

Real Time Recognition of Human Activities from Wearable Sensors by Evolving Classifiers

Javier Andreu, Rashmi Dutta Baruah, Plamen Angelov

School of Computing and Communications

Infolab21, Lancaster University

Lancaster, LA1 4WA, United Kingdom

{j.andreu,r.duttabaruah,p.angelov}@lancaster.ac.uk

Abstract—A new approach to real-time human activity recognition (HAR) using evolving self-learning fuzzy rule-based classifier (eClass) will be described in this paper. A recursive version of the principle component analysis (PCA) and linear discriminant analysis (LDA) pre-processing methods is coupled with the eClass leading to a new approach for HAR which does not require computation and time consuming pre-training and data from many subjects. The proposed new method for evolving HAR (eHAR) takes into account the specifics of each user and possible evolution in time of her/his habits. Data streams from several wearable devices which make possible to develop a pervasive intelligence enabling them to personalize/tune to the specific user were used for the experimental part of the paper.

Keywords— human activity recognition, fuzzy rule-based classifiers, evolving systems, wearable sensors, accelerometers.

I. INTRODUCTION

Nowadays due to the increase of the average age of the population in the world as well as due to the dramatic increase of the capacities and availability (including lower costs, power consumption, size) of versatile computing and communication platforms there is a rising interest in areas of research such as Ambient Intelligence (AmI), Ubiquitous and Pervasive Computing, autonomous machine learning etc. Traditional machine learning and intelligent systems can only partially address all of the challenges because of the requirements such as personalization, adaptation, human-intelligible (often linguistic) interface and transparency (reluctance to use 'black-box' solutions even if they perform well. Inside this AmI there is an important interest in the Human Activity Analysis. It may serve as a basis for a better understanding of human habits and relations.

Trying to address all of these challenges, an emerging trend in computational intelligence which borrows heavily from adaptive systems, systems identification, fuzzy rule-based models, machine learning and pattern recognition have recently been prominent – the development of evolving systems [1]. They are characterized by the ability to self-learn and self-adapt their structure as well as to tune their parameters. They are also, usually, human-intelligible (transparent). Evolving systems, and classifiers, in particular [2] can evolve their structure to reflect the habits and behavior of a specific human user. In this way, they can be personalized. This is very useful because, for example, same type of activities (such as *walking*) is usually performed by

younger individuals in a quite different manner as compared to more mature individuals. Users with different complexions perform activities with a different intensity or energy.

Evolving systems is a new paradigm which requires a different type of real-time, on-line learning which combines structure learning with the parameter learning. In this study we will take the evolving fuzzy rule-based (FRB) type classifier (eClass) as a starting point and will develop a more advanced pre-processing mechanism specifically suitable for the activity recognition task that we have.

HAR is also an area of intensive research recently [3] [4]. Very often it relies on cameras and video signals [4], however these methods often have problems related to person occlusion in the camera range, selecting a good viewpoint to track the person, privacy connotations, etc. More recently, wearable sensors have offered a very promising and attractive opportunity to collect information directly from the human body. This approach has the advantage of monitoring the human/subject continuously and without invading his privacy.

HAR problem was already addressed in several works in real-time mode using wearable sensors. For example, in [5] an activity recognition system which retrieves information from several wearable sensors is used. In [6] an *iPhone* is used to recognize simple activities such as *running*, *walking*, *bicycling*. In these works the training is carried out collecting expert knowledge from many subjects and creating a general model or creating a personal model from training provided from a particular user. However, a cross-personal model which is just trained from one subject and evolves itself with the next one, without expert knowledge or large training sets of different users would be more attractive.

A previous work [8] has involved evolving classifier in the HAR field for the first time, but it was limited to using binary data from light power switches. However, evolving system has not been involved yet in a HAR context using a complex signal domain such as acceleration from wearable sensors.

This paper is organized as follows; section II provides a description of various sensors used; in section III the pre-processing and feature selection approach, that is particularly suitable for this problem, is presented; in section IV the eHAR system is described along with the mechanism of its self-learning and adaptation, and in section V the proposed approach is being evaluated and compared with the possible alternatives. Finally, section VI concludes and outlines the future perspectives.

II. WEARABLE SENSORS

The miniaturization of electronic circuits has made possible increasing the number of devices that one can pervasively wear or even carry in a pocket or as a part of another device. One of the most prominent pervasive micro sensors which can be applied close to the body or as a wearable item is the accelerometer. A large number of manufactured products bring on board accelerometers such as smart phones, high-tech shirts, sports watches, hearings and so on. Hence, in this research we will consider accelerometers as preferred devices of study across in the wearable context.

There are analogue and digital accelerometers; the former produce a voltage that is proportional to the acceleration while the latter outputs pulse with modulated (PWM) signal. Accelerometers can be classified based on the number of axes they consider. For easy recognition tasks, two axes are often enough. However, the higher the number of axes the more sources of information we get for the motion of the object. Although, accelerometer data seems to be highly correlated, the influence of centripetal and gravitational forces leads to randomisation of the correlation either negatively or positively according to the movement. A well planned pre-processing procedure (see next section) can further significantly minimise the correlation effect. Another characteristic to bear in mind is the maximum swing of the accelerometer. It can be computed using the tilt which is the inverse of the tangent of the quotient of two vectors. In order to measure the tilt, only a swing of $\pm 1.5g$ can be enough; in order to conduct very basic recognition of activities such as walking, running or stopping a swing of $\pm 2g$ may be required; however, in order to measure more complex activities such as sudden switches between different types of motion (sudden stop - abrupt movement, etc.) the accelerometer have to handle a swing of $\pm 5g$ or more. Sensitivity is also very important, but it directly depends on the quality of the device itself or the inter-mediation of other devices; for example, when an accelerometer is embedded in a smart phone, its accuracy could be reduced by conflicts with other in-built devices or can be influenced by the position of the sensor inside the phone. Finally, the bandwidth is, perhaps, the most critical aspect of the experiment, because it can restrict or facilitate the level of cognition of a movement. In order to take into account the vibration as well as the acceleration changes or tilt during the movement the bandwidth should be set up accordingly (for example, at least 150 Hz).

Accelerometers use a 3D Cartesian coordinate system to represent the direction of the vector of acceleration. When a device is moving along an axis, the acceleration is positive is moving towards positive coordinates, on the contrary, the acceleration is negative.

A. Placement of the Sensors on the Body

Devices that a user can carry such as phones, MP3 players, etc. can have different placement on the body. The location on the human body of the sensors which are part of these devices is very important in respect to facilitating the recognition of specific activities. For example, in case we aim to recognise if

someone is *standing up*, it is best to attach the sensor to the leg of the subject; however, in case we aim to recognise if an individual is "*writing on the white board*", a combination of sensors placed on body's arms and legs would be required. Hence, the position of the sensor on the body is of a crucial importance in order to collect informative signal for activity recognition using wearable sensors. The need to add multiple sensing points increases with the complexity of the activity.

A lot of research has been carried out recently aiming to clarify the number of sensors and locations where sensors can be applied aiming to maximise the recognition rate. For example, statistical analysis has been used in [7] to discover the most significant placements of sensors as well as a heuristic method have been defined to cope with this problem. However, in real life applications activity recognition should be made as pervasively as possible avoiding affecting individual's behaviours and habits. For example, it is expected not to force an individual to wear a sensitive object every time in the same place and same position in order to keep certain level of freedom of movement for the user. Moreover, this individual cannot have the same motion habits/patterns as the other. Consequently, the data that are collected from the sensor will not be the same for each individual. Therefore, personalisation is an important issue in human activity recognition and self-adaptive/evolving models [8] can address this problem by self-learning classifiers aiming to recognise human activity of a specific individual adapting to the personal specifics and self-developing their mechanism of classification. In evolving classifiers learning can take place on-line and the classifier structure evolves adapting to an individual's specifics

B. Hardware Instrumentation

In order to get data samples and test the algorithms, we have used several portable devices which can be considered as wearable. Some of these sensors have processing capacities on-board and others are data transducers. In particular, we considered:

1) *SunSPOT*: It is a hardware/software platform developed by Oracle/Sun Microsystems in 2007 [9]. It has characteristics of a wireless sensor, i.e data can be relayed to a base station via sockets and also forwarded to all sensors. The radio range is 10 meters (USA version allows up to 100m). SunSPOT sensors have embedded a 3-axial accelerometer that can be set to measure accelerations over a scale of $\pm 6g$. The device is programmable in Java ME through an embedded virtual machine (Squawk VM), Figure 1.



Fig 1. SunSPOT (on a palm and attached to the arm; size 63x38x25mm)

2) *Nokia N97*: It is a Smart Phone which can be used as a data collection platform. The reason we consider *Nokia N97* is to study a commonly used device as a sensor platform. Usually mobile/cell phones are carried in different locations: a pocket, a bag, on a mobile belt bag, etc. They are completely personal which allows training procedures to be adapted to the specific phone owner (personalised). Consequently, the data collected from these devices is highly representative. *Nokia N97* has a fully programmable Symbian Environment on board and a 3-axial accelerometer with a scale of $\pm 8g$, Figure 2.



Fig 2. Nokia 97 Smartphone. Size 117 x 55 x 16 mm.

3) *Porcupine*: This device is a small wearable sensing module which was developed initially at Lancaster University, UK. A further, improved version of *Porcupine* was developed at the Embedded Sensing Systems group, Technical University, Darmstadt, Germany. *Porcupine* is based on a Zigbee board [10]. The latest version of the *Porcupine* has up to 4GB of memory, 6 tilt switches, 2 light sensors, a temperature sensor, real time clock and a low power 3-axis 3-D accelerometer. The sampling frequency of the accelerometer can be tuned in the range between 0.5Hz to 1.6kHz for *X* and *Y* axes, and a range of 0.5Hz to 550Hz for the *Z* axis, Figure 3.

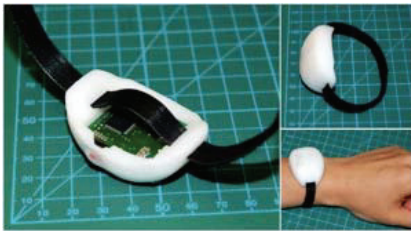


Fig 3. Porcupine Sensor v2.5 with modular sensor and plastic cases (Image Source: Embedded Sensing System Group, TU Darmstadt, Germany).

III. PRE-PROCESSING AND FEATURE SELECTION

A. Pre-processing

As a result of the chaotic nature of acceleration data, it is extremely complex to recognise human activities based on the raw data. The main characteristic of a human activity is often hidden in the raw data. In addition, due to the non-stationary nature of the motion and other characteristics of human activity an adaptive approach (including in respect to feature selection) requires online update of classifier structure as well as of its parameters. This means the classifier structure (including its inputs – features) to be updated in parallel with the recognition task which means to evolve.

Pre-processing is an important stage of any machine learning and pattern recognition algorithm which provides a valuable level of interpretability from raw data in respect to

the input data. This stage is usually done off line on the basis of a large number of training data from a number of subjects. This approach has obvious disadvantages in respect of the representativeness of the training data, the large amount of time and memory needed to process the data. It includes several tasks, such as; i) outliers and missing data detection, ii) data cleaning and filtering, iii) data normalisation or standardisation, iv) feature extraction and feature selection. Pre-processing is often done off-line. It is important to mention that pre-processing can increase the recognition rate by up to 40%. In this study we propose and test several pre-processing methods which are particularly suitable for adaptive and evolving classifiers.

B. Feature Selection

Feature selection aims to reduce the computation level and enhance recognition characteristics based on the analysis of the importance of each feature (usually in terms of statistical characteristics) [7]. Although real-time algorithms work in a sample-per-sample basis, for recognizing human activity from motion sensor it is recommendable to use short time windows as instances [11]. For example, if the sampling frequency is 50Hz a data instance represents 20ms which is a very short period to get a representative motion pattern.

The features for the classifier that is supposed to recognise the human activity are usually different statistical characteristics of the signal. For example, [5] suggest using the following features:

- *Mean value of the signal*; this represents the local posture of the body;
- *Standard deviation*: It corresponds to the amount of motion presented in the signal. This is very valuable information to differentiate activities with a very different pattern such as *walking* and *running*, for example;
- *Energy of the signal*: It represents the stress. The energy of the signal indicates the dynamics of the motion.
- *Correlation*: It helps differentiate simple from complex movements. For example, both *walking* and *running* have similar acceleration pattern in all dimensions whereas *climbing stairs* have a very different pattern in two dimensions.

All these features are computed for a window with size of 268 samples with an overlap of 128 samples, Figure 4.

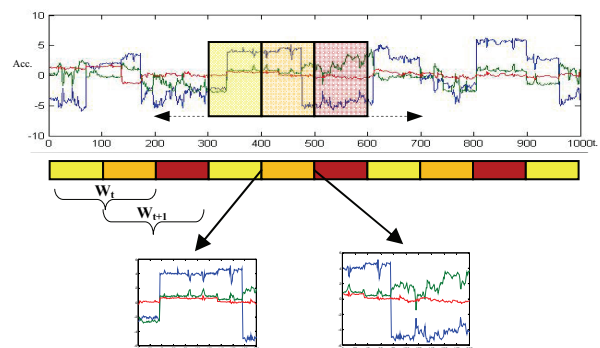


Fig.4 Sampling Windows

C. Feature Extraction.

Due to the same physical principles, different accelerometer axes exhibit some correlation regardless of which human activity is being monitored. For example, significant values of the correlation has been found between axes ($R^2=0.4$) in human motion.

1) *Principal Component Analysis (PCA)*: One of the most widely used approaches for feature extraction which successfully addresses the issue of correlation of raw input data by orthogonalisation is the Principal Component Analysis (PCA). PCA aims to transform the raw data that are, usually, correlated into a smaller set of uncorrelated variables. In this transformation the new variables are linear combinations of the original (measured) variables and in this sense, the overall number of variables that are required to be measured is not being reduced; what is reduced is the number of inputs to the classifier (manipulated variables by the classifier) by preserving the majority of the information in terms of data variance which is useful for correct classification.

PCA has been applied widely in various areas, including spectroscopy, computing vision, intelligent sensors. It is based on producing new variables maximising the variance of the data. PCA is very helpful in revealing the hidden dynamics that is often 'hidden' in a raw signal. It relies on so called singular value decomposition (SVD) by which the most efficient direction in terms of classifying the data is being determined. This efficient representation is achieved by projection in a least-squares (LS) sense. To get these projections a covariance matrix, C is computed and thereafter a new matrix, V of eigen-values and eigen-vectors, which diagonalise the previously computed matrix, are worked out. Once the matrix V has been computed, eigenvectors, denoted as matrix w are sorted in decreasing order regarding their respective eigen-values. Data is standardized and projected onto the matrix, w .

2) *Linear Discriminant Analysis (LDA)*: PCA is very useful in reducing the correlation and reducing the number of inputs to the classifier (thus the computational complexity). However, PCA does not take into account what activity corresponds to certain motion pattern, i.e. it takes only input data into account and not the class labels (outputs in terms of the classifier).

If data is arranged based on the classes (activity types), that we expect to have, so called Linear Discriminant Analysis (LDA) approach can be applied. While PCA get projections of input data in a way that maximises their separability, LDA helps finding these directions which will be best to distinguish between different classes (activities). A combination of both, PCA and LDA has proven to be an effective pre-processing technique in spectroscopy. Sample variance (σ^2) and sample variance per class (σ_c^2) are used to form scatter matrices (S_b and S_w). S_w denotes the **within**-class scatter matrix.

Likewise, S_b represents a singular matrix denoted as **between**-classes scatter matrix. The separation between those

matrices is defined as a measure of the signal-to-noise ratio for class labelling.

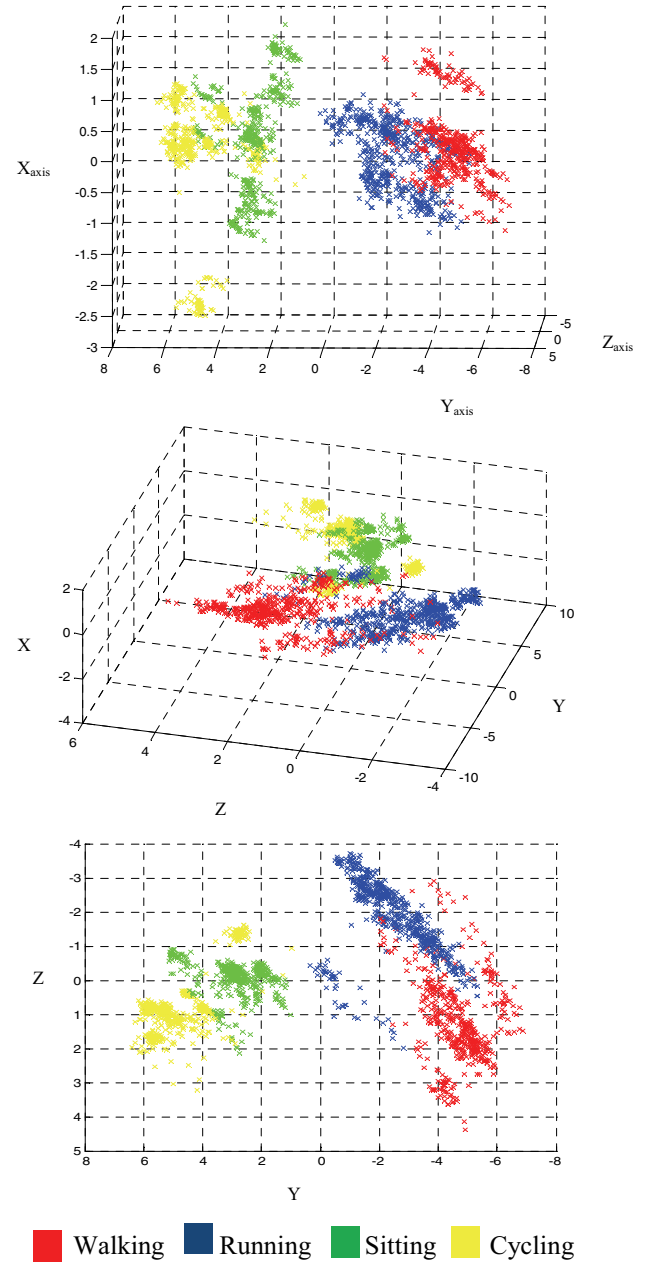


Fig. 5 Features grouped by labels after rLDA-PCA applied

Hence, a criterion function J is computed on the basis of these two matrices, in the following form:

$$J(w) = \frac{w^T S_b w}{w^T S_w w} \quad (1)$$

Values of w which maximizes J are taken as desirable projections of data regarding not just their features (inputs to the classifier) but the class label (output) such as activity type.

3) *On-line, recursive PCA, rPCA*: All the procedures described above regarding pre-processing techniques such as PCA and LDA are calculated off-line and require, therefore, all the training data to be available beforehand, high memory/storage and high computational power capacities. They have been applied in face recognition problems [12], but are not suitable for real-time, on-line applications in their original form. In this study we need a recursive implementation of the combination of PCA-LDA [12]. This approach is applicable on-line and in real-time. PCA and LDA are compatible techniques because whereas PCA tries to find better projections of data according features, LDA finds what projections discriminate classes better.

PCA projections represented by the eigen-values, v can be calculated expressed as suggested by [11] as follows:

$$v_k = \frac{1}{k} \sum_{i=1}^k u_i \cdot u_i^T \cdot \frac{v_{i-1}}{\|v_{i-1}\|} \quad (2)$$

where u denotes an orthogonal projection and k is the current time instant/sample.

This expression is based on the definition of v as $\lambda \cdot x = C \cdot x$ and $v = \lambda \cdot x$ where λ is the magnitude of the eigen-value, $\|v\|$ and C the covariance matrix.

This expression, (2) can be expressed recursively by weighting its current and last estimates by $\frac{k-1}{k}$ and $\frac{1}{k}$ respectively; then we get:

$$v_k = \frac{k-1}{k} \cdot v_{k-1} + \frac{1}{k} \cdot u_k \cdot u_k^T \cdot \frac{v_{k-1}}{v_{k-1}} \quad (3)$$

The algorithm estimates ortho-normalized vectors to ensure maximum independence of the directions. Thus, from the first order eigenvector a new one is computed by subtracting its orthogonal projection. The formula which describes this projection u_j is as follows:

$$u_k = u_{k-1} - u_{k-1}^T \cdot \frac{v_{k-1}}{\|v_{k-1}\|} \cdot \frac{v_k}{\|v_k\|} \quad (4)$$

4) *On-line recursive LDA, rLDA*: In order to get the desired projections in terms of LDA we have to calculate the between-, and within- classes scatter, S_b and S_w recursively:

$$S'_{wk} = S'_{wk-1} + (Y_k - (\sigma_c^2 - \sigma^2)) \cdot (Y_k - (\sigma_c^2 - \sigma^2))^T \quad (5)$$

$$S_{wk}^{-1} = \frac{S_{w(k-1)}^{-1} + (Y_k - (\sigma_c^2 - \sigma^2)) \cdot (Y_k - (\sigma_c^2 - \sigma^2))^T S_{w(k-1)}^{-1}}{1 + (Y_k - (\sigma_c^2 - \sigma^2))^T \cdot S_{w(k-1)}^{-1} \cdot (Y_k - (\sigma_c^2 - \sigma^2))} \quad (6)$$

$$S_{wk} = A_{k-1} \cdot S_{wk}^{-1} \cdot A_{k-1}^T \quad (7)$$

$$S_{wk}^{-1} = A_{k-1} \cdot S_{wk}^{-1} \cdot A_{k-1}^T \quad (8)$$

$$S_b = A_{k-1} \cdot \left(\sum_{i=1}^c n_i \cdot (\sigma_c^2 - \sigma^2) \cdot (\sigma_c^2 - \sigma^2)^T \right) \cdot A_{k-1}^T \quad (9)$$

where $Y_k = x_k - \mu$ is the variance; A_k is the matrix of desired orthogonal projections (eigenvectors) S_w^{-1} and S_b .

An explanatory diagram of the combination of both methods in a same time step, k is shown in Fig. 6. At the end of the algorithm desired eigenvectors which guarantee best projections and features that enhance the separation between classes of activities are provided. Consequently, the raw data, x are being mapped onto the feature space, f (see Fig. 5).

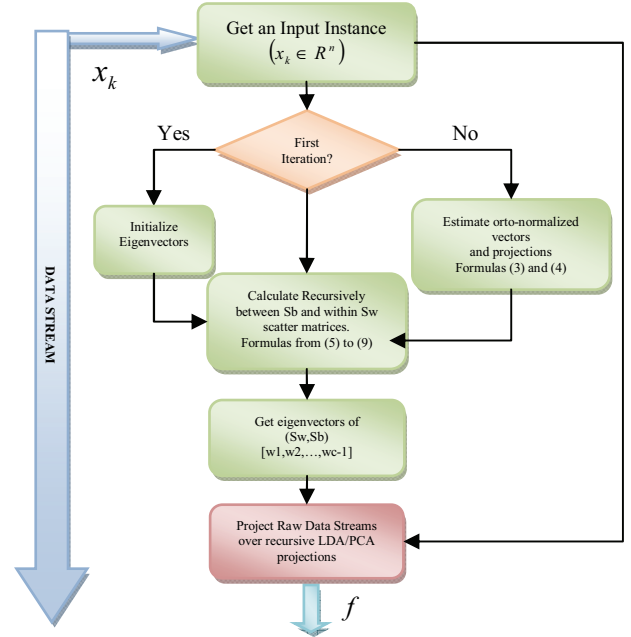


Fig. 6 rLDA-PCA scheme

IV. EVOLVING CLASSIFIER

A classifier is a mapping from the feature space to the class label space. In this study we will develop and apply a self-learning evolving fuzzy rule-based (FRB) classifier, eClass [2]. The structure of the FRB classifiers consists of a set of fuzzy logic (IF-THEN) rules, the fuzzy sets that define the linguistic terms such as 'Low', 'Medium', 'High' etc., aggregation operators such as 'AND', 'OR', 'NOT' and number of inputs/features, n . In our study we consider an evolving FRB classifier, *eClass* which is characterised by having an evolving (not fixed) structure of the following form [1]:

$$R^i: \text{IF}(f_1 \text{ is } \sim f_1^{i*}) \text{ AND } \dots (f_n \text{ is } \sim f_n^{i*}) \text{ THEN } (L^i) \quad (10)$$

where $f = [f_1, f_2, \dots, f_n]^T$ is the vector of features – for our study we consider $n=16$; R^i denotes the i^{th} fuzzy rule; $i=[1, N]$; N is the number of fuzzy rules; $(f_j \text{ is } \sim f_j^{i*})$ denotes the j^{th} fuzzy set of the i^{th} fuzzy rule; $j=[1, n]$; f^{i*} is the focal point of the i^{th} rule antecedent (note that this is a prototype - a real, existing data sample not the mean). L^i is the label of the class of the i^{th} prototype (focal point).

This set of fuzzy rules (note, that due to its evolving nature their number, N is not fixed) defines with its antecedent/premise (IF) part a $16-D$ feature space, $f \in R^{16}$ and with its consequent part it determines in a fuzzy way (with a degree of fulfilment) the class label (activity), $L_i, i=[1, K]$.

The inference in *eClass* is produced by ‘winner takes all’ rule [1]:

$$L = L^*; i^* = \arg \max_{i=1}^N (\tau^i) \quad (11)$$

where τ^i denotes the firing level of (degree of confidence in) the i^{th} fuzzy rule, which is determined as a product of the membership values, μ_j^i of the j^{th} feature, $j=[1, n]$ to the fuzzy set (f_j is $\sim f_j^{i*}$):

$$\tau^i = \prod_{j=1}^n \mu_j^i(f_j); i=[1, N] \quad (12)$$

The membership functions that describe the degree of association with a specific prototype are usually of Gaussian form (characterized by good generalization capabilities and coverage of the whole feature space):

$$\mu_j^i = e^{-\frac{1}{2} \left(\frac{d_j^i}{r_j^i} \right)^2}; i=[1, N]; j=[1, n] \quad (13)$$

Where d_j^i is the distance between a sample and the prototype (focal point) of the i^{th} fuzzy rule; r_j^i is the spread of the membership function, which also represents the radius of the zone of influence of the fuzzy rule.

A. Learning *eClass*

Normally, classifiers use evolutionary algorithms or gradient-based based schemes to be trained-offline. Some examples of this are NN combined with back propagation. Evolving intelligent systems such as *eClass* are designed to be fully on-line in estimating and training. This is achieved by means of an evolving (self-developing) FRB structure. This FRB structure is created from data-stream inputs $z = [f^T, L]^T$. Focal points are generated to describe the nature and dynamics of the signal.

This process of self-calibration resembles adaptive control and estimation. Two phases are formed during the process:

- i) class estimation (classification);
- ii) Update or evolution of the model.

In the first phase the class label is unknown, consequently it is estimated. During the second phase this same class label is used as a supervisory information to update the model structure (so named process of **structure evolution**), as well as its internal parameters.

There are a couple of differences between *eClass* and conventional classifiers:

- i) Starting from scratch, the fuzzy rule-base structure self-develops itself. On the other hand, the

structure of a conventional classifier is determined off-line (and then fixed).

- ii) Also the on-line learning process considers this flexible rule-base structure.

The overall fuzzy rule base structure has a K number of sub-rule based with the same number of consequents. However, the number of rules must be not less than the number of classes ($N \geq K$). That's to say, every data from a non-previously seen label, automatically becomes a prototype. Model updating procedures (replacement and removal of rules) makes this situation temporal. Later, this prototype is usually replaced by more descriptive prototypes. In this sense, *eClass* learns from the scratch and a fixed class number does not need to be set.

The partitioning model is generated by a density model which defines the potential or density of each point in the feature space. For this reason, a Cauchy function of the sum of distance between a particular data sample and all other data samples of its same class label (L), in the feature space [2];

$$D_k^i(z_k) = \frac{1}{2 - \left[1 / \sqrt{\sum_{j=1}^{n+1} (z_k^j)^2} \right] \sum_{j=1}^{n+1} z_k^j b_{k-1}^i} \quad (14)$$

$$k = 2, 3, \dots \quad D_1(z_1) = 1$$

$$\text{where } b_k^i = b_{k-1}^i + \frac{(z_k^i)}{\sqrt{\sum_{j=1}^{n+1} (z_k^j)^2}}, b_1^i = \frac{(z_1^i)^2}{\sqrt{\sum_{j=1}^{n+1} (z_1^j)^2}}$$

$i=[1, n+1]$ and $z = [f^T, L]^T$ is the input/output vector.

The use of a density model comes from the following principle: ‘every point with the highest density is chosen to be the focal point (prototype) for the antecedent of a fuzzy rule’. By means of this procedure, it is possible to define rules with high descriptive power.

The data density model needs to be updated each time a new data sample is added; logically, density values of the existing prototypes have to be updated. This procedure is also done in a recursive way:

$$D_k^L(z_l^{i*}) = \frac{(N_k^L - 1) D_{k-1}^L(z_l^{i*})}{N_k^L - 1 + \left[(N_k^L - 2) \left(\frac{1}{D_{k-1}^L(z_l^{i*})} - 1 \right) + d(z_l^{i*}, z_k) \right]} \quad (15)$$

The updating of the model procedure is as follows: The density of the new data sample is calculated recursively using formula (14) and after this calculation, updating of the density/potential of each existing prototype is computed recursively using (15). After both densities have been computed/updated, conditions are inspected. If the density of the new data sample is higher than any other previously existing prototypes of the same class L ; then it is a good candidate to become a prototype of a new rule. It means that this point has enough descriptive power.

$$D_k^L(z_k) > D_k^L(z_L^{i*}); \forall i^* \in N^L \quad (16)$$

In the previous case, a new fuzzy rule around the new added

prototype is formed. This condition leads to a gradual increase of the sub-rule-base. This is one of the fundaments why this computational process is named “evolving”.

$$z^{(N^L+1)*} \leftarrow z_k \quad (17)$$

One the new generated prototype has been created; its new density is set to 1. Its density value is updated, according to the influence of this prototype on each new collected data sample (15):

$$D_k^L(z^{(N^L+1)*}) \leftarrow 1 \quad (18)$$

It is possible that after adding a rule from a new prototype to the model, this rule would be ambiguous or existing prototypes already describe this behaviour/characteristic. Hence, each time a fuzzy rule is added it is also verified whether any of the existing prototypes of the current sub-rule base are already described by this rule to a degree higher than e^{-1} :

$$\exists i, i = [1, N^L]; \mu_i^j(z^{N^L+1}) > e^{-1} \quad \forall j, j = [1, n] \quad (19)$$

If any of the existing prototypes satisfy condition (19), all rules that correspond to them are removed from the sub-rule-base (indeed, they are replaced for the new formed rule).

The spread of membership functions of the sub-rule-base of the respective class are also recursively updated on the basis of the data distribution:

$$r_k^i = \rho r_{k-1}^i + (1 - \rho) \sigma_{k-1}^i; \quad i = [1, N^L] \quad (20)$$

Note ρ as learning parameter; it sets how fast the spread of the membership functions will concur to the local scatter of that prototype; the normal used value is 0.5.

V. EXPERIMENTAL RESULTS AND EVALUATION

In this section the proposed classification scheme for human activity recognition (*eClass_HAR*) which includes recursive PCA and LDA as well as the evolving FRB Classifier, *eClass* is compared with several well known classifiers. These classifiers are grouped in two groups: i) static, off-line classifiers, and ii) on-line and evolving classifiers. The former, generally, provide higher classification rate, but impose heavy requirements on the memory and computational resources. They keep most of the data in the memory and process all of them at once. On the other hand, the latter, process the data sample by sample (on-line) and may be able to adapt their structure (evolve) as a reaction to the variance in the data pattern. As a result of this adaptation and evolution they result in less complex models and due to the on-line processing manner they require significantly less computational resources. For this reason, they use to have a better performance in terms of CPU and memory usage which compensate the somewhat lower accuracy.

All the results of the comparison are tabulated in Table I. As it can be seen, the proposed *eClass_HAR* classifier is very competitive. It has very low memory requirements and CPU usage. Classification rate is also higher than that of any other on-line classifier and very competitive with the off-line

classifiers.

TABLE I COMPARISON TABLE

Off-line Classifiers	Memory, KB	Max. CPU, %	Class. rate
Hoeffding Adaptive Tree	2.3	10%	81%
Incremental Naïve Bayes	4.868	14%	76.53%
Nearest Neighbour (Euclidean Distance)	3.64	47%	72%
Kstar (Entropic Distance Measure)	4.624	50%	83.55%
Locally-Weighted Learning + Naïve Bayes	1.853	48%	75.16%
On-line Classifiers			
OzaBagASHT (Adaptative Tree)	0.105	18%	70.53%
OzaBagADWIN (Adaptative Windowing + Adaptative Tree)	0.992	13%	70.61%
SingleClassifierDrift (Drift Detection+ Adaptative Tree)	0.012	8%	49.99%
SingleClassifierDrift (Drift Detection + Incremental Naïve Bayes)	0.01	6%	63.41%
AdaHoeffdingOptionTree (Adaptative Tree + Incremental Naïve Bayes)	0.01	8%	70.45%
eClass HAR (proposed)	0.02	5%	71.4%

Figure 7 shows an evolution graph of the classification rate produced by *eClass_HAR*. The top plot shows that recognition rate may drop during the first subject of test (the classifier can start with a pre-training on a different set of subjects or ‘from scratch’). Following the first subject, the classifier self-tunes its parameters as well as its self-learns its structure (evolves) and, as a result, raises its classification rate. After an increase to values around 70-80% there is some saturation (and even reverse). This is due to the known phenomena in signal and data processing that ‘more data not always brings more useful information’.

The bottom plot illustrates the effect of self-learning by temporal re-training. Because of the low computational requirements of the algorithm, it is also possible to update the classifier online by providing in real-time ground truth (activity labels). Several such update periods (with duration of 1 to 2 min) were used in real-time. It is obvious that they helped *eClass_HAR* to increase the classification rate to 83-84% which surpasses all off-line classifiers. It is due to the fact that the classifier adapts its structure (fuzzy rules) to match better the nature and dynamic of the data stream.

In Table II, a confusion matrix is presented which includes all activities that has been considered. It can be seen that the general activities such as *walking*, *running*, *lying down* or even sporty ones (such as *bicycling*) are classified with higher rates. On the other hand, there are also two intermediate activities that are composed of several sub-activities. These are *vacuuming* and *stretching*. The classification rates for these activities are fluctuating because they involve several general activities. In order to overtake this issue, larger windows or

feedback from objects via integrated sensors would help to increase the classification rates. This will be studied and

reported in a further publication.

TABLE II. CONFUSION MATRIX

	Walking	Running	Bicycling	Standing	Lay down	Vacuuming	Stretching	Sitting
Walking	89.42	0	9.12	1.46	0	0	0	0
Running	18.11	76.38	3.15	2.36	0	0	0	0
Bicycling	4.67	0	89.25	5.61	0	0	0.47	0
Standing	0	0	0	99.14	0.43	0	0.43	0
Lay down	0	0	0	9.02	90.98	0	0	0
Vacuuming	7.25	0	19.85	10.31	2.29	58.78	1.53	0
Stretching	4.17	0	23.33	27.50	0.00	37.50	7.50	0
Sitting	0	0	0	1.33	23.11	0	4.44	71.11

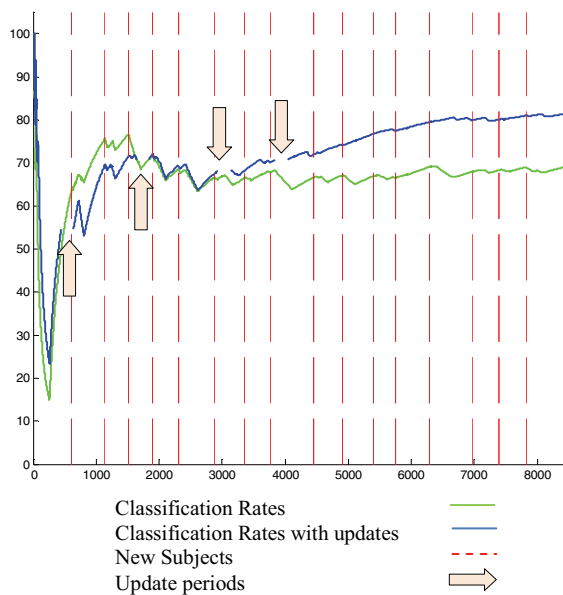


Fig 7. Classification rate evolution graph (the effect of update by eClass_HAR is clearly seen to improve the classification rate despite switching to new subjects).

VI. CONCLUSION

This work presents the ground of a new paradigm of processing HAR data - eHAR. Evaluation and experimentation has shown that results are very promising. This new evolving paradigm applied to the HAR systems provides self-adaptation and self-learning capabilities. In this sense, the classifier is able to evolve its FRB structure to reflect the evolution of the habits and behaviour of a specific human user or a change to a new user (see Figure 7). In this way, eHAR enabled intelligent devices can be personalized. For example, same type of activities (such as *walking*) is usually performed in a different manners by different people depending on age, habits etc. Users with different complexions perform activities with a different intensity or energy. The performance of eHAR is very competitive in comparison with existing off-line and incremental approaches (see Table I). Off-line approaches, however, require a huge amount of training data, time and

computational resources and less relevant for new users (see also Figure 7). Incremental approaches are outperformed by eHAR, but more importantly, they do not keep an evolving structure as the proposed approach does. The proposed approach is also linguistic (IF-THEN rules can be easily extracted that are human-intelligible).

REFERENCES

- [1] P. Angelov, *Evolving Rule-based Models: A Tool for Design of Flexible Adaptive Systems*, Heidelberg: Springer, ISBN 3-7908-1457-1, 2002
- [2] P. Angelov, X. Zhou, "Evolving Fuzzy-Rule-based Classifiers from Data Streams" in *IEEE Trans. on Fuzzy Systems*, vol. 16, no. 6, 2008, pp. 1462-1475,
- [3] N. Robertson and I. Reid, "A general method for human activity recognition in video" in *Comput. Vis. Image Underst.*, Vol. 104, 2006, pp. 232-248.
- [4] W. Chen, H. Aghajan, "Real-Time human pose estimation: a case study in algorithm design for smart camera networks" in *Proceedings of the IEEE*, vol. 96, no. 10, 2006, pp.1715-1732.
- [5] E.M Tapia, S.S. Intille, W. Haskell, K. Larson, J. Wright, A. King, R. Friedman, "Real-Time recognition of physical activities and their Intensities using wireless accelerometers and a heart rate monitor" in *11th IEEE Intern. Symposium on Wearable Computing*, 2007, pp.37-40.
- [6] S. Saponas, J. Lester, J. Froehlich, J. Fogarty and J. Landay "iLearn on the iPhone: Real-Time Human Activity Classification on Commodity Mobile Phones". Univ. of Washington, Tech. Report, UW-CSE-08-04-02, 2008.
- [7] U. Maurer, A. Smailagic, D.P. Siewiorek and M. Deisher, "Activity recognition and monitoring using multiple sensors on different body positions" in *Intern. Workshop on Body Sensor Networks*, 2006, pp.104-116.
- [8] J. Andreu, and P. Angelov, "Real-time human activity recognition from wireless sensors using evolving fuzzy systems" in *IEEE World Congress on Computational Intelligence*, 2010, pp. 18-23.
- [9] SunSPOT World [Online]. Available: <http://sunspotworld.com/>
- [10] K. Van Laerhoven, G. Hans-Werner and Y.G. Malliaris, "Long-term activity monitoring with a wearable sensor node." in *Intern. Workshop on Wearable and Implantable Body Sensor Networks*, 2006, pp. 171-174.
- [11] N. Ravi, N. Dandekar, P. Mysore, and M.L. Littman, "Activity recognition from accelerometer data" in *IAAI'05*, vol. 3, 2005, pp. 1541-1546.
- [12] I. Dagher, "Incremental PCA-LDA algorithm" in *IEEE Intern. Conf. on Computational Intelligence for Measurement Systems and Applications*, 2010, pp. 97-101.