

This is a preprint of the accepted paper: García-Cuesta, E., & Iglesias, J. A. (2012). User modeling: Through statistical analysis and subspace learning. *Expert Systems with Applications*, 39(5), 5243-5250 accessible in its published version through the link doi: <https://doi.org/10.1016/j.eswa.2011.11.015>

# User modeling: Through statistical analysis and subspace learning

Esteban García-Cuesta<sup>a,\*</sup>, José Antonio Iglesias<sup>b</sup>

<sup>a</sup>Department of Physics at Carlos III University, Avda. Universidad, 30 Leganés, Madrid, Spain

<sup>b</sup>Department of Computer Science at Carlos III University, Avda. Universidad, 30 Leganés, Madrid, Spain

## ARTICLE INFO

### Keywords:

User modeling  
Behavior recognition  
Dimensionality reduction  
High dimensional data  
Pattern recognition

## ABSTRACT

One of the challenges which must be faced in the field of the information processing is the need to cope with huge amounts of data. There exist many different environments in which large quantities of information are produced. For example, in a command-line interface, a computer user types thousands of commands which can hide information about the behavior of her/his. However, processing this kind of streaming data on-line is a hard problem.

This paper addresses the problem of the classification of streaming data from a dimensionality reduction perspective. We propose to learn a lower dimensionality input model which best represents the data and improves the prediction performance versus standard techniques. The proposed method uses maximum dependence criteria as distance measurement and finds the transformation which best represents the command-line user. We also make a comparison between the dimensionality reduction approach and using the full dataset. The results obtained give some deeper understanding in advantages and drawbacks of using both perspectives in this user classifying environment.

## 1. Introduction

Recognizing the behavior of a user by observation is an interesting work which can be applied, for example, to behavior prediction tasks. Specifically, if we observe the way that ordinary computer users use the computer, we can get information about them which can be used in different areas, such as interaction with users appropriately (Nasoz & Lisetti, 2007) or computer-aided instruction (Barrow et al., 2008). Computer user modeling is the process of learning about ordinary computer users by observing the way they use the computer. The result of a user modeling process in computer systems is usually stored in user profiles that contain information that characterizes the usage behavior of a computer user.

Users themselves do not know how to articulate what they do, especially if they are very familiar with the tasks they perform. Computer users, like all of us, leave out activities that they do not even notice they are doing. Thus, only by observing users we can model his/her behavior correctly (Hackos & Redish, 1998). However, the construction of effective computer user profiles needs to cope with huge amounts of data.

Most existing techniques for computer user modeling assume the availability of carefully hand-crafted user profiles, which encode the *a priori* known behavior repertoire of the observed user.

Unfortunately, techniques for automatically acquiring user profiles from observations are only beginning to emerge.

A user model may take different forms depending on the purposes for which it is created. As Webb et al. propose (Webb, Pazzani, & Billsus, 2001), user models may seek to describe:

1. the cognitive processes that underlie the user's actions;
2. the difference between the user's skills and expert skills;
3. the user's behavioral patterns or preferences; or
4. the user characteristics.

Recent research has predominantly pursued the third approach, focusing on users' behaviors as proposed by Webb (1993), rather than on the cognitive processes that underlie that behavior. This research is also focused on finding relevant user's behavioral patterns in order to classify a user behavior. We will use the approach presented in Iglesias, Ledezma, and Sanchis (2009) for automatically creating the profile of a user based on the analysis of the sequence of commands s/he typed. However, as this approach generates huge amounts of data, we need to face this aspect by applying dimensionality reduction (DR) techniques. This paper proposes a novel method for reducing this amount of data considerable but without losing relevant information.

The paper is organized as follows; Section 2 provides a brief overview of the background and related work relevant to this research. How the computer user profiles are created from a sequence of commands and the detailed proposed dimensionality reduction method are explained in Section 3. Section 4 describes

\* Corresponding author.

E-mail addresses: [esteban.garcia@uc3m.es](mailto:esteban.garcia@uc3m.es) (E. García-Cuesta), [jiglesia@inf.uc3m.es](mailto:jiglesia@inf.uc3m.es) (J.A. Iglesias)

the experimental setting and the results obtained. Finally, Section 5 contains future work and concluding remarks.

## 2. Background and related work

To model, recognize, or classify the behavior of a computer user is very useful in many different computer areas: Discovery of navigation patterns (Spiliopoulou & Faulstich, 1998), Web recommender systems (Macedo, Truong, Camacho-Guerrero, & da GracCa Pimentel, 2003) or Computer security (Pepyne, Hu, & Gong, 2004). For this reason, the literature of agent modeling is truly vast. Frias-Martinez, Magoulas, Chen, and Macredie (2005) examine how soft computing techniques, including fuzzy logic, neural networks, genetic algorithms, fuzzy clustering and neuro-fuzzy systems, have been used, alone or in combination with other machine learning techniques, for user modeling during the last years.

However, in this research we focus on discovering computer user patterns from the sequence of commands that a user types. The problem of behavior classification is examined as a problem of learning to characterize the behavior of a user in terms of sequences of commands. However, as the information obtained is really large, we also need to apply a reduction of the dimensionality. We thus focus here on the most related work in behavior modeling based on the analysis of sequences of events and how to reduce the number of dimensions to get a better classification rate and a more efficient method.

Kaminka, Fidanboyly, Chang, and Veloso (2002) recognize basic actions based on descriptive predicates, and learn relevant sequences of actions using a statistical approach. Horman and Kaminka (2007) expanded on this approach. A similar process is also used in Huang, Yang, and Chen (2003) to create frequent patterns in dynamic scenes. However, these previous works focused on unsupervised learning, with no ability to classify behaviors into classes.

Han and Veloso (2000) recognize behaviors of robots in a real world scenario using *Hidden Markov Models*. In this case, states in the *HMMs* correspond to an abstracted decomposition of a robot's behavior. The observations of a robot represent its physical state and the corresponding set of Markov states represents its *mental state*. Then, as the intermediate states probabilities of a HMM indicate a behavior in progress, they can be used in anticipating the future behavior (states) of the robot. However, this approach makes a Markovian assumption (the probability of moving from the current state to another is independent of its previous states) in modeling an agent, whereas our proposal takes into account a short sequence of events to incorporate some of the historical context of the agent.

As in this research, there are other areas in which sequential data are analyzed in order to solve a specific problem. In general, the sequence learning problem can be categorized in four basic categories: sequence prediction, sequence generation, sequence classification and sequential decision making. In this paper, the sequence classification is the category analyzed and developed.

A very important issue in sequence learning is temporal dependencies. The following aspect is essential in our research: a current situation or the action that an agent performs usually depend on what has happened before. This aspect is taken into account in our research and in models such as *HMMs*; however, there are some other models which have problems dealing with such dependencies. For example, recurrent neural network models (Giles & Gori, 1998) or reinforcement learning cannot manage efficiently the long-range dependencies. Due to this sequential data approach, thousands of new features are created using different temporal ranges and therefore we have to deal with this huge amount of features to find any pattern associated to the different types of users.

Usually that large amount of features degrades the performance of practical algorithms in a supervised machine learning context: regression or classification. The algorithms have to face with many features that are not necessary for predicting the desired output and they could perform an overkill adjustment of the model parameters leading to overfitting, and reducing the prediction performance on new data sets.

For this reason, a dimensionality reduction approach is therefore needed to transform the high dimensional data into a meaningful representation of smaller dimensionality. Ideally, the reduced representation has a dimensionality that corresponds to the intrinsic dimensionality of the problem. The intrinsic dimensionality of data is the minimum number of parameters needed to account for the observed properties of the data (Fukunaga, 1990).

Principal Component Analysis (PCA) (Jolliffe, 2002) is a standard linear dimensionality reduction technique that performs a dimensionality reduction by projecting the original data onto the linear subspace spanned by the first  $l$  eigenvectors and therefore decorrelating the data. It is optimal for Gaussian distributed classes and it captures the directions of maximum variance in the data using the correlation or covariance matrix. Although it is a very powerful tool and has been used widely, for instance in computer vision to extract the "eigenfaces" (Turk & Pentland, 1991) or as standard DR technique for blog visualization Tsai (2011), it does not necessarily provide meaningful representation of the data. The supervised alternative method is linear discriminant analysis (LDA) (Duda, Hart, & Stork, 2000) which is also used in that context and usually outperforms PCA whenever the label information is available.

More recently, several techniques which try to preserve the local properties of samples to guide the dimensionality reduction have been proposed. LPP (He & Niyogi, 2003) is an alternative tool to PCA and it tries to preserve the locality by incorporating a graph with neighborhood information. Also in He, Yan, Hu, Niyogi, and Jiang Zhang (2005) the authors use the same idea and define the Laplacian matrix to model the local similarities between samples.

Despite of the successful results of those techniques in computer vision, they do not treat specifically the known information about the outputs (side information or labels) which can be very important for other classification applications as in the user modeling application which we are dealing with. We introduce in this research a novel method for dimensionality reduction which introduces the side information to find a lower subspace where the characteristics of the different users lay on.

## 3. User modeling and dimensionality reduction

In this section we explain in detail our proposal for classifying user behavior profiles from a dimensionality reduction perspective. It consists of two main parts: The creation of a computer user profile (sub Section 3.1), and The novel dimensionality reduction technique (sub Section 3.2).

### 3.1. Creating a computer user profile

We consider that the commands typed by a user are usually influenced by past experiences. This aspect motivates the idea of automated sequence learning for behavior classification; if we do not know the features that influence the behavior of an agent, we can consider a sequence of past actions to incorporate some of the historical context of the agent. Indeed, sequence learning is arguable the most common form of human and animal learning. Sequences are absolutely relevant in human skill learning (Sun, Merrill, & Peterson, 2001) and in high-level problem solving and

reasoning (Anderson, 1995). Taking this aspect into account in this paper, the computer user modeling is transformed into a sequence analysis problem where a sequence of commands represents a specific behavior. This transformation can be done because it is clear that any behavior has a sequential aspect, as actions are performed in a sequence.

The commands typed by a computer user are inherently sequential, and this sequentiality is considered in the modeling process (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, in order to get the most representative set of subsequences from a sequence, we propose the use of a *trie* data structure (Fredkin, 1960). This *trie* data structure was also used in Iglesias, Ledezma, and Sanchis (2007) and in Kaminka et al. (2002) where a team behavior was learned as well as in (Iglesias, Ledezma, & Sanchis, 2006) to classify the behavior patterns of a *RoboCup* soccer simulation team.

The construction of a user profile from a single sequence of commands is done as it is explained in Iglesias et al. (2009) for any sequence of events. This process consists of three steps: 1. Segmentation of the sequence of commands, 2. Storage of the subsequences in a *trie*, and 3. Creation of the user profile. These steps are detailed in the following 3 subsections.

### 3.1.1. Segmentation of the sequence of commands

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

In the proposed sample sequence ( $\{ls \rightarrow date \rightarrow ls \rightarrow date \rightarrow cat\}$ ), let 3 be the subsequence length, then we obtain:

$\{ls \rightarrow date \rightarrow ls\}, \{date \rightarrow ls \rightarrow date\}, \{ls \rightarrow date \rightarrow cat\}$

### 3.1.2. Storage of the subsequences in a *trie*

The subsequences of commands are stored in a *trie* in a way that all possible subsequences are accessible and explicitly represented. When a new model needs to be constructed, we create an empty *trie*, and insert each subsequence of events into it. Every *trie*-node represents a command appearing at the end of a sub-sequence, and the node's children represent the commands that have appeared following this command. Also, each node keeps track of the number of times a command has been inserted into it. When a new subsequence is inserted into a *trie*, the existing nodes are modified and/or 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.

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

### 3.1.3. Creation of the user profile

Once the *trie* is created, the subsequences that characterize the user profile and its relevance need to be obtained (where a sub-sequence is a path from the *root* node to any other node of the *trie*). Thus, the *trie* is traversed to calculate the relevance of each sub-sequence. For this purpose, frequency-based methods are used. In particular, to evaluate the relevance of a subsequence, its relative frequency or support (Agrawal, 1995) 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* to the total number of subsequences of equal size inserted.

Thus, in this step, the *trie* can be transformed into a set of subsequences labeled with a value (support). This set of subsequences is represented as a distribution of relevant subsequences. Note that this step does not need to be carried out separately, after creating the *trie*. In the previous example, the *trie* consists of 9 nodes; therefore, the corresponding profile consists of 9 different subsequences which are labeled with its support. Fig. 2 shows the distribution of these subsequences.

Once a user behavior profile has been created, it is classified using the full profile or using the new features obtained after reducing the dimensionality. It is worth it to highlight that, creating the *trie*, a new set of features with temporal information is built although it does not distinguish between relevant or useless information. Then, the main purpose of the dimensionality reduction technique is to extract the temporal patterns which are significantly different for each type of user.

## 3.2. Supervised local maximum variance preserving

The proposed Supervised Local Maximum Variance Preserving (SLMVP) dimensionality reduction method uses the full or partially labeled user dataset. The input data  $\mathbf{X} \in \mathbb{R}^{n \times m}$  are the different tries (relative frequency values) associated to each sample, and  $\mathbf{Y} \in \mathbb{R}^{1 \times m}$  is the label associated to each sample (for instance: novice programmer or experienced programmer).

Previously (García-Cuesta, Galván, & de Castro, 2009) presented a global joint distribution, between inputs and outputs, variance preserving algorithm for a remote sensing application. The main difference between that approach and our proposal is that we take into account the local behavior of the data to find the different

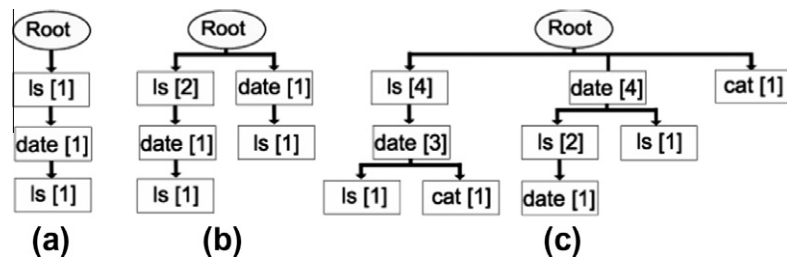


Fig. 1. Steps of creating an example trie.

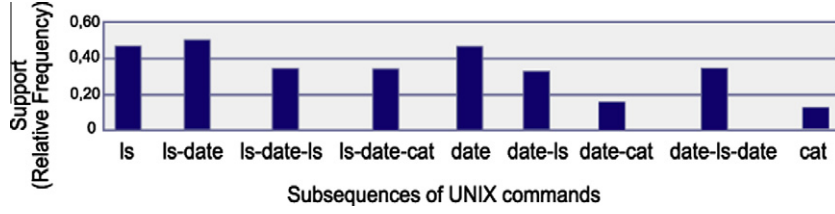


Fig. 2. Distribution of subsequences of commands - Example.

groups using a graph embedding viewpoint. To review the graph embedding framework for dimension reduction lets firstly introduce the general problem of linear dimensionality reduction. Given a set  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m \in \mathbb{R}^n$ , the objective is to find a transformation matrix  $\mathbf{A}$  that maps these  $m$  points to a set of points  $\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_m$  in  $\mathbb{R}^l$  ( $l \ll n$ ), such that  $\mathbf{y}_i$  "best represents"  $\mathbf{x}_i$ , where  $\mathbf{y}_i = \mathbf{A}^T \mathbf{x}_i$ .

Following Belkin and Niyogi (2002) and based on standard spectral graph theory we define a weighted graph  $G = (V, E)$  with edges connecting nearby points to each other. Now, the dimension reduction problem can be consider as mapping the weighted connected graph  $G$  to a line so that connected points stay as close together as possible. As LPP (He & Niyogi, 2003), we wish to choose  $\mathbf{y}_i \in \mathbb{R}^l$  which minimizes the distance with its neighbors following the next objective function

$$\sum_{ij} (y_i - y_j)^2 w_{ij} \quad (1)$$

where  $\mathbf{W} \in \mathbb{R}^{m \times m}$  is the connection (edges of the graph) or similarity matrix. The element  $w_{ij}$  indicates if the vertex pair  $i$  and  $j$  are connected, or measures their similarity.

In the case of multidimensional data the function to optimize can be expressed by

$$\frac{1}{2} \sum_{ij} \|\mathbf{y}_i - \mathbf{y}_j\|^2 w_{ij}. \quad (2)$$

To overcome the drawback of LPP which does not take into account the output information we propose to include the output local information into the input space.

Therefore our proposal is based on two different graphs, one for the inputs and another one for the outputs, and the goal to achieve is to maintain the local properties of both. Then, we set the next two graphs, input and output, as follows:

- Input graph  $\{\mathbf{G}, \mathbf{U}\}$ . In LPP (He & Niyogi, 2003), firstly an adjacency graph is constructed using the  $\epsilon$ -neighborhood of the  $k$ -nearest samples and then it is weighted by a function which is usually an exponential one. Instead of that we assumed that the graph is fully connected and the weighting function is going to model the local behavior. We want to point out that the weighting function implicitly contains the similarities between samples and therefore it must be a function which has metric properties and is positive. This two constraints are also accomplished by kernels Schölkopf (2000), and therefore we can use them as weighting functions too. The linear kernel is the most simple but some other widely used kernels are polynomial, or Gaussian kernels, given by the following expressions:

$$K(x, y) = (1 + x \cdot y)^p, \quad (3)$$

$$K(x, y) = e^{-\frac{\|x - y\|^2}{2\sigma^2}}. \quad (4)$$

- Output graph  $\{\mathbf{H}, \mathbf{V}\}$ . It is constructed following the same procedure that for the input matrix. It is worth it to highlight that the type of kernel used for the input and output graphs does not have to be the same. If the input and output implicit subspaces are different then it would be necessary to use different kernels for each one.

Because we are only interested in maintaining the local properties of both spaces simultaneously, we propose the minimization of the next cost function

$$J_1 = \frac{1}{2} \sum_{ij} \|\mathbf{y}_i - \mathbf{y}_j\|^2 z_{ij} \quad (5)$$

where,  $z_{ij} = u_{ij} \circ w_{ij}$  or  $z_{ij} = \sum_{k=1}^m u_{ik} w_{kj}$ , upon it is a classification or a regression problem.

The Eq. (5) can be solved equivalently as in (He et al., 2005) by

$$\min_{\mathbf{Y}^T \mathbf{D} \mathbf{Y} = \mathbf{I}} \text{tr}(\mathbf{Y}^T \mathbf{L} \mathbf{Y}) \quad (6)$$

where  $D_{ii} = \sum_j z_{ij}$  and  $\mathbf{L} = \mathbf{D} - \mathbf{Z}$ .

If we express the input and output graphs as kernel functions ( $\mathbf{K}_x(\mathbf{x}, \mathbf{x}')$ ) and  $\mathbf{K}_y(\mathbf{y}, \mathbf{y}')$ , then  $\mathbf{L}$  can be derived as:  $\mathbf{L} = \mathbf{D} - \mathbf{Z} = \mathbf{D} - \mathbf{K}_x(\mathbf{x}, \mathbf{x}') \mathbf{K}_y(\mathbf{y}, \mathbf{y}') = \mathbf{D} - \phi(\mathbf{X})^T \phi(\mathbf{X}) \phi(\mathbf{Y})^T \phi(\mathbf{Y})$ , and the minimization problem can be transformed into a maximization one:

$$\begin{aligned} \min_{\mathbf{Y}^T \mathbf{D} \mathbf{Y} = \mathbf{I}} \text{tr}(\mathbf{Y}^T (\mathbf{D} - \mathbf{Z}) \mathbf{Y}) &= \min_{\mathbf{Y}^T \mathbf{D} \mathbf{Y} = \mathbf{I}} \text{tr}(\mathbf{Y}^T \mathbf{D} \mathbf{Y} - \mathbf{Y}^T \mathbf{Z} \mathbf{Y}) \\ &= \max \text{tr}(\mathbf{Y}^T \mathbf{Z} \mathbf{Y}) = \max \text{tr}(\mathbf{Y}^T \mathbf{K}_x \mathbf{K}_y \mathbf{Y}) \end{aligned} \quad (7)$$

where the criteria to maximize turns out to be the covariance operator between the feature representations of each space.

Because the subspace  $\mathbf{Y}$  that we want to learn is a linear combination of the input parameters  $\mathbf{X}$ ,  $\mathbf{Y}$  can be expressed as  $\mathbf{Y} = \mathbf{B}^T \mathbf{X}$  and incorporating this information to Eq. (7) we obtained the next objective function

$$\max \text{tr}(\mathbf{B}^T \mathbf{X} \mathbf{K}_x \mathbf{K}_y \mathbf{X}^T \mathbf{B}) \quad (8)$$

which can be solved as an eigenvector problem,

$$\mathbf{X} \mathbf{K}_x \mathbf{K}_y \mathbf{X}^T \mathbf{B} = \lambda \mathbf{B}. \quad (9)$$

In case that the number of samples is large this eigenvector problem is intractable. Then, we can avoid solving the complete eigenproblem and solving a reduced rank one in two steps. Observe that Eq. (8) can be transformed to

$$\max \text{tr}(\mathbf{B}^T \mathbf{U} \Sigma \mathbf{V}^T \mathbf{K}_x \mathbf{K}_y \mathbf{V} \Sigma^T \mathbf{U}^T \mathbf{B}) \quad (10)$$

using the singular value decomposition  $\mathbf{X} = \mathbf{U} \Sigma \mathbf{V}^T$ , and letting be  $\mathbf{C} = \mathbf{V} \Sigma^T \mathbf{U}^T$  it allows to solve Eq. (10) as an eigenvector problem in  $\mathbf{C}$  (first step) and then calculate  $\mathbf{B}$  as  $\mathbf{B} = \mathbf{U} \Sigma^{-1} \mathbf{C}$  (second step). This eigenproblem in  $\mathbf{C}$  is done after the dimensionality is reduced selecting only the  $k$  first columns of  $\mathbf{V}$ .

#### 4. Experimental setup and results

In order to evaluate our approach in a command-line interface, we use a data set with the UNIX commands typed by 168 real users and labeled in 4 different groups. In this section we will detail the experimental results obtained and the comparison with other different classification methods.

#### 4.1. Data set

For these experiments, we use the command-line data collected by [Greenberg \(1988\)](#) using *UNIX csh* command interpreter. In this data, four target groups were identified, representing a total of 168 male and female users with a wide cross-section of computer experience and needs. Salient features of each group are described below, and the sample sizes (the number of people observed) are indicated in [Table 1](#).

- **Novice Programmers:** The users of this group had little or no previous exposure to programming, operating systems, or *UNIX*-like command-based interfaces. These users spent most of their time learning how to program and use the basic system facilities.
- **Experienced Programmers:** These group members were senior Computer Science undergraduates, expected to have a fair knowledge of the *UNIX* environment. These users used the system for coding, word processing, employing more advanced *UNIX* facilities to fulfill course requirements, and social and exploratory purposes.
- **Computer Scientist:** This group, graduates and researchers from the Department of Computer Science, had varying experience with *UNIX*, although all were experts with computers. Tasks performed were less predictable and more varied than other groups, research investigations, social communication, word-processing, maintaining databases, and so on.
- **Non-programmers:** Word-processing and document preparation was the dominant activity of the members of this group, made up of office staff and members of the Faculty of Environmental Design. Knowledge of *UNIX* was the minimum necessary to get the job done.

#### 4.2. Experimental design

In order to measure the performance of the proposed dimensionality reduction method using the above data, a 3-fold cross-validation is used. In addition, we need to define several parameters for creating the user profile process and the number of dimensions needed to reach a good performance.

The number of *UNIX* commands analyzed per user is very relevant for the classification results. Although we have performed several experiments with different number of *UNIX* commands per user, for this experiment we show the results obtained with 100 commands per user - these commands are selected from the last commands typed by a user. In addition, in the phase of behavior model creation, the length of the subsequences in which the original sequence is segmented (used for creating the *trie*) is a relevant parameter. Using a longer length, the time consumed for creating the *trie* and the number of relevant subsequences in the corresponding distribution increase drastically. In order to evaluate the relevance of this parameter, three different segmentation values for the sequence (subsequence lengths) are considered: 2, 4 and 6. Both the number of commands per user and the

**Table 1**  
Sample group sizes and command lines recorded.

Group of users name	Sample size	Total number of command lines
Novice Programmers	55	77423
Experienced Programmers	36	74906
Computer Scientists	52	125691
Non-Programmers	25	25608
Total	<b>168</b>	<b>303628</b>

subsequence lengths have been chosen taking into account the results in ([Iglesias et al., 2009](#)).

Regarding the features used for the classification we chose the projections  $\mathbf{P}$  of the original features over the new learned bases  $\mathbf{B}$ . Recall that the projected data  $\mathbf{P} = \mathbf{B}^T \mathbf{X}$  where  $\mathbf{B}$  is the solution obtained solving Eq. (9), and  $\mathbf{X}$  is the original data. We want to highlight that the number of the selected projected features is very relevant for the classification result. Therefore, in order to analyze what is the optimal number of dimensions to use we have performed several experiments using 3, 10, 50 and 100 projected features.

#### 4.3. Results

The creation of the user profile with 100 commands per user generates 21.762 different subsequences. The percentage of users correctly classified by different classification methods (C4.5, K-Nearest Neighbor, Naive Bayes and SVM) using all these subsequences as attributes are listed in [Table 2](#). The different classification methods are trained using a feature vector for each computer user (168 samples). This vector consists of the support value of all the different subsequences obtained for all the users. For this reason, there are subsequences which do not have a value because the corresponding user did not type those commands. In this case, the value considered is 0.

We have performed several experiments using the values 2, 4 and 6 for subsequence length. The results obtained using all the attributes for classifying a user using these different length values are listed in [Table 2](#). It is remarkable that in this case, without dimensionality reduction, we need to cope with huge amounts of data. In addition, its time consumption is very high. This aspect is a main drawback in the context of the proposed application because processing streaming data on-line and in real-time is usually needed as in [Nasoz and Lisetti \(2007\)](#).

These experiments have been repeated using different techniques for reducing the number of attributes. For comparison purposes we have chosen PCA and LPP. PCA has been chosen because it is the standard dimensionality reduction technique, and LPP because the proposed method SLMVP is based on it.

The results obtained with these techniques for the different subsequences lengths are shown in [Figs. 3–5](#). We observe that the results for each experiment are quite homogeneous. For instance, in [Table 3](#) we can see the best results for each technique for a subsequence of length 2 is about 75% (PCA 73.21%, LPP 75.59% and SLMVP 76.78%). It also occurs the same for a subsequence of length 4 (PCA 61.90%, LPP 69.64% and SLMVP 63.69%) with an accuracy about 65%, and for a subsequence length of 6 (PCA 68.45%, LPP 64.68% and SLMVP 70.23%) with an accuracy about 67%. Therefore, we can conclude that the best results for every technique are obtained with a subsequence of length 2. Then, in the next we will compare and explain in detail the results obtained for this best subsequence length although the same analysis could be extended to the other ones.

These results using a subsequence length of 2 for the three dimensionality reduction techniques are shown in the graph [Fig. 3](#). In each graph, the x-axis corresponds to a attribute-reduction size being 3, 10, 50 and 100 the number of selected features,

**Table 2**  
Classification results (%) using all the attributes.

Subsequence length	Attributes	Classification algorithm (%)			
		C4.5	K-NN	Naive Bayes	SVM
2	4.214	72.03	42.85	79.16	85.71
4	21.762	66.97	37.50	75.00	75.59
6	47.723	65.47	32.72	72.03	69.64

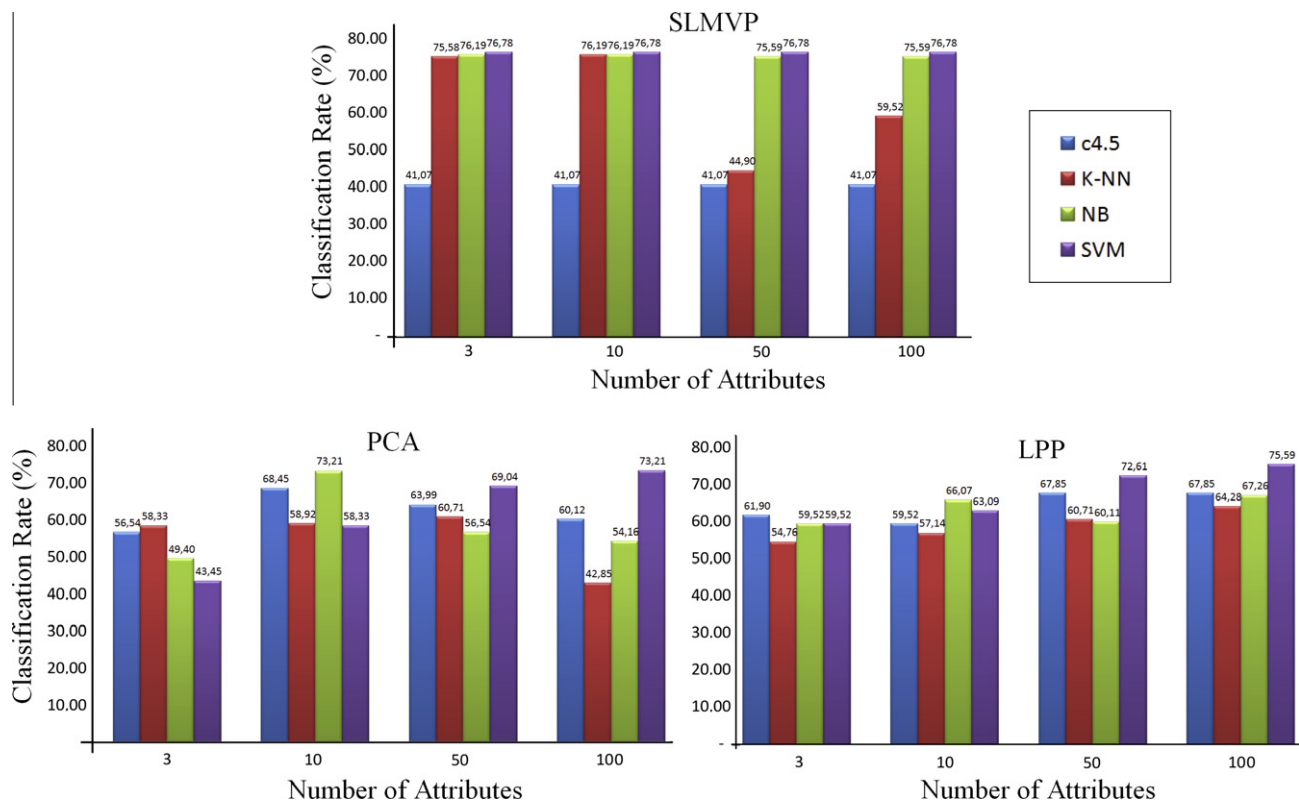


Fig. 3. Comparison of SLMVP, PCA and LPP using a subsequence length of 2.

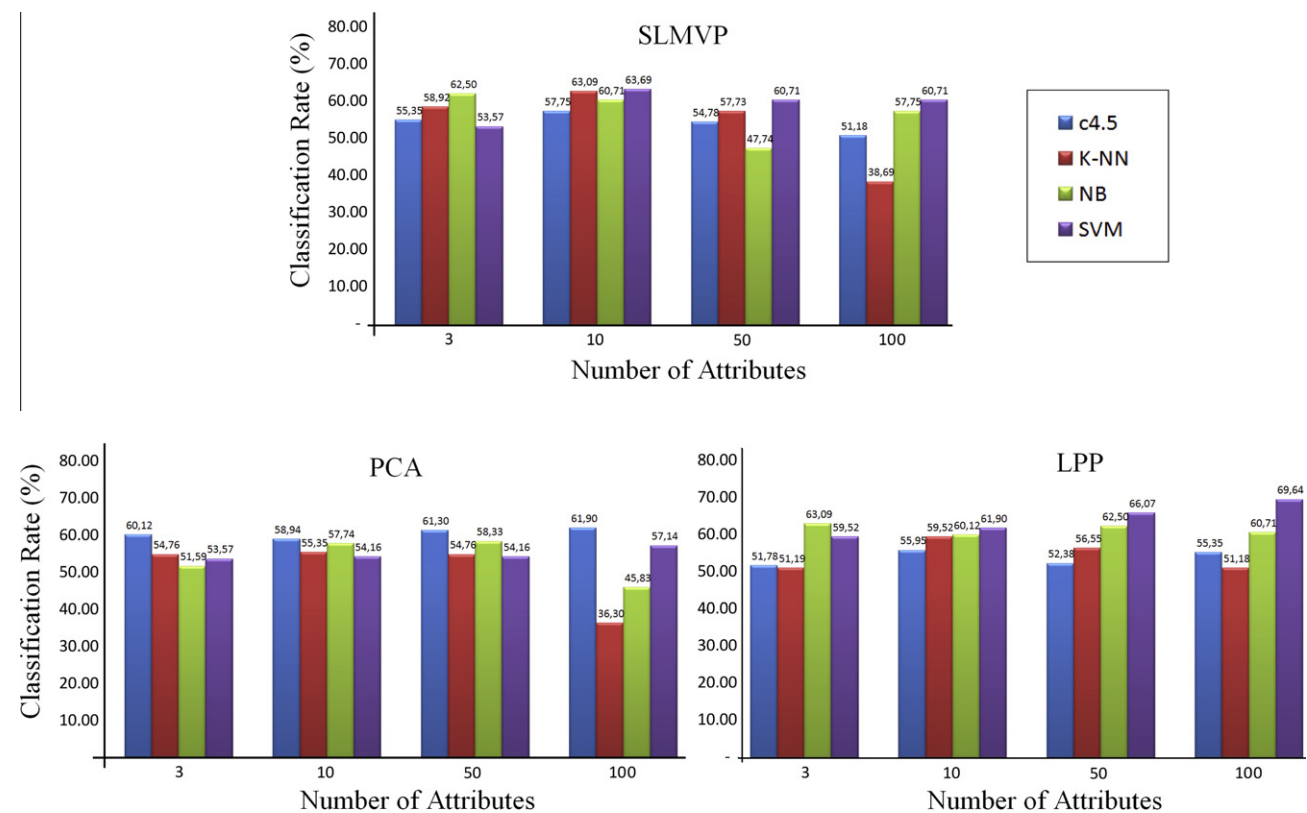


Fig. 4. Comparison of SLMVP, PCA and LPP using a subsequence length of 4.

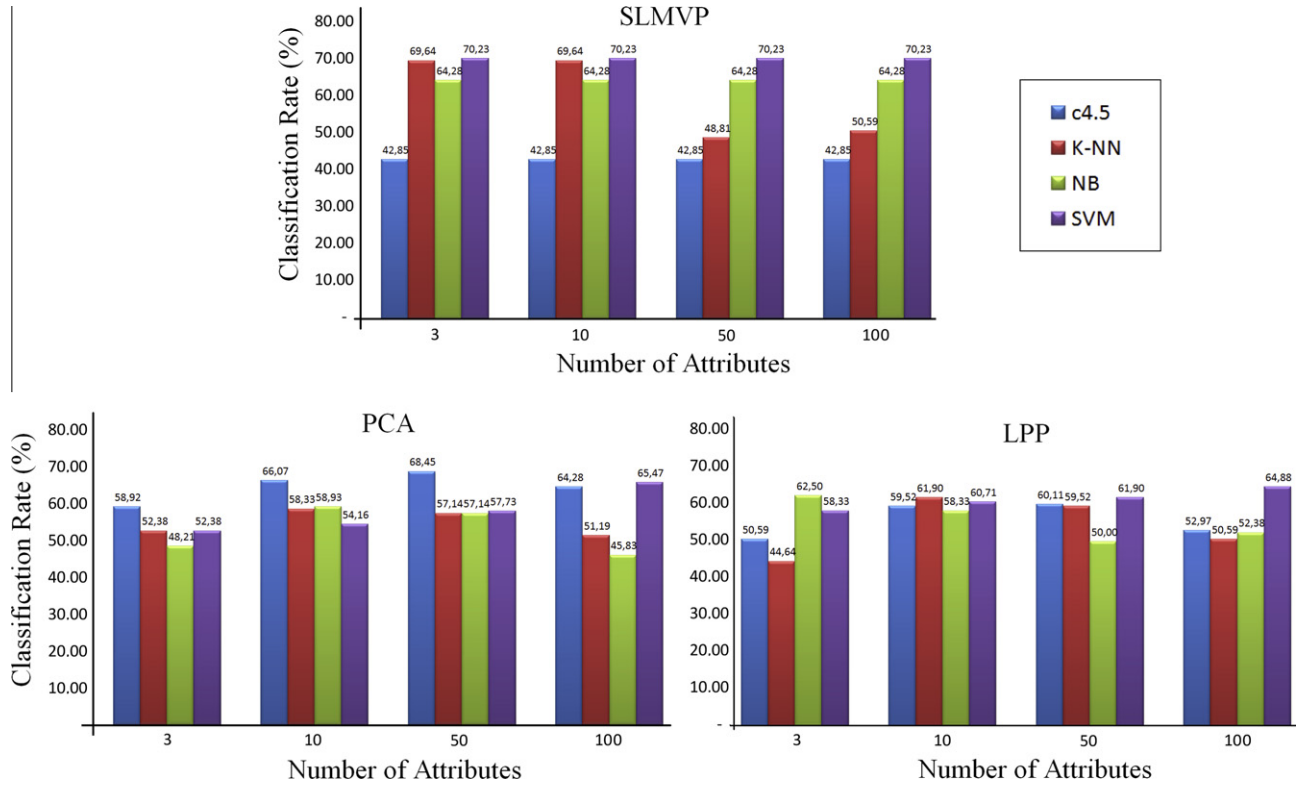


Fig. 5. Comparison of SLMVP, PCA and LPP using a subsequence length of 6.

**Table 3**  
Classification results (%) with dimensionality reduction – subsequence length of 2.

DR Technique	Attributes	Classification algorithm (%)			
		C4.5	K-NN	Naive Bayes	SVM
PCA	3	56.54	58.33	49.40	43.45
	10	68.45	58.92	73.21	58.33
	50	63.99	60.71	56.54	69.04
	100	60.12	42.85	54.16	<b>73.21</b>
LPP	3	61.90	54.76	59.52	59.52
	10	59.52	57.14	66.07	63.09
	50	67.85	60.71	60.11	72.61
	100	67.85	64.28	67.26	<b>75.59</b>
SLMVP	3	41.07	75.58	76.19	<b>76.78</b>
	10	41.07	76.19	76.19	76.78
	50	41.07	44.90	75.59	76.78
	100	41.07	59.52	75.59	76.78

and the y-axis shows the average classification success for four different classification algorithms: C4.5, K-Nearest Neighbor, Naive Bayes, and SVM.

The best results are obtained with the proposed reduction method SLMVP and using SVM as classifier. It gets a 76.78% of success versus a 73.21% of PCA + NB, or 75.59% of LPP + SVM. It makes sense to think that the difference (above 1%) between SLMVP + SVM and LPP + SVM is not significant but there are other terms which have to be taken into account. We want to point out that SLMVP + SVM accuracy is obtained using only 3 features which is a reduction of factor 7254, but for the other techniques the reduction is to 10 dimensions in the case of PCA + NB or to 100 for LPP + SVM. It is also important to highlight that the results obtained using the SLMVP reduction and KNN or NB are also very similar to the best one (SLMVP + SVM) with a 75.58% and 76.19% of success respectively. It is especially important in the case of

using the approach SLMVP + KNN because the complexity of the classifier is very low in comparison with the SVM. We can conclude that the approach of SLMVP + KNN using 3 features has similar results than LPP + SVM approach using 100 features, which means undoubtedly that our proposal is much more efficient in the dimensionality reduction process and classifying than the other approaches.

As can be seen in the graph Fig. 3 for the classifiers: NB, KNN and SVM, the results obtained using the reduced features with the proposed method improve the results obtained by PCA and LPP. The only exceptions are the results reported using the method C4.5. The main reason for this is that our method is supervised and therefore is able to obtain a better compression of the data by reducing more efficiently the number of dimensions needed to classify the samples. In this case the optimal number of features needed is 3. This is not a good property for C4.5 because it uses a binary tree to create different subgroups of data and then it applies different models to each one. Therefore, there are only four different leafs/models for the whole problem and is not taking advantage of the richer information of the features by combining them or using different range values to create more complicated models.

Also, using PCA tends to obtain the best results between 10–50 features while for our method it is between 3–10. This is also due as result of introducing information of the labels into the reduction process. Then, the relevant information for classification is compressed better into the first dimensions and therefore the classification performance improves for lower number of features.

According to the experiments, we can see that the percentages of users correctly classified by Naive Bayes and SVM using 21.762 attributes is higher than the results obtained after applying attribute-reduction methods. However, the number of attributes used in the classification algorithms is extremely lower using these (attribute-reduction) techniques. Thus, the difference in the percentage of users correctly classified can be assumed as a minor

drawback because this attribute reduction allows using simpler classification algorithms which is more efficient from a computational point of view and allows to tackle real time applications. Although the results obtained using the complete number of features gets better performance when it is used jointly with SVM (85%), we also want to point out the bad results obtained by a KNN classifier (42.85%) compare to our method (76.19%).

## 5. Conclusions and future work

In this paper we propose a generic approach to model and classify sequences of commands which represent a certain computer user profile. The underlying assumption in this approach is that the user information collected can be transformed into a sequence of commands. This sequence is segmented and stored in a *trie* and then the relevant sub-sequences are evaluated by using a frequency-based method. However, this process needs to cope with huge amounts of data.

For this reason, we propose the application of the maximum dependence criteria to find a lower dimension which represents better the data and improves the performance over the standard techniques of principal components analysis and the locally preserving projection. The proposed method is a novel dimensionality reduction approach based on maximum variance dependence between inputs and outputs in the kernel space.

The two most important contributions of this paper are the proposed method and its application to the user information retrieval environment.

The test results with a data set of 168 real UNIX users demonstrate that the dimensionality reduction is correctly accomplished by our method. It has the best results with between 3–10 features while the standard PCA has its best results with between 10–30 features, and the LPP between 50–100 which proofs the better reduction properties of our proposal.

Although is not addressed in this paper, the proposed method can be also applied to other different information processing, classification, or regression problems, in which the need to cope with huge amounts of data is an important challenge.

## References

Agrawal, R., Srikant, R. (1995). Mining sequential patterns. In *Proceedings of the Eleventh International Conference on Data Engineering* (pp. 3–14). Washington, DC, USA: IEEE Computer Society.

Anderson, J. (1995). *Learning and memory: An integrated approach*. New York: John Wiley and Sons.

Barrow, L., Markman, L., Rouse C.E. (2008). Technology's edge: The educational benefits of computer-aided instruction, Working Paper 14240, National Bureau of Economic Research.

Belkin, M., & Niyogi, P. (2002). *Advances in Neural Information Processing Systems*, 14.

Duda, R. O., Hart, P. E., & Stork, D. G. (2000). *Pattern classification*. Wiley-Interscience.

Fredkin, E. (1960). *Comm. A.C.M.*, 3, 490–499.

Frias-Martínez, E., Magoulas, G. D., Chen, S. Y., & Macredie, R. D. (2005). Modeling human behavior in user-adaptive systems: Recent advances using soft computing techniques. *Expert Systems with Applications*, 29, 320–329.

Fukunaga, K. (1990). *Introduction to statistical pattern recognition*. New York: Academic Press.

García-Cuesta, E., Galván, I. M., & de Castro, A. J. (2009). In *Proceedings of the Third International Workshop on Knowledge Discovery from Sensor Data, KDD 2009*. ACM Press.

C.L. Giles, M. Gori (Eds.), *Adaptive Processing of Sequences and Data Structures*, International Summer School on Neural Networks, “E.R. Caianiello, Vietri sul Mare, Salerno, Italy, September 6–13, 1997, Tutorial Lectures, volume 1387 of Lecture Notes in Computer Science, Springer, 1998.

S. Greenberg, *Using Unix: Collected traces of 168 users*, Master's thesis, Department of Computer Science, University of Calgary, Alberta, Canada, 1988.

Hackos, J. T., & Redish, J. C. (1998). *User and task analysis for interface design*. Wiley.

Han, K., & Veloso, M. (2000). Automated robot behavior recognition applied to robotic soccer. In J. Hollerbach, & D. Koditschek (Eds.), *Robotics research: the Ninth International Symposium* (pp. 199–204). London: Springer-Verlag.

He, X., & Niyogi, P. (2003). *Advances in neural information processing systems* (Vol. 16). MIT Press.

He, X., Yan, S., Hu, Y., Niyogi, P., & Jiang Zhang, H. (2005). Face recognition using laplacianfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27, 328–340.

Horman, Y., & Kaminka, G. A. (2007). Removing biases in unsupervised learning of sequential patterns. *Intelligent Data Analysis*, 11, 457–480.

Huang, Z., Yang, Y., Chen, X. (2003). An approach to plan recognition and retrieval for multi-agent systems. In M. Prokopenko (Ed.), *Workshop on Adaptability in Multi-Agent Systems, First RoboCup Australian Open 2003 (AORC2003)*. CSIRO.

Iglesias, J. A., Ledezma, A., & Sanchis, A. (2006). In *MDAI. LNCS* (Vol. 3885, pp. 117–128). Springer.

Iglesias, J. A., Ledezma, A., & Sanchis, A. (2007). In *IDA. LNCS* (Vol. 4723, pp. 207–218). Springer.

Iglesias, J. A., Ledezma, A., & Sanchis, A. (2009). In *UMAP 2009: Proceedings of the 1st and 7th international conference on user modeling, adaptation, and personalization. LNCS* (Vol. 5535, pp. 90–101). Springer.

I.T. Jolliffe, *Principal Component Analysis* (2nd Ed.), Springer Series in Statistics Springer-Verlag (Chap.8), New York, 2002.

Kaminka, G. A., Fidanboyu, M., Chang, A., & Veloso, M. M. (2002). In *RoboCup. Lecture Notes in Computer Science* (Vol. 2752, pp. 111–125). Springer.

Macedo, A. A., Truong, K. N., Camacho-Guerrero, J. A., & da GracCa Pimentel, M. . In *HYPERTEXT '03* (pp. 48–56). New York, NY, USA: ACM.

Nasoz, F., & Lisetti, C. L. (2007). In J. A. Jacko (Ed.), *HCI (3). Lecture Notes in Computer Science* (Vol. 4552, pp. 421–430). Springer.

Pepyne, D. L., Hu, J., & Gong, W. (2004). User profiling for computer security. In *Proceedings of the American control conference* (pp. 982–987).

Schölkopf, B. (2000). *Technical report MSR-TR-2000-51*. Microsoft Research.

Spiliopoulou, M., & Faulstich, L. C. (1998). In *Proceedings of EDBT workshop WebDB98* (pp. 109–115). Springer Verlag.

Sun, R., Merrill, E., & Peterson, T. (2001). From implicit skills to explicit knowledge: a bottom-up model of skill learning. *Cognitive Science*, 25, 203–244.

Tsai, F. S. (2011). Dimensionality reduction techniques for blog visualization. *Expert Systems with Applications*, 38, 2766–2773.

Turk, M., & Pentland, A. P. (1991). *IEEE Conference on Computer Vision and Pattern Recognition*.

Webb, G. I. (1993). Feature based modelling: A methodology for producing coherent, consistent, dynamically changing models of agents competency. In P. Brna, S. Ohlsson, & H. Pain (Eds.), *Proceedings of the 1993 world conference on artificial intelligence in education (AI-ED'93)* Edinburgh, Scotland. (pp. 497–504). Charlottesville, VA: AACE.

Webb, G. I., Pazzani, M. J., & Billsus, D. (2001). Machine learning for user modeling. In *User Modeling and User-Adapted Interaction 11* (pp. 19–20). Netherlands: Springer.