HUMAN ACTIVITY RECOGNITION BASED ON EVOLVING FUZZY SYSTEMS

JOSE ANTONIO IGLESIAS
Carlos III University of Madrid, Avda. Universidad, 30
Leganes, Madrid, 28914, Spain
jiglesia@inf.uc3m.es

PLAMEN ANGELOV
InfoLab21, Lancaster University, South Drive
Lancaster, LA1 4WA, United Kingdom
p.angelov@lancaster.ac.uk

AGAPITO LEDEZMA* and ARACELI SANCHIS†
Carlos III University of Madrid
**ledezma@inf.uc3m.es*
†*masm@inf.uc3m.es*

Environments equipped with intelligent sensors can be of much help if they can recognize the actions or activities of their users. If this activity recognition is done automatically, it can be very useful for different tasks such as future action prediction, remote health monitoring, or interventions. Although there are several approaches for recognizing activities, most of them do not consider the changes in how a human performs a specific activity. We present an automated approach to recognize daily activities from the sensor readings of an intelligent home environment. However, as the way to perform an activity is usually not fixed but it changes and evolves, we propose an activity recognition method based on Evolving Fuzzy Systems.

Keywords: Activity recognition; evolving fuzzy systems; Fuzzy-Rule-Based (FRB) classifiers.

1. Introduction

Activity recognition is an important task which can be applied to many real-life problems such as healthcare or eldercare.[1] However, while collecting sequences of sensor readings in an intelligent home environment is valuable, determining what activities these sequences represent is a more challenging task. This task needs also to take into account the following aspects: human behavior is often erratic, and sometimes individuals behave differently because of a change in their goals. Thus, it is necessary to update models that describe the way an individual performs a specific activity.

In recent years, significant work has been carried out for recognizing human activities; however, most of the models created for an activity do not change according to the moods and habits of the individual who performs that activity. In this paper, we propose an adaptive approach for creating models of Activities of Daily Living (ADLs) and recognizing them from the sensor readings collected in an intelligent environment using a Fuzzy-Rule-Based (FRB) system. It is difficult or, in general, impossible, to build a classifier that will have a full description of all possible ways to perform an activity because they are not static and new patterns may emerge as well as an old habit may be forgotten or stopped to be used. Thus, as the description of the performance of a particular activity itself may also evolve, we assume that each ADL is described by one or more fuzzy rules. In addition, in order to achieve this task we need to cope with large amounts of data and process this information in real time because analyzing it in an offline mode would be impractical. In order to take into account these aspects, we propose an *Evolving Fuzzy Systems*-based approach that allows to create dynamic and evolving models

of ADLs. An Evolving Fuzzy System (EFS) is a self-developing, self-learning fuzzy rule-based system that have both their parameters but also their structure self-adapting on-line.

This paper is organized as follows: In the next section it is discussed the background and related work on the proposed problem. Section 3 details the structure of the proposed evolving approach. How the model of an ADL is created from a sequence of sensor readings is explained in Sec. 4. Section 5 details the evolving ADL classifier, which is based on *Evolving Fuzzy Systems*. Section 6 describes the experimental settings and results obtained. Finally, Sec. 7 contains future work and concluding remarks.

2. Background and Related Work

Over the last decade there has been growing interest in the interpretation of human behavior, including in video.[2] However, most of these works are limited because human activity is a complex process which is difficult to model, especially its dynamic aspect. In this research, we present an approach to recognize a human activity directly from the sensors readings collected in an intelligent home environment, without complex, off-line feature extraction, model/classifier pre-training and using user- and problem-specific parameters and thresholds.

There are many research studies about activity recognition in intelligent environments which can be applied to many real-life, human-centric problems.[3,4] However, several challenges need to be addressed before intelligent home environments technologies can be deployed. One of these main challenges is the design of powerful activity recognition algorithms. Most of the current research studies focus on recognizing simple human activities and the recognition of complex activities are only beginning to emerge. Some of the recent approaches to activity recognition are based on probabilistic models, such as Hidden Markov Models[5] or Bayesian Networks,[6,7] because sensor readings are noisy and activities are usually performed in a non-deterministic way. Other models for recognizing activities are logic based[8] or hand-crafted.[9]

It should be emphasized that the above approaches ignore the fact that the way to perform an activity can change and evolve according to the moods and habits of the individual. Thus, in this research, the activity recognition is considered, treated and modeled as a dynamic and evolving phenomenon. This is the most important contribution of this paper.

There exist several FRB[7,10–12] and Neural Networks[13–18] systems which have been applied to different tasks such as modeling,[19–22] monitoring,[23–27] detection,[28–31] clustering,[32–34] or classification.[35–37] However, the FRB classifier proposed in this paper has an open structure of rule-base and the on-line learning mechanism.

3. Our Approach

This section introduces the proposed approach for classifying a sequence of sensor readings into an ADL. The input of the proposed method is a sequence of sensor readings, but this method could be applied for modeling any task represented by a sequence of events. Although, the sensor readings are usually tagged with the time and date of the event, this information was not used in this study. This approach proposes a method to create the model of an ADL only from its corresponding sequence of ordered sensor readings (data streams) (Sec. 4). In addition, an evolving FBR classifier (called EvAClass - Evolving ADL Classifier) is presented which takes into account the fact that the way an individual performs an ADL is not fixed, but is rather changing, evolving (Sec. 5).

4. Creation of the Model of an ADL

In order to learn and construct the model of a certain ADL in an intelligent environment, we analyze the sequence of sensor readings collected while the ADL is done by a human. As the order of the sequence readings is essential in this task, the date and time of a sensor readings are used for creating the ordered sequence, but that information is not included in it. Sequences play a crucial role in human skill learning[38] and in high-level problem solving and reasoning.[39] Thus, in this research, the human activity modeling is transformed into a sequence analysis problem where a sequence of sensor readings represents a specific activity. This transformation can be done because it is clear that any activity has a sequential aspect, as sensors are activated in a sequence.

When a person activates a sensor, it usually depends on the previous sensors activated and it is

related to the following ones. In order to get the most representative and relevant set of sensor readings subsequences from a sequence, we propose the use of a *trie* data structure.[40] This structure has been also used for classifying behavior patterns in different domains.[41–44]

The process of creation of the model of an ADL from a sequence of sensor readings, which is explained in detail in Ref. 45, consists of three steps: 1. Segmentation of the sequence of sensor readings, 2. Storage of the subsequences in a *trie*, and 3. Creation of the model of the ADL. These steps are detailed in the following three subsections.

For the sake of simplicity, let us consider an example in which a specific ADL (*ADL-1*) is represented by the readings of two different motion sensors (*S1* and *S3*) symbolized by *S1.On*, *S1.Off* and *S3.On* (where, for example, *S1.On* expresses that the sensor *S1* has detected motion and it has been activated). In order to simplify the example, the sequence consists of only 5 readings. The example sequence represents in chronological order the sensor readings activated while an individual performs *ADL-1*: $\{S1.On \rightarrow S1.Off \rightarrow S1.On \rightarrow S1.Off \rightarrow S3.On\}$.

4.1. *Segmentation of the sequence of sensor readings*

First, the sequence is segmented into subsequences of equal length from the first to the last element. Thus, the sequence $A = A_1 A_2 \cdots A_n$ (where n is the number of sensor readings of the sequence) will be segmented in the subsequences described by $A_i \cdots A_{i+length} \, \forall i, i = [1, n\text{-}length + 1]$, where *length* is the size of the subsequences created and this value determines how many sensor readings are considered as dependent (how many sensor readings are usually activated consecutively as part of a pattern). In the remainder of the paper, we will use the term *subsequence length* to denote this value.

In the proposed sample sequence, let 3 be the subsequence length, then we obtain: $\{S1.On \rightarrow S1.Off \rightarrow S1.On\}$, $\{S1.Off \rightarrow S1.On \rightarrow S1.Off\}$, $\{S1.On \rightarrow S1.Off \rightarrow S3.On\}$.

4.2. *Storage of the subsequences in a trie*

The subsequences of sensor readings are stored in a special data structure, called *trie*, in which all possible subsequences are accessible and explicitly represented. Every *trie*-node represents a sensor reading and its *children* represent the sensor readings that follow it. Also, each node keeps track of the number of times a sensor reading has been recorded into it. As the dependencies of the sensor readings are relevant in the model of the ADL, the subsequence suffixes (subsequences that extend to the end of the given sequence) are also inserted.

Considering the previous example, the first subsequence ($\{S1.On \rightarrow S1.Off \rightarrow S1.On\}$) is added as the first branch of the empty *trie* (Fig. 1(a)). Each node is labeled with the number 1 which indicates that the sensor reading has been inserted in the node once (in Fig. 1, this number is enclosed in square brackets). Then, the suffixes of the subsequence ($\{S1.Off \rightarrow S1.On\}$ and $\{S1.On\}$) are also inserted (Fig. 1(b)). Finally, after inserting the three subsequences and its corresponding suffixes, the completed *trie* is obtained (Fig. 1(c)).

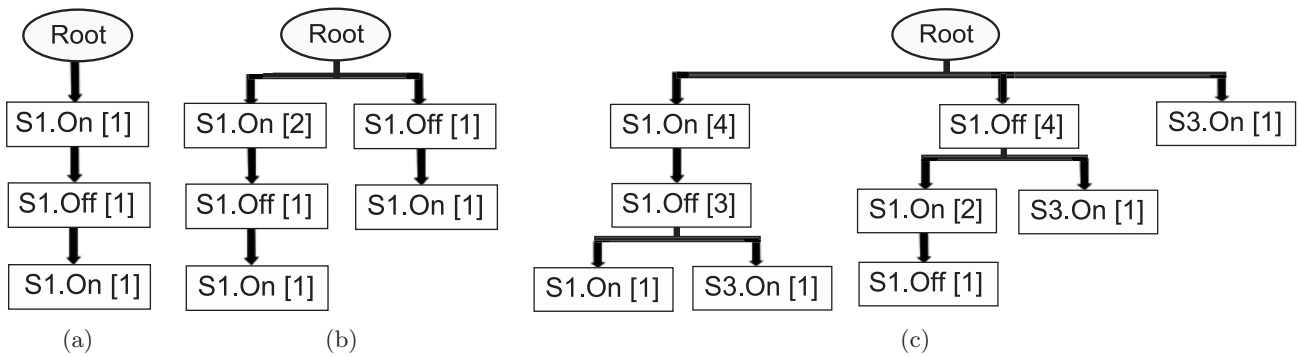

Fig. 1. Steps of creating an example trie.

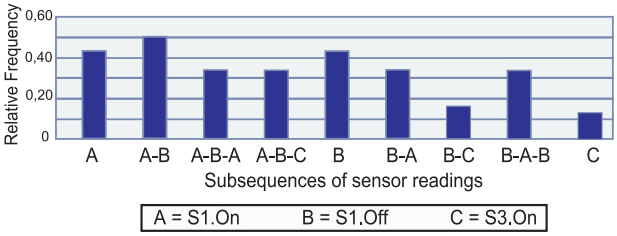

Fig. 2. Distribution of subsequences of sensor readings — Example.

4.3. *Creation of the model of the ADL*

Once the *trie* is created, the relevance of the subsequences that characterize the ADL can be measured by its relative frequency or support.[46] In this case, the support of a subsequence is defined as the ratio of the number of times the subsequence has been inserted into the *trie* and the total number of subsequences of equal size inserted.

Thus, in this step, the *trie* can be transformed into a set of subsequences labeled by its support value. This set of subsequences is represented as a distribution of relevant subsequences. In the previous example, the *trie* consists of 9 nodes; therefore, the corresponding model of the ADL consists of 9 different subsequences which are labeled with its support (Fig. 2). Once the model of an ADL has been created, it is classified and used to update the *Evolving ADL Library* (EvALib), as explained in the next section.

5. Evolving FRB Classifier (EvAClass)

A classifier is a mapping from the feature space to the class label space. In the proposed evolving FRB classifier, the feature space is defined by distributions of subsequences of sensor readings. On the other hand, the class label space is represented by the most representative distributions. Thus, a distribution in the class label space represents a specific model of an ADL which is one of the prototypes of the evolving library EvALib. These prototypes are not fixed and evolve taking into account the new information collected on-line from the data stream — this is what makes the classifier *Evolving*. The number of these prototypes is not pre-fixed but it depends on the homogeneity of the collected sequences. The differences between EvAClass and a conventional FRB classifier are the open structure of the rule-base and the on-line learning mechanism. These aspects make that the classifier can be adapted to the new ways of doing an ADL. The following subsections describe how the model of an ADL is represented to be used by EvAClass, and how this classifier is created in an evolving manner.

5.1. *ADL representation*

EvAClass receives, in an on-line manner, a distribution of subsequences (obtained from the sequence of sensor readings) which represents an ADL. In order to classify an ADL, these distributions must be represented in a data space. For this reason, each distribution is considered as a data vector that defines a *point* in the data space. The data space in which these *points* can be represented should consist of n dimensions, where n is the number of the different subsequences obtained. However, this value is unknown and the creation of this data space from the beginning is not efficient. For this reason, in EvAClass, the dimension of the data space is incrementally growing according to the different subsequences that are represented in it.

5.2. *Structure of the classifier* EvAClass

Once the corresponding data vector, which represents the distribution of a specific ADL, has been created from the sequence of sensor readings, it is processed by the classifier. As it is explained in the next five subsections, the structure of this classifier includes five steps: (1) **Classify the new sample** in a group represented by a prototype. (2) **Calculate the** *potential* **of the new data sample** to be a prototype. (3) **Update all the prototypes** considering the new data sample. (4) **Insert the new data sample as a new prototype** if needed. (5) **Remove existing prototypes** if needed.

5.2.1. *Classify the new sample*

In order to classify a new data sample, we compare it with all the prototypes stored in EvALib. This comparison is done using cosine distance and the smallest distance determines the closest similarity. This aspect is considered in Eq. (1).

$$Class(x_z) = Class(Prot^*); \quad Prot^* = MIN_{i=1}^{NumProt}(cosDist(Prot_i, x_z)) \tag{1}$$

where x_z represent the sample to classify, $NumProt$ determines the number of existing prototypes in the EvALib, $Prot_i$ represents the ith prototype, and $cosDist$ represents the cosine distance between two samples in the data space.

The time-consumed for classifying a new sample depends on the number of prototypes and its number of attributes. However, we can consider, in general terms, that both the time-consumed and the computational complexity are reduced and acceptable for real-time applications (in order of milliseconds per data sample) because the cosine distance is calculated recursively, as it is explained in the next subsection.

5.2.2. *Calculate the potential of the new data sample*

As in Ref. 47, a prototype is a data sample (the model of an ADL represented by a distribution of subsequences of sensor readings) that groups several samples which represent a certain way to perform an ADL. The classifier is initialized with the first data sample, which is stored in the EvALib. Then, each data sample is classified into one of the prototypes defined in the classifier. Based on the *potential* of the new data sample to become a prototype,[48] it could form a new prototype or replace an existing one.

The potential (P) of the kth data sample (x_k) is calculated by the Eq. (2), which represents a function of the accumulated distance between a sample and all the other $k-1$ samples in the data space.[47] 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.

In Ref. 49 the potential is calculated using the euclidean distance and in Ref. 47 it is calculated using the cosine distance. EvAClass uses the cosine distance ($cosDist$) to measure the similarity between two samples, as it is described in Eq. (3), because it tolerates different samples to have different number of attributes (in this case, an attribute is the support value of a subsequence of sensor readings).

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

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 Eq. (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 Ref. 47 it is developed a recursive expression for the cosine distance. This formula is as follows:

$$P_k(z_k) = \frac{1}{2 - \frac{1}{(k-1)\sqrt{\sum_{j=1}^{n}(z_k^j)^2}} B_k}; \quad k = 2, 3 \cdots$$

where

$$B_k = \sum_{j=1}^{n} z_k^j b_k^j; \quad b_k^j = b_{(k-1)}^j + \sqrt{\frac{(z_k^j)^2}{\sum_{l=1}^{n}(z_k^l)^2}}$$

and

$$b_1^j = \sqrt{\frac{(z_1^j)^2}{\sum_{l=1}^{n}(z_1^l)^2}}; \quad j = [1, n+1]; \quad P_1(z_1) = 1 \tag{4}$$

where z_k represents the kth data sample (x_k) and its corresponding label ($z = [x, Label]$). Using this expression, it is only necessary to calculate $(n+1)$ values where n is the number of different subsequences obtained; this value is represented by b, where $b_k^j, j = [1, n]$ represents the accumulated value for the kth data sample.

5.2.3. *Update all the prototypes*

Once the potential of the new data sample has been calculated, all the existing prototypes in the EvALib are updated taking into account this new data sample. It is done because the *density* of the data space surrounding certain data sample changes with the insertion of each new data sample. This operation is done really fast and it requires very little memory space because of the use of the recursive Eq. (4).

5.2.4. *Insert the new data sample as a new prototype*

EvAClass can start '*from scratch*' (without prototypes in the library) in a similar manner as *eClass* evolving fuzzy rule-based classifier proposed in Ref. 49 and further developed in Ref. 47. In this step, the potential of the new sample (z_k) is

compared with the potential of the existing prototypes. A new prototype is created if its value is higher than any other existing prototype, as shown in Eq. (5).

$$\exists\, i,\ i = [1, NumProt] :\ P(z_k) > P(Prot_i) \quad (5)$$

Thus, if the new data sample is not relevant, the overall structure of the classifier is not changed. Otherwise, if the new data sample has high descriptive power and generalization potential, the classifier evolves by adding a new prototype in the EvALib which represents a part of the obtained data samples.

5.2.5. *Removing existing prototypes*

After adding a new prototype, we check whether any of the already existing prototypes in the EvALib are described *well* by the newly added prototype.[47] By *well* we mean that the value of the membership function that describes the closeness to the prototype is a Gaussian bell function chosen due to its generalization capabilities:

$$\exists\, i, \quad i = [1, NumPrototypes] :\ \mu_i(z_k) > e^{-1} \quad (6)$$

The membership function between a data sample and a prototype is calculated as follows:

$$\mu_i(z_k) = e^{-\frac{1}{2}[\frac{cosDist(z_k, Prot_i)}{\sigma_i}]}, \quad i = [1, NumProt] \quad (7)$$

where $cosDist(z_k, Prot_i)$ represents the cosine distance between a data sample (z_k) and the ith prototype ($Prot_i$); σ_i represents the spread of the membership function, which also symbolizes the radius of the zone of influence of the prototype. This spread is determined based on the scatter[50] of the data. In order to calculate the scatter without storing all the received samples, this value can be updated (as shown in Ref. 49) recursively by:

$$\sigma_i(k) = \sqrt{[\sigma_i(k-1)]^2 + \frac{[cosDist^2(Prot_i, z_k) - [\sigma_i(k-1)]^2]}{k}} \quad (8)$$

where k is the number of data samples inserted in the data space; $cosDist(P_i, z_k)$ is the cosine distance between the ith prototype and the new data sample.

5.3. *Pseudo-code and properties of the classifier* EvAClass

5.3.1. *Pseudo-code of* EvAClass

Begin EvAClass
Initialization: $z_1 = [x_1, Label_1]$ and $P_1 = 1$
DO for a data sample WHILE data stream ends
Read the new sample, x_k (sequence of sensor readings)
Classify x_k in a class using equation 1.
Calculate $P(x_k)$ using the recursive formula 4.
Update all the prototypes (considering x_k) by 4.
IF (5 holds) THEN
Insert x_k as a new prototype.
IF (6 holds) THEN
Remove the corresponding prototype/s.
End DO
End EvAClass

5.3.2. *Properties of* EvAClass

The proposed classifier faces an important challenge in the human activity recognition: to evolve the created classifier according to the new sequences of sensor readings collected in the intelligent environment. This approach does not require pre-traning and it starts "from scratch".

As the information of the sensors readings collected from an intelligent home environment is usually quite large, EvAClass can also cope with huge amounts of data and process streaming data in real-time and on-line. In an intelligent environment, storing the complete data set and analyzing the data streams in off-line mode could be impractical.

6. Experimental Setup and Results

In order to evaluate EvAClass, we use a dataset with the sensor readings activated by a person while s/he is doing a specific ADL. Thus, the sequence of sensor readings is labeled.

6.1. *Dataset*

The dataset used in this research was created by the *CASAS Smart Home project*, which is a multidisciplinary research project at Washington State University focused on the creation of an intelligent home environment.[51] The data represents sensor

Table 1. Example of a sequence of events which represents the ADL "*Cook*".

ADL: "*Cook*"	
Sensor Readings	*Sequence*
2008-02-29 13:25:05.527 I01 ABSENT	*I01-ABSENT*
2008-02-29 13:25:09.190 M16 OFF	*M16-OFF*
2008-02-29 13:25:10.513 M17 ON	*M17-ON*
2008-02-29 13:25:11.979 I07 ABSENT	*I07-ABSENT*
...	...

readings collected in a smart apartment testbed. Sensor information includes motion, temperature, water, burner, phone usage, and item sensor readings. The data represents 24 participants performing the following five ADLs: (1) *Make a phone call*, (2) *Wash hands*, (3) *Cook*, (4) *Eat*, and (5) *Clean*. Thus, the dataset consists of 120 different samples labeled with its corresponding ADL. Each sample consists of a sequence from about 30 to 150 sensor readings.

The first column of the Table 1 shows a portion of the sensor readings that are generated by the activity "Cook" where M16 and M17 represent motion sensors and I01 AND I07 represent item sensors (these readings report *PRESENT* when an item is placed on the sensor and *ABSENT* otherwise). The second column of this table reports the sequence of sensor readings which is used for the proposed approach.

6.2. *Experimental design*

In the creation of the ADL model, the length of the subsequences in which the original sequence is segmented (used for creating the *trie*) is an important parameter: using long subsequences, the time consumed for creating the *trie* and the number of relevant subsequences of the corresponding distribution increase drastically. In the experiments presented in this paper, the *subsequence length* varies from 2 to 6.

Although, EvACLass has been designed to be used on-line, in order to have comparable results with other different classifiers using the above dataset, 3-fold cross validation is used. It should be emphasized that EvACLass does not need to work in this mode.

For evaluating the performance of EvACLass, we compare it with the following classifiers: Naive Bayes (NB) and k-Nearest Neighbor (k-NN) (incremental and non-incremental), C5.0, Bayesian Networks (BN), Support Vector Machine (SVM), Learning Vector Quantization (LVQ) and Artificial Neural Networks (ANN). The k-NN needs to be parameterized with the number of neighbors (k) used for classification; in this case, the better results are obtained using $k = 1$. Using a NB classifier, the numeric estimator precision values are chosen based on analysis of the training data; however, the NB incremental uses a default precision of 0, 1. The algorithm implementation used in the LVQ classifier is the enhanced version of LVQ1, the OLVQ1 implementation. The type of ANN used is a multi-layer perceptron trained with the back-propagation algorithm, in which the number of neurons is modified according to the number of inputs.

For this comparison, these classifiers were trained using a feature vector for each activity done by a resident. Therefore, each of the 12 sample vectors are labeled with its corresponding ADL. The corresponding vector consists of the support value of all the different subsequences of sensor readings obtained for all the ADLs. There are a lot of subsequences which do not have a value because the corresponding sensor readings subsequence was not activated doing that ADL. In this case, in order to be able to use this data for training the classifiers, we consider the value 0 (although its real value is *null*).

6.3. *Results*

Table 2 shows the average classification success of different classifiers using different subsequences lengths for segmenting the initial sequence (from 2 to 6). According to these data, the percentages of sequences correctly classified by our approach are better than the obtained by *K-NN* and very similar to the obtained by the other classifiers. In general, the difference between EvACLass and the NB, SVM and ANN is considerable using long subsequences (5 or 6), but this difference decreases when this length is smaller. Thus, using an appropriate subsequence length, the proposed classifier can compete well with off-line approaches. The ANN-based classifier needs a great deal of neurons because the number of inputs is usually very large. Thus, the training process is very time consuming, computationally expensive and impractical in implementation in the environment proposed in this research.

Table 2. Classification rates using different subsequence lengths for the creation of the model of the ADL.

	Classifiers and classification rate (%)									
		Incremental classif.		Non-incremental classifiers						
Subsequence lengths	EvACLass	NB increm.	k-NN increm. ($k=1$)	C5.0	NB	k-NN ($k=1$)	BN	SVM	LVQ	ANN
2	92.5	93.3	91.6	94.1	97.5	95.0	98.3	97.5	91.6	98.3
3	**94.2**	**88.3**	**81.5**	**95.0**	**97.5**	**86.6**	**97.5**	**97.5**	**92.5**	**97.5**
4	87.5	87.5	53.3	93.3	95.8	59.1	97.5	96.6	90.0	97.5
5	79.2	84.1	40.8	93.3	93.3	43.3	97.5	98.3	84.1	97.5
6	78.3	82.5	33.3	92.5	92.5	34.1	97.5	98.3	85.0	97.5

It should be noticed that only our approach can evolve the created classifier according to the new sequences collected in the intelligent environment. The proposed environment needs a classifier able to process steaming data in on-line and in real-data. Only the incremental classifiers satisfy this requirement, but unlike EvACLass, they assume a fixed structure. Taking into account the incremental classifiers, the average classification success is very similar; besides EvACLass can cope with huge amounts of data because it does not need to store the entire data stream in the memory and disregards any sample after being used. In addition, the strucuture of EvACLass is open and the rule-base evolves in terms of creating new prototypes as they occur automatically. For this reason, EvACLass is computationally simple and efficient as it is recursive and one pass. In fact, since the number of attributes is very large in a real environment and it changes frequently, the proposed approach EvACLass is the only working alternative.

Considering the same goal proposed in this paper, Cook and Schmitter[52] implemented both a naive Bayesian classifier and a Markov Model (MM) to recognize an ADL using the data set described above. Using 3-fold cross validation on the 120 sequences of reading sensors (activity traces), the naive Bayes algorithm achieved 91% classification accuracy while the MM achieved 98%.

7. Conclusions

In this paper we propose a generic and evolving approach (EvACLass) to model and classify sequences of sensor readings which represent a certain ADL. EvACLass is based on *Evolving Fuzzy Systems* and it is one pass, non-iterative, recursive and it has the potential to be used in an interactive mode; therefore, it is computationally very efficient and fast. The test results with a real dataset demonstrate that EvACLass performs as well as other well established off-line classifiers in terms of correct classification on validation data. However, the environment proposed in this paper changes rapidly and EvACLass takes into account this aspect since its structure is open and its rule-base evolves.

Although it is not addressed in this paper, it could be interesting to include, in the ADL model, information about the date and the time when the sensor readings have been activated. In addition, the proposed algorithm should be included in a complete system that performs functional assessment of individuals in their environments.

Acknowledgements

This work has been partially supported by the Spanish Government under project TRA2007-67374-C02-02.

References

1. J. Boger and C. Boutilier, A decision-theoretic approach to task assistance for persons with dementia, in *Proceedings of International Joint Conferences on Artificial Intelligence* (2005) 1293–1299.
2. J. Hoey and J. J. Little, Value-directed human behavior analysis from video using partially observable markov decision processes, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **29** (2007) 1118–1132.
3. J. Sarkar, Y.-K. Lee and S. Lee, A smoothed naive bayes based classifier for activity recognition, *IETE Technical Review (SCIE)* **27**(2) (2010) 109–119.

4. E. Kim, S. Helal and D. Cook, Human activity recognition and pattern discovery, *IEEE Pervasive Computing* **9** (2010) 48–53.
5. D. Sanchez, M. Tentori and J. Favela, Activity recognition for the smart hospital, *IEEE Intelligent Systems* **23**(2) (2008) 50–57.
6. D. H. Wilson and C. Atkeson, Simultaneous tracking and activity recognition (star) using many anonymous, binary sensors (2005) 62–79.
7. E. Castillo, I. Gallego, J. M. Menendez and S. Sanchez-Cambronero, Traffic estimation and optimal counting location without path enumeration using bayesian networks, *Computer-Aided Civil and Infrastructure Engineering* **23**(2) (2008) 198–207.
8. N. Landwehr, B. Gutmann, I. Thon, L. De Raedt and M. Philipose, Relational transformation-based tagging for activity recognition, *Fundamenta Informaticae* **89**(1) (2008) 111–129.
9. J. Fogarty, Sensing from the basement: A feasibility study of unobtrusive and low-cost home activity recognition, in *Symposium on User interface Software and Technology* (2006) 91–100.
10. Y. Li and T. Zhang, Global exponential stability of fuzzy interval delayed neural networks with impulses on time scales, *International Journal of Neural Systems* **19**(6) (2009) 449–456.
11. A. Haidar, A. Mohamed, M. Al-Dabbagh, A. Aini Hussain and M. Masoum, An intelligent load shedding scheme using neural networks & neuro-fuzzy, *International Journal of Neural Systems* **19**(6) (2009) 473–479.
12. J. Villar, E. de la Cal and J. Sedano, A fuzzy logic based efficient energy saving approach for domestic heating systems, *Integrated Computer-Aided Engineering* **16**(2) (2009) 151–163.
13. T. Jorgensen, B. Haynes and C. C. C. F. Norlund, Pruning artificial neural networks using neural complexity measures, *International Journal of Neural Systems* **18**(5) (2008) 389–403.
14. H. Huynh, Y. Won and J. Kim, An improvement of extreme learning machine for compact single-hidden-layer feedforward neural networks, *International Journal of Neural Systems* **18**(5) (2008) 433–441.
15. J. Liang, Z. Wang and X. Liu, Global synchronization in an array of discrete-time neural networks with mixed coupling and time-varying delays, *International Journal of Neural Systems* **19**(1) (2009) 57–63.
16. S. Ghosh-Dastidar and H. Adeli, Spiking neural networks, *International Journal of Neural Systems* **19**(4) (2009) 295–308.
17. M. Asaduzzaman, M. Shahjahan and K. Murase, Faster training using fusion of activation functions for feed forward neural networks, *International Journal of Neural Systems* **19**(6) (2009) 437–448.
18. J. Menke and T. T. Martinez, Improving supervised learning by adapting the problem to the learner, *International Journal of Neural Systems* **19**(1) (2009) 1–9.
19. H. Adeli and X. Jiang, Neuro-fuzzy logic model for freeway work zone capacity estimation, *Journal of Transportation Engineering* **129**(5) (2003) 484–493.
20. S. Suresh, N. Kannan, N. Sundararajan and P. Saratchandran, Neural adaptive control for vibration suppression in composite fin-tip of aircraft, *International Journal of Neural Systems* **18**(3) (2008) 219–231.
21. W. Pedrycz, R. Rai and J. Zurada, Experience-consistent modeling for radial basis function neural networks, *International Journal of Neural Systems* **18**(4) (2008) 279–292.
22. P. Anand, B. Siva Prasad and C. Venkateswarlu, Modeling and optimization of a pharmaceutical formulation system using radial basis function network, *International Journal of Neural Systems* **192**(2) (2009) 127–136.
23. E. Carden and J. Brownjohn, Fuzzy clustering of stability diagrams for vibration-based structural health monitoring, *Computer-Aided Civil and Infrastructure Engineering* **23**(5) (2008) 360–372.
24. G. Rigatos, Adaptive fuzzy control with output feedback for h-infinity tracking of sisi nonlinear systems, *International Journal of Neural Systems* **18**(4) (2008) 305–320.
25. H. Elragal, Improving neural networks prediction accuracy using particle swarm optimization combiner, *International Journal of Neural Systems* **19**(5) (2009) 387–393.
26. X. Jiang and H. Adeli, Dynamic fuzzy wavelet neuroemulator for nonlinear control of irregular high-rise building structures, *International Journal for Numerical Methods in Engineering* **74**(7) (2008) 1045–1066.
27. Y. Tang, Q. Miao and J. Fang, Synchronization of stochastic delayed chaotic neural networks with markovian switching and application in communication, *International Journal of Neural Systems* **19**(1) (2009) 43–56.
28. H. Adeli and A. Karim, Fuzzy-wavelet rbfnn model for freeway incident detection, *Journal of Transportation Engineering* **126**(6) (2000) 464–471.
29. K. Perusich, Using fuzzy cognitive maps to identify multiple causes in troubleshooting systems, *Integrated Computer-Aided Engineering* **15**(2) (2008) 197–206.
30. A. Samant and H. Adeli, Enhancing neural network incident detection algorithms using wavelets, *Computer-Aided Civil and Infrastructure Engineering* **16**(4) (2001) 239–245.
31. A. Karim and H. Adeli, Comparison of the fuzzy wavelet rbfnn freeway incident detection model with

the california algorithm, *Journal of Transportation Engineering* **128**(1) (2002) 21–30.
32. X. Gao, L. Vuong and M. Zhang, Detecting data records in semi-structured web sites based on text token clustering, *Integrated Computer-Aided Engineering* **15**(4) (2008) 297–311.
33. W. Chen and C. Zhang, A hybrid framework for protein sequences clustering and classification using functional signature motif information, *Integrated Computer-Aided Engineering* **16**(4) (2009) 353–365.
34. A. Khashman and B. Sekeroglu, Document image binarisation using a supervised neural network, *International Journal of Neural Systems* **18**(5) (2008) 405–418.
35. M. Ahmadlou and H. Adeli, Enhanced probabilistic neural network with local decision circles: A robust classifier, *Integrated Computer-Aided Engineering* **17**(3) (2010).
36. S. Buchholz and N. Bihan, Polarized signal classification by complex and quaternionic multilayer perceptrons, *International Journal of Neural Systems* **18**(2) (2008) 75–85.
37. W. Zou, Z. Chi and K. Lo, Improvement of image classification using wavelet coefficients with structured-based neural network, *International Journal of Neural Systems* **18**(3) (2008) 195–205.
38. R. Sun, E. Merrill and T. Peterson, From implicit skills to explicit knowledge: A bottom-up model of skill learning, *Cognitive Science* **25**(2) (2001) 203–244.
39. J. Anderson, *Learning and Memory: An Integrated Approach*. New York: John Wiley and Sons (1995).
40. E. Fredkin, Trie memory, *Communications of the ACM* **3**(9) (1960) 490–499.
41. J. A. Iglesias, A. Ledezma and A. Sanchis, Sequence classification using statistical pattern recognition, in *Intelligent Data Analysis* **4723** (2007) 207–218.
42. G. A. Kaminka, M. Fidanboylu, A. Chang and M. M. Veloso, Learning the sequential coordinated behavior of teams from observations, in *RoboCup*, Vol. 2752, Springer (2002) 111–125.
43. J. A. Iglesias, A. Ledezma and A. Sanchis, A comparing method of two team behaviours in the simulation coach competition, in *Proceedings of the Modeling Decisions for Artificial Intelligence* (2006) 117–128.
44. J. A. Iglesias, A. Ledezma, A. Sanchis and G. Kaminka, A plan classifier based on chi-square distribution tests, *Intelligent Data Analysis* **15**(2) (2011).
45. J. A. Iglesias, A. Ledezma and A. Sanchis, Creating user profiles from a command-line interfaces: A statistical approach, in *Proceedings of the International Conference on User Modeling, Adaptation, and Personalization* (2009) 90–101.
46. R. Agrawal and R. Srikant, Mining sequential patterns, in *International Conference on Data Engineering* (1995) 3–14.
47. P. Angelov and X. Zhou, Evolving fuzzy rule-based classifiers from data streams, *IEEE Transactions on Fuzzy Systems: Special issue on Evolving Fuzzy Systems* **16**(6) (2008) 1462–1475.
48. P. Angelov and D. Filev, An approach to online identification of takagi-sugeno fuzzy models, *Systems, Man, and Cybernetics, Part B, IEEE Transactions on* **34**(1) (2004) 484–498.
49. P. Angelov, X. Zhou and F. Klawonn, Evolving fuzzy rule-based classifiers, *IEEE Symposium on Computational Intelligence in Image and Signal Processing* (2007) 220–225.
50. P. Angelov and D. Filev, Simpl_ets: A simplified method for learning evolving takagi-sugeno fuzzy models, in *IEEE International Conference on Fuzzy Systems* (2005) 1068–1073.
51. School of Electrical Engineering and Computer Science — Washington State University, CASAS Smart Home Project (http://ailab.wsu.edu/casas/) (2010).
52. D. Cook and M. Schmitter-Edgecombe, Assessing the quality of activities in a smart environment, *Methods of Information in Medicine* **48**(5) (2009) 480–485.