Bibliographic Essay on Data Fusion

Clément Hérouard and Marie-Morgane Paumard

CONTENTS

Abstract

TACOMA is working on behaviors recognition in smart home, and notably on context awareness. To compute the context, the team uses belief function theory, which implies to figure out mass functions. However, such computation is complex. In this bibliographic essay, we will present two clustering methods to calculate mass functions. We present two clustering methods: k-Nearest-Neighbors (k-NN) and Overlapping K-Means (OKM). The former is based on supervised learning and the latter deals with unsupervised learning. We also provide preliminary results using k-NN showing that the methods is interresting to generate mass function.

Index Terms

Smart home, belief function theory (BFT), mass function, clustering, k-Nearest-Neighbors (k-NN), Overlapping K-Means (OKM)

I. INTRODUCTION

Born in the 1980s, smart home aims to improve the comfort and the security of inhabitants. Earlier approaches were based on pre-set scenarios, but proved to be helpless to analyse and adapt to user behaviors, even for one of the most common feature of smart home —automatic lighting. Those were supplanted by context-based approaches. Dey [1] defined the context as *"any information that can be used to characterize the situation of an entity"*. To distinguish between cooking and washing dishes, a computer will use the context: it can be the water consumption or the electric burner temperature. The fusion of those data is what makes recognition a difficult process. Still, humans use those information to better understand their environment. This is why context-based approaches are prominent in smart home nowadays. It will allow us to provide innovative services.

Context awareness is a cornerstone of high-tech industry, especially in domains such as smart home, where the system must adapt to the inhabitants actions and where captors are numerous and heterogeneous. Nevertheless, this

F. Weis, Y. Maurel and A. Richez are with TACOMA, INRIA Rennes.

growing number of sensors causes data profusion. To be useful, data needs to be filtered, aggregated and merged. The team we work with, TACOMA, notably studies how to quantify the interest of each.

TACOMA aims to provide a user-friendly domotic system which is non intrusive: that means no microphone nor camera will be used. Such choice is motivated by user acceptance; one can mention costs, privacy protection or freedom to not wear sensors (even a watch or a smartphone). Other expected qualities are the ease of use and the ease of maintenance and configuration. The former involves that any user can use it on daily basis: the system will not need cumbersome configuration. The latter induces that any installer can make required adjustments without arduous manipulations. Moreover, to avoid data leakage, context evaluation will be done locally.

Though, this initialisation phase is burdensome and can induce a lot of potential mistakes. To work well, sensors need to be calibrated to adapt to the room. Because each place is different, an installer must know how many sensors he wants to and where to place them. Admitting that the calibration is optimal, the system will need time to learn habits of the users. It must know how correct are its guesses on the activities of the inhabitants to make the right decision. To do so, it may use learning algorithms.

Tacoma proposes to use the belief function theory to address the context recognition problem. Although this theory proves to be effective, it requires an heavy calibration process. Our goal is to ease this process by generating automatically the mass function, which are necessary tools to compute the belief function theory.

Among machine learning algorithms that are suited for this purpose, unsupervised learning seems to be a good starting point, since the system will learn by its own. Although, to be proficient, such system will make many errors and will need very large amount of data. Even if this approach is optimal, the industry will find no interest in such time-consuming solution: there is no chance a client want to use it. A second approach is supervised learning. This method is based on giving the system sample of data, where sensors values are associated with their context, letting the system draw conclusions. A third approach is reinforcement learning, but we will not examine it.

In this bibliographic essay, we develop the state of the art in data fusion and compare the different methods to cluster data. First, we describe the research of TACOMA group and present Pietropaoli's thesis [2], we then resume and analyse three papers that deals with the clustering methods: K-Means [3] is a well-tried method of unsupervised learning, Overlapping K-Means [4] add interesting functionnalities and k-Nearest-Neighbors [5] is an easy to implement algorithm of supervised learning. Last, we present few tools we may use further.

II. Building context awareness

To adapt properly to the inhabitants, we want to be able to recognize various activities: sleeping, reading, cooking, washing dishes et cæterea. Some of them looks very similar : how can we discern if someone is reading on a bed from someone which is sleeping? However, a smart home should be able to determine if it must turn the light off. To discriminate two similar action, a domotic system should use the context awareness: it leads to smoother and less frustrating services. Pietropaoli [2] suggests method: first of all, it proposes a way to represent data from sensors. Then, it convert those representation into mass functions. Finally, the mass functions are merged, allowing to decide what context aware action is pertinent.

A. Representing the data from a sensor, belief function theory

To extract the context from the data acquired by the sensors, data is collected from each sensors and then merged. This is easier than managing all the sensors simultaneously and it simplifies the integration of new sensors. Hence, we need a method to represent acquired data. The chosen representation must comply with the following conditions:

- **Measure of incertitude:** some sensors do not produce relevant data or they may be defective. They should provide data on how credible are the acquired information.
- **Measure of imprecision:** defining the exact context may be harsh and unnecessarily ambitious. For example, if the system needs to know if someone is sleeping, knowing that this person is sitting or standing is precise enough to compute that this person is not sleeping.

These two properties are required to be able to provide better and more adapted services: the level of incertitude and imprecision allows a service to take better decision depending on the context. For instance, a critical service may require a higher level of certitude and precision than a classical service (e.g. comfort management, ...)

To satisfy these constraints, we use a mathematical framework named Belief Function Theory. In his thesis, Pietropaoli [2] explains in those fundamental concepts, such as mass functions.

1) Definition of mass functions: Let Ω be the set of possible states; an element of Ω is called a focal element. A mass function is a function which gives some weight to each subset of Ω.

Example of mass function: Let $\Omega = \{A, B, C\}$ and $X \in \mathcal{P}(\Omega)$.

$$m(X) = \begin{cases} 0.1 & \text{if } X = \{A\} \\ 0.3 & \text{if } X = \{B, C\} \\ 0.6 & \text{if } X = \{A, B, C\} \\ 0 & \text{else} \end{cases}$$

Since the element A can occur in both sets $\{A\}$ and $\{A, B, C\}$, the present mass function does **not** indicate that the focal element A has only 0.1 chance to occurs but somewhere between 0.1 and 0.7 chances. Indeed, because A is member of a third-element set, and because we cannot determine which focal element will occur, we can only say that the chances that A occurs are greater than 0.1.

$\mathcal{P}(\Omega)$ has a particular point which is Ω, called total uncertainty: all the states are possible It means that the more important $m(\Omega)$ is, the less relevant m is.

2) Properties of mass functions: A mass function m verifies some properties :

$$\sum_{A \subseteq \Omega} m(A) = 1$$

$$m(\varnothing) = 0$$

Intuitively, a high value $m(A)$ means that the context is one of the items of A, but we do not have any clue about which element of A it is.

Mass functions can be mistaken with probabilities; however those concepts are very distinct. In probabilities, $\mathbb{P}(A \cup B) = \mathbb{P}(A) + \mathbb{P}(B)$. Here, $m(A \cup B)$ has no link with $m(A)$ and $m(B)$. For instance, if $m(\{Sitting, Lying\})$ is high, the person is sitting or lying, and we cannot discern a case from another. Moreover, if we note Ω the set of possible contexts, we would have $\mathbb{P}(\Omega) = 1$; with the mass functions, the mass of the Ω set is not equal to 1, since $m(\Omega)$ is the uncertainty. When $m(\Omega)$ is high, the data is not pertinent.

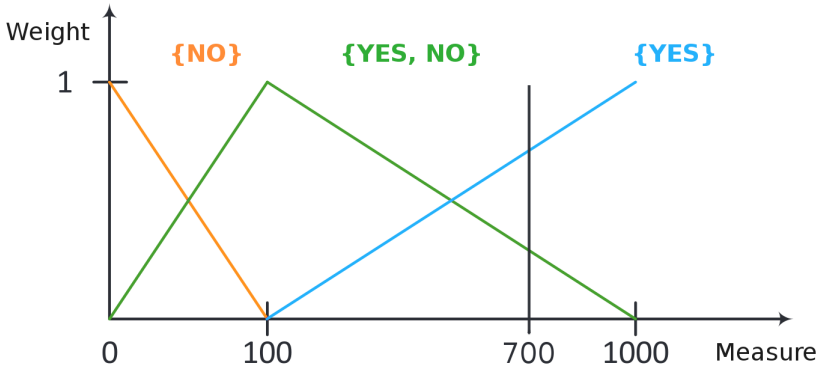

Fig. 1. *Image from Pietropaoli's thesis [2].* Example of mass function for a given presence sensor, that can send values from 0 to 1000. The frame of discernment is $\{Yes, No\}$; depending on the sent values, we can calculate the weight of a specific state. For instance, if a sensor sent 700, we have $m(Yes) = 0.7$ and $m(Yes \cup No) = 0.3$.

3) Use of mass functions: The sensors send a numeric value that we want to associate with a mass function. This association is represented such as in Fig. 1. The abscissa axis corresponds to the sensor value, so each curve shows the value of the mass function of each item of $\mathcal{P}(\Omega)$. For example, if the sensor registers a value of 700, the mass function will be 70 % of YES and 30% of uncertainty.

Henceforth, this method is well-suited to represent data from our sensor and to transform a sensor value into a mass function. What is lacking is a way to to merge mass function and to decide if it is pertinent to act, according to a deduced context.

B. Merging informations

A mass function represents the data from a sensor. Indeed, we have seen that those mass function depict the contexts a sensor seems to perceive and estimate how credible this information is. In order to decide on the context, we want to merge this information. We will present two methods to merge mass function.

The first merging method was created by Dempster [7]. It is associative: to merge several functions, the order has no importance. This method allows to create a function $m_{1\cap2}$ from two mass functions m_1 and m_2. For all set A, we have :

$$m_{1\cap2}(A) = \sum_{B\cap C=A} m_1(B)m_2(C)$$

This method is associative. This means that to merge several functions, it possible to merge them in any order with the previous method.

The second method was created by Dubois and Prade. This method allows to merge a high number of mass functions in a single calculation. Compared to Dempster's, this method favor uncertainty of the mass function, but is far more complex. We will decide what method to use based on the results of the experiment we will conduct.

Thanks to those methods to merge mass function, we can decide the context and the certainty from some criteria[1]. From those results, a system may base its actions on the recognized context and the precision of this recognition. For instance, a critic system —where a mistake would be alarming— will not take action if the precision is not high enough. For lighter application, their tolerance level is lower; we can choose context with low precision.

It is possible to manually determine a mass function for each sensor, however this is a long and cumbersome process. Mapping of mass functions and sensor values cannot be provided per sensor because the environment, the characteristics of each sensors and their position have large influence on the functions. It implies that if an object is moved, the mass function must change. Hence, we must be able to compute a function for each sensor in a specific environment. Our aim is to automate this process. To automate this process, we need to group the data according to the context they induced. To do so, we will use clustering algorithms.

III. Clustering Methods

To generate a mass function, we will use clustering algorithms. Intuitively, we want to combine the data we received from our sensors in order to determine the context. This problem looks like a clustering issue, where the classes are the states of the frame of discernment. Then, the clustering methods seems promising to compute mass functions.

In this section, we present two algorithms to obtain a mass function from both clustering sample and sensors values. These algorithms are based on different clustering methods: k-Nearest-Neighbors (k-NN) and Overlapping K-Means (OKM). The former is based on supervised learning, when the latter deals with unsupervised learning. Supervised leaning means that the users must give himself information to the system to compute data; unsupervised learning means that the system should learn without any intervention of the user.

To obtain belief function from the data, we first record a very large sample of possible contexts. Such sample may need the user to do some specific tasks during the data acquisition. Then, we apply a clustering algorithm to separate data in few clusters, depending on their meaning. Next, we deduce the mass function.

A. K-Means

When it comes to unsupervised learning, **K-Means** [3] is one of the most famous clustering algorithm. It has been designed to separate data into k partitions. Nota that this algorithm is not a good candidate for the generation of mass functions, as explained later. Nevertheless, it is basic and shares many principles with the two thereafter described methods.

[1]Those may be found in Pietropaoli's thesis[2]

Let X be the set of n input values (Fig. 2). We will create k clusters $(\mathcal{C}_j)_{j=1}^k$. To characterize a good clustering, we compute the center μ_j of each cluster $\mathcal{C}_j$. Then, we want to minimize the distances between the points and the center of their cluster:

$$SEE = \sum_{j=1}^{k} \sum_{x \in \mathcal{C}_j} \|x - \mu_j\|^2$$

To minimize the former distance, we randomly generate k points, called μ_j. Then we do the following step until we obtain a sufficient convergence:

- Compute the clusters $(\mathcal{C}_j)_{j=1}^k$ from the μ_j points;
- Update the position of the μ_j points, so they correspond to the center of the newly computer clusters $\mathcal{C}_j$.

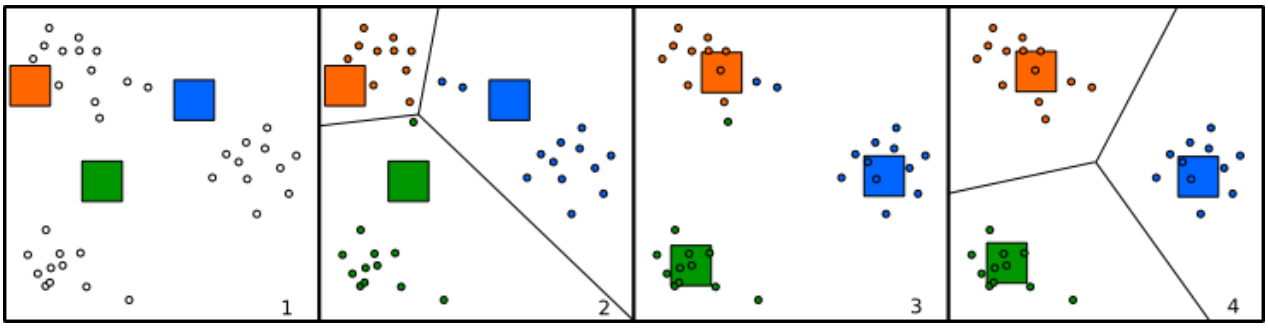

Fig. 2. Example of application of k-Means. 1. Given a set of points; we are looking for a 3-partition clustering. The randomly-cast μ_j points are the squares. 2. The clusters are computed from the μ_j points. 3. The position of the μ_j points are updated. 4. The final clusters are computed from the μ_j points.

This algorithm is rather effective, however the convergence depends on the initialization of the position of the μ_j points. The algorithm may be sluggish, which impose to use a timeout, and may converge to local and inefficient solutions. This last issue may be partially solved by using heuristics on initial points.

This algorithm is too basic for our purpose because it only allows to associate the mass to one single class. It is then impossible to generate mass functions with imprecision, so it does not fulfill our expectations.

B. Overlapping K-Means

In this section, we will present an unsupervised clustering method, which is derived from the K-Means algorithm. It will allow us to obtain automatically generated mass function. It is worth repeating that our mass function handle set that are the union of focal element and that a cluster is the representation of a focal element. Then, we will use the clusters to overlap: the algorithm we present is known as **Overlapping K-Means** (OKM) [4]. For example, rather than trying to determine the absence or presence, we would like to have an overlapping set that would be *absence or presence* (i.e. the intersection of presence and absence clusters). An elegant solution is to allow the overlapping of clusters.

The main difference between K-Means and OKM is the fact a point may be part of several clusters (Fig. 3). We note A_i the set of clusters that contain the points x_i. As before, we note μ_j the average of the position of elements of a cluster $\mathcal{C}_j$. Then, we define the image of the point x_i, which is noted $im(x_i)$. It represents the image we can compute from x_i if we do not know the cluster it is part of. Therefore, $im(x_i)$ is the average of the average of the cluster containing x_i :

$$im(x_i) = \frac{1}{|A_i|} \sum_{\mathcal{C}_j} \mu_j$$

We want to minimize the distance between the points and their images (Fig. 4). The lesser the distance is, the better the point is represented by the clusters. To reach that goal, we adapt the K-Means algorithm. Initially, we randomly choose k points $(\mu_j)_{j=1}^k$ that will be the initial center of our clusters. Then, we execute the following steps until convergence:

- Compute the clusters $(\mathcal{C}_j)_{j=1}^k$ from the points μ_j;

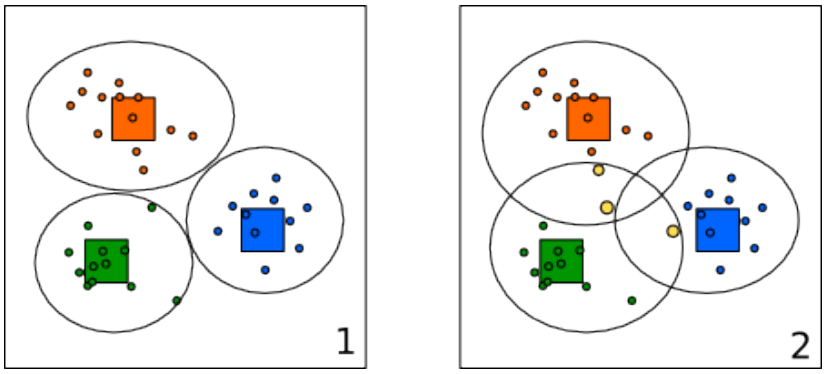

Fig. 3. Differences between K-Means (1) and OKM (2). With OKM, three points (yellow and bigger) are part of two clusters: this method allows uncertainty which is represented by overlapping.

- Update the position of the points μ_j, so they correspond to the center of the newly computer clusters $\mathcal{C}_j$.

To determine the new clusters from the points μ_j, the algorithm proceeds point by point. Given a point x_i, it defines the set A_i of clusters which it belongs. It takes the point μ_j which is the nearest of x_i. Then, it adds $\mathcal{C}_j$ to A_i. Next, it chooses the point $\mu_{j'}$ which is the second nearest point to x_i, after μ_j. It calculates the distance between x_i and its image with A_i then with $A_i \cup \{\mathcal{C}_{j'}\}$. If the distance decrease, it returns A_i. Else, it means that adding $\mathcal{C}_{j'}$ to the set A_i would improve the image of x_i. Therefore, the algorithm adds this cluster to A_i, and it loops in order to add a new cluster to A_i.

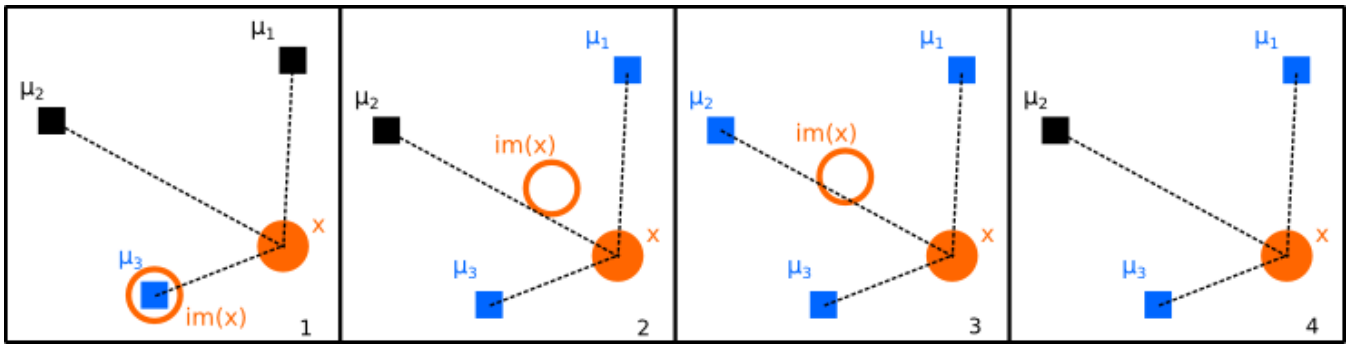

Fig. 4. Steps for associating the point x (orange) to few clusters and building the A set (in blue), given μ_j the center of the clusters $\mathcal{C}_j$. 1. The cluster whose the center is the nearest to x, μ_3, becomes part of A. The image of x, $im(x)$ is computed. 2. The second nearest cluster to x, μ_1, is added to A. Then, image is closer than before. 3. The last step is repeated, however $im(x)$ is farther so 4. the previous clustering is validated.

This algorithms allows to create a overlapping clustering. We have deduced the k centers of the clusters from a dataset. We may then compute a mass function, whose weight is m_{x_0}, from the centers μ_j and the value x_0 acquired by a sensor. The function must represent the chances that x_0 is part of a given cluster. Since the clusters correspond to an element of the frame of discernment, an union of clusters is a focal element of our mass function. Let A be this union of clusters. We note $c(A)$ the centroid of the centers of those clusters. We define m_{x0} as :

$$m_{x_0}(A) = \frac{exp(\|x_0, c(A)\|)}{\sum_{B \in \Omega} exp(\|x_0, c(B)\|}$$

Then, the nearest of the center $c(A)$ the point x_0, the bigger $m_{x_0}(A)$ is. However, this function is one of the possible example. We may replace the exponential function by another decreasing function. This kind of setting will be chosen based on the experiments.

This algorithm is more adapted to our own setting K-Means, because it allows overlapping. However, it has the following drawback. First of all, it is complex to implement. Besides, since the result depends on the initialization, this algorithm is non determinist. Moreover, the method is unsupervised so it is arduous to interpret the clusters: we can say which data are part of the same cluster, but we must find what cluster it is. A possible solution is to

have some reference points whose the cluster they belong is known; yet an unsupervised method does not easily allow it.

C. k-Nearest-Neighbors

We wanted to supersede the drawback of unsupervised learning so we also investigate the use of supervised learning. We studied the k-Nearest-Neighbors algorithm (kNN) used by Denœux in [5]. Supervised means that, during the setup, sensors need the user to provide preliminary information to guide the calibration process. For example, the user had a presence sensor set up. Such sensors distinguish the *absent* state from the *present* state. Therefore, the user needs to be in the room for a while and then to step outside. The gathered information constitutes a base of trusted data, where each acquired data corresponds to a given context.

We have a set of points grouped into clusters and a other point p (Fig. 5). This point p is the acquired value and from which we must deduce the mass function. It will be construct based on the k-nearest points from p, and on the clusters they are part of. The nearest to a cluster a point is, the higher the chance that this point belongs to the cluster. Note that a variable k is not the same that in K-Means: in the later, we have k clause; here we examine k points. So k must be seen as a precision index. From these k elements, we will build k mass functions as shown below. Let a be the point near of p, d the distance between them, $\mathcal{C}$ the context of a, Ω the set of contexts and α a decreasing function. We define f the function so that $f(\mathcal{C}) = \alpha(d)$ and $f(\Omega) = 1 - \alpha(d)$.

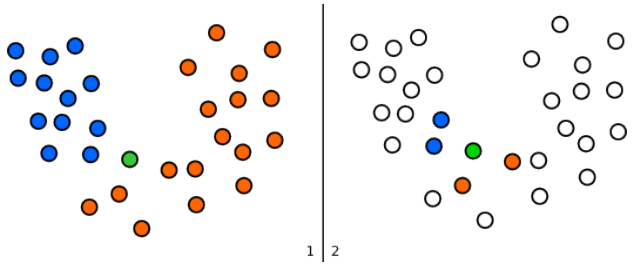

Fig. 5. 1. Given two sets (blue and orange), we want to associate the green point to one of them. 2. To do so, the algorithm selects the k (here, $k = 4$) nearest points to the green one. It computes a simple mass function for each and fusion those. Then the algorithm gives a mass function which indicates the belonging of the green point to each cluster.

If a point from cluster $\mathcal{C}$ is near to the one we want to cluster, the weight of our mass function is on the cluster $\mathcal{C}$ and on the total uncertainty Ω. The uncertainty grows as the distance between points: if their are apart, their are probably not bound. Once we have those k mass functions, we merge them to obtain a single mass function. We may use Dempster's method or Dubois and Prade's. k-NN is configurable: when k is big, the examined points and the merged functions are numerous, so the computation is complex. Still, it allows a high calculation accurency. Moreover, because this method is supervised, the user must specify a context during the acquisition process.

Preliminary results: k-NN was applied to all the possible values of a motion sensor. Its data were acquired without and with someone in the room. We want to define if a person is present. To do so, we will merge the mass function based on Dempster's method (Fig. 6) and Dubois and Prade's (Fig. 7).

The results are encouraging: we get coherent mass functions because they looks like those of Fig. 1, which are the expected functions for a motion sensor. Compared to manually obtained function, the peak is quite high. If our sensor sent a very close to zero value, it does not detect a movement. Those values correspond to a mass function that apply a high weight to a subject absence. This is not desirable because such behavior is too categorical: this could produce a conflict during the fusion of mass functions, making them unusable. We get higher uncertainty with Dubois and Prade's method, but this method is far more time consuming. So, even if the results are encouraging, this method must be tested on other kind of sensors to ensure that it is effective. The method should also be improved to get less categorical functions.

IV. Tools

In addition to the bibliographic study, we learn how to use few effective tools.

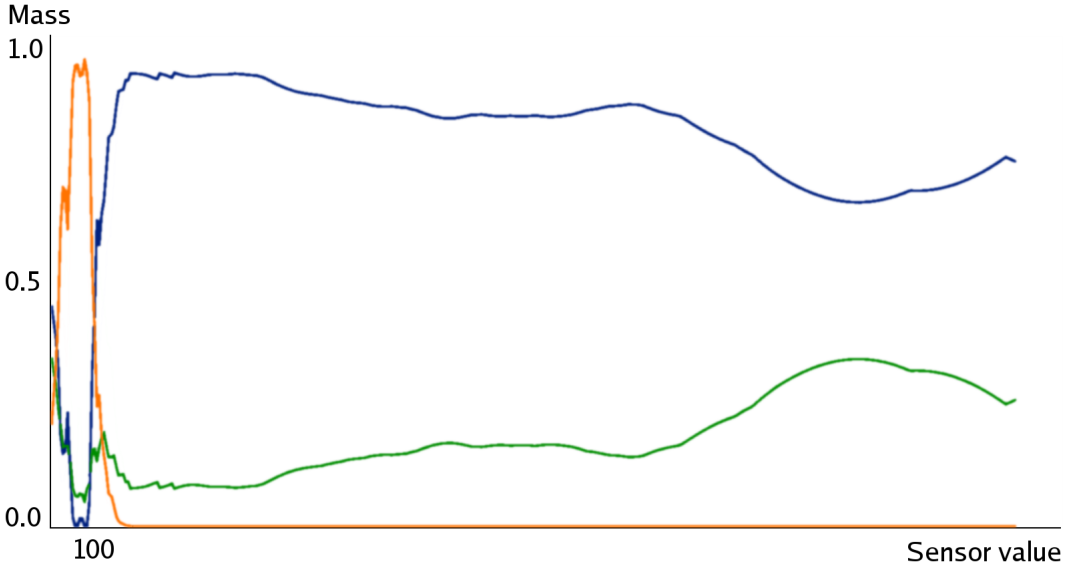

Fig. 6. Application of Dempster's method: mass depending on values acquired by a motion sensor. The orange curve represents the *absence* of a person, the blue one the *presence* and the green one the *presence or absence*. When the sensor value is 100, the blue curve intersects the orange one. Compared to Fig. 1, this graph has a very low mass for the *presence or absence* state. This means the uncertainty is not well-represented. Having such harsh separation between the *absence* and the *presence* can induce errors in the system: it will suppose that there is no uncertainty where it is, in fact, high.

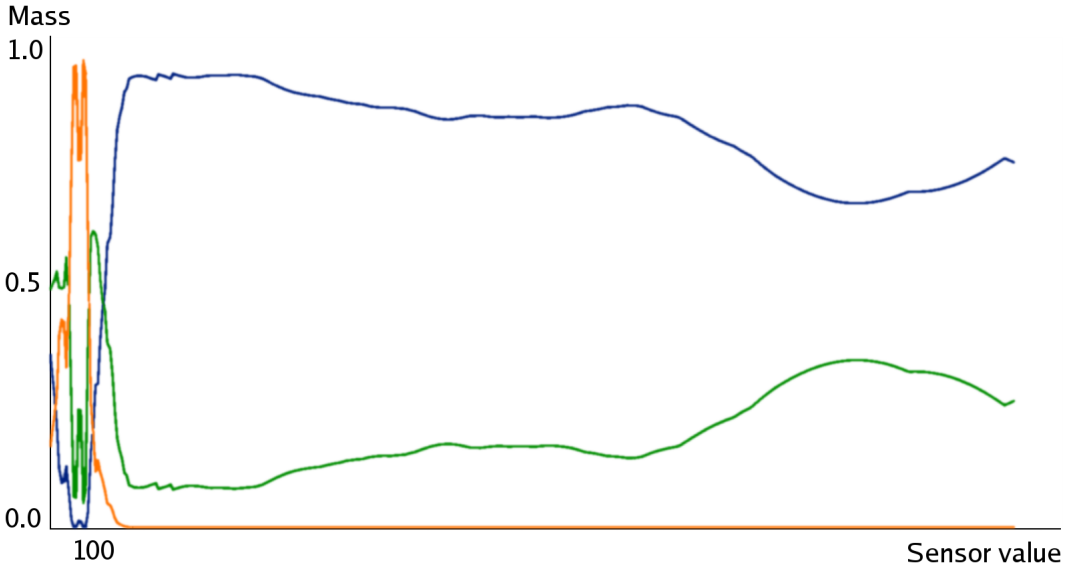

Fig. 7. Application of Dubois and Prades's method: mass depending on values acquired by a motion sensor. The same colors are use than in Fig. 6. However, when the sensor value is 100, we have a high value for the *presence or absence*, meaning that the uncertainty is represented.

a) R language: R is a software environment for statistical computing [6]. It is widely used among statisticians and data miners: indeed, it comes with a tremendous number of libraries. We use R because it includes libraries for both KNN and OKM which will allow us to quickly prototype. In the book we read, example for those methods were given and clearly explained.

b) Java: We use java to perform few computation on our data. Our choice was made because of its properties on dynamic loading at runtime.

V. Conclusion

In this bibliographic essay, we have done a brief state of the art on context awareness. In particular, we discuss on the automatic construction of mass function. To take up this challenge, we need to combine data so we choose to study two clustering methods based on different kind of machine learning algorithms. In supervised learning, we want to infer a function from a training set, containing both input object and a desired output value. On the contrary, the problem of unsupervised learning is that of trying to find hidden structure in unlabeled data. In smart home, we give priority to supervised learning: we have to give plenty of information —which needs tiresome and accurate settings for sensors. However, it is highly improbable that a computer can extract a context from unlabelled data in a short period of time.

We have presented three methods of clustering.

- The first one is based on unsupervised learning. **K-Means** does not fit with our expectations, since it does not allow overlapping.
- The second one is based on the previous method. **Overlapping K-Means** will not require any manipulation from the user during the learning process. However, it is very complex to implement and to interpret. In addition to that, the examples that were proposed in the paper are far from our issues: maybe this algorithm will deceive our expectations.
- The third one uses supervised learning. Since it needs a sample set, **k-Nearest-Neighbors** is harder to setup. However, it allows the system to interpret the meaning of the values faster than with a unsupervised algorithm. We have already tried it on data and the results are promising. Nevertheless, we must proceed to adjustements to obtain a fine configuration. It is also possible that those encouraging results depend on the sensor we used: maybe with another kind of sensor, the result would have been bad.

Schedule

Our goal for the next few months is to apply OKM and kNN on a given context. We will study a particular sensor, a motion sensor and we will evaluate the listed above methods and compare them. We then would like to try different sensors in order to compare their behaviors depending on whether algorithm is used.

We consider following this schedule :

Week 1: Elaboration of a scenario and the experimental protocol. We will especially take care of the following points: make replication of the experiment easy and figure out a way to determine the accuracy of our measure.

Week 2: Study of the tools: R and belief function theory library.

Weeks 3-5: Experiment with kNN. We would like to find the best way to initialize the training set and to adjust the sensors. We will test again Dubois and Prades.

Weeks 6-8: Experiment with OKM. We are notably interested in the initialization and in the changes involved by overlapping.

Week 9: Comparison of kNN and OKM, overall evaluation of our method.

References

[1] Dey, Anind K. (2001). *Understanding and Using Context*. Personal Ubiquitous Computing.

[2] Bastien Pietropaoli. *Reconnaissance de contexte stable pour l'habitat intelligent*. Ubiquitous Computing. Universite Rennes 1, 2013.

[3] Lloyd., S. P. (1982). *Least squares quantization in PCM*. IEEE Transactions on Information Theory 28 (2): 129–137.

[4] Sebastien Regis, Andrei Doncescu, Makoto Takizawa and Guillaume Cleuziou. *Initialization of masses by Okm for the belief function theory. Application to System Biology*.

[5] Thierry Denoeux. *A k-nearest Neighbor Classification Rule Based on Dempster-Shafer Theory*

[6] Brian S. Everitt and Torsten Hothorn. "A Handbook of Statistical Analyses Using R". Second Edition, CRC Press, 2010.

[7] Dempster. *Upper and lower probabilities induced by a multivalued mapping*. The Annals of Mathematical Statistics, 38(2), 1967.