Hybrid system to analyze user's behaviour

Valérian Guivarch
Institut de Recherche en Informatique de Toulouse
Toulouse, France
valerian.guivarch@irit.fr

Javier Bajo
Artificial Intelligence Department. Polytechnic University of Madrid
Madrid, Spain
jbajo@fi.upm.es

Juan F. De Paz
University of Salamanca, BISITE Research Group,
Edificio I+D+I, 37008 Salamanca, Spain
fcofds@usal.es

André Péninou
Institut de Recherche en Informatique de Toulouse
Toulouse, France
andre.peninou@irit.fr

Gabriel Villarrubia
University of Salamanca, BISITE Research Group,
Edificio I+D+I, 37008 Salamanca, Spain
gvg@usal.es

Valerie Camps
Institut de Recherche en Informatique de Toulouse
Toulouse, France
valerie.Camps@irit.fr

Abstract— The evolution of ambient intelligence systems has allowed for the development of adaptable systems. These systems trace user's habits in an automatic way and act accordingly, resulting in a context aware system. The goal is to make these systems adaptable to the user's environment, without the need for their direct interaction. This paper proposes a system that can learn from users' behavior. In order for the system to perform effectively, an adaptable multi agent system is proposed and the results are compared with the use of several classifiers.

Keywords; classifiers, multi-agent systems, ambient intelligence

I. INTRODUCTION

Ambient intelligence is widespread nowadays; intelligent systems are common in use and facilitate our work. Some systems can monitor user behavior, detect strange behavior patterns and send alerts to control centers. Since it is not possible to tackle the variety of behaviors and situations and include their prediction in the system, ambient intelligence systems are designed to learn and evolve on their own based on the users' behavior. With the evolution of ambient intelligence, the internet of things will attain great importance as devices will be able to interact with each other and with their users in their houses. In context-sensitive systems, the environment should be able to adapt to the environment with minimal user interaction, for that reason in this paper the use of adaptive multi-agent systems to learn and predict user behavior is analyzed.

Currently, data mining is the most common technique for detecting patterns from different data bases, among them classifiers, neural networks or statistics. With regard to classifiers, different techniques can be applied, depending on the decision rules, functions, bagging, boosting etc. In the case of a neural network it is possible to apply a multilayer perceptron or an RBF neural network [20]. In order to create regression models a statistical test could be used for the extraction of relevant parameters. The main disadvantage of these systems is that each of them has to start its learning process from the beginning and cannot learn from one which has already evolved.

The aim of this study is to create a context aware system which is able to learn from users' behavior. Classifiers have been used for this purpose. The context obtained from different sensors and actuators is used by classifiers in order to create and train algorithms. In this way, the system obtains the value (number?) of different devices and actions made by the user. If the accuracy is low, system learning must be recalculated again. Together with the adaptive multi-agent system it will be able to adjust itself continuously and learn dynamically, in accordance with the perceptions received. In this work, Amadeus; an Adaptive Multi-Agent System (AMAS) is presented. The proposal is also proved with more conventional algorithms in order to show their accuracy.

This paper is divided into the following sections: section two contains the state of the art about conventional techniques and

AMAS, section three contains the proposal and section four contains the results and conclusions.

II. State of the art

A. Classifiers

In the literature, in many cases algorithms are used for making predictions. The training of some techniques, such as decision rules and decision trees, can be easily interpreted and explained [15]. Decision rules are used in many case studies where they are applied to the analysis of bioinformatics.–The decision rules are used in many cases studies since analysis accident [14] to bioinformatics [16][17]. Decision rules ranging from accidents to bioinformatics. In other cases, it is possible to use other classifiers based on statistics (e.g. Bayesian networks [18]). However, such predictions are black boxes and they don´t allow classification. Besides these classifiers have a problem because they require the discretization of data. Support Vector Machine (SVM) are very popular in bioinformatics [19], since they transform original data into a high dimensional space. SVMs use kernel functions in order to separate data linearly SVMs are efficient compared to distance based classification algorithms. Some studies combine several classifiers with experts knowledge and other techniques such as neural networks [20], decision tree, Random Forest[21] or ensembles methods [10][11].

B. Adaptive Multi-Agent System

The systems based on Artificial Intelligence, are used to solve complex problems, where an initial solution is not known. The use of Multi Agent Adaptive Systems (AMAS) combines different self-organizing techniques in order to reach the goal successfully. The different entities that make up the system, must communicate using a common language, one which can be understood by all and can ensure collective behavior. Cooperation between agents is based on the application of three rules, applied accordingly, depending on the problem that needs to be solved:

- cper: Any entry must be identified, understood and treated unequivocally.
- cdec: Any information coming from their perception should
- aid reasoning and decision making.
- cact: Their reasoning capacity must be trained so that it leads to actions that are useful for the environment or for the rest of the entities that make up the system.

Another aspect of AMAS that needs to be understood is the ability of agents or elements that make up the system to detect or prevent non-cooperative situations (NCS). An NCS occurs when one of the system entities does not verify one of the three rules mentioned above. The following describe the NCS that can take place if one of the rules is not verified:

- (cper) Situation of ambiguity and misunderstanding.
- (cdec) Situation of incompetence and unproductivity.
- (cact) Situation of conflict.

When using an AMAS based system, it is vital to design and apply the different cooperation methodologies between agents. In doing so, we determine the role each agent has in ensuring that the three AMAS rules are fulfilled.

When designing a self-adaptive multi agent system, special attention should be given to the following tasks:

Determine the entities that compose the MAS and determine the main capabilities of each one and the functionality it offers to other components in the system.

Predict any occurrences that can lead to non-cooperative situations (NCS).

Once the situation is detected (NCS), it is necessary to determine the set of actions that should be performed in the system to undo the non-cooperative state and transform it into a totally cooperative state. This method has helped to solve complex computing problems, for example, problems related to process execution, response times, real-time processing, bioinformatics, etc. .

The capacities of AMAS, characterized mainly by the execution of tasks in a distributed and dynamic way, make it possible to solve complex problems such as predictive learning, or applications related to environmental intelligence.

III. Proposed System

In order to evaluate the system proposed in this paper, a multi-agent architecture (AMAS) called Amadeus has been designed based on a MAS called PANGEA. The main objective is to be able to interconnect a set of hardware devices in a simple way, adapting the behavior in function of the context through techniques of environmental intelligence.

Amadeus uses unsupervised observation and learning techniques, memorizing the different operations a user makes with each physical device. If the user performs the same action or task repeatedly, the system can learn that these tasks constitute habitual behaviors of the user and can use these for the realization of future prognosis.

As can be seen in Figure 1, each device has a case of Amadeus. Data agents are responsible for capturing the information collected by the different sensors, the user agent represents the user´s level of satisfaction, expressed by a value between 0 and 1.

The operation of the context agents and the controlling agent is described below.

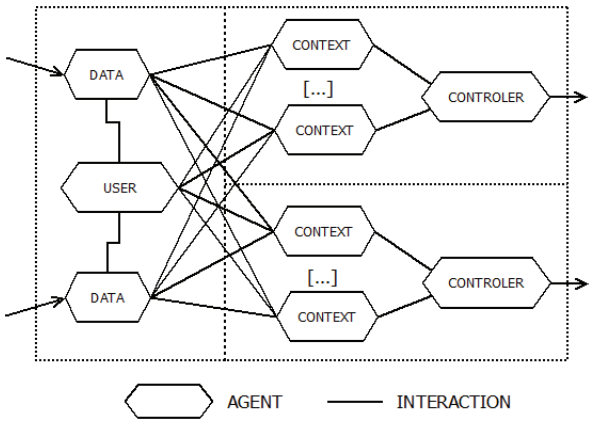

Fig 1. General structure of the system

- ***General Operation***

The main role of the controlling agent is to predict the user´s actions. A controlling agent works in association with several context agents, specifying the potential actions each of them can take. Context agents will be in charge of evaluating these actions to determine which is the best option for end user satisfaction. These agents must have the computational capabilities to detect when those assessments have had an undesirable effect on the system and should not be considered valid in the future.

- **Controller *agent***

The operation of the controlling agent is described in detail below. Their behavior is based on a 3 phase cycle. The first phase, called the perception phase, consists of receiving service requests from the context agent. In each request, fields note the service that is demanded, the description of the required function, as well as forecast the impact this action will have on user satisfaction. This forecast is always accompanied by a confidence margin that gives information on its accuracy.

Subsequently, the controlling agent evaluates which action is best for user satisfaction. It may happen that two agents make a service request with the same characteristics, in this case they will opt for the request that has the greatest certainty. This phase is called the evaluation phase.

Finally, the execution phase occurs, where the controlling agent opts for the context agent which sends the most satisfactory service request and executes a corresponding action.

- **Context *agent***

Each context agent that makes up the AMAS, presents a set of values that make up the validity of the agent. An agent is in a valid and secure state when the value it receives from each of the remaining elements of the system is contained in this range or set of values. Because of this, each time an agent invokes a certain functionality, a contextual validation process begins, which can lead to a safe or unsafe state. The upper and lower limits of the set of possible input values of each agent are delimited by an Adaptive Value Tracker (AVT), which allows the determination of the value of a dynamic variable in a given space.

Once the validation process is completed and the agent has determined that it is in a safe state, the controller notifies of a proposed execution of a terminated service. This request contains the description of the requested service as well as the different external parameters that are necessary to invoke the functionality. This request has an indicator, a numerical value that ranges from -1 to 1 that indicates the impact that the execution of the service has on user satisfaction. In addition, the request has a numerical value ranging from 0 to 1 corresponding to the (AVT), offering a prognosis about the security that the agent has that the proposal of execution is carried out correctly. A context agent only communicates functionality demand proposals to the controlling agent when it resides in a secure state.

In an AMAS system, different context agents can coexist and be in charge of different services. When a system entity requests a service from an agent, the controlling agent determines the best candidate.

The criteria that the controlling agent can apply to determine the candidate agent are based on various factors, for example, the hardware characteristics, where the context agent resides, the CPU time used, the demand for services, etc. Only one candidate agent can be active for each service that is offered in the system at any given time. When the offer of service is concluded, the level of satisfaction of the user is stored, in order to compare this value obtained with the initial prediction that was made. From this comparison three different scenarios are drawn:

In an AMAS system, different context agents can coexist and be in charge of different services. When a system entity requests a service from an agent, the controlling agent determines the best candidate.

The initial prediction of the context agent coincides with the final value obtained from user experience, in this case the context agent increases its confidence margin.

The prediction was not exactly the same as the one offered by the user, however the prediction is correct, in this case the context agent adapts the predicted value to the value supplied by the user and increases his confidence margin.

In the latter case, the context agent has made an incorrect prediction, it knows that the service request has not been executed as expected, which is why it must readjust the set of input values in order to increase forecast accuracy in the future. In this case, the context agent's confidence margin will decrease.

Context agents, as mentioned above, must evaluate their prediction of user satisfaction and evaluate whether it is correct or not. The higher the confidence index of a context agent, the greater the possibility that the prediction has been correctly performed. To calculate the confidence, the following formula is used:

$$T_{t+1} = T_t * \lambda + F * (1 - \lambda)$$

T in t+1 is calculated according to Tt, F feedback in range [0,1], and λ determined experimentally.

When an AMAS system entity requests a service or functionality for the first time, the system automatically instantiates a context agent. The instantiated agent starts with a confidence index of 0.5. The one with a set of values of the context agent is initialized to the set of real values supplied in the initial request.

The interaction produced between the agents of context and the controlling agents endow the system with a learning model since there is a continuous and dynamic adaptation of the confidence level of the context agent. This learning model is based on the modification of the range of input values that make the agent in a safe state, through the validation of the three meta-rules.

IV. CASE STUDY

The aim of the case study was to analyze the performance of multi-agent systems with conventional algorithms, for example, classifiers such as Bayes, LMT, decision tree or SVM. In the case study, the system was used to predict user´s behavior. The data from the sensor and the user's behavior are stored each time as cases while in Amadeus these data are considered a contextual situation.

The system data are simulated by an ambient intelligence system that defines the user's behavior in a virtual environment. In this way, it is possible to obtain more data than in a real environment. The sitting room in the simulator contains an electrical lamp and a window shutter. It will also contain a sensor which will record the luminosity and presence. There will only be one user in the house and they will be able to use the different rooms. When they arrive to the sitting room they will begin an interaction with the simulator. However, we have to consider that the user may want to save energy. The user will intend to use natural light for as long as possible and will only turn on the lamp if it gets too dark. If the room is too bright then the user will try to save energy by turning off the lamp. They may also use the shutter to reduce the brightness of the room, this will help to keep the luminosity of the room in random value in an interval when the user is inside.

The user will move randomly between different parts of the house. When in the sitting room, they will turn on the light if the luminosity is lower than 55. The system will always select a random value between 50 and 60, in order to avoid simple patterns in the user's behavior. The simulator was used for 13 days and the system generated one data per minute. Then, using various techniques, these data were used to predict user behavior. Two experiments were carried out and they had different arrangements. In the first experiment, one day was dedicated for training and the other for data testing. In the second experiment the system was trained within the first three days and the other days were spent on studying system performance and comparisons with classifiers.

Table 1 shows an example of simulator generated data. The table contains the day, the number of cycles in a concrete day, user's satisfaction, the luminosity and presence, where 1 means presence of the light. Value 1 for blind is when the shutter is open, value 1 for light when the lamp is turned on. In this case the user has defined maximum luminosity to 90, with more luminosity the user's satisfaction decreases. We can see that user satisfaction is the lowest at a threshold of 26%, when the user acts and turns off the light, then user satisfaction increases.

The objective of these tests is to foresee users´ behavior. When we are using classifiers, the first two columns are ignored to prevent them from predicting according to these columns.

Day	Cycle	User Satisfaction	Luminosity	Presence	Blind	Light	User Decision
0	480	34%	91,97	1	1	1	1
0	481	32%	92,06	1	1	1	1
0	482	31%	92,16	1	1	1	1
0	483	29%	92,26	1	1	1	1
0	484	27%	92,35	1	1	1	1
0	485	26%	92,45	1	1	1	0
0	486	100%	74,03	1	1	0	0
0	487	100%	74,11	1	1	0	0
0	488	100%	74,18	1	1	0	0
0	489	100%	74,26	1	1	0	0
0	490	100%	74,33	1	1	0	0

TABLE I. EXAMPLE OF DATA, AS PERCEIVED BY THE DIFFERENT LEARNING ALGORITHMS.

V. RESULTS AND CONCLUSIONS

The system was tested with data obtained during 13 days, the data were collected for each minute. The objective of the tests was to analyze the adaptability of the system. The system was trained with the data obtained on the first day. Then new cases were introduced into the adaptive multi agent system and the classifiers and user's behavior were predicted. After prediction, new case was introduced into the system so that it could learn it. This process was repeated until the 13th day. Given classifiers, the cases were rebuilt if the classification was wrong.

To analyze the adaptably and evolution of the system two parameters were considered. In order to determine the changes in user preferences, the number of days and the case in day, the number of case is restarted each day.

The adaptive multi agent system was compared with classifiers J48, the Naive Bayes, SVM and LMT, the results are shown in the figure 1. The figure shows the prediction of the light state according to the parameters shown in table 1. The Y axes contain the number of errors, and the X axes the cases. As can be seen, the figure shows accumulated errors for each technique. It is visible that the errors are more or less similar until the multi agent system increases the errors and after this moment continues with a similar efficiency than the other methods. LMT had the highest accuracy.

A new text was introduced within 3 training days. Figure 2 contains the results obtained. We can see that SMA provides better results than LMT, until the users change its behavior, then LMT provides the best results. However, main problem with this algorithm is that it is not efficient enough because the execution time is very high. The execution time of LMT was 5177 seconds as compared to other techniques which had their execution time below 60 milliseconds.

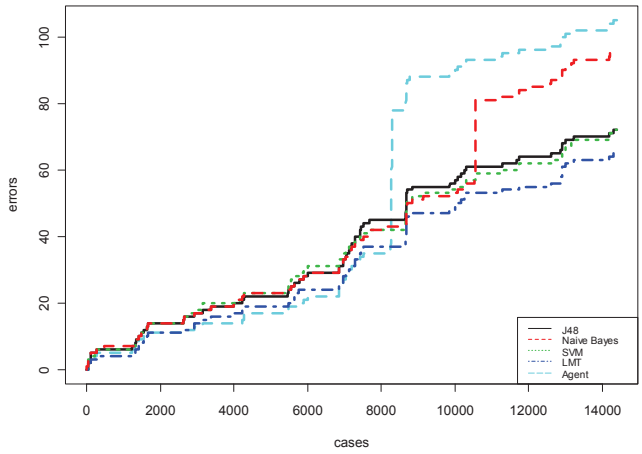

Fig 2. Evolution of the number of errors made by the CBR with learning based on one day.

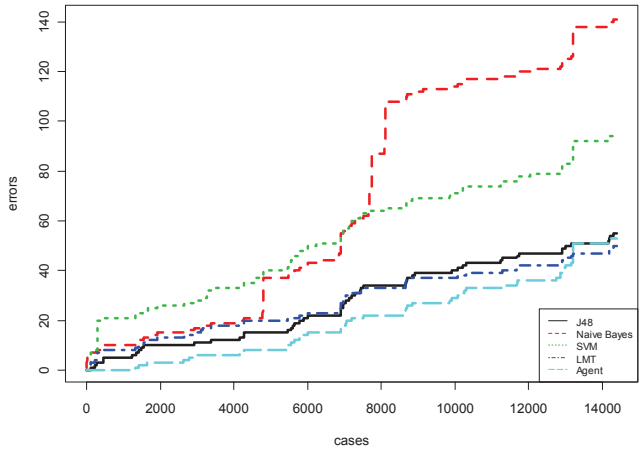

Fig 3. Evolution of the number of errors made by the CBR during a three day learning.

In conclusion, the multi agent system does not have a great capacity of generalization but it provides better performance when new cases arrive to the system. The best characteristic of the multi agent system is that it doesn't have to store the cases in order to retrain them in the system. This makes data processing easier because it is not necessary to store them.

ACKNOWLEDGMENT

This work has been supported by the European Commission H2020 MSCA-RISE-2014: Marie Skłodowska-Curie project DREAM-GO Enabling Demand Response for short and real-time Efficient And Market Based Smart Grid Operation - An intelligent and real-time simulation approach ref 641794.

REFERENCES

1. Aha D., Kibler D., Albert, M.K.: Instance-based learning algorithms. Machine Learning. vol. 6, 37-66 (1991).
2. Bouckaert, R.R. (1995). Bayesian Belief Networks: from Construction to Inference. Utrecht, Netherlands.
3. Cohen, W.W. (1995). Fast effective rule induction. In Proceedings of the 12th International Conference on Machine Learning. Morgan Kaufmann. 115-123 (1995).
4. Holmes, G., Hall, M. and Prank, E. (2007). Generating Rule Sets from Model Trees. Advanced Topics in Artificial Intelligence. 1747/1999, 1-12.
5. Holte, R.C. (1993). Very simple classification rules perform well on most commonly used datasets. Machine Learning. 11, 63-91.
6. Quinlan, J.R. (1993). C4.5: Programs For Machine Learning. Morgan Kaufmann Publishers Inc.
7. Breiman, L., Fried, J.H., Olshen, R.A. and Stone, C.J. (1984). Classification and regression trees. Wadsworth International Group.
8. Duda R.O. and Hart P. (1973). Pattern classification and Scene Analysis. New York: John Wisley & Sons.
9. Vapnik, V. and Lerner, A. (1963). Pattern Recognition Using Generalized Portrait Method, in, 1963, 774–780.
10. Breiman, L. (1984). Bagging predictors. Machine Learning, 24 (2), 123-140.
11. Freund, Y. and Schapire, R.E. (1996). Experiments with a new boosting algorithm. In Thirteenth International Conference on Machine Learning. 148-156.
12. Juan A. Fraile, Yanira de Paz, Javier Bajo, Juan Francisco de Paz, Belén Pérez Lancho: Context-aware multiagent system: Planning home care tasks. Knowl. Inf. Syst. 40(1): 171-203 (2014)
13. Sara Rodríguez, Juan Francisco de Paz, Gabriel Villarrubia, Carolina Zato, Javier Bajo, Juan M. Corchado: Multi-Agent Information Fusion System to manage data from a WSN in a residential home. Information Fusion 23: 43-57 (2015)
14. Joaquín Abellán, Griselda López, Juan de Oña, (2013) Analysis of traffic accident severity using Decision Rules via Decision Trees, Expert Systems with Applications, 40 (15), 6047–6054.
15. Rodrigo C. Barros, 1, , Pablo A. Jaskowiak, Ricardo Cerri, Andre C.P.L.F. de Carvalho, (2014) A framework for bottom-up induction of oblique decision trees, Neurocomputing, 135 (5), 3–12.
16. Mevlut Ture, Fusun Tokatli, Imran Kurt (2009) Using Kaplan–Meier analysis together with decision tree methods (C&RT, CHAID, QUEST, C4.5 and ID3) in determining recurrence-free survival of breast cancer patients, Expert Systems with Applications, 36 (2-1) 2017–2026.
17. Juan F. De Paz, Javier Bajo, Vicente Vera, Juan M. Corchado (2011) MicroCBR: A case-based reasoning architecture for the classification of microarray data, Applied Soft Computing, 11(8) 4496–4507.
18. Seung-Hyun Lee, Kyon-Mo Yang, Sung-Bae Cho (2015) Integrated modular Bayesian networks with selective inference for context-aware decision making, Neurocomputing, 163 (2) 38–46.
19. Juan Francisco de Paz, Javier Bajo, Angélica González, Sara Rodríguez, Juan M. Corchado: Combining case-based reasoning systems and support vector regression to evaluate the atmosphere-ocean interaction. Knowl. Inf. Syst. 30(1): 155-177 (2012)
20. Gabriel Villarrubia, Juan F. De Paz, Dechen Pelki, Fernando de la Prieta, Sigeru Omatu, (2016) Virtual organization with fusion knowledge in odor classification, Neurocomputing, In Press.
21. L. Breiman, Random forestsMach. Learn., 45 (2001), pp. 5–32