Iglesias, J. A.; Ledezma, A.; Sanchis, A. (2013). Evolving systems for computer user behavior classification. *Evolving and Adaptive Intelligent Systems (EAIS), 2013 IEEE Conference on*, Singapore, pp. 78-83.
DOI: 10.1109/EAIS.2013.6604108

Evolving Systems for Computer User Behavior Classification

José Antonio Iglesias
Carlos III University of Madrid
Madrid, Spain
Email: jiglesia@inf.uc3m.es

Agapito Ledezma
Carlos III University of Madrid
Madrid, Spain
Email: ledezma@inf.uc3m.es

Araceli Sanchis
Carlos III University of Madrid
Madrid, Spain
Email: masm@inf.uc3m.es.

***Abstract*—A computer can keep track of computer users to improve the security in the system. However, this does not prevent a user from impersonating another user. Only the user behavior recognition can help to detect masqueraders. Under the UNIX operating system, users type several commands which can be analyzed in order to create user profiles. These profiles identify a specific user or a specific computer user behavior.**

In addition, a computer user behavior changes over time. If the behavior recognition is done automatically, these changes need to be taken into account. For this reason, we propose in this paper a simple evolving method that is able to keep up to date the computer user behavior profiles. This method is based on Evolving Fuzzy Systems. The approach is evaluated using real data streams.

I. Introduction

Observations of human-computer interaction can give us insight into the behavior of the computer users. One of the simplest environments from which we can obtain these observations is UNIX operating systems.

As Greenberg described more than 10 years ago [1], studying UNIX is attractive for many reasons: First, UNIX is widely used, very powerful and potentially complex, and has a broad range of users. Second, if UNIX findings could not be generalized, they would still be valuable in their own right. Also, UNIX has already been studied extensively. Finally, as large groups of diverse people use it at many different sites, studying UNIX is relatively easy to do. Although these reasons were proposed in 1998, nowadays UNIX is still an interesting environment for many different researchers, specially in user behavior modeling.

On the other hand, taking into account the study of Webb et al [2], user models may seek to describe:

1) the cognitive processes that underlie the users actions;
2) the differences between the users skills and expert skills;
3) the users behavioral patterns or preferences; or
4) the users characteristics.

In this research, the user modeling refers to the description of the users behavior patterns. Thus, if we can obtain these patterns, we can create computer user models. Then, observing a new user, we can conclude which is his/her model and to detect if it is similar to any other already seen. In this sense, we can classify users taking into account the created models. In this paper, a user model is acquired implicitly by making inferences about the users from their interaction with the computer.

The goal of this research is to present and to evaluate an easy method for classifying the behavior of a user based on the commands that s/he types. However, as a user behavior changes over time, we propose a method based on Evolving Fuzzy Systems (EFS) which keep up to date the computer user behavior profiles. The approach is evaluated using real UNIX data streams. This mehod can be very useful, for example, in computer intrusion detection.

The evolving classifiers used in this research (*eClass*) were proposed by Angelov et al [3] and it has been applied to a wide range of problems, both benchmarks and real. The use of these classifiers allows us to cope with huge amounts of data, process streaming data on-line in real time, and evolve the structure of a computer user model based on the observed changes. Thus, the created user models are designed and treated as changing models which constantly reflect the changes in the way a user interacts with a command-line interface.

eClass is a fuzzy rule-based (FRB) classifier which uses (fuzzy) rules that evolve from streaming data. An *eClass* (which can start learning ”from scratch”) learns new rules from new data gradually preserving/inheriting the rules learned already. In addition, *eClass* can be defined as a self-developing classifier which has both their parameters but also (more importantly) their structure self-adapting on-line.

This paper is organized as follows: Section 2 provides a brief overview of the background and related work of behavior recognition and EFS. Section 3 explains the structure of our proposal. Section 4 describes the experimental setting and results obtained. Finally, Section 5 contains future work and concluding remarks.

II. Background and Related Work

There is much varied research which models and classifies the behavior of other humans, robots or agents. In some works, a team of a competitive domain is modeled and classified using different methods, such as *Hidden Markov Models* [4], *Deterministic Finite Automatons* [5] or decision trees [6].

In addition, to find out relevant information under the human behavior, many methods have been used: Macedo et al. [7] propose a system (*WebMemex*) that provides recommended information based on the captured history of navigation from a list of known users. Gody and Amandi [8] present a technique to generate readable user profiles that accurately capture interests by observing their behavior on the Web. Pepyne et al. [9] propose a method using queuing theory and logistic regression modeling methods for profiling computer users based on simple temporal aspects of their behavior.

In the computer intrusion detection problem, Coull et al. [10] propose an algorithm that uses pairwise sequence alignment to characterize similarity between sequences of commands. The algorithm produces an effective metric for distinguishing a legitimate user from a masquerader. Schonlau et al. [11] investigate a number of statistical approaches for detecting masqueraders.

Similar to this research, Iglesias et al. proposed an approach for modeling and classifying behaviors from observations (called *ABCD*)[12]. In order to use that approach, the observed behavior needs to be transformed into a sequence of ordered atomic behaviors. Then, the sequence is segmented and stored in a *trie* and the relevant subsequences are evaluated by using frequency.-based methods. *ABCD* was experimentally evaluated in the same UNIX domain proposed in this paper. However, there are two important differences between ABCD and the proposed method in this paper:

1) *ABCD* is based on temporal dependences, and the order of the different commands is essential for the result.
2) In *ABCD* the created user models are fixed and it is not considered that a user computer behavior changes over time.

In order to solve the second of these aspects, the method proposed in [12] is modified by Iglesias et al. [13]. In that research, as a user behavior is not fixed but rather it changes and evolves, the proposed classifier is able to keep up to date the created profiles by using an *Evolving Classifier*. Thus, the idea proposed in [13] is the same that is proposed in this paper; however, the method for obtaining the user models and how they are keep up to date (although both methods are based on evolving systems), is different. The method proposed in [13] is extended to other different domains in [14].

eClass (evolving Classifier) family was introduced in [15] and further developed in [16]. *eClass* is a set of evolving neuro-fuzzy classifiers which take its roots in evolving Takagi-Sugeno (eTS). A set of fuzzy rules that describes the most important features of each class is formed during the training process. Then, these rules are constantly adjusted to the available training data. It is important to highlight that *eClass* does not require parameter optimization as its only parameter 'scale' can be directly inferred from the training data. This technique [17] is based on partitioning the data space into overlapping local regions through *Recursive Density Estimation* (RDE) and associating clusters (respectively fuzzy sets) to them.

As it is explained in [3], the main differences between *eClass* family and a conventional Fuzzy Rule-Based (FRB) classifier are:

- the open structure of the rule-base: *eClass* self-develops on-line starting from scratch, while in a conventional FRB classifier it is determined offline and then fixed.
- the online learning mechanism which takes into account this flexible rule-base structure.

eClass family includes two different architectures and on-line learning methods:

- *eClass0* with the classifier consequents representing class label.
- *eClass1* for regression over the features using first order eTS fuzzy classifier.

Both classifiers (*eClass0* and *eClass1*) are recursive, non-iterative incremental and thus computationally light and suitable for real-time applications. Thus, they been applied in many different areas such as autonomous landmark recognition [18], self-localization and mapping [19], object detection and tracking [20][21], collision avoidance [22], IR spectral data of exfoliative cervical cytology [23], activity recognition from sensor streams [24][25] and, as we already have mentioned, user modeling [26][13].

III. Computer User Behavior Classification Based on EFS

This section introduces the proposed method based on evolving classifiers. The architecture of the proposed method is shown in Figure 1. The following subsections details the different parts of this architecture:

A. Obtaining the UNIX User Models

In order to classify a UNIX user, her/his profile must be created in advance. To apply the proposed classifier, a profile based on *Term Frequency* (TF) is created for each UNIX user. Thus, the frequency of each command (the number of times that a user has typed that command) will be used to describe

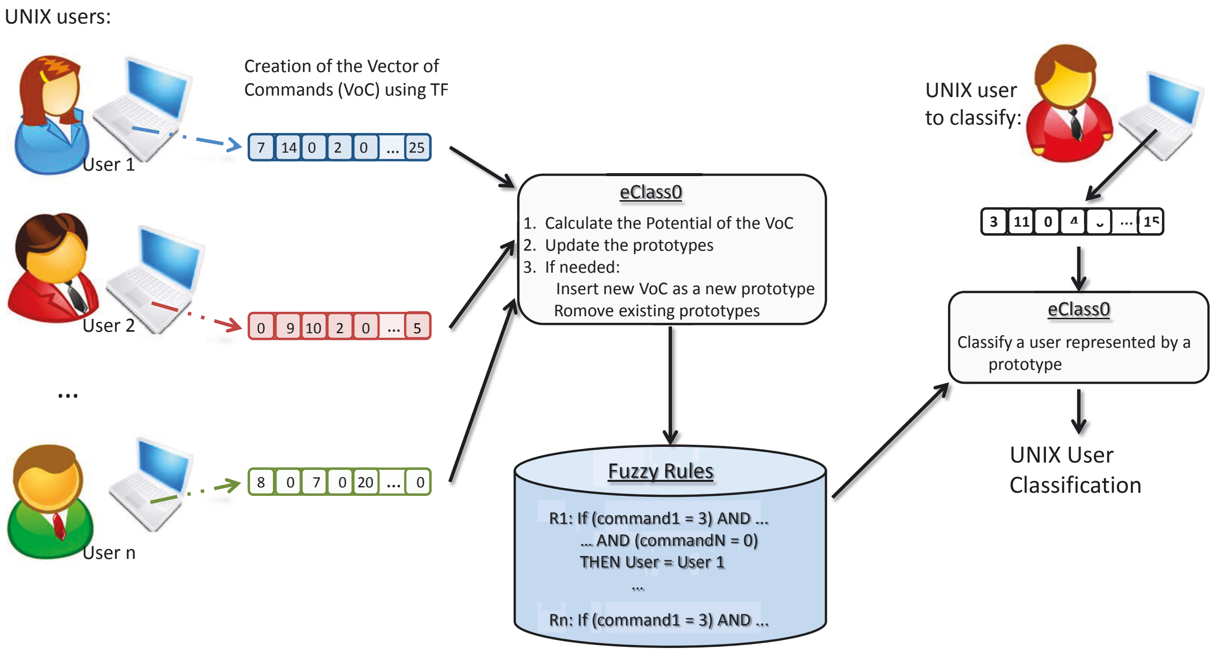

Fig. 1: Architecture of the proposed method

and identify an specific user. Although this is a very simple measure, we will prove that the results are good. Also, and it is more important, we want to create a model as easy and fast as possible. Thus, TF is a really simple measure which will be the base of the created models. We can observe that this measure is a common method often used in Information Retrieval (IR). However, the commands that a user types during an specific shell session, could be considered as the words that appear in a specific document (equivalent to the 'vocabulary' present in that document).

Although it was not the purpose of this research, we could apply other more complex measures. For example, in [27] the TF-IDF (*Term Frequency Inverse Document Frequency*) measure is used in the same domain. However, for applying that measure, we need to know the number of users we are treating, and the number of users who have typed an specific command at least once.

Once the TF of each command is calculated, the model of a UNIX user is represented by the distribution of these values. This model is represented by a vector of values indicating how many times a command has been typed by the users. This vector will be called Vector of Commands (*VoC*). We need to take into account that this representation needs to create and update the different commands typed by the users (it could be defined as *vocabulary* - number of unique commands). However, the vocabulary is easily obtained and it can be updated removing those commands which are not relevant (or are becoming "out of date"). Note that this *VoC* could also be created when the user has finished the shell session on the computer. But in that case, the TF should be normalized taking into account the total number of commands that s/he has typed during that session.

B. *Creating the Fuzzy Rules using* eClassO

eClass0 possesses a zero-order Takagi-Sugeno consequent, so a fuzzy rule in the *eClass0* model has the following structure:

$$Rule_i = IF(X_1\ is\ P_1)\ AND \ldots AND\ (X_n\ is\ P_n)$$
$$THEN\ UnixUser\ =\ UnixUser_i \quad (1)$$

where i represents the number of rule; n is the number of input variables (number of different commands); the *VoC* X stores the TF (Term Frequency) of the input commands, and the *VoC* P stores the TF of the commands of one of the prototypes (cluster centre) of the corresponding class (user). $UnixUser \in \{$set of different Users$\}$.

The *eClass0* model is composed of several fuzzy rules per class (the number of rules depends on the heterogeneity of the input data of the same class). In this case, each class represents an specific user. Although it is not considered in this work, a class could also represent a set of users with similar characteristics.

During the training process, a set of rules is formed "from scratch" using an evolving clustering approach to decide when to create new rules. The inference in *eClass0* is produced using the "winner takes all" rule and the membership functions that describe the degree of association with a specific prototype are of Gaussian form. The *potential* (Cauchy function of the sum of distances between a certain data sample and *all* other data samples in the feature space) is used in the partitioning algorithm. However, in these classifiers, the potential (P) is calculated recursively (which makes the algorithm faster and more efficient). The potential of the k^{th} data sample (x_k) is calculated [3] by the equation 2. The result of this function represents the *density* of the data that surrounds a certain data sample.

$$P(x_k) = \frac{1}{1 + \frac{\sum_{i=1}^{k-1} distance(x_k, x_i)}{k-1}} \tag{2}$$

where *distance* represents the distance between two samples in the data space. Also, as it is described in equation 3, the distance (similarity) between two samples is measured by the cosine distance(*cosDist*).

$$cosDist(x_k, x_p) = 1 - \frac{\sum_{j=1}^{n} x_{kj} x_{pj}}{\sqrt{\sum_{j=1}^{n} x_{kj}^2 \sum_{j=1}^{n} x_{pj}^2}} \tag{3}$$

where x_k and x_p represent the two samples to measure its distance and *n* represents the number of different attributes in both samples.

Note that the expression in the equation 2 requires all the accumulated data sample available to be calculated, which contradicts to the requirement for real-time and on-line application needed in the proposed problem. For this reason, in [3] it is developed a recursive expression cosine distance.

All details about the *eClass0* model and the learning algorithm can be found in [17].

The procedure of this classifier for creating and updating the Fuzzy Rules are:

1) Calculate the potential of the *new VoC* to be a prototype. This calculation is done by using a function of the accumulated distance between a sample and all the other *VoC* in the data space [3]. The result represents the *density* of the data that surrounds a certain data sample (*VoC*).
2) Update all the prototypes considering the new *VoC*. The *density* of the data space surrounding certain *VoC* changes with the insertion of each new *VoC* and the existing prototypes need to be updated.
3) Insert the new *VoC* as a new prototype if needed. The potential of the new *VoC* is calculated recursively and the potential of the other prototypes is updated.
4) Remove existing prototypes if needed. After adding a new prototype, we check whether any of the already existing prototypes are described *well* by the newly added prototype.

More details about this procedure and the learning algorithm can be found in [17].

As it is shown in Figure 1, this procedures update the Fuzzy Rules that define the different users.

C. Classification of a new user

The first step in the process of classification a new user is the creation of the corresponding *VoC* as it has been previously explained. Then, it is classified in a specific Unix user (class) represented by a prototype. For this task, we compare this new *VoC* with all the prototypes stored as Fuzzy Rules. This comparison is done in this case using cosine distance and the smallest distance determines the closest similarity (equation 4).

$$\begin{aligned} &Class(x_z) = Class(Prot^*);\\ &Prot^* = MIN_{i=1}^{NumProt}(cosDist(Prototype_i, x_z)) \end{aligned} \tag{4}$$

where x_z represents the z^{th} *VoC* to classify, $NumProt$ determines the number of existing prototypes, $Prototype_i$ represents the i_{th} prototype, and $cosDist$ represents the cosine distance between two vectors (*VoC*) in the data space.

D. Characteristics of the Classification Method

It is important to highlight that the proposed classification process keeps up to date the user behavior models. The Fuzzy Rules that represent these models change/evolve according to the changes in the behavior of the user (represented by her/his *VoC*). This is one of the most important characteristics in this approach.

In addition, the proposed method is really fast and computationally very simple as the *VoC* is created instantaneously and no complex operations are needed. Also, the proposed classifier can cope with huge amounts of data and it does not need to store all the commands typed by the user in memory. In addition, it is very efficient as it recursive and one pass.

IV. Experimental Setup and Results

In order to evaluate the proposed method, we conducted the following experiments with real-data of UNIX users.

A. UNIX Users Data

In this research, we have used the Data[1] drawn from the command histories of 9 UNIX computer users at Purdue University over 2 years [28]. This history files were parsed and sanitized to remove filesnames, user names, directory structures, web addresses, host names, and other possibly identifying items. Command names, flags, and shell metacharacters were preserved. Additionally, **SOF** and **EOF** tokens were inserted at the start and end of shell sessions, respectively.

[1] Available from:http://archive.ics.uci.edu/ml/datasets/UNIX+User+Data

Sessions are concatenated by date order and tokens appear in the order issued within the shell session, but no timestamps are included in this data.

B. Experiment Design

Although the proposed method has been designed to be used in real-time, we have used the above datasets in order to have comparable results with the established off-line techniques. For this reason, we have modified these data as follows:

- The training set contains 72 instances (9 users * 8 instances/user). Each instance consists of 100 commands and the user who typed that commands.
- The test set contains 27 instances (9 users * 3 instances/user).

However, in order to obtain the relevance of the size of the training set in the results, its size has been modified from 9 instances to 72. Then, we can have an idea about how many commands are necessary to classify a Unix user in this case. We should also take into account that in this case, we have obtain 100 commands per user; but this value could change according the environment.

After obtaining the *VoC* per instance (in this case the amount of different commands is 333), we can evaluate the performance of the proposed classifiers. We compare the proposed method with different classifiers which are detailed as follows:

- *C4.5* [29] is a well-known decision tree classifier.
- *PART* [30] is a rule based classifier which produces a set of it-then rules.
- *Nearest Neighbor (1-NN)* [31] classifies objects based on closest training examples in the feature space.
- *Naive Bayes* (NB) classifier [32], in which it is used a default precision of 0.1 for numeric attributes when it is created with zero training instances.
- *Support Vector Machine* Classifier (SVM) relies on the statistical learning theory [33].

C. Results

Figure 2 shows the percentage of instances correctly classified into its corresponding user using different number of instances as training set.

According to these data, we can see that even when the number of instances in the training set is very reduced, *eClass* works quite well. However, C4.5 and PART need much more instances for obtaining similar results. In general, *eClass* obtains comparable results to the obtained by the Naive Bayes.

Taking into account these results, we can concluded that the proposed classifier (*eClass0*) is comparable in this environments with other well established as Naive Bayes or SVM. However, due to the characteristics of the domain, *eClass0* is very suitable since it does not need to store the entire data streams in the memory and disregards any sample after being used.

V. Conclusions and Future Work

We have presented in this paper an evolving method to create Unix users models and to keep these models up to date. The proposed method is very simple as it has to be work very fast. The model only takes into account the frequency of the different commands a user types. The most important characteristic of the proposed user classifier is that it is able to change/evolve the models according to the changes in the behavior of the users. This classifier is one pass, non-iterative, recursive and it can be used in an interactive mode.

In addition, this method can cope with huge amounts of data and process streaming data quickly. Although the amount of commands that a user types in a command-line interface is huge, the proposed method is able to extract the most important characteristic with no need to store all the commands in memory. The approach has been evaluated using real data streams and the results are comparable to well established classifiers.

The domain used in this paper is UNIX; however, there are other areas in which a user can also be represented by a set of words. For example, this method could be used for modeling and classifying Twitter users taking into account the TF of the words of the tweets they post.

Acknowledgment

This work has been supported by the Spanish Government under i-Support (Intelligent Agent Based Driver Decision Support) Project (TRA2011-29454-C03-03).

References

[1] S. Greenberg, "Using unix: Collected traces of 168 users," Tech. Rep., 1988.

[2] "Machine Learning for User Modeling," *User Modeling and User-Adapted Interaction*, vol. 11, no. 1-2, 2001.

[3] P. P. Angelov and X. Zhou, "Evolving fuzzy-rule-based classifiers from data streams," *IEEE T. Fuzzy Systems*, vol. 16, no. 6, pp. 1462–1475, 2008.

[4] K. Han and M. Veloso, "Automated robot behavior recognition applied to robotic soccer," in *IJCAI-99*, 1999.

[5] D. Carmel and S. Markovitch, "Opponent modeling in multi-agent systems," in *Adaptation and Learning in Multi-Agent Systems*. Springer-Verlag: Heidelberg, Germany, 1996, pp. 40–52.

[6] P. Riley and M. M. Veloso, "On behavior classification in adversarial environments," in *DARS*, 2000, pp. 371–380.

[7] A. A. Macedo, K. N. Truong, J. A. Camacho-Guerrero, and M. da GraÇa Pimentel, "Automatically sharing web experiences through a hyperdocument recommender system," in *HYPERTEXT 2003*. New York, NY, USA: ACM, 2003, pp. 48–56.

[8] A. Godoy, D.; Amandi, "User profiling for web page filtering," *Internet Computing, IEEE*, vol. 9, no. 4, pp. 56–64, 2005.

[9] H. J. G. W. Pepyne, D.L., "User profiling for computer security," in *American Control Conference*, 2004, pp. 982–987.

[10] S. Coull, J. Branch, B. Szymanski, and E. Breimer, "Intrusion detection: A bioinformatics approach," in *ACSAC '03: Proceedings of the 19th Annual Computer Security Applications Conference*. Washington, DC, USA: IEEE Computer Society, 2003, p. 24.

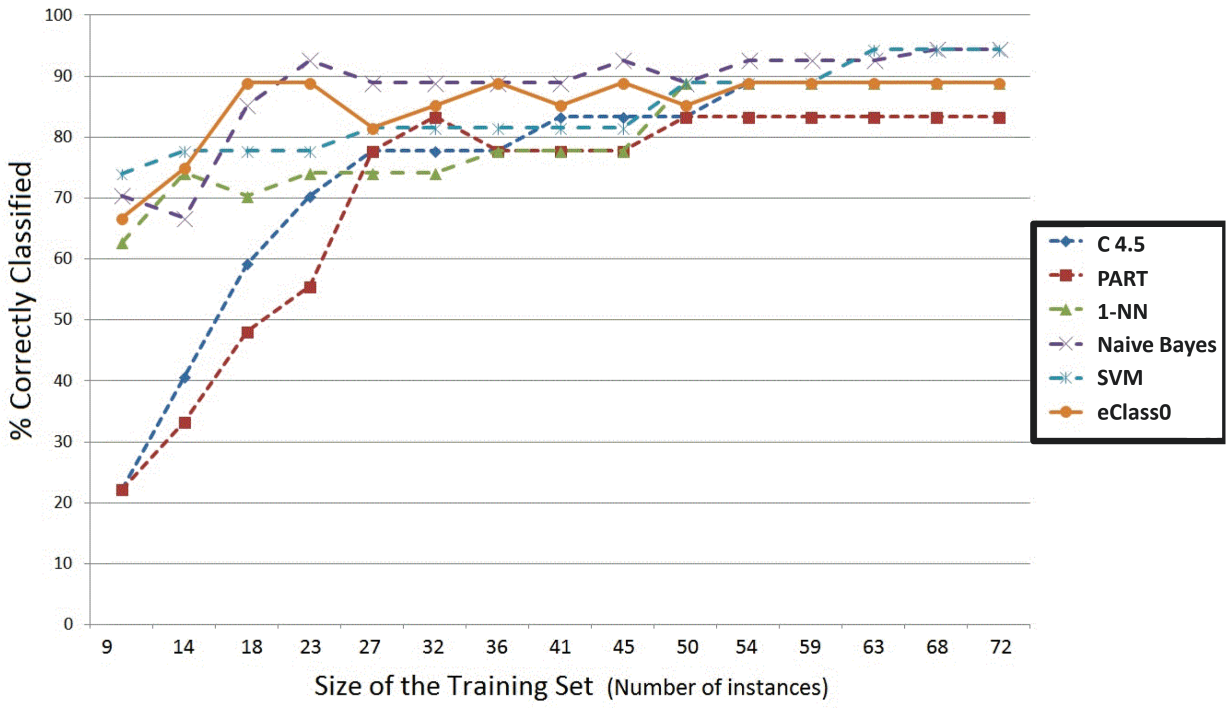

Fig. 2: *eClass0* vs. other well established classifiers.

[11] M. Schonlau, W. DuMouchel, W. Ju, A. Karr, M. Theus, and Y. Vardi, “Computer intrusion: Detecting masquerades,” *Statistical Science*, vol. 16, no. 1, pp. 58–74, 2001.

[12] J. A. Iglesias, A. Ledezma, A. Sanchis, and G. A. Kaminka, “A plan classifier based on chi-square distribution tests,” *Intelligent Data Analysis*, vol. 15, no. 2, pp. 131–149, 2011.

[13] J. A. Iglesias, P. P. Angelov, A. Ledezma, and A. S. de Miguel, “Creating evolving user behavior profiles automatically,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 5, pp. 854–867, 5 2012.

[14] J. A. Iglesias, A. L. Plamen Angelov, and A. Sanchis, “Evolving classification of agents behaviors: a general approach,” *Evolving Systems Journal*, vol. 1, pp. 161–171, 2010.

[15] P. P. Angelov, *Evolving rule-based models: a tool for design of flexible adaptive systems*. London, UK: Springer-Verlag, 2002.

[16] P. Angelov and X.-W. Zhou, “Evolving fuzzy systems from data streams in real-time,” in *Proceedings of the Internat. Symp. on Evolving Fuzzy Systems*, 2006, pp. 29–35.

[17] P. Angelov and D. Filev, “An approach to online identification of takagi-sugeno fuzzy models,” *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 34, no. 1, pp. 484 – 498, feb. 2004.

[18] X. Zhou and P. Angel, “Real-time joint landmark recognition and classifier generation by an evolving fuzzy system,” in *Fuzzy Systems, 2006 IEEE International Conference on*, 2006, pp. 1205 –1212.

[19] P. P. Angelov and X. Zhou, “Evolving fuzzy classifier for novelty detection and landmark recognition by mobile robots,” in *Mobile Robots*, 2007, pp. 89–118.

[20] P. Angelov, R. Ramezani, and X. Zhou, “Autonomous novelty detection and object tracking in video streams using evolving clustering and takagi-sugeno type neuro-fuzzy system,” in *Neural Networks, 2008. IJCNN 2008. (IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, june 2008, pp. 1456 –1463.

[21] Y. Liu, H. Meng, D. Wang, and X. Wang, “Adaptive staggering time estimation for target tracking in periodic nonuniform sampling system,” *Electronics Letters*, vol. 43, no. 24, pp. 1385–1387, 2007. [Online]. Available: http://link.aip.org/link/?ELL/43/1385/1

[22] P. Angelov, C. D. Bocaniala, C. Xideas, C. Patchett, D. Ansell, M. Everett, and G. Leng, “A passive approach to autonomous collision detection and avoidance,” *Computer Modeling and Simulation, International Conference on*, vol. 0, pp. 64–69, 2008.

[23] J. G. Kelly, P. P. Angelov, J. Trevisan, A. Vlachopoulou, E. Paraskevaidis, P. L. Martin-Hirsch, and F. L. Martin.

[24] J. A. I. annd Javier Ordoez, A. Ledezma, P. de Toledo, and A. Sanchis, “Evolving activity recognition from sensor streams,” in *Proceedings of the 2012 IEEE Evolving and Adaptive Intelligent Systems (EAIS-2012)*, May, pp. 96–101.

[25] J. Ordóñez, J. Iglesias, P. de Toledo, A. Ledezma, A. Sanchis *et al.*, “Online activity recognition using evolving classifiers,” *Expert Systems with Applications*, 2012.

[26] E. Garca-Cuesta and J. A. Iglesias, “User modeling in changeable environments,” in *Proceedings of the 2012 IEEE Evolving and Adaptive Intelligent Systems (EAIS-2012)*, May, pp. 182–185.

[27] J. A. Iglesias, A. Ledezma, and A. Sanchis, “Using well-known techniques for classifying user behavior profiles,” vol. 5, 2008, pp. 18–22.

[28] A. Frank and A. Asuncion, “UCI machine learning repository,” 2010. [Online]. Available: http://archive.ics.uci.edu/ml

[29] J. R. Quinlan, *C4.5: programs for machine learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

[30] E. Frank and I. H. Witten, “Generating accurate rule sets without global optimization,” in *Proceedings of the Fifteenth International Conference on Machine Learning*, ser. ICML ’98. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998, pp. 144–151.

[31] D. Aha and D. Kibler, “Instance-based learning algorithms,” *Machine Learning*, vol. 6, pp. 37–66, 1991.

[32] I. Rish, “An empirical study of the naive Bayes classifier,” in *Proceedings of IJCAI-01 Workshop on Empirical Methods in Artificial Intelligence*, 2001.

[33] J. Platt, “Machines using sequential minimal optimization,” in *Advances in Kernel Methods - Support Vector Learning*, B. Schoelkopf, C. Burges, and A. Smola, Eds. MIT Press, 1998.