International Engineering Journal For Research & Development

E-ISSN No:2349-0721

Volume 2 Issue 4

# EVOLVING USER BEHAVIOUR PROFILES RECURSIVELY USING STATISTICAL COSINE DISTANCE

**Rahul Patil [1], Prof.Vinay S Kapse [2]**

**[1]M Tech Student, VIT, Nagpur, [2]Asst.Prof. VIT, Nagpur,**

**[1]rahulpatil6040@gmail.com, [2]vinaykapse82@gmail.com**

---

## ABSTRACT:

**Recognizing the behaviour of others in real time is a significant aspect of different human tasks in many different environments. When this process is carried out by software agents or robots, it is known as user modeling. The recognition of users can be very beneficial for assisting them or predicting their future actions. Most existing techniques for user recognition assume the availability of handcrafted user profiles, which encode the a-priori known behavioural repertoire of the observed user. However, the construction of effective user profiles is a difficult problem for different reasons: human behaviour is often erratic, and sometimes humans behave differently because of a change in their goals. This last problem makes necessary that the user profiles we create evolve.**

**Keywords: ANN, LVQ, Bayesian, SVM, Decision Tree.**

## 1. INTRODUCTION

Recognizing the behaviour of others in real time is a significant aspect of different human tasks in many different environments. When this process is carried out by software agents or robots, it is known as user modeling. The recognition of users can be very beneficial for assisting them or predicting their future actions. Most existing techniques for user recognition assume the availability of handcrafted user profiles, which encode the a-priori known behavioural repertoire of the observed user. However, the construction of effective user profiles is a difficult problem for different reasons: human behaviour is often erratic, and sometimes humans behave differently because of a change in their goals. This last problem makes necessary that the user profiles we create evolve.

The user-profiling or user-modelling task involves inferring unobservable information about users from observable information about them, for example their actions or utterances (Zukerman & Albrecht, 2001). In order to perform this task, agents must deal with the uncertainty of making inferences about the user in the absence of complete information. Like other knowledge-based approaches, user modelling is concerned with two main issues: acquisition and representation.

The acquisition of user profiles is related to the mechanisms an agent has to formulate assumptions about users. In this regard, users provide valuable information about themselves during interaction with computers. Both user behaviour and user judgments about agent actions are the most trustworthy sources of information for acquiring profiles and, from their successful interpretation, information agents will be able to tailor actions to user idiosyncrasies. In addition, user profiles can be refined, boot-strapped or even exclusively built based on a number of additional sources, such as the characteristics and shared beliefs of groups of users with similar tastes, domain knowledge or level of expertise.

The most usual approach to profile acquisition, however, is the application of learning mechanisms. Learning of user profiles based on the observation of user behaviour leads to explicit representations of user interests that enable agents to make decisions about future actions. As profiles in this approach are usually expressed using the

representational formalisms of specific learning algorithms (e.g. decision trees, rules, etc.), the learning process and the content of the resulting profiles are, consequently, interlinked issues.

A user profile allows agents to make decisions about actions to be carried out with individual, previously unseen pieces of information. If user profiles are acquired using a learning algorithm, decision making is directly supported by the learning method. In other cases, several methods can be used in order to compare user interests and information items. Besides profile-item matching, agents acting collaboratively with other agents toward a common goal also need to be endowed with a method to match user profiles in order to find users with similar interests.

The adaptation of user profiles is also an important factor in user profiling. Assuming that certain user interests persist for a long time (e.g. the interest in painting in an adult), an agent should become increasingly better at satisfying user needs, i.e. an agent should be able to gradually converge to the part of user needs that is predictable and consistent over time. However, since the interaction may extend over a long period of time, the user interests cannot be assumed to remain constant during such a time. A change in user interests can be anything from a slight shift in relative priorities to a complete loss of interest in some domains or an increase of interest in others. An agent must be able to detect such changes and respond to them by adapting user profiles in order to refine future agent behaviour.

## 2. BACKGROUND AND RELATED WORK

Different techniques have been used to find out relevant information related to the human behaviour in many different areas. The literature in this field is vast.

Macedo et al. [5] propose a system (WebMemex) that continuously captures users' Web surfing history and uses this history to provide the users and their friends with suggestions of related Web pages to what they are currently viewing. This system acts as an instantiation of an architecture for capturing and asynchronously sharing experiences for the automated recommendation of related information. Web application is adopted by users. Users are allowed to create many profiles. Thus, they can use this application in different ways: (a) as a personal application which only suggests information from her own personal Web history; or (b) as a collaborative application where information is shared between groups of friends and recommends related URLs from the collective Web history.

Pepyne et al. [6] describe a method using queuing theory and logistic regression modeling methods for profiling computer users based on simple temporal aspects of their behaviour. Instead of digging deep into the OS commands users generate, we instead simply look at the busy period structure of the sample path they generate, i.e., when they are busy, how long they are busy, and roughly how active they are while busy. This leads to a very fast and simple implementation, requiring only simple time-stamping and counting. ' Being so simple and fast, another potential application, outside of user profiling, is for making a quick first pass over audit data, either throughout the day, or to aid forensic investigation after an attack by locating traces for further more detailed examination. The limitation is that the approach does not provide real-time detection, since it only makes its decisions when a session ends-although by simple modification one can imagine a system that dynamically comes to a decision as features reveal themselves.

Gody and Amandi [7] present a technique to generate readable user profiles that accurately capture interests by observing their behaviour on the web. There is a lot of work focusing on user profiling in a specific environment, but it is not clear that they can be transferred to other environments. However, the approach we propose in this paper can be used in any domain in which a user behaviour can be represented as a sequence of actions or events. Because sequences play a crucial role in human skill learning and reasoning [8], the problem of user profile classification is examined as a problem of sequence classification.

According to this aspect, Horman and Kaminka [9] present a learner with unlabeled sequential data that discover meaningful patterns of sequential behaviour from example streams. Unsupervised sequence learning is important to many applications. A learner is presented with unlabeled sequential data, and must discover sequential patterns that characterize the data. Popular approaches to such learning include (and often combine) frequency-based approaches and statistical analysis. However, the quality of results is often far from satisfactory. Though most previous investigations seek to address method-specific limitations, we instead focus on general (method-neutral) limitations in current approaches. This paper takes two key steps towards addressing such general quality-reducing flaws. First, we carry out an in-depth empirical comparison and analysis of popular sequence learning methods in terms of the quality of information produced, for several synthetic and real-world datasets, under controlled settings of noise. We find that both frequency-based and statistics-based approaches (i) suffer from common statistical biases based on the length of the sequences considered; (ii) are unable to correctly generalize the patterns discovered, thus flooding the results with multiple instances (with slight variations) of the same pattern. They had shown empirically that the relative quality of different approaches changes based on the noise present in the data: Statistical approaches do better at high levels of noise, while frequency-based approaches do better at low levels of noise. As our second contribution, we develop methods for countering these common deficiencies. They had shown how to normalize rankings of candidate patterns such that the relative ranking of different-length patterns can be compared. They had additionally shown the use of clustering, based on sequence similarity, to group together instances of the same general pattern, and choose the most general pattern that covers all of these.

Lane and Brodley [10] present an approach based on the basis of instance-based learning (IBL) techniques, and several techniques for reducing data storage requirements of the user profile. There work has demonstrated a technique for mapping the temporal data occurring in the anomaly detection task onto a space in which IBL learning can be formulated. the system is biased toward detecting anomalous conditions quickly, but generating false alarms rarely. a new clustering technique is based on sequential, greedy selection of clusters. The greedy clustering technique was able to produce a large saving in storage requirements, with an overall small loss in accuracy. K-centers clustering was unable to match the performance of greedy clustering in this domain in either convergence rate or detection accuracy. The algorithms employed here could be implemented as single detection elements in an overall anomaly detection scheme that also employs alternative data sources such as biometric measurements, resource consumption measurements, and activity time-of-day. Such ensemble classification methods are well known in the machine learning community.

## 3. PROBLEM FORMULATION

This paper addresses the problem of user behaviour detection as exemplified in the task of classifying sequences of user-command data into either of five categories: novice user, non programmer user, programmer user, scientist user and expert users. The data used are those of Schonlau et al., described in Section 5. These data consist of 15,000 commands from each of 50 different users. The data are configured in two ways: (1) randomly injected with data from users outside the community of 50 and (2) each user crossed with every other user to compare the effects of every user against all other users.

There is a lot of work focusing on user profiling in a specific environment, but it is not clear that they can be transferred to other environments. However, the approach we propose in this paper can be used in any domain in which a user behaviour can be represented as a sequence of actions or events. Because sequences play a crucial role in human skill learning and reasoning [8], the problem of user profile classification is examined as a problem of sequence classification. According to this aspect, Horman and Kaminka [9] present a learner with unlabeled sequential data that discover meaningful patterns of sequential behaviour from example streams. Popular approaches to such learning include statistical analysis and frequencybased methods. Lane and Brodley [10] present an approach based on the basis of instance-based learning (IBL) techniques, and several techniques for reducing data storage requirements of the user profile.

It should be emphasized that all of the above approaches ignore the fact that user behaviours can change and evolve. However, this aspect needs to be taken into account in the proposed approach. In addition, owing to the characteristics of the proposed environment, we need to extract some sort of knowledge from a continuous stream of data. Thus, it is necessary that the approach deals with the problem of classification of streaming data. Incremental algorithms build and refine the model at different points in time, in contrast to the traditional algorithms which perform the model in a batch manner. It is more efficient to revise an existing hypothesis than it is to generate hypothesis each time a new instance is observed. Therefore, one of the solution to the proposed scenario is the incremental classifiers.

An incremental learning algorithm can be defined as one that meets the following criteria [15]:

- It should be able to learn additional information from new data.
- It should not require access to the original data, used to train the existing classifier.
- It should preserve previously acquired knowledge.
- It should be able to accommodate new classes that may be introduced with new data.

## 4. PROPOSED APPROACH

### 4.1 Brief Overview about problem definition

Recognizing the behaviour of others in real time is a significant aspect of different human tasks in many different environments. The recognition of users can be very beneficial for assisting them or predicting their future actions. Most existing techniques for user recognition assume the availability of handcrafted user profiles, which encode the a-priori known behavioural repertoire of the observed user. However, the construction of effective user profiles is a difficult problem for different reasons, human behaviour is often erratic and sometimes humans behave differently because of a change in their goals. This last problem makes necessary that the user profiles we create and evolve. Previous research studies in this environment [20] focus on detecting masquerades (individuals who impersonate other users on computer networks and systems) from sequences of UNIX commands. However, EVABCD creates evolving user profiles and classifies new users into one of the previously created profiles. Thus, the goal of EVABCD in the UNIX environment can be divided

- Creating and updating user profiles from the commands the users typed in a UNIX shell.
- Classifying a new sequence of commands into the predefined profiles.

The creation of a user profile from a sequence of UNIX commands should consider the consecutive order of the commands typed by the user and the influence of user past experiences. This aspect motivates the idea of automated sequence learning for computer user behaviour classification; if we do not know the features that influence the behaviour of a user, we can consider a sequence of past actions to incorporate some of the historical context of the user. However, it is difficult, or in general, impossible, to build a classifier that will have a full description of all possible behaviours of the user, because these behaviours evolve with time, they are not static and new patterns may emerge as well as an old habit may be forgotten or stopped to be used. The descriptions of a particular behaviour itself may also evolve, so we assume that each behaviour is described by one or more fuzzy rules. A conventional system do not capture the new patterns (behaviours) that could appear in the data stream once the classifier is built. In addition, the information produced by a user is often quite large. Therefore,[21] we need to cope with large amounts of data and process this information in real time, because storing the complete data set and analyzing it in an offline (batch) mode, would be impractical. In order to take into account these aspects, we use an evolving EVABCD system that allows for the user behaviours to be dynamic, to evolve. These proposed approach agent used as a monitor, analyze, and detect abnormalities based on time varying to detect masqueraders applied for any

command-line argument. The proposed approach for designing classifier having automatic clustering, classifier designing and classification of user behaviour profile.

- Creating and evolving the classifier
  - Creating the user behaviour profile
  - Evolving the classifier
- User classification

### 4.2 Architecture Block diagram of system

Primary objectives of the proposed system can be summarized as creating user behaviour profile in terms of classifying relevant sequence of events. Knowledge about computer [22] user with increased complexity of thinking user behaviour and segmentation of subsequent relevant events evaluated by using automatic clustering, Classifier design and classification method.

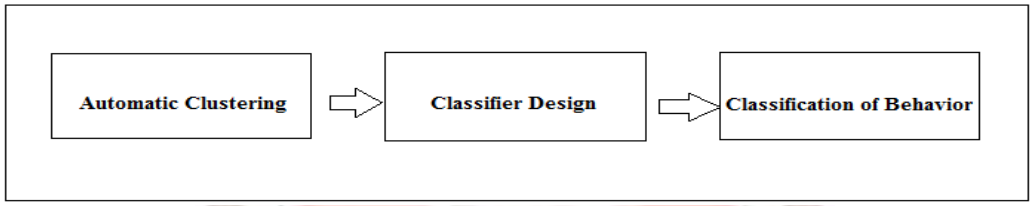


**Fig.4.2 Architecture Block Diagram of EVABCD**

In order to construct a user behaviour profile in online mode from a data stream, we need to extract an ordered sequence of recognized events. These commands are inherently sequential and this sequentiality is considered in the modeling process. According to this aspect based on the work done in order to get the most representative set of subsequences from no of sequence. The proposed approach make use of trie data structure [23]. Above architecture gives us the creation of a user profile from a sequence of UNIX commands. This motivates the idea of automated sequence of learning for computer user commands typed by different UNIX users.

### 4.3 Description about each block and connectivity between each block

If the dependencies of the commands are relevant in the user profile, the suffixes subsequence of subsequences are extend to the end of the given sequence which are already inserted. Considering the previous example, the first subsequence (fls-date-lsg) is added as the first branch of the empty trie. Each node is labeled with the number 1 which indicates that the command has been inserted in the node once, this number is enclosed in square brackets. The suffixes of the subsequence (fdate-lsg and-ls) segmentation of the sequence of commands which store the subsequences in a trie. During creation of the user profile, after inserting the three subsequences and its corresponding suffixes a completed trie is obtained. A new subsequence are inserted into a trie, the existing nodes are modified by new nodes which are created in ABCD profile.

### 4.4 Evolving ABCD

EVABCD classifier is a mapping from the feature space to the class label space. In the proposed classifier, the feature space is defined by distributions of subsequences of events. On the other hand, the class label space is represented by the most representative distributions . Thus, a distribution [24] in the class label space represents a specific behaviour which is one of the prototypes of the EPLib. The prototypes are not fixed and evolve taking into account the new information collected online from the data stream this is what makes the classifier Evolving. The number of these prototypes is not prefixed but it depends on the homogeneity of the observed behaviours. The following sections describes how a user behaviour is represented by the proposed classifier, and how this classifier is created in an evolving manner.

### 4.5 Automatic Clustering

Bayesian classifier is an effective and fundamental methodology for solving classification problems. However, it is computationally efficient when all features are considered simultaneously. But sometimes all the features do not contribute significantly to classification. Evolving connectionist systems (ECOS) are systems that evolve their structure and functionality over time through interaction with the environment. They have some "genetically" predefined parameters knowledge but they also learn and adapt as they operate. They emerge, evolve, develop, unfold through learning, and through changing their structure in order to better represent incoming data. A block diagram of the EVABCD framework. Also the noisy attributes sometimes may decrease the accuracy of classifier. So before classification feature selection is used as a pre-processing step [25].

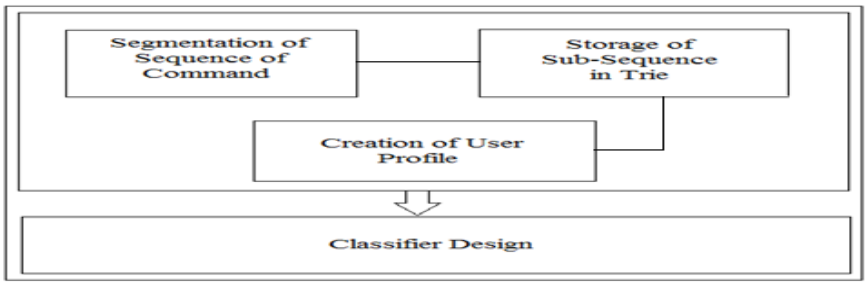


**Fig.4.3 Architecture of Automatic Clustering**

### 4.6 Classifier Designing

The proposed incremental Bayesian classifier is computationally efficient over batch Bayesian classifier in terms of time. The effectiveness of the proposed incremental Bayesian classifier has been demonstrated through experiments on different datasets. It is found on the basis of experiments that the incremental Bayesian classifier has an equivalent power compared to batch Bayesian classifier in terms classification accuracy. However, the proposed incremental Bayesian classifier has very high speed efficiency in comparison to batch Bayesian classifier.

EVABCD receives observations in real time from the environment to analyze. In our case, these observations are UNIX commands and they are converted into the corresponding distribution of subsequences online. In order to classify a UNIX user behaviour, these distributions must be represented in a data space. For this reason, each distribution is considered as a data vector that defines a point that can be represented in the data space. The data space in which we can represent these points should consist of n dimensions, where n is the number of the different subsequences that could be observed. It means that we should know all the different subsequences of the environment a priori. However, this value is unknown and the creation of this data space from the beginning is not efficient. For this reason, in EVABCD, the dimension of the data space also evolves, it is incrementally growing according to the different subsequences that are represented in it.

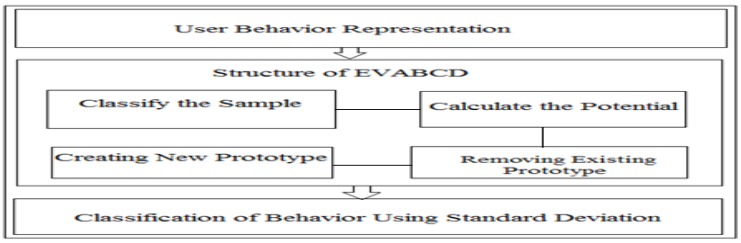


**Fig.4.4 Architecture of Classifier Design**

## 4.6 Classification of User Behaviour

### 4.6.1 Segmentation of the sequence of commands

First, the sequence is segmented into subsequence's of equal length from the first to the last element. Thus, the sequence A1, A2 . . . An where n is the number of commands of the sequence will be segmented in the subsequence's described by Ai . . . where length is the size of the subsequence's created. In the remainder of the paper, we will use the term subsequence length to denote the value of this length. This value determines how many commands are considered as dependent. In the proposed sample sequence (fls-date-ls-date-catg), let 3 be the subsequence length, then we obtain (fls-date-lsg), (fdate-ls-dateg), (fls-date-catg).

### 4.6.2 Storage of the subsequence's in a trie

The subsequence's of commands are stored in a trie data structure. When a new model needs to be constructed, we create an empty trie. And it insert each subsequence of events into it, such that all possible subsequence's are accessible and explicitly represented. Every trie node represents the subsequence's of commands are stored in a trie data structure. When a new model needs to be constructed, we create an empty trie and insert each subsequence of events into it. The subsequence's are accessible and explicitly represented. Every trie node represents an event appearing at the end of a subsequence and the nodes. In order to construct a user behaviour profile in online mode from a data stream. We need to extract an ordered sequence of recognized events in UNIX commands. These commands are inherently sequential and this [26] sequentiality is considered in the modelling process. The construction of a user profile from a single sequence of commands is done by a three step process. Children represent the events that have appeared following this event. Also, each node keeps track of the number of times a command has been recorded into it. When a new subsequence is inserted into a trie, the existing nodes are modified new nodes are created. As the dependencies of the commands are relevant in the user profile, the subsequence suffixes subsequences that extend to the end of the given sequence are also inserted.

### 4.6.3 Creation of the user profile

Once the trie is created, the subsequence's that characterize the user profile and its relevance are calculated by traversing the trie. For this purpose, frequency-based methods are used. In particular, in EVABCD, to evaluate the relevance of a subsequence, its relative frequency or support [27] is calculated. 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 subsequence's of equal size inserted. In this step, the trie can be transformed into a set of subsequence's labeled by its support value. In EVABCD, this set of subsequence's is represented as a distribution of relevant subsequence's. Thus, we assume that user profiles are n-dimensional matrices, where each dimension of the matrix will represent a particular subsequence of commands. In the previous example, the trie consists of nine nodes; therefore, the corresponding profile consists of nine different subsequence's which are labeled with its support. It shows the distribution of these subsequence's. Once a user behaviour profile has been created, it is classified and used to update the Evolving-Profile-Library, as explained in the next section.

### 4.6.4 Mathematical steps in implementation of each block

In this phase, the first step is to extract the significant pieces of the sequence of commands that can represent a pattern of behaviour. When a user types a command, it usually depends on the previous typed commands and it is related to the following commands. According to this aspect, it was used in order to get the most representative set of subsequence's from the acquired sequence, the use of a trie data structure is proposed. This structure is also proposed in to learn a team behaviour and to classify the behaviour patterns of a RoboCup soccer simulation team. Therefore, as we can see, the classifier does not need to be configured according to the

environment where it is used because it can start "from scratch." Also, the relevant information of the obtained samples is necessary to update the library. It is done because the density of the data space surrounding certain data sample changes with the insertion of each new data sample. Following are the mathematical steps in implementation of each block as given:

- Calculate the Potential of Data Sample
- Calculating the Potential Recursively
- Simplifying the Potential Expression
- Classification Method
- Supervised and Unsupervised Learning

**4.6.5 Calculate the Potential of Data Sample**

The potential (P) of the $k^{th}$ data sample is calculated by (1) which represents a function of the accumulated distance between a sample and all the other k-1 samples in the data space [28]. The result of this function represents the density of the data that surrounds a certain data sample where distance represents the distance between two samples in the data space.

The potential is calculated using the Euclidean distance and [29] it is calculated using the cosine distance. Cosine distance has the advantage that it tolerates different samples to have different number of attributes. An attribute is the support value of a subsequence of commands.

Also, cosine distance tolerates that the value of several subsequences in a sample can be null is different than zero. Therefore, EVABCD uses the cosine distance (cos Dist) to measure the similarity between two samples.

Once the corresponding distribution [30] has been created from the stream, it is processed by the classifier. The structure of this classifier includes

- Classify the new sample in a class represented by a prototype.
- Calculate the potential of the new data sample to be a prototype.
- Update all the prototypes considering the new data sample.
- Remove any prototype if needed.

**4.6.6 Calculating the Potential Recursively**

As a prototype is a data sample (a behaviour represented by a distribution of subsequences of commands) that represents several samples which represent a certain class. The classifier is initialized with the first data sample, which is stored in EPLib. Then, each data sample is classified to one of the prototypes (classes) defined in the classifier. Finally, based on the potential of the new data sample to become a prototype [31], it could form a new prototype or replace an existing one.

$$pk(zk) = \frac{1}{2 + \left[\frac{1}{h(k-1)}\left[[-2Bk] + \left[\frac{1}{h}Dk\right]\right]\right]}$$

K=2, 3 …….. ; $p_{1(z_{1)=1}}$

$$Bk = \sum_{j-1}^{n} z_k^j \, b_k^j ; b_k^j = b_{(k-1)}^j + \sqrt{\frac{(z_k^j)2}{\sum_{l-1}^{1}(z_1^l)2}}$$

$$b_1^j = \sqrt{\frac{(z_k^j)2}{\sum_{l-1}^{1}(z_1^l)2}} ; j = [1, n+1] \qquad (3)$$

$$Dk = \sum_{j-1}^{n} z_k^j \sum_{p-1}^{n} z_k^p d_k^{jp}$$

$$d_1^{jp} = \frac{z_1^j z_1^p}{\sum_{l-1}^{n}(z_1^l)2} ; j = [1, n+1]$$

Where $d_k^{ij}$ represents this accumulated value for the k[th] data sample considering the multiplication of the i[th] and j[th] attributes of the data sample. Thus, to get recursively the value of the potential of a sample using (1), it is necessary to calculate different accumulated values which store the result of the multiplication of an attribute value of the data sample by all the other attribute values of the data sample.

### 4.6.7 Simplifying the Potential Expression

Note that the expression in (1) requires all the accumulated data sample available to be calculated, which contradicts to the requirement for real time and online application needed in the proposed approach. For this reason, we need to develop a recursive expression of the potential, in which it is not needed to store the history of all the data. In (3), the potential is calculated in the [32] input-output joint data space, where therefore, the k[th] data sample ($x^k$) with its corresponding label is represented as $z^k$. we develop a recursive expression similar to the recursive expressions developed in using euclidean distance and in using cosine distance. This formula is as follows:

$$Bk = \sum_{j-1}^{n} z_k^j \, b_k^j ; b_k^j = b_{(k-1)}^j + \sqrt{\frac{(z_k^j)2}{\sum_{l-1}^{1}(z_1^l)2}} \qquad (5)$$

$$b_1^j = \sqrt{\frac{(z_k^j)2}{\sum_{l-1}^{1}(z_1^l)2}} ; j = [1, n+1],$$

Using this expression, it is only necessary to calculate $(n-1)$ values where 'n' is the number of different obtained subsequences. This value is represented by b, where $b_k^j, j = [1, n]$ represents the accumulated value for the k[th] data sample. Different accumulated values which store the result of the multiplication of an attribute value by replacing the data sample of other attribute. However, since the number of needed accumulated values is very large, so we [33] simplify this expression in order to calculate it faster and with less memory.

### 4.6.8 Creating New Prototypes

In our particular application, the data are represented by a set of support values and are thus positive. To simplify the recursive calculation of the expression (1), we can use simply the distance instead of square of the distance. For this reason, we use in this case (4) instead of (1)

$$i = [1, NumPrototypes]: p(zk) > P(Proti). \quad (6)$$

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 which represents a part of the observed data samples.

**4.6.9 Removing Existing Prototypes**

After adding a new prototype, we check whether any of the already existing prototypes are described well by the newly added prototype. 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 After adding a new prototype, we check whether any of the already existing prototypes are described well by the newly added prototype. 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

$$i = [1, NumPrototypes]: \ni (zk) > e^1 \quad (7)$$

For this reason, we calculate the membership function between a data sample and a prototype which is defined as where it represents the cosine distance between a data sample $z^k$ and the $i^{th}$ prototype P, 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 of the data. The equation to get the spread of the $k^{th}$ data sample is defined as:

$$Bk = \sum_{j-1}^{n} z_k^j \, b_k^j; b_k^j = b_{(k-1)}^j + \sqrt{\frac{\left(z_k^j\right)2}{\sum_{l-1}^{1}(z_1^l)2}} \quad (8)$$

where represents the cosine distance between a data sample ($z^k$) and the $i^{th}$ prototype (P); 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 of the data. The equation to get the spread of the $k^{th}$ data sample is defined as:

$$\in (k) = \sqrt{\frac{1}{k}\sum_{j=1}^{k} cosDist(Proti, zk)\,; \in i(0) = 1,} \quad (9)$$

**4.6.10 Classification Method**

In order to classify a new data sample, we compare it with all the prototypes stored in EPLib. This comparison is done using cosine distance and the smallest distance determines the closest similarity. This comparison is shown as:

$$Prot *$$

$$class(x2) = class$$

$$Prot * MINT_{i=1}^{NumProt} \left(cosDist(xPrototypei, xz)\right) \qquad (10)$$

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 [34] the time consumed and the computational complexity are reduced and acceptable for real-time applications in order of milliseconds per data sample. The time consumed for classifying a new sample depends on the number of prototypes and its number of attributes.

## 5. EXPERIMENTAL DESIGN

In order to measure the performance of the proposed classifier using the above data, the well-established technique cross validation is used. For this research, 10-fold cross-validation is used: We remove a 10% of the commands from the initial data of each user and the corresponding distributions are calculated (Training Distributions ). Then, the portion of data originally taken out of each user data is analyzed and its corresponding distribution is created (Test Distribution). Using the proposed statistical method, these distributions are compared and the user is classified. As 10-fold cross validation is used, this process is repeated 10 times per user.

## 6. RESULT

In this research, a UNIX command sequence (Test Distribution) is classified into the user behaviour (Training Distribution) with the smallest deviation.

**TABLE 7.2.(a) Total Number of Different Subsequences Obtained**

| No. of Command per User | Sub-Sequence Length | No. of Different Sub-Sequences |
|---|---|---|
| **100** | 3 | 799 |
| | 4 | 799 |
| | 5 | 799 |
| | 6 | 799 |
| **500** | 3 | 1416 |
| | 4 | 1416 |
| | 5 | 1417 |
| | 6 | 1416 |
| **1000** | 3 | 1595 |
| | 4 | 1726 |
| | 5 | 1696 |
| | 6 | 1786 |

The [38] self evolving system makes use of recursive formula to find the potential of a new data sample to form new prototype or replace existing prototype. It is based on distribution of large user profile data to simple and easy to implement. Response time is high to continuous stream of data. It ignore the fact that user behaviour cannot change and evolve. Evolving Profile Library Classifier (EPLib) selection based on distribution of large user profile data. Simple and easy to implement response time is high.

We can see from the set of users results Table.7.2.(b) results obtained with different subsequence lengths for creating the trie (3, 4, 5 and 6) show that the higher classification rates are not obtained using a higher length. The

higher classification rate is usually obtained using subsequences of length 5; this number determines the number of commands considered as dependent for a UNIX user.

**TABLE 7.2.(b) EVABCD Number of Prototypes Created per Group Using 10-Fold Cross Validation**

| Group | Number of Prototype in each of the 10 runs | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| **Normal User** | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| **Masquerades User** | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 2 |

## 7. CONCLUSION

EVABCD is a realization that one of the main modules in a framework of evolving connectionist systems. It is a model to classify user behaviours from a sequence of events. The underlying assumption in this approach is that the data collected from the corresponding environment can be transformed into a sequence of events. This sequence is segmented and stored in a trie and the relevant sub-sequences are evaluated by using a frequency-based method. Then, a distribution of relevant subsequences is created. However, as a user behaviour is not fixed but rather it changes and evolves, the proposed classifier is able to keep up to date the created profiles using an Evolving Systems approach. EVABCD is one pass, noniterative, recursive, and it has the potential to be used in an interactive mode; therefore, it is computationally very efficient and fast. In addition, its structure is simple and interpretable.

A significant advantage of Evolving Agent Behaviour Classification based on Distribution of relevant event is the local learning which allows for a fast adaptive learning that is robust to catastrophic forgetting. EVABCD incorporate important features such as

1) Adaptive learning

2) Non-monotonic reasoning

3) Knowledge manipulation in the presence of imprecision and uncertainties

4) Knowledge acquisition and explanation

EVABCD have all of the features of knowledge-based systems, logic systems, case-based reasoning systems, and adaptive connectionist-based systems combined. Through self-organization and self-adaptation during the learning process. they allow for solving difficult engineering tasks as well as for simulation of emerging and evolving biological and cognitive processes. The self evolving system makes use of recursive formula to find the potential of a new data sample to form new prototype or replace existing prototype.

## REFERENCES

[1] D. Godoy and A. Amandi, “User Profiling in Personal Information Agents: A Survey,” Knowledge Eng. Rev., vol. 20, no. 4, pp. 329 361, 2005.
[2] J.A. Iglesias, A. Ledezma, and A. Sanchis, “Creating User Profiles from a Command Line Interface: A Statistical Approach,” Proc. Int’l Conf. User Modeling, Adaptation, and Personalization (UMAP), pp. 90 101, 2009.
[3] M. Schonlau, W. Dumouchel, W.H. Ju, A.F. Karr, and Theus, “Computer Intrusion: Detecting Masquerades,” Statistical Science, vol. 16, pp. 58 74, 2001.
[4] R.A. Maxion and T.N. Townsend, “Masquerade Detection Using Truncated Command Lines,” Proc. Int’l Conf. Dependable Systems and Networks (DSN), pp. 219 228, 2002.

[5] A. Alaniz Macedo, K.N. Truong, J.A. Camacho Guerrero, and M. Graca Pimentel, "Automatically Sharing Web Experiences through a Hyperdocument Recommender System," Proc. ACM Conf. Hypertext and Hypermedia (HYPERTEXT '03), pp. 48 56, 2003.
[6] D.L. Pepyne, J. Hu, and W. Gong, "User Profiling for Computer Security," Proc. Am. Control Conf., pp. 982 987, 2004.
[7] D. Godoy and A. Amandi, "User Profiling for Web Page Filtering," IEEE Internet Computing, vol. 9, no. 4, pp. 56 64, July/Aug. 2005.
[8] J. Anderson, Learning and Memory: An Integrated Approach. John Wiley and Sons, 1995.
[9] Y. Horman and G.A. Kaminka, "Removing Biases in Unsupervised Learning of Sequential Patterns," Intelligent Data Analysis, vol. 11, no. 5, pp. 457 480, 2007.
[10] T. Lane and C.E. Brodley, "Temporal Sequence Learning and Data Reduction for Anomaly Detection," Proc. ACM Conf. Computer and Comm. Security (CCS), pp. 150 158, 1998.
[11] S.E. Coull, J.W. Branch, B.K. Szymanski, and E. Breimer, "Intrusion Detection: A Bioinformatics Approach," Proc. Ann. Computer Security Applications Conf. (ACSAC), pp. 24 33, 2003.
[12] P. Angelov and X. Zhou, "Evolving Fuzzy Rule Based Classifiers from Data Streams," IEEE Trans. Fuzzy Systems: Special Issue on Evolving Fuzzy Systems, vol. 16, no. 6, pp. 1462 1475, Dec. 2008.
[13] M. Panda and M.R. Patra, "A Comparative Study of Data Mining Algorithms for Network Intrusion Detection," Proc. Int'l Conf. Emerging Trends in Eng. and Technology, pp. 504 507, 2008.
[14] A. Cufoglu, M. Lohi, and K. Madani, "A Comparative Study of Selected Classifiers with Classification Accuracy in User Profiling," Proc. WRI World Congress on Computer Science and Information Eng. (CSIE), pp. 708 712, 2009.
[15] R. Polikar, L. Upda, S.S. Upda, and V. Honavar, "Learn++: An Incremental Learning Algorithm for Supervised Neural Net works," IEEE Trans. Systems, Man and Cybernetics, Part C (Applications and Rev.), vol. 31, no. 4, pp. 497 508, http:// dx.doi.org/10.1109/5326.983933, Nov. 2001.
[16] D. Kalles and T. Morris, "Efficient Incremental Induction of Decision Trees," Machine Learning, vol. 24, no. 3, pp. 231 242, 1996.
[17] F.J. Ferrer Troyano, J.S. Aguilar Ruiz, and J.C.R. Santos, "Data Streams Classification by Incremental Rule Learning with Para meterized Generalization," Proc. ACM Symp. Applied Computing (SAC), pp. 657 661, 2006.
[18] J.C. Schlimmer and D.H. Fisher, "A Case Study of Incremental Concept Induction," Proc. Fifth Nat'l Conf. Artificial Intelligence (AAAI), pp. 496 501, 1986.
[19] P.E. Utgoff, "Id5: An Incremental Id3," Proc. Int'l Conf. Machine Learning, pp. 107 120, 1988.
[20] P.E. Utgoff, "Incremental Induction of Decision Trees," Machine Learning, vol. 4, no. 2, pp. 161 186, 1989.
[21] G.A. Carpenter, S. Grossberg, and D.B. Rosen, "Art2 a: An Adaptive Resonance Algorithm for Rapid Category Learning and Recognition," Neural Networks, vol. 4, pp. 493 504, 1991.
[22] G.A. Carpenter, S. Grossberg, N. Markuzon, J.H. Reynolds, and D.B. Rosen, "Fuzzy Artmap: A Neural Network Architecture for Incremental Supervised Learning of Analog Multidimensional Maps," IEEE Trans. Neural Networks, vol. 3, no. 5, pp. 698 713, Sept.1992.
[23] N. Kasabov, "Evolving Fuzzy Neural Networks for Supervised/ Unsupervised Online Knowledge Based Learning," IEEE Trans. Systems, Man and Cybernetics Part B: Cybernetics, vol. 31, no. 6, pp. 902 918, Dec. 2001.
[24] T. Seipone and J.A. Bullinaria, "Evolving Improved Incremental Learning Schemes for Neural Network Systems," Proc. IEEE Congress on Evolutionary Computation, pp. 2002 2009, 2005.
[25] T. Kohonen, J. Kangas, J. Laaksonen, and K. Torkkola, "Lvq pak: A Program Package for the Correct Application of Learning Vector Quantization Algorithms," Proc. IEEE Int'l Conf. Neural Networks, pp. 725 730, 1992.
[26] F. Poirier and A. Ferrieux, "Dvq: Dynamic Vector Quantization An Incremental Lvq," Proc. Int'l Conf. Artificial Neural Networks, pp. 1333 1336, 1991.
[27] R.K. Agrawal and R. Bala, "Incremental Bayesian Classification for Multivariate Normal Distribution Data," Pattern Recognition Letters, vol. 29, no. 13, pp. 1873 1876, http://dx.doi.org/10.1016/j.patrec.2008.06.010, 2008.