ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 12, December 2014

Recognizing user activities in smart environments with automatically constructed fuzzy finite automata

H. Karamath Ali[1], Dr. D. I. George Amalarethinam[2]

Associate Professor in Computer Science, Periyar E.V.R. College, Trichy, India[1]

Associate Professor in Computer Science & Director-MCA, Jamal Mohamed College, Trichy, India[2]

Abstract: In order to provide personalized services, an activity recognition system has to decide the current activity performed by the user before the user finishes the activity, and it has to predict the next likely activity. This requirement strongly suggests the need for online recognition of activities to provide context-aware assistance or guidance. For online recognition the system must keep track of the changes in the sensing environment, and for each change in the sensor outputs, it has to decide whether there is any change in the activity performed by the user. The system can use the previous inputs upto the most recent one to decide which activity is performed. But, the system should not wait for future inputs for making decisions. This paper proposes an extension of the earlier methods for automatically constructing an automaton for online recognition of user activities. When tested with a publicly available data set, the proposed methods achieve highly promising results.

Keywords: Pervasive Computing, Context Awareness, Activity Recognition, Online recognition, Finite Automata, Environmental Sensors, Activities Of Daily Living, Smart Environments.

I. INTRODUCTION

One of the main objectives of pervasive computing is to offer context aware services to users. A context aware system needs to know the activities being performed by the user. Automatic and unobtrusive recognition of human activities in a smart environment can be used for offering a wide range of services from healthcare and surveillance to luxurious services like automatically adjusting room ambience to suit the mood of the user[1].In a smart environment there are a large number of sensors embedded in every imaginable object. The sensor outputs provide information about the user's interaction with the objects. Using this information the activity recognition system has to decide and, if need be, predict the activity performed by the user. Based on this decision appropriate services may be provided to the user.

Deciding the activity becomes difficult because of the large volume of data to be handled and the lack of preciseness on the part of users in performing activities. There happens to be considerable variation in the number, order and duration of the constituent steps of an activity even if the same user is performing the same activity at different times. Moreover, the need for online recognition of user activities requires the system to identify the intended activity without waiting for future inputs.

A number of probabilistic and structural methods have been used to address the problem of activity recognition. Use of fuzzy automata is one of the structural methods. Manually analyzing the activity to be recognized and constructing the automata has been the approach used by researchers. While this is suitable for simple, well-structured and repetitive activities like walking and running, manual construction of the automata for other activities of daily living(ADLs) like washing, watching TV, etc. becomes very tedious. Algorithms for automatic construction of fuzzy finite automata for offline and online recognition of activities are proposed in our earlier papers[2][3]. In this paper extended versions of the algorithms are presented. These algorithms produce highly promising results when tested with publicly available data sets.

The remainder of this paper is organized as follows: section II gives an overall view of the related work found in the literature. The problem is defined in section III. Section IV proposes an algorithm named Auto-Fuzzy Automata Algorithm(AFAA) which is an extension of the method for offline recognition of activities proposed in [2]. In section V, an algorithm named Online Recognition Algorithm(ORA) that is an extension of the method for online recognition of activities proposed in [3] is described. Conclusion is given in section VI.

II. RELATED WORK

Researchers have used a number of models to develop activity recognition systems. Some of the widely used models are Hidden Markov Models(HMMs), dynamic and naïve Bayes networks, decision trees, nearest neighbour algorithms and Support Vector Machines (SVMs). A number of variations of these models have also been used. Detailed description of these models can be found in [4][5].

The main objective of this research work is to explore the use of finite automata for activity recognition. So, earlier works that used finite automata are given below.

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 12, December 2014

One of the earlier works that used fuzzy finite automata(FFA) for activity recognition was presented by Friedrich Steimann and Klaus-Peter Adlassnig[6]. They presented a framework for an intelligent monitor that derived the current status of a patient by fuzzy state transitions on pre-processed input continuously observed by clinical instrumentation.

Dhruv Mahajan et al.[7] presented a framework for activity recognition and detection of unusual activities in video data. The framework was based on a model of physical, logical and event layers of finite state machines. The usual patterns of activities were learned by the finite state machine layers, in an unsupervised mode. In the recognition phase, activities that did not conform to the learned activity patterns were flagged as abnormal.

Thiago Teixeira et al.[8] presented an activity recognition system for assisted living applications and smart homes. Camera nodes placed on the ceiling were used to locate the user. An inertial sensor, an accelerometer, a gyroscope and a magnetometer which were worn by the user provided direction and motion information. These four measurements were parsed using a lightweight hierarchy of finite state machines. The finite state machines were manually constructed by "a field expert - someone with enough insight to be able to dissect the activity of interest into a finite state machine(FSM)".

A Fuzzy Rule based Classifier and two Fuzzy Finite State Machines(FFSM) were used by A. Alvarez-Alvarez, et al.[9] to recognize activities like working in the desk room, crossing the corridor, having a meeting, etc. Using the classifier an approximate position of the user at the level of discrete zones such as office, corridor and meeting room was obtained. One of the FFSM was used for human body posture recognition. Localization and posture recognition were combined by the other FFSM.

Gonzalo Bailador and Gracián Triviòo[10] proposed a syntactic pattern recognition approach based on fuzzy automata, which could cope with the variability of patterns by defining imprecise models. The approach was called temporal fuzzy automata as it allowed the inclusion of time restrictions to model the duration of the different states. This approach was used for recognizing hand gestures.

A technique based on a hierarchical finite state machine, to detect check-out related primitive activities in a retail store, was proposed by Hoang Trinh et al.[11] . Their approach used visual features and predefined spatial constraints on the hand motion to capture particular motion patterns performed in primitive activities. The approach was applied to the problem of retail fraud detection. The FSM was constructed manually. Obviously, the number of component activities of hand motion in a check-out counter was very small, and were often repeated in the same order.

The use of fuzzy finite state systems for human gait modeling was demonstrated in [12]. In their approach, the fuzzy states and transitions were defined by the expert while the fuzzy rules and membership functions regulating the state changes were derived automatically by a genetic fuzzy system. The gait cycle was defined as the interval between two successive events (usually heel contact) of the same foot. So manually deciding the states and transitions was not that tough.

Nattapon Noorit[13] proposed a human activity recognition method based on FSM model. The basic actions with their properties for each person in the interested area were extracted and calculated. The action stream with related features (movement, referenced location) was recognized using predefined FSM recognizers.

In these studies, fuzzy or ordinary finite automata are used to deal with variability and impreciseness of human activities. But, the states of the fuzzy automata and the transitions between them have to be manually defined by an expert. This will be tedious for activities like cooking or cleaning which are not so structured. Also, users perform activities in different ways at different times. Moreover, having to deal with large number of states manually becomes onerous and cumbersome.

So, automatic construction of fuzzy finite automata for recognizing activities was proposed in our earlier papers [2][3]. In this paper two algorithms namely, Auto-Fuzzy Automata Algorithm(AFAA) and Online Recognition Algorithm(ORA) are described which are the extensions of the methods for offline and online recognition of activities in [2] and [3] respectively.

III. THE PROBLEM

The objective is to recognize activities from sensor readings in a smart environment. For this, the time series data of sensor readings is divided into time slices of constant length. Each time slice is labeled with the activity performed during that time slice. A vector $\vec{x}_t = (x_t^1, x_t^2, \ldots, x_t^N)^T$ is used to represent the sensor readings at time slice t, where x_t^i represents the input from the i^{th} sensor x^i during the time slice and 'N' is the number of sensors. The activity performed during time slice 't' is represented by y_t. So, the task of the activity recognition system is to find an association between a sequence of observation vectors x = $\{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n\}$ and a sequence of activity labels y = $\{y_1, y_2, \ldots, y_n\}$. Figure 1 illustrates this setup, with Δt representing a time slice[14].

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 12, December 2014

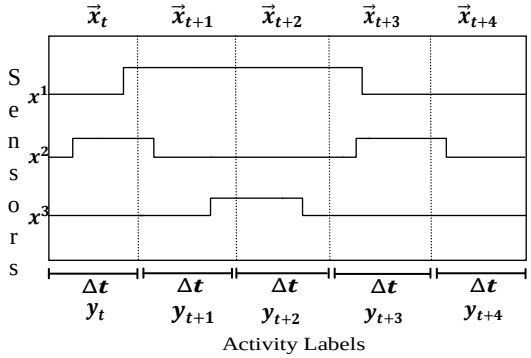


Figure 1. Sensor readings and Time slices

IV. THE AUTO-FUZZY AUTOMATA ALGORITHM(AFAA)

Figure 2 gives an overall view of the AFA algorithm for recognizing activities represented by test data. The sensor output data is preprocessed to obtain the string equivalents of event sequences of the activity. Using the SL-infer algorithm[15] an FA that accepts these strings is constructed. By using appropriate membership assignment function(F_1) and multi-membership resolution function(F_2)[16], fuzziness is incorporated into the constructed automata. The resulting fuzzy finite automaton is used to decide the level of acceptance of the test data.

Data And Experiment

There are a number of publicly available data sets collected in experimental smart environments. One such data set collected and made public by Tim van Kasteren, et al.[17] was used to test the AFAA. The data set has been collected by observing the behavior of inhabitants inside their homes using wireless sensor networks. Output of the binary sensors have been annotated with the activities performed by the subjects during predefined time intervals. A data set so collected for 25 days of 10 activities such as preparing dinner, using washroom and sleeping, was used to conduct the experiment. Three different representations, namely, raw, change point and last-fired, of the sensor data are available in the data set[17]. The raw sensor representation uses the sensor data directly as it was received from the sensors. The change point representation indicates when a sensor event takes place. The last-fired sensor representation indicates which sensor fired last. For each of the representations, six different time slices (600, 300, 60, 30, 10 and 1 seconds) have been used. Sensor data for each of the 10 activities and for each representation were extracted from the data set.

A fuzzy automaton for recognizing a particular activity is constructed as follows. A set of strings representing the time series data for the activity is generated and using the SL-infer algorithm a DFA is constructed. Similarly, a DFA for each of the activities in the data set is constructed.

Fuzzy characteristic is incorporated in the generated DFAs by introducing a membership assignment function($\boldsymbol{F_1}$) and a multi-membership resolution function($\boldsymbol{F_2}$). F_1 and F_2 may be defined to suit the application[16]. The proposed algorithm is tested with $\boldsymbol{F_1}$ defined as

$$\boldsymbol{F_1 : (mf * mvc) + (df * wot)}$$

where $\boldsymbol{0 < mf, df < 1}$, such that $\boldsymbol{mf + df = 1}$,

$\boldsymbol{mvc}$ – the membership value of the current state and

$\boldsymbol{wot}$ (weight of transition) = $(\boldsymbol{1.0 - dist / ns})$, where

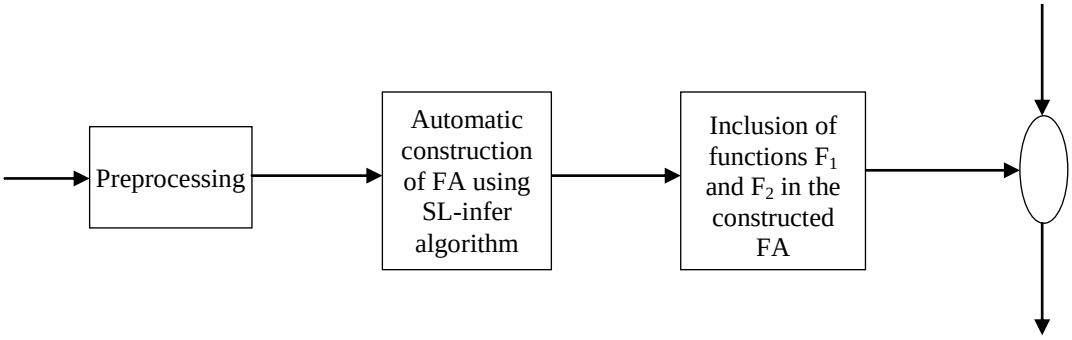
$\boldsymbol{dist}$ – distance between the expected and obtained inputs, and $\boldsymbol{ns}$ – number of sensors in the environment.

The value of $\boldsymbol{mf}$ and $\boldsymbol{df}$ were varied from 0 to 1 in steps of 0.25.

For resolving multi-membership, function $\boldsymbol{F_2}$was defined as follows:

$\boldsymbol{F_2 : \underset{i=1\ to\ n}{Max}\left[F_1(\mu(q_i),\ \delta(q_i, \vec{x}_t, q_m))\right]}$ where $\boldsymbol{n}$ is the number of transitions to a state $\boldsymbol{q_m}$ on input vector $\vec{\boldsymbol{x}}_{\boldsymbol{t}}$ at time $\boldsymbol{t}$, $\boldsymbol{\mu(q_i)}$ is membership value of $\boldsymbol{q_i}$ and $\boldsymbol{\delta(q_i, \vec{x}_t, q_m)}$ is weight of transition from $\boldsymbol{q_i}$to $\boldsymbol{q_m}$ on $\vec{\boldsymbol{x}}_{\boldsymbol{t}}$.

Let $\vec{\boldsymbol{x}} = \vec{\boldsymbol{x}}_1\vec{\boldsymbol{x}}_2 \ldots \vec{\boldsymbol{x}}_n$ be the given sequence of input vectors representing an activity. Initially the list of active states consists only of the initial state, and its membership value is set to 1.0. Suppose $\boldsymbol{q}$ is an active



ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 12, December 2014

state, and $\vec{x}_t$ is an input vector at time $t, 1 \le t \le n$. If $\delta(q, \vec{x}_t)$ is defined by the generated automaton then the transition is carried out and the membership value of next state is calculated using the function F_1. Otherwise, an $\vec{x}_p$ that has minimum distance with $\vec{x}_t$ and for which $\delta(q, \vec{x}_p)$ is defined, is identified. The distance between $\vec{x}_t$ and $\vec{x}_p$ is the number of positions in which they differ. This distance information is used to decide the membership value of the corresponding next possible active state by using F_1. Naturally, a next state caused by an $\vec{x}_p$ with lesser distance from $\vec{x}_t$ has greater membership value than the one caused by an $\vec{x}_p$with greater distance with $\vec{x}_t$.

In short, the membership value of a next state is inversely proportional to the distance between an allowable input and $\vec{x}_t$. Lesser the distance, greater the membership value and vice versa. The problem of multi-membership is solved by choosing the maximum of the membership values. This is summarized in Figure 3. This method may be called 'single transition (ST) function', since there will be only one transition from q even if $\delta(q, \vec{x}_t)$ is not defined. Proceeding in this way, after the membership value of each active state for the last input vector in the given test string is calculated, if the conditional acceptance set $Q_{act}(\vec{x}, \tau_1/\tau_2) \neq \emptyset$ where $0 < \tau_1 \le \tau_2 \le 1$, then the given string is accepted; otherwise, it is not accepted.

An alternative method of checking string acceptance, named 'multi-transition (MT) function' is given in Figure 4. The main difference between the methods is that in single transition method, if $\delta(q, \vec{x}_t)$is not defined, then only the transition from q on an input that has minimum distance with $\vec{x}_t$ is considered; whereas in multi-transition method, all available possibilities from state q are considered. So, there may be multiple transitions from the current active state if $\delta(q, \vec{x}_t)$is not defined. To study the effectiveness of the Auto-Fuzzy Automata Algorithm, ten fold cross validation was used. That is, the data set was divided into ten subsets each of size N/10, where N is the size of the data set.

```
For each q in the list of active states and input x_t
  If δ(q, x_t) is defined then
    calculate mv(δ(q, x_t)) using F_1;

include δ(q, x_t) in the new list of active states;
  else
    Find an x_p such that δ(q, x_p) is defined
and the
      distance between x_t and x_p is minimum;
    calculate mv(δ(q, x_p)) using F_1;
    include δ(q, x_p) in the new list of active stat
  end if
end for
```

Figure 3. Single Transition Function

An FFA for an activity was first constructed using nine subsets and the remaining one subset was used to test the constructed FFA. This was repeated ten times and average values were calculated. The value of mf and df was varied from 0 to 1 in terms of 0.25. Since there are no final states in the constructed FFA, a threshold value is used for conditional acceptance of input strings by the FFA. The threshold value is called *minimum acceptance value*(mav). *mav* was varied from 0.5 to 0.9 in steps of 0.1.

For each possible combination of the three types of data representations(raw, changepoint and lastfired), transition function(single or multi), membership assignment function F_1(with varying mf and df values) and mav, the experiment was carried out. So, the experiment was repeated 150 times = (3 data representations × 2 transition functions × 5 different values for *mf* and *df* × 5 different values for *mav*) and average of true and false positives and negatives were calculated.

The performance of the built fuzzy finite automata was measured by calculating $recall, precision$ and $F-measure$ as follows.

$$recall = tp / (tp + fn)$$
$$precision = tp / (tp + fp)$$

```
For each q in the list of active states and an input x_t
  If δ(q, x_t) is defined then
    calculate mv(δ(q, x_t)) using F_1;

include δ(q, x_t) in the new list of active states;
  else
    For each x_p for which δ(q, x_p) is defined
      calculate mv(δ(q, x_p)) as a function of

mv(q) and the distance between x_t and x_p;

include δ(q, x_p) in the new list of active states;
    endfor
  end if
```

Figure 4. Multi Transition Function

$$F-measure = \frac{2(recall * precision)}{(recall + precision)}$$

where tp, fp and fn represent the number of true positives, false positives and false negatives respectively.

In the results of the experiments it was observed that, for *mf* values 0 and 0.25(and the corresponding *df* values) the obtained average F-measure value was less than that for *mf* = 0.5; also, the average F-measure value for *mf* = 1.0(and *df* = 0) was almost the same as that of *mf* = 0.75(and *df* = 0.25). So, only the results for *mf* = 0. 5(and *df* = 0.5) and *mf* = 0.75(and *df* = 0.25) are presented in this paper. Therefore, in the rest of the paper, F_1^1 and F_1^2 denote $F_1^1 = (0.75 * mvc) + (0.25 * wot)$ and $F_1^2 = (mvc + wot)/2$ respectively.

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 12, December 2014

The average F-measure obtained for the raw, change-point and last-fired feature representations of the data set, are summarized in Table 1. From Table 1, it can be observed that for all the combinations of the single/multi transition functions and membership value assignment functions $\boldsymbol{F_1^1}$ and $\boldsymbol{F_1^2}$, maximum average F-measures are obtained for minimum acceptance value($\boldsymbol{mav}$) 0.8. This is true irrespective of the feature representation method used. The reason for this pattern of results can be explained as follows: when the minimum acceptance value is set to 0.9, it becomes too rigid a condition by resulting in relatively less number of true positives; this reduces the recall and precision rates which in turn reduce the F-measure value. On the other hand, when the $\boldsymbol{mav}$ is varied from 0.5 through 0.7, the acceptance condition becomes too lenient by allowing too many false positives; this also results in reduced recall and precision rates which implies smaller F-measure values.

When $\boldsymbol{mav}$ is set to 0.8, it proves to be moderate and reasonable by producing more true positives and less number of false positives. This explains the highest F-measure rates for $\boldsymbol{mav}$ = 0.8.

Also it can be seen from Table 1 that the highest F-measure values obtained by the multi-transition function((0.84, 0.82, 0.86) and (0.83, 0.82, 0.84)) are slightly higher than that obtained by their single transition counterparts((0.81, 0.79, 0.82) and (0.78, 0.78, 0.80)). This is because when there is no transition from an active state $\boldsymbol{q}$ for an input $\boldsymbol{x}$, the single transition method considers only one – the one with minimum distance from $\boldsymbol{x}$ – of the possible alternative paths from $\boldsymbol{q}$. In some such cases, the chosen path may not match better with the subsequent inputs of the event string. So the membership value may get reduced with each input and may become smaller than the $\boldsymbol{mav}$, and the string gets rejected. The multi-transition method, on the other hand, considers each of the paths from $\boldsymbol{q}$. As a result, the path that matches better – if there is one such path – will result in the acceptance of the string.

Table 1. Average of the F-measure values achieved by AFAA

ST/ MT	$\boldsymbol{mv}$ fn.	*mav*	Raw Data	Change Point	Last Fired
Single Transition	$\boldsymbol{F_1^1}$	0.5	0.75	0.75	0.74
		0.6	0.76	0.75	0.75
		0.7	0.76	0.76	0.75
		0.8	0.81	0.79	0.82
		0.9	0.74	0.77	0.75
	$\boldsymbol{F_1^2}$	0.5	0.75	0.75	0.74
		0.6	0.76	0.75	0.75
		0.7	0.76	0.75	0.75
		0.8	0.78	0.78	0.80
		0.9	0.67	0.72	0.65
Multi Transition	$\boldsymbol{F_1^1}$	0.5	0.78	0.77	0.77
		0.6	0.79	0.78	0.78
		0.7	0.79	0.78	0.78
		0.8	0.84	0.82	0.86
		0.9	0.79	0.80	0.80
	$\boldsymbol{F_1^2}$	0.5	0.78	0.77	0.77
		0.6	0.79	0.78	0.78
		0.7	0.79	0.78	0.78
		0.8	0.83	0.82	0.84
		0.9	0.72	0.76	0.71

Further, for the minimum acceptance value of 0.8 that produces the maximum F-measure value for all combinations, the value obtained for LastFired representation(0.82, 0.80, 0.86 and 0.84) is consistently the highest, although the difference is small. This is followed by RawData and ChangePoint representatations in that order.

Within the same transition functions viz. single or multi transitions, $\boldsymbol{mv}$ assignment function $\boldsymbol{F_1^1}$ and $\boldsymbol{F_1^2}$ produce almost equal results for $\boldsymbol{mav}$s 0.5 through 0.7. But for $\boldsymbol{mav}$ = 0.9, $\boldsymbol{F_1^1}$ produces higher F-measure values than $\boldsymbol{F_1^2}$ for all the three feature representation methods.

The average F-measure values are comparable to the class accuracies obtained by the methods described by Kasteren, et al.[Kas 08]. So, the proposed algorithm AFAA, can be used to effectively recognize user activities in smart environments.

V. ONLINE RECOGNITION ALGORITHM

For online recognition of activities, a Fuzzy Finite Automaton to accept the set of string equivalents of event sequences of all the activities was constructed as explained in section IV. Using the automaton, the activity represented by an event string can be decided only after scanning the last symbol in the string, which is not suitable for online recognition. To recognize activities online, for each occurrence of a sensor event or for an input vector of each timeslice, the activity must be decided without depending on future inputs. So the SL-infer algorithm was extended such that in each state $\boldsymbol{I^a}$ of the constructed automaton the label(s) of the activity(or activities) corresponding to the code sequence in which '$\boldsymbol{a}$' appears get stored. So given the current state of the DFA, the activity corresponding to the input that led to the state can be decided by accessing the activity labels stored in that state.

If there are no common sub-sequences of events among the observed activities, then in the constructed automaton, each activity will have a separate path of states right from the initial state, and in each state of a path only one activity label belonging to that path will be stored. In such cases, the activity can be identified at the occurrence of the first input vector itself. If two or more activities begin with some common event sub-sequences, then more than one activity label will be stored in the states of the corresponding paths of the automaton. The number of

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 12, December 2014

activity labels stored in such a state will be equal to the number of activities that share the event sub-sequence leading to the state. As the control progresses through the successive states of such a path in the automaton, the number of activities stored in the states gets reduced. After passing through a few states the number of which depends upon the length of common sub-sequences between activities, a stage will be reached from where onwards the successive states will have only one activity label. At reaching such a stage, the activity being performed can be easily decided.

To resolve ambiguity when more than one activity label are stored in a state, two tables namely $\boldsymbol{av}$ and $\boldsymbol{ar}$, are maintained. $\boldsymbol{av}$ is of size $\boldsymbol{v} \times \boldsymbol{n}$, where $\boldsymbol{v}$ is the number of different input vectors encountered in the training data, and $\boldsymbol{n}$ is the number of activities. $\boldsymbol{av}[\boldsymbol{i}][\boldsymbol{j}]$ is the number of times vector $\boldsymbol{i}$ appeared in activity $\boldsymbol{j}$. $\boldsymbol{ar}$ is of size $\boldsymbol{n} \times \boldsymbol{n}$. $\boldsymbol{ar}[\boldsymbol{i}][\boldsymbol{j}]$ is the number of times activity $\boldsymbol{i}$ is followed by activity $\boldsymbol{j}$ in the training data. The proposed Online Recognition Algorithm, given in Figure 5 explains how the information in the two tables $\boldsymbol{av}$ and $\boldsymbol{ar}$ is used for resolving ambiguity.

To decide the probable activity for an input say $\vec{\boldsymbol{x}}$, the ORA decides the list of states that become active after $\vec{\boldsymbol{x}}$. For this either the single or multi transition function explained in section IV can be used. In the ORA, '$\boldsymbol{\eta}$' represents either $\vec{\boldsymbol{x}}$ or the minimum distance alternative in case the single transition function is used; $\boldsymbol{preact}$ represents previous activity which means the activity decided to be corresponding to the vector that occurred immediately before $\vec{\boldsymbol{x}}$. Obviously, there is no previous activity for the very first input vector. So, initially $\boldsymbol{preact}$ is set to -1, indicating there is no previous activity and the list of active states consists only of the initial state with membership value set to 1.0.

For the next input $\vec{x}_t$ decide the list of active states using either single or multi transition function;
Let η denote either $\vec{x}_t$ or its closest match as decided in the transition function;
Find the active state q that has maximum membership value;
For each activity label 'i' in state q
if ($preact == -1$) then $w_{q_i} = av[\eta][i]$;
else $w_{q_i} = rf * ar[preact][i] + vf * av[\eta][i]$;
endfor
Activity for $\vec{x}_t$ is the one with maximum w value;

Figure 5. Online Recognition Algorithm

From the decided list of active states, the state $\boldsymbol{q}$ with the maximum membership value is selected. Activity label corresponding to $\vec{\boldsymbol{x}}$ is decided from the list of activity labels stored in $\boldsymbol{q}$. To do this, for each activity label '$\boldsymbol{i}$' stored in '$\boldsymbol{q}$' a weight value $\boldsymbol{w_{q_i}}$ is calculated. $\boldsymbol{w_{q_i}}$ is set to $\boldsymbol{av}[\boldsymbol{\eta}][\boldsymbol{i}]$ if there is no previous activity; otherwise, $\boldsymbol{w_{q_i}}$ is calculated as the sum of the two fractional portions $\boldsymbol{rf}$ and $\boldsymbol{vf}$ of the elements $\boldsymbol{ar}[\boldsymbol{preact}][\boldsymbol{i}]$ and $\boldsymbol{av}[\boldsymbol{\eta}][\boldsymbol{i}]$, respectively. That is, $\boldsymbol{w_{q_i}}$ depends on two factors: the first is the number of times activity '$\boldsymbol{i}$' follows $\boldsymbol{preact}$ in the training data; the second is the number of times $\boldsymbol{\eta}$ appears in activity '$\boldsymbol{i}$' in the training data. The $\boldsymbol{rf}$ and $\boldsymbol{vf}$ values are chosen such that $\boldsymbol{0} < rf, df < 1$ and $\boldsymbol{rf} + \boldsymbol{df} = \boldsymbol{1}$. The activity label '$\boldsymbol{i}$' for which the calculated $\boldsymbol{w_{q_i}}$ is maximum, is decided to be the activity corresponding to the input $\vec{\boldsymbol{x}}$.

To test the Online Recognition Algorithm the data set described in section IV, was used with 'leave one day out' approach[Kas 10]. In this approach, one full day of sensor readings is used for testing and the remaining days are used for training. This is repeated for all the days, for each representation with 60 seconds time slice.The ORA was tested using both the single and multi transition functions with both the membership calculation functions $\boldsymbol{F_1^1}$ and $\boldsymbol{F_1^2}$, explained in section IV. So the experiment was repeated 300 (= 25 days * 3 representations * 4 combinations of transition and $\boldsymbol{mv}$ functions) times, and the performance was measured by calculating the average of $\boldsymbol{recall, precision}$ and $\boldsymbol{F-measure}$. In addition $\boldsymbol{accuracy}$ was calculated as $(\boldsymbol{tp})/(\boldsymbol{tp} + \boldsymbol{tn} + \boldsymbol{fp} + \boldsymbol{fn})$ where $\boldsymbol{tp}$, $\boldsymbol{fp}$, *tn* and $\boldsymbol{fn}$ represent the number of true positives, false positives, true negatives and false negatives.

Average of these measures obtained for the two transition methods and $\boldsymbol{mv}$ calculation functions are summarized in Table 2. From Table 2, it can be observed that generally the multi-transition (MT) function produces better results than the single transition (ST) irrespective of the $\boldsymbol{mv}$ calculation function, except in two cases. First, for changepoint, the recall, F-measure and accuracy values for ST are consistently higher than their MT counterparts, even though the difference is very small. Second, for last fired representation, precision achieved by ST with $\boldsymbol{F_1^2}$ is 94.19±3.1. This is greater than 93.82±4.4 produced by MT with $\boldsymbol{F_1^2}$ by less than one percent.

The different combinations of the transition functions and the $\boldsymbol{mv}$ functions, produce better results for raw data and last-fired representations than for change-point representation. Within the same transition function, $\boldsymbol{F_1^1}$ and $\boldsymbol{F_1^2}$ result in almost the same results. For example, in single transition, with both $\boldsymbol{F_1^1}$ and $\boldsymbol{F_1^2}$ same accuracy values are obtained wheareas for precision, recall and F-measure there are only very small differences most of them less than 0.25%.

When the contents of Table 2 are compared with the results obtained by Kasteren et al.[Kas 10] for the same data set using Naïve Bayes(NB), Hidden Markov Model(HMM), Hidden Semi-Markov Model(HSMM) and Conditional Random Field(CRF),the following observations can be made.

For all the three feature representations, the results obtained by ORA are far better than that obtained by Naïve Bayes(NB) method

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 12, December 2014

Table 2. Performance Measure of ORA

Transition function	mv function	Feature Representation	Precision	Recall	F-measure	Accuracy
Single Transition	F_1^1	RawData	93.14±5.3	64.04±12.6	75.41±9.2	90.66±6.0
		ChangePoint	88.93±5.9	62.79±12.3	73.23±9.2	77.41±9.6
		LastFired	92.91±5.6	68.51±11.3	78.55±8.3	96.92±2.1
	F_1^2	RawData	93.43±5.1	63.91±12.7	75.43±9.3	90.66±6.0
		ChangePoint	88.94±5.9	63.02±12.2	73.42±9.1	77.41±9.6
		LastFired	94.19±3.1	68.47±11.2	78.95±7.7	96.92±2.1
Multi Transition	F_1^1	RawData	94.24±4.8	64.05±12.3	75.78±8.9	90.71±6.0
		ChangePoint	89.57±5.2	62.19±12.1	73.04±8.9	77.40±9.6
		LastFired	94.66±3.0	68.91±11.2	79.42±7.8	96.95±2.1
	F_1^2	RawData	93.51±5.1	64.39±12.3	75.79±8.8	90.70±5.9
		ChangePoint	89.07±5.7	62.20±12.2	72.86±9.0	77.39±9.6
		LastFired	93.82±4.4	68.74±11.2	78.99±8.0	96.92±2.1

ORA produces far better results than the Hidden Markov Model(HMM) for raw data representation. For change-point representation, precision and F-measure values obtained by ORA are consistently greater than those of HMM.

The values obtained by HMM for recall and accuracy are better than that of ORA. For last-fired representation, ORA achieves much better precision, F-measure and accuracy values than HMM. Recall obtained by HMM is slightly better than that of ORA.

When compared with Hidden Semi-Markov Model(HSMM), ORA produces far better results for raw data representation. For change-point representation, ORA produces better precision and F-measure values, whereas the recall and accuracy of HSMM are better than those of ORA. For last-fired representation, ORA generates better precision, F-measure and accuracy values. Recall by HSMM is better than that of ORA.

As far as Conditional Random Field(CRF) is concerned, ORA gives better precision, recall, F-measure and accuracy values for raw data representation. For change-point representation precision and F-measure by ORA are greater than those of CRF. The recall and accuracy values of CRF are better than those of ORA. For last-fired represenations all values obtained by ORA are much better than that of CRF.

The above discussion can be summarized as follows: ORA is far better than NB irrespective of feature representation methods. It is better than all the four methods, if only raw-data representation is considered. If only the last-fired representation is taken into account ORA is better than CRF. ORA gives performance that is comparable with that of HMM, HSMM and CRF as far as change-point representation is concerned.

VI. Conclusion

Relatively less attention has been paid by researchers to automatic construction and use of fuzzy finite automata for activity recognition. In this paper an algorithm named Auto-Fuzzy Automata Algorithm demonstrated that a DFA can be automatically constructed and fuzziness incorporated into it for recognizing activities. Since the results are promising, the AFA algorithm can be tested with any other data sets. The AFA algorithm was extended by the Online Recognition Algorithm, to recognize activities in real time by incorporating the ability to decide which activity is performed by the user in each time slice. The OR algorithm will be very useful in providing context aware personalized services to users in smart environments. Both the algorithms were tested with a publicly available data set and found to achieve very promising results. The algorithms can be easily applied to recognize user activities in smart environments with environmental sensors.

References

[1] Clemens Lombriser, Oliver Amft, Piero Zappi, Luca Benini and Gerhard Tröster, "Benefits of Dynamically Reconfigurable Activity Recognition in Distributed Sensing Environments", Activity Recognition in Pervasive Intelligent Environments, Atlantis Press, pp.265-290, 2011.

[2] H. Karamath Ali and D. I. George Amalarethinam, "Activity Recognition with Fuzzy Finite Automata", Proceedings of World Congress on Computing and Communication Technologies, IEEE Xplore, ISBN: 978-1-4799-2876-7, pp. 222-227, 2014.

[3] H. Karamath Ali and D. I. George Amalarethinam, "Automatic Construction of Finite Automata for Recognizing User Activities in Smart Environments", IOSR Journal of Engineering, Volume 3, Issue 12, pp. 40-45, December 2013.

[4] Liming Chen and Ismail Khalil, "Activity Recognition: Approaches, Practices and Trends", Activity Recognition in Pervasive Intelligent Environments, Atlantis Ambient and Pervasive Intelligence, Atlantis Press, Volume Editors: Liming Chen, Chris Nugent, Jit Biswas, Jesse Hoey, pp.1-31, 2011.

[5] Oscar D Lara and Miguel A Labrador, "A Survey of Human Activity Recognition using wearable sensors", IEEE Communications Survey and Tutorials, Volume 15, Issue 3, pp. 1192-1209, November 2012.

[6] Friedrich Steimann & Klaus-Peter Adlassnig, "Clinical Monitoring with Fuzzy Automata", Journal of Fuzzy Sets and Systems, Volume 61 Issue 1, pp. 37-42, January 1994.

[7] Dhruv Mahajan, Nipun Kwatra, Sumit Jain, Prem Kalra and Subhashis Banerjee, "A framework for activity recognition and detection of unusual activities", Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing, pp. 15–21, 2004.

[8] Thiago Teixeira, Deokwoo Jung, Gershon Dublon and Andreas Savvides, "Recognizing Activities from Context and Arm Pose using Finite State Machines", Third IEEE/ACM International Conference on Distributed Smart Cameras, pp. 1-8, 2009.

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

[9] A. Alvarez-Alvarez, J. M. Alonso, G. Trivino, N. Hern´andez, F. Herranz, A. Llamazares and M. Oca˜na, "Human Activity Recognition applying Computational Intelligence techniques for fusing information related to WiFi positioning and body posture", 2010 IEEE International Conference on Fuzzy Systems (FUZZ), pp.1- 8, July 2010.

[10] Gonzalo Bailador, GraciánTriviòo, "Pattern recognition using temporal fuzzy automata", Journal of Fuzzy Sets and Systems, Volume 161, Issue 1, pp. 37-55, January 2010.

[11] Hoang Trinh, Quanfu Fan, Jiyan Pan, Prasad Gabbur, Sachiko Miyazawa and Sharath Pankanti, "Detecting Human Activities In Retail Surveillance Using Hierarchical Finite State Machine", IEEE International Conference on Acoustics, Speech and Signal Processing(ICASSP), pp. 1337-1340, May 2011.

[12] Alberto Alvarez-Alvarez, GracianTrivino, Oscar Cord´on, "Human Gait Modeling Using a Genetic Fuzzy Finite State Machine", IEEE T. Fuzzy Systems, 20(2), pp. 205-223, 2012.

[13] Nattapon Noorit and Nikom Suvonvorn, "Human Activity Recognition From Basic Actions Using Finite State Machine", Proceedings of the First International Conference on Advanced Data and Information Engineering (DaEng-2013) Lecture Notes in Electrical Engineering, Volume 285, pp. 379-386, 2014.

[14] Tim van Kasteren, Athanasios Noulas, Gwenn Englebienne and Ben Kr¨ose, "Accurate Activity Recognition in a Home Setting", Proceedings of the 10th international conference on Ubiquitous computing, pp. 1-9, 2008.

[15] Henning Fernau, "Algorithms for learning regular expressions from positive data", Journal of Information and Computation, Volume 207, Issue 4, pp. 521-541, April 2009.

[16] Mansoor Doostfatemeh and Stefan C. Kremer, "New directions in fuzzy automata", International Journal of Approximate Reasoning, pp.175–214, 2005.

[17] T.L.M. van Kasteren, G. Englebienne and B.J.A. Kröse, "Human Activity Recognition from WirelessSensor Network Data: Benchmark and Software", Chapter 8, Activity Recognition in Pervasive Intelligent Environments, Atlantis Press, pp. 165-185, 2010.