

(16) $$N_\epsilon(p) = \{q \in D | dist(p,q) \leq \epsilon\}$$

This equation describes that q is in the $\epsilon - neighborhood$ of p if the distance between them is smaller than ϵ. An intuitive notion of a dense cluster would be to say that each point has at least *MinPoints* in their $\epsilon - neighborhood$, but it would fail because there are core points and border points in a cluster. The second definition introduced by the team of Ester describes the notion of directly density-reachable point p from a point q:

(17) $$p \in N_\epsilon(q) \; and \; |N_\epsilon(q)| \geq MinPoints$$

That means that p is directly density-reachable from q if it is in its neighborhood and q is a core point (second condition). The relation is symmetric if both points are core type. Third, the point p is density-reachable from a point q if there is a chain of points $P_1, \dots, P_n, P_1 = q, P_n = p$ such that P_{i+1} is directly density-reachable from P_i. Finally, A point p is density-connected to a point q if there is a point o such that both, p and q are density-reachable from o. Using these four definitions, the authors define a dense cluster as a set of density-connected points. A special set is used to comprise the noise. It includes the points that do not belong to any cluster. Figure 10 shows visually the concept density reachability and density-connectivity. Algorithm 6 gives the general idea of the clustering from the concepts presented.

Overall, DBSCAN possesses two important advantages. First, it can be used for applications with noisy data. Second, the clusters can be of varied shape: circular, rectangular, elongated, concave, etc. There is also a Generalized version (GDBSCAN) [43] which allows to use the algorithm with different distances and with two dimensional-shapes. DBSCAN possesses some limitations for AR. It is not fast enough for online use. Additionally, it is made for static spatial information rather than changing spatial information such as what we get in smart homes. Therefore, it cannot extract the patterns of movement of the various objects in the realization of ADLs.

Algorithm 6: DBSCAN algorithm

Input: S the dataset, $mpts$ the minimum number of points, ϵ the neighborhood

```
Set clusterID = nextID(NOISE)
For (i = 1 to |S|)
    Set p = S[i]
    If (p.ClID = UNCLASSIFIED) Then
        If (ExpendCluster(S, p, clusterID, ϵ, mpts)) Then
            Set clusterID = nextID(clusterID)
        End
    End
End
```

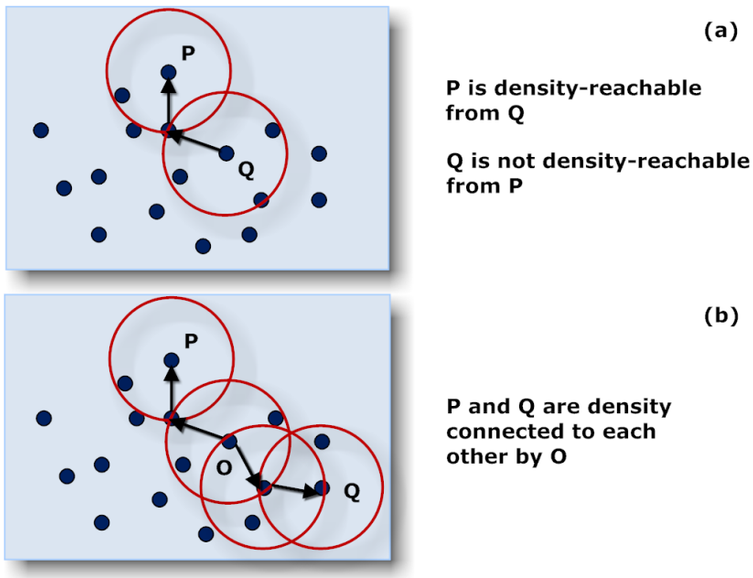

Figure 10: (a) Density-reachability. (b) Density-connectivity.

5.3 Moving clustering

The last algorithm we present is a very different approach to clustering. The algorithm presented in [44] aims to develop a mobility based clustering for the monitoring of vehicles' crowdedness in metropolis. Their idea is to use solely the current speed of vehicles since a high mobility means low crowdedness. The main challenge of their approach is not one of clustering in fact; it is to deal with contextual information (e.g. red light, etc.) and the imprecision of GPS data. Otherwise, most of their work is based on statistical methods. There are many advantages to mobility based clustering. First, it is little sensitive to the size of the sample. Second, it does not require precise position and support errors in positioning. Finally, it naturally incorporates the mobility of different objects such as vehicles. Even through the model is not general enough to be directly applied to our problematic, we found the idea innovating and it inspired us in the quest for a new spatial data mining method. Still, mobility based clustering is new and much work remains to do to obtain interesting accuracy.

5.4 The Bottom Line

Clustering seems to be a very good opportunity for AR, but only few approaches have successfully exploited it [16]. Moreover, every time it is with a small number of low granularity activities. For example, Palme et al. [45] used completely unsupervised method that extracted the most relevant object to represent an ADL (key object). It is limited by the uniqueness requirement of the key object. In general, however, there are many reasons that explain why very few approaches exist. First of all, the complexity of information gathered from multiple sensors in smart home limit the ability of a standard clustering algorithm to spit correctly the data. In fact, an algorithm such as K-Means is not able to distinguish noise from interesting information. Second,

most of the clustering algorithms need the initial number of clusters to work correctly and those that do not are very slow (high complexity). Finally, standard clustering algorithms do not fully exploit the ADL information embedded in the dataset. For example, they ignore many fundamental spatial aspects such as the topological relationships or the movement of entities.

6 References

1 Robles, R.J., and Kim, T.-h.: 'Applications , Systems and Methods in Smart Home Technology : A Review', Int. Journal of Advanced Science And Technology, 2010, 15, pp. 37-48

2 Ricquebourg, V., Durand, D., Delahoche, L., Menga, D., Marhic, B., and Loge, C.: 'The Smart Home Concept : our immediate future', in Editor (Ed.)^(Eds.): 'Book The Smart Home Concept : our immediate future' (2006, edn.), pp. 23-28

3 Katz, S., Ford, A.B., Moskowitz, R.W., Jackson, B.A., and Jaffe, M.W.: 'Studies of Illness in the Aged', JAMA: The Journal of the American Medical Association, 1963, 185, (12), pp. 914-919

4 Schwartz, M.F., Segal, M., Veramonti, T., Ferraro, M., and Buxbaum, L.J.: 'The Naturalistic Action Test: A standardised assessment for everyday action impairment' (Psychology Press, 2002. 2002)

5 Baum, C., and Edwards, D.F.: 'Cognitive performance in senile dementia of the Alzheimer's type: the Kitchen Task Assessment' (1993. 1993)

6 Nugent, C., Mulvenna, M., Moelaert, F., Bergvall-Kareborn, B., Meiland, F., Craig, D., Davies, R., Reinersmann, A., Hettinga, M., Andersson, A.-L., Droes, R.-M., and Bengtsson, J.E.: 'Home based assistive technologies for people with mild dementia'. Proc. Proceedings of the 5th international conference on Smart homes and health telematics, Nara, Japan2007 pp. Pages

7 Kasteren, T.v., Noulas, A., Englebienne, G., and Krose, B.: 'Accurate activity recognition in a home setting'. Proc. Proceedings of the 10th international conference on Ubiquitous computing, Seoul, Korea2008 pp. Pages

8 Bouchard, B., Roy, P., Bouzouane, A., Giroux, S., and Mihailidis, A.: 'An Activity Recognition Model for Alzheimer's Patients: Extension of the COACH Task Guidance System'. Proc. ECAI2008 pp. Pages

9 Cohen, P.R.: 'The pragmatics of referring and the modality of communication', Computational Linguistics, 1984, 10, (2), pp. 97-146

10 Kautz, H.A.: 'A formal theory of plan recognition and its implementation': 'Reasoning about plans' (Morgan Kaufmann Publishers Inc., 1991), pp. 69-124

11 Bouchard, B., Giroux, S., and Bouzouane, A.: 'A keyhole plan recognition model for Alzheimer's patients: First results', Journal of Applied Artificial Intelligence, 2007, 21, (7), pp. 623-658

12 Chen, L., Nugent, C.D., and Wang, H.: 'A Knowledge-Driven Approach to Activity Recognition in Smart Homes', IEEE Trans. on Knowl. and Data Eng., 2012, 24, (6), pp. 961-974

13 Nguyen, N., Bui, H., Venkatesh, S., and West, G.: 'Recognising and Monitoring High-Level Behaviours in Complex Spatial Environments', in Editor (Ed.)^(Eds.): 'Book Recognising and Monitoring High-Level Behaviours in Complex Spatial Environments' (2003, edn.), pp. 620-625

14 Albrecht, D.W., Zukerman, I., and Nicholson, A.E.: 'Bayesian Models for Keyhole Plan Recognition in an Adventure Game', User Modeling and User-Adapted Interaction, 1998, 8, (1-2), pp. 5-47

15 Turaga, P., Chellappa, R., Subrahmanian, V.S., and Udrea, O.: 'Machine Recognition of Human Activities : A Survey', Anglais, 2008, (11), pp. 16

16 Wyatt, D., Philipose, M., and Choudhury, T.: 'Unsupervised activity recognition using automatically mined common sense'. Proc. Proceedings of the 20th national conference on Artificial intelligence - Volume 1, Pittsburgh, Pennsylvania2005 pp. Pages

17 Gu, T., Chen, S., Tao, X., and Lu, J.: 'An unsupervised approach to activity recognition and segmentation based on object-use fingerprints', Data Knowl. Eng., 2010, 69, (6), pp. 533-544

18 Komninos, A., and Stamou, S.: 'HealthPal: an intelligent personal medical assistant for supporting the self-monitoring of healthcare in the ageing society', Proceedings of the UbiHealth, 2006

19 Ian H. Witten, and Franck, E.: 'Data mining: practical machine learning tools and techniques', Elsevier editor, 2010

20 Quinlan, J.R., and Ghosh, J.: 'Top 10 Algorithms in Data Mining', in Editor (Ed.)^(Eds.): 'Book Top 10 Algorithms in Data Mining' (IEEE, 2006, edn.), pp.

21 Barlow, H.B.: 'Unsupervised learning', Neural computation, 1989, 1, (3), pp. 295-311

22 Chaudhuri, S., Dayal, U., and Narasayya, V.: 'An overview of business intelligence technology', Commun. ACM, 2011, 54, (8), pp. 88-98

23 Jakkula, V.R., and Cook, D.J.: 'Enhancing Smart Home Algorithms Using Temporal Relations', in Mihailidis, A., Boger, J., Kautz, H., and Normie, L. (Eds.): 'Technology and Aging' (IOS Press, 2008), pp. 3-10

24 Allen, J.F., and Hayes, P.J.: 'A common-sense theory of time'. Proc. Proceedings of the 9th international joint conference on Artificial intelligence - Volume 1, Los Angeles, California1985 pp. Pages

25 Carbonell, J., Michalski, R., and Mitchell, T.: 'An Overview of Machine Learning', in Michalski, R., Carbonell, J., and Mitchell, T. (Eds.): 'Machine Learning' (Springer Berlin Heidelberg, 1983), pp. 3-23

26 Jakkula, V., and J. Cook, D.: ‘Mining Sensor Data in Smart Environment for Temporal Activity Prediction’, in Editor (Ed.)^(Eds.): ‘Book Mining Sensor Data in Smart Environment for Temporal Activity Prediction’ (ACM, 2007, edn.), pp.
27 Stankovski, V.: ‘application of decision trees to Smart Homes’: ‘Designing Smart Homes’ (Springer, 2006)
28 Bao, L., and Intille, S.S.: ‘Activity recognition from user-annotated acceleration data’: ‘Pervasive Computing’ (Springer, 2004), pp. 1-17
29 Ravi, N., Dandekar, N., Mysore, P., and Littman, M.L.: ‘Activity recognition from accelerometer data’, in Editor (Ed.)^(Eds.): ‘Book Activity recognition from accelerometer data’ (Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2005, edn.), pp. 1541
30 Meir, R., and Rätsch, G.: ‘An introduction to boosting and leveraging’: ‘Advanced lectures on machine learning’ (Springer, 2003), pp. 118-183
31 Quinlan, J.R.: ‘Bagging, boosting, and C4. 5’, in Editor (Ed.)^(Eds.): ‘Book Bagging, boosting, and C4. 5’ (1996, edn.), pp. 725-730
32 Todorovski, L., and Džeroski, S.: ‘Combining classifiers with meta decision trees’, Machine learning, 2003, 50, (3), pp. 223-249
33 Gaddam, S.R., Phoha, V.V., and Balagani, K.S.: ‘K-means+ id3: A novel method for supervised anomaly detection by cascading k-means clustering and id3 decision tree learning methods’, Knowledge and Data Engineering, IEEE Transactions on, 2007, 19, (3), pp. 345-354
34 Agrawal, R., and Srikant, R.: ‘Fast Algorithms for Mining Association Rules in Large Databases’. Proc. Proceedings of the 20th International Conference on Very Large Data Bases1994 pp. Pages
35 Srikant, R., and Agrawal, R.: ‘Mining Sequential Patterns: Generalizations and Performance Improvements’. Proc. Proceedings of the 5th International Conference on Extending Database Technology: Advances in Database Technology1996 pp. Pages
36 Weiss, G.: ‘Multiagent Systems’ (MIT Press, 2000. 2000)
37 Jakkula, V., and Cook, D.J.: ‘Learning Temporal Relations in Smart Home Data’, in Editor (Ed.)^(Eds.): ‘Book Learning Temporal Relations in Smart Home Data’ (2007, edn.), pp.
38 Gopalratnam, K., and Cook, D.J.: ‘Active LeZi: An Incremental Parsing Algorithm for Sequential Prediction’. Proc. FLAIRS Conference2003 pp. Pages
39 Bouchard, K., Bouchard, B., and Bouzouane, A.: ‘Discovery of Topological Relations for Spatial Activity Recognition’, in Editor (Ed.)^(Eds.): ‘Book Discovery of Topological Relations for Spatial Activity Recognition’ (IEEE, 2013, edn.), pp.
40 Egenhofer, M.J., and Franzosa, R.D.: ‘POINT-SET TOPOLOGICAL SPATIAL RELATIONS’, International Journal of Geographical Information Systems, 1991, 5, (2), pp. 161-174
41 S. Z. Selim, and Ismail, M.A.: ‘K-means type algorithms: a generalized convergence theorem and characterization of local optimality’, IEEE Transactions on Pattern Analysis and Machine Intelligence 6, 1984, pp. 81–87
42 Ester, M., Kriegel, H.-P., Sander, J., and Xu, X.: ‘A density-based algorithm for discovering clusters in large spatial databases with noise’, in Editor (Ed.)^(Eds.): ‘Book A density-based algorithm for discovering clusters in large spatial databases with noise’ (Kdd, 1996, edn.), pp.
43 Sander, J., Ester, M., Kriegel, H.-P., and Xu, X.: ‘Density-based clustering in spatial databases: The algorithm gdbscan and its applications’, Data Mining and Knowledge Discovery, 1998, 2, (2), pp. 169-194
44 Liu, S., Liu, Y., Ni, L.M., Fan, J., and Li, M.: ‘Towards mobility-based clustering’. Proc. Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, Washington, DC, USA2010 pp. Pages
45 Palmes, P., Pung, H.K., Gu, T., Xue, W., and Chen, S.: ‘Object relevance weight pattern mining for activity recognition and segmentation’, Pervasive Mob. Comput., 2010, 6, (1), pp. 43-57