An approach to feature selection for keystroke dynamics systems based on PSO and feature weighting

Gabriel L. F. B. G. Azevedo, George D. C. Cavalcanti and E. C. B. Carvalho Filho
Center of Informatics
Federal University of Pernambuco
P.O. Box 7851, Cidade Universitaria
Recife – PE, Brazil
E-mail: {glfbga, gdcc, ecdbcf}@cin.ufpe.br

Abstract— **Techniques based on biometrics have been successfully applied to personal identification systems. One rather promising technique uses the keystroke dynamics of each user in order to recognize him/her. In the present study, we present the development of a hybrid system based on support vector machines and stochastic optimization techniques. The main objective is the analysis of these optimization algorithms for feature selection. We evaluate two optimization techniques for this task: genetic algorithms (GA) and particle swarm optimization (PSO). We use the standard GA and we created a PSO variation, where each particle is represented by a vector of probabilities that indicate the possibility of selecting a particular feature and directly affects the original values of the features. In the present study, PSO outperformed GA with regard to classification error, processing time and feature reduction rate.**

I. Introduction

Information security systems based on keystroke dynamics have been currently attracting considerable attention as a cheap solution that is more transparent to the user than fingerprint or iris recognition [1]. In this type of system, a user is identified by a login, password and keystroke pattern, the latter of which is a unique trait that is far more difficult to copy than the others and runs no risk of ever being lost. This system could be used in any setting where the user needs a password in order to gain access to particular information and/or functionalities, including Internet access [2].

Personal verification systems through some type of behavioral biometric characteristic, such as signature, voice or keystroke dynamic, are normally constructed from a set of examples that define the user. In the case of keystroke dynamic, it would be necessary for the user to type his/her name or password a number of times in order for the system to be able to extract the relevant features that uniquely represent the user. However, the task of typing one's name over and over is both tiring and tedious, which could lead users to alter their normal pattern. Thus, most systems based on biometrics are required to work with a summarized set of information with little information from which to extract knowledge. In order to reduce this problem, we could use subsets of data that better represent the classes responsible for identifying the different users of a system. Along with the reduction in the number of features that need to be learned (thereby increasing the accuracy of the system), we also diminish the processing time. One way to reduce this set of information would be to have a specialist delete the less representative data, but such a task is subjective and subject to errors [1]. It is expected that a system of this type should be executed automatically, where the user only has the task of typing his/her data repeatedly. The entire recognition process is accomplished by the system automatically. In the present work, we use a technique based on genetic algorithms for the feature selection researched in previous investigations, such as [1], and we propose a new technique based on PSO (Particle Swarm Optimization) [3], [4], [5], which is an evolutionary computation optimization technique based on the flocking behavior of birds. This new technique is a variation of that presented in [6].

In the next section, the SVM model is introduced and in the section III, o PSO algorithm is introduced. In Section IV of this article, we present the method used to extract features. Section V presents the methods used in the pre-processing of the data. Section VI describes the acquisition and division of the database. The experiments performed and their results are presented in Sections VII and VIII, respectively. Finally, in Section IX, we have the conclusions and suggestions for further studies.

II. Support Vector Machine

In recent work, such as [7], [8], SVM models have been proposed to solve novelty detection problems. The basic idea of this approach is the creation of a region that encloses the majority of data pertaining to a particular class, making it possible that some authentic patterns of the class are not contained within this region. The model would generate the smallest possible region that could enclosing the majority of data and would define a particular threshold that indicated to what extent a particular pattern could be outside this region and still be considered an authentic pattern. Further detailing the way in which the SVM would perform this task, the idea would be to map the input set of data in a space of high-dimensionality features corresponding to the kernel and then separate them from the origin with a maximum margin. The algorithm would find a function *f* that returned "+1" in the region it found that enclosing the majority of data contained

1-4244-1340-0/07$25.00 ©2007 IEEE

in the input set (normal data), and "-1" for data outside this region (novel data). After finding this function, any new pattern that was presented to this algorithm would have its value f(x) determined by the indication of which side of the hyperplane it was found in the feature space (returning +1 for patterns considered authentic or normal and -1 for novel patterns), as shown in Figure 1.

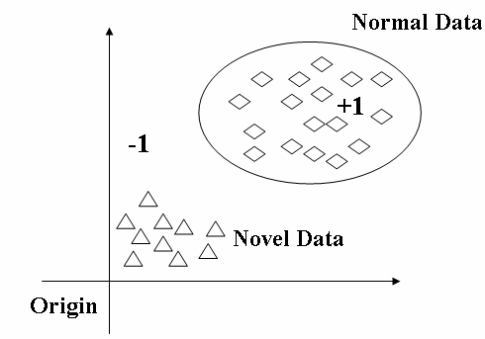

Figure 1 – SVM model for novelty detection.

The most used kernel function in novelty detection problems is Gaussian, given by Equation 1:

$$K(x, y) = \exp\left(-\left\|x - y\right\|\right)^2 / 2\sigma^2, \qquad (1)$$

where the parameter σ is responsible for the variation in the different estimates of regions in which the authentic data would be contained. It is important to point out that the Gaussian kernel ensures Mercer's condition, in which, for every value of σ, K(x,y) corresponds to the internal product between patterns x and y, after these patterns have been mapped for the set of features, as we can see in Equation 2.

$$K(x, y) = (\Phi(x) \cdot \Phi(y)). \qquad (2)$$

The regions found by the SVM algorithm are determined by a subset of data that form the training set and are found on the boundary between being within the regions formed by authentic data or being outside it and considered novelty. The data that compose this subset are called support vectors.

Without going into great detail, in order to separate authentic data from unauthentic data, the SVM algorithm attempts to optimize the following expression:

$$\min_{\alpha} \sum \alpha_i \alpha_j k(x_i, x_j), \qquad (3)$$

subject to the following constraints:

$$\sum_i \alpha_i = 1, \quad \forall i, 0 \le \alpha_i \le \frac{1}{vd}, \qquad (4)$$

where d represents the dimensionality of the data and v is a free parameter that acts as both the upper bound for patterns considered novel and the lower bound for the support vectors [8]. v can take on values between $1/n$ and 1, where n is the number of patterns. The α_i variables represent the Lagrange multipliers; there is one for each pattern contained in the original set. The task of the algorithm is to find the values of these coefficients. It is important to stress that only the Lagrange multipliers associated to the set of support vectors have values different from zero. When these multipliers are obtained, the decision function is defined as:

$$h(x) = \sum_{i \in SV} \alpha_i k(x_i, x) - \rho, \qquad (5)$$

where the sum is only valid for the multipliers associated to the support vectors. ρ is the threshold, which is calculated during training (for further details, see [8]).

III. Particle Swarm Optimization

PSO was introduced in the mid 1990s by [4] and is a stochastic search technique that aims to optimize an objective function. It was developed through attempts to graphically simulate the movement patterns of birds in search of food. Later, in the search for theoretical foundations, studies were carried out on the manner in which individuals in society generally interact, exchanging information and reviewing their concepts in the search for better solutions to their problems [5].

In PSO, a population (swarm) of solutions (particles) is maintained. Let s be the size of the swarm, n be the dimension of the problem and t be the current instant, each particle $1 \le i \le s$ has a position $x_i(t) \in R^n$ in the space of solutions and a velocity $v_i(t) \in R^n$ that governs the direction and range of its movement. Moreover, each particle has a memory $y_i(t) \in R^n$ of the best individual position visited until instant t, and the swarm has a memory of the best position it has visited so far.

As the algorithm runs, the velocity of each particle is calculated according to two main references: the best individual position visited $y_i(t)$ (cognitive term of optimization) and the best position visited by the swarm $\hat{y}(t)$ (social term of optimization). After calculating the velocity by Eq.(6), the position of the particle is updated by Eq. (7).

$$v_i(t+1) = \omega v_i(t) + c_1 r_1 (y_i(t) - x_i(t)) + c_2 r_2 (\hat{y}(t) - x_i(t)), \qquad (6)$$

$$x_i(t+1) = x_i(t) + v_i(t+1). \qquad (7)$$

Item w is the weight of inertia (term of momentum), which multiplies the velocity of the previous instant t and makes the search more explorative at the beginning and more exploitive at the end, as its value generally drops linearly from 0.9 to 0.4. The terms r1 and r2 are random variables taken from U1(0, 1) and U2(0, 1), respectively. Both possess the function of randomizing the influences of each term of expression (individual and global). The

coefficients of personal and global acceleration, 0 < c1, c2 ≤ 2, respectively, generally possess fixed, equal values and are responsible for controlling how far a particle will move in a single iteration.

The best position visited $y_i(t)$ of each particle is updated by Eq. (8), whereas the updating of the best position visited by the swarm is accomplished by Eq. (9).

$$y_i(t+1) = \begin{cases} y_i(t), & se\ f(x_i(t+1)) \geq f(y_i(t)) \\ x_i(t+1), & se\ f(x_i(t+1)) \langle\ f(y_i(t)) \end{cases}, \quad (8)$$

$$\hat{y}(t+1) = \arg\min\ f(y_i(t+1)), \quad 1 \leq i \leq s\,. \quad (9)$$

The calculated velocity $v_i(t + 1)$ in each cycle is generally limited to the interval [−vmax, vmax], with $v_{max} = k.x_{max}$, $0.1 \leq k \leq 1.0$, which does not ensure that the particle is always within the boundaries of the search space, but limits the maximum distance the particle can move in a single iteration. Figure 2 displays the complete algorithm of the PSO pattern. The characteristics frequently attributed to PSO are rapid convergence into unimodal functions, with a good success rate, and the premature convergence into multimodal functions, with a high correlation between components.

IV. Feature Extraction

To capture keystroke dynamic, it would be necessary for users to type their own name a number of times. The system would set about capturing these data using two methods regarding the time (in milliseconds) that a particular user maintains the key pressed and the time elapsed between releasing one key and pressing the next (latency time). Hence, we have, for example, a pattern that represents the user "ABCD", illustrated in Figure 3, formed by a size 7 timing vector. As we can see, the latency time in some cases can be negative if one key is pressed before the previous one is released. The first element of the vector is the length of time in which the first key remains pressed; the second element is the latency time between releasing the
first key and pressing the second; the third element is the length of time in which the second key remains pressed, and so on. In general, for a word of size n, a vector will be formed made up of features the size of 2n-1. In this case, we have n features that represent the lengths of time in which the keys remained pressed and n-1 features that represent the time elapsed between consecutive keys. In the present work, each pattern is formed by a vector with these lengths of time in milliseconds.

Create and initialize a swarm of particles
repeat
  for each particle *i* of swarm **do**
    if $f(x_i(t)) < f(y_i(t))$ **then**
      $y_i(t) = x_i(t)$;
    end if
    if $f(y_i(t)) < f(\hat{y}(t))$ the
      $\hat{y}(t) = y_i(t)$;
    end if
  end for
  Perform PSO updates of which particle using equations (6) and (7)
until stopping criterion is met

Figure 2 – PSO algorithm.

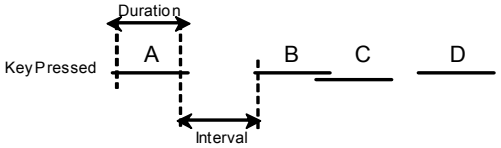

Figure 3 – Timing vector corresponding to a pattern "ABCD".

V. Pre-Processing of the Data

In the pre-processing step of the features, a clean-up of the data is performed in order to improve the performance of the system. In the present work, this step includes data normalization and feature selection. It is important to point out that these methods are performed by class in order to create a system that is customized to the user.

A. Data Normalization

In the present work, we use the sample standard deviation of the features to normalize the data. As we know, the standard deviation is calculated by the following expression:

$$\sigma = \left[\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2\right]^{1/2}, \quad where\ \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \quad (10)$$

Calculating this deviation for each of the features, we have $X_{ni} = X_i/\sigma$, where X_{ni} and X_i represent the i^{th} normalized feature and the original feature of a particular keystroke pattern, respectively.

B. Feature Selection

As mentioned above, most pattern recognition systems based on behavioral biometrics (online and offline signatures, keystroke dynamic, etc.) have few data with which to represent the classes and a considerable amount of features to learn. The idea of the selection task is to obtain a subset of features that allows better rates of correct identification (with increased system performance) than with the entire set of features. This subset is obtained

automatically. The user only has the work of presenting the training data to the system. The system then generates the feature selection model.

Feature selection is an optimization problem, since the aim is to obtain any subset that minimizes a particular measure (classification error, for instance). As we know, an optimization problem can be solved through stochastic algorithms. In [6], we can see the use of genetic algorithms (GA) and the particle swarm optimization (PSO) algorithm. In the present work, we suggest a variation of the PSO algorithm used in [6].

In [9], it was shown that the feature selection task can be classified into two categories: filter and wrapper. The filter approach does not depend upon the algorithm used for verification, whereas the wrapper approach does depend upon this algorithm. Therefore, the filter approach is far more efficient (in terms of processing time) than the wrapper approach, as there is no need to train a particular algorithm numerous times with various possible subsets of features. Selection through a filter simply discards irrelevant features through a given measure. In [10], for instance, the features that presented the largest coefficients of variance (CV = $\sigma/\bar{x}$) were discarded because they exhibited considerable variation between patterns, which tends to hinder the recognition task. In the wrapper approach, the idea is to generate several subsets of features, evaluate them through a given algorithm and choose the subset that represents the best solution. This approach achieves better results than the filter approach, but is only viable if the algorithm used to train the classifier is efficient.

In the present work, as with [6], we present a feature selection model that uses PSO as the optimization algorithm and SVM as the verification algorithm due to its known accuracy in pattern recognition problems [11]. The proposed model follows the structure displayed in Figure 4.

In [1] and [6], GA was used to optimize a vector with binary features, where 1 indicates the presence of the feature and 0 indicates its absence. Each particle of a PSO is formed by a vector of real numbers and the algorithm optimizes this vector. In [6], therefore, we decided to use this vector of real numbers as the likelihood of a feature being present in the subset, because these numbers represents the influence about the features. Thus, we normalize their values between 0 and 1. In [6], we also define a likelihood threshold that indicates that the feature is present in the subset if it has a value equal to or greater than this threshold and is absent otherwise. Thus, the final vector is formed by binary characters, as presented in [1] and [6].

In the present work, we do not use the original value of the features, as in [6]. If a particular feature has a probability represented by 0.8, the value used is 80% of its original value. In the first moment, we do not use a threshold to eliminate features. As the results were unsatisfactory (especially with regard to time), we eliminate the probabilities that were below a certain threshold, as done in [6]. The rest will have their influences affected by the probability, as described above.

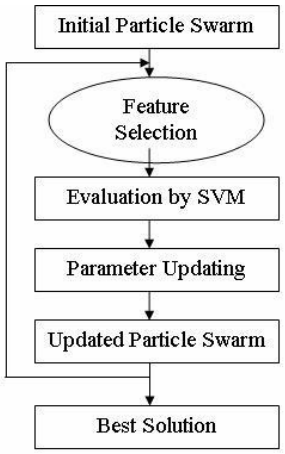

Figure 4 – Hybrid model proposed for feature selection through SVM and PSO.

VI. Database Acquisition and Division

The database used in the present work was the same used in [6] and [10]. The data were obtained through a software program developed in the Visual Basic language, which retrieves the necessary information that defines the keystroke dynamic of each user. The database used in this project is made up of information from 24 different users. Each user entered his/her complete name approximately 60 times. To obtain the data that represent frauds against the system, some users entered the complete names of other users approximately 5 times. The database generated here has approximately 60 authentic patterns and 40 false patterns for each of the classes.

In the present work, each user entered his/her complete name, which resulted in large vectors. For some classes, the input vectors had sizes greater than 70. The database used here was divided into 40 training patterns, 10 validation patterns and the remaining patterns were used for testing. As the database had a few classes containing less than 60 patterns (some with as little as 56 patterns, resulting in just 6 test patterns, which would have been of little significance), we opted to discard classes with less than 59 patterns in the data set. Thus, four classes were discarded, resulting in a total of 20 classes used.

VII. Experiments

As in [6], we used a basic GA and tested two possible PSO configurations. The verifier was the SVM as proposed in [7], [8]. After specifying the parameters of the verification and optimization algorithms, we performed 30 executions of the experiment, which included the presentation of the training data, feature extraction, pre-processing of the data (normalization and feature selection) and the verification step (through the test set). The system evaluation proposed

was performed using three measures: classification error, processing time in seconds and the reduction percentage of the set of features.

The classification error is presented by two measures: FAR (False Acceptance Rate), when the system accepts an imposter (or fraud) as authentic, and FRR (False Rejection Rate), when the system rejects an authentic user.

Another evaluation measure used in the present work regards the feature reduction capacity of the entry set. For example, if we have a pattern that initially has 60 features and after concluding the selection step, this pattern has 42 features, we have achieved a reduction of 30%.

As mentioned above, the verifier used in the present work was based on support vector machines. The kernel function used is Guassian function, the most commonly employed for pattern recognition work. The value of the parameter σ used in this equation was obtained empirically and is 0.015. The value of v was also obtained empirically and is also 0.015.

In the present work, the population size took on values of 20, 30, 40, 50, 60, 70, 80, 90 and 100 individuals (particles, in the case of PSO). This was done to determine at what population size the classification error stabilizes in both the experiments performed with GA and with PSO.

For the GA algorithm, each individual of the population was represented by binary vector. The crossover rate was 0.8 and the function used was the two-point crossover. The mutation rate was 0.1 and its function was uniform.

For the PSO algorithm, we used two configurations: one that considers the personal global acceleration coefficients, C_1 e C_2, to be 1.5 and another that considers them to be 2. Values were obtained empirically. For the inertia weight w (momentum term), we used the values used in most work with PSO, which drop linearly from 0.9 to 0.4 [3]. As with GAs, the PSO algorithm can be executed a maximum of 1000 iterations. The threshold values, which indicate the minimum likelihood that a feature must possess in order to make up part of the feature subset, were 0.5, 0.6, 0.7, 0.8 and 0.9.

VIII. Results

In this section, we present the results obtained from the experiments. Figure 5 displays the variation in the classification error as we increased the size of the population to be trained as well as the results from the genetic algorithms and PSO, obtained using the same configurations as in [6]. We can see that, for the genetic algorithms, the classification error does not diminish when the population goes from 90 to 100 individuals; there is stabilization in the error reduction curve, indicating that an increase in population size does not necessarily imply a reduction in classification error. The same occurs for the PSO algorithm with a 0.5 threshold. For the PSO algorithms with thresholds of 0.6 and 0.7, we can see that despite some drops in performance with the increase in population size, the classification error continues to fall until reaching population size 100, where it achieves the best results in both cases. As a population that contains 100 individuals is already quite large, we decided to set it as the maximum limit, whereas the minimum limit was set at 20 individuals. Figure 5 only displays the PSO performance with personal and global accelerations set at 1.5, as this was the configuration that obtained the best results. Moreover, PSO performance with personal and global accelerations set at 2.0 was similar.

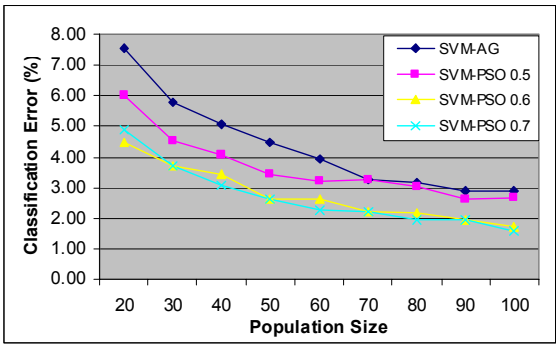

Figure 5 – Variation in classification error per population size with GA and PSO 1.5.

After defining the limits for the population size, we begin the actual experiments. Table 1 displays the results from a SVM verifier combined with genetic algorithms to carry out the task of selecting features. The best result achieved was for the population composed of 90 individuals, as we had seen in Figure 5. The classification error was 2.87%, with an FRR of 2.50% and an FAR of 0.37%. The lowest processing time of 1.93s was also for the population with 90 individuals. Table 1 shows that as population size increases, processing time diminishes until stabilizing at a population size of 90, as occurred with the classification error. There was also a considerable improvement in processing time, going from 7.54s to 1.93s and representing an improvement of around 74.4%. There was not much variation regarding the feature reduction rate and the best result achieved was 52.59%, which was slightly higher than the 52.54% achieved in the best GA configuration.

TABLE I
RESULTS OBTAINED FROM THE GA PATTERN.

Pop	FRR(%)		FAR(%)		class.error (%)		Time (s)		Reduction(%)	
	mean	s.dev	mean	s.dev	mean	s.dev	mean	s.dev	mean	s.dev
20	7,00	6,34	0,54	6,66	7,54	6,66	7,90	11,76	52,39	2,08
30	5,31	5,28	0,48	0,76	5,79	5,51	5,70	11,97	52,15	2,00
40	4,63	4,95	0,42	0,81	5,05	5,14	4,23	8,92	52,18	2,55
50	4,08	4,54	0,37	0,75	4,45	4,78	2,03	3,18	52,11	2,06
60	3,64	4,22	0,30	0,49	3,94	4,35	2,38	4,25	52,04	2,37
70	2,92	3,90	0,33	0,60	3,25	4,07	2,45	4,34	52,07	2,30
80	2,83	4,03	0,34	0,57	3,17	4,22	2,19	1,86	52,59	2,19
90	**2,50**	**3,67**	**0,37**	**0,71**	**2,87**	**3,91**	**1,93**	**0,63**	**52,54**	**2,40**
100	2,52	3,85	0,36	0,75	2,88	4,00	2,02	0,62	52,51	1,81

In the experiments using PSO for the feature selection task, we begin by executing the proposed model above any threshold. There is no elimination of features; they are all weighed. However, the results were rather weak, with substantial increases in the classification error and,

especially, processing time. Thus, these results are not presented here.

The second stage of experiments with PSO was carried out the same way as the first, but with the elimination of features when their influences were below a certain threshold. The thresholds tested were 0.5, 0.6, 0.7, 0.8 and 0.9. As we can see in Tables II, III, IV, V, VI and VII, the results obtained from threshold 0.8 and 0.9 are not displayed, as they too were quite inferior to the others.

Tables II, III and IV display the results obtained for PSO with personal and global accelerations of 1.5. The best result obtained for the 0.5 threshold (Table II) had 90 particles in the swarm and achieved a classification error of 2.62%, with an FRR of 2.10% and an FAR of 0.52%. Processing time was 1.66s, which was the second worst performance. As we can see, processing time increased with the increase in swarm size, which was different from the behavior presented by the GA. The feature reduction rate remained stable and its best result was 54.10% for swarm size 30.

Table III displays the results from the PSO 1.5 and threshold of 0.6. The best result was obtained with a swarm of 100 particles (the maximum limit used in the present work). The classification error was 1.74%, with an FRR of 1.14% and an FAR 0.60%. Processing time was 1.37s, which was the worst for this configuration. The feature reduction rate remained stable, with an average of 66.51%, ranging from 65.93% to 66.96%.

TABLE II
RESULTS OBTAINED WITH PSO 1.5 AND THRESHOLD 0.5.

Thr-Pop	FRR(%)		FAR(%)		class.error (%)		Time (s)		Reduction(%)	
	mean	s.dev	mean	s.dev	mean	s.dev	mean	s.dev	mean	s.dev
0.5-20	5,34	5,14	0,67	1,06	6,01	5,66	0,52	0,42	53,10	6,01
0.5-30	3,85	4,14	0,66	0,85	4,51	4,57	0,60	0,22	54,10	5,57
0.5-40	3,50	3,81	0,56	0,80	4,06	4,32	0,79	0,32	52,83	3,71
0.5-50	2,83	3,28	0,62	0,95	3,45	3,79	0,95	0,36	52,68	5,23
0.5-60	2,75	3,60	0,46	0,67	3,21	4,05	1,08	0,31	53,01	5,00
0.5-70	2,68	3,43	0,57	0,84	3,25	3,93	1,23	0,29	53,37	4,45
0.5-80	2,55	3,74	0,47	0,79	3,02	4,17	1,47	0,45	53,10	4,35
0.5-90	**2,10**	**2,81**	**0,52**	**0,78**	**2,62**	**3,25**	**1,66**	**0,61**	**52,37**	**4,32**
0.5-100	2,20	2,76	0,45	0,74	2,65	3,18	1,74	0,38	52,88	4,65

TABLE III
RESULTS OBTAINED WITH PSO 1.5 AND THRESHOLD 0.6.

Thr-Pop	FRR(%)		FAR(%)		class.error (%)		Time (s)		Reduction(%)	
	mean	s.dev	mean	s.dev	mean	s.dev	mean	s.dev	mean	s.dev
0.6-20	3,49	3,36	1,01	1,37	4,50	4,25	0,36	0,24	66,96	4,27
0.6-30	2,73	3,01	0,96	1,35	3,69	3,75	0,47	0,18	66,84	4,09
0.6-40	2,45	2,95	0,98	1,34	3,43	3,85	0,58	0,15	65,98	3,57
0.6-50	1,76	2,66	0,86	1,06	2,62	3,40	0,71	0,18	66,54	3,31
0.6-60	1,85	2,41	0,78	1,10	2,63	3,25	0,83	0,17	65,95	3,31
0.6-70	1,50	2,41	0,72	1,07	2,22	3,19	0,97	0,19	65,93	4,64
0.6-80	1,42	2,35	0,73	1,15	2,15	3,17	1,11	0,24	66,77	3,95
0.6-90	1,29	2,26	0,66	0,89	1,95	2,88	1,22	0,19	66,27	4,78
0.6-100	**1,14**	**1,75**	**0,60**	**0,86**	**1,74**	**2,43**	**1,37**	**0,21**	**66,51**	**4,33**

Table IV displays the results obtained from PSO 1.5 and a threshold of 0.7. The lowest classification error was 1.57%, with an FRR of 0.81% and a FAR of 0.76%, using a swarm with 100 particles. Processing time was 1.13s and reduction rate was 77.04%. From Tables II, III and IV, we can see that the classification error diminishes as the threshold is raised. This was also observed with regard to processing time and feature reduction rate. However, as mentioned earlier, when the threshold is above 0.7, the results worsen considerably. The greatest feature reduction rate achieved here was 77.44%, which is an enormous reduction of the original set. For example, if we originally had 60 features, the resulting set after applying the PSO algorithm would be 14 features.

TABLE IV
RESULTS OBTAINED WITH PSO 1.5 AND THRESHOLD 0.7.

Thr-Pop	FRR(%)		FAR(%)		class.error (%)		Time (s)		Reduction(%)	
	mean	s.dev	mean	s.dev	mean	s.dev	mean	s.dev	mean	s.dev
0.7-20	3,07	3,13	1,79	2,19	4,86	4,86	0,33	0,38	77,22	3,10
0.7-30	2,17	2,71	1,53	1,87	3,70	4,07	0,39	0,20	77,44	2,65
0.7-40	1,72	2,08	1,35	1,49	3,07	3,33	0,50	0,27	77,34	2,76
0.7-50	1,51	2,04	1,10	1,40	2,61	3,26	0,63	0,33	77,18	3,29
0.7-60	1,13	1,58	1,15	1,65	2,28	3,10	0,70	0,20	76,85	3,75
0.7-70	1,25	1,87	0,97	1,36	2,22	2,91	0,82	0,29	77,32	3,18
0.7-80	1,02	1,63	0,92	1,18	1,94	2,65	0,92	0,23	76,82	3,42
0.7-90	0,97	1,72	0,97	1,42	1,94	2,76	1,09	0,28	77,38	3,10
0.7-100	**0,81**	**1,42**	**0,76**	**1,11**	**1,57**	**2,26**	**1,13**	**0,16**	**77,04**	**3,36**

Tables V, VI and VII display the results obtained for PSO with personal and global acceleration of 2.0 and thresholds of 0.5, 0.6 and 0.7, respectively. In Table V, the lowest classification error was 2.52%, with an FRR of 2.01% and an FAR of 0.51%. This value was achieved for the swarm with 100 particles. The time obtained was 1.69s, which was the worst among the experiments with this configuration. The reduction rate was 54.93%, which was the best rate achieved with this configuration.

TABLE V
RESULTS OBTAINED WITH PSO 2.0 AND THRESHOLD 0.5.

Thr-Pop	FRR(%)		FAR(%)		class.error (%)		Time (s)		Reduction(%)	
	mean	s.dev	mean	s.dev	mean	s.dev	mean	s.dev	mean	s.dev
0.5-20	5,17	4,73	0,80	1,12	5,97	5,55	0,49	0,29	53,97	6,77
0.5-30	4,18	4,63	0,73	0,98	4,91	5,27	0,64	0,33	53,75	7,36
0.5-40	3,65	3,88	0,64	0,97	4,29	4,58	0,82	0,45	53,98	6,77
0.5-50	3,01	3,48	0,75	1,05	3,76	4,13	0,93	0,35	54,42	5,86
0.5-60	3,07	3,60	0,61	0,88	3,68	4,07	1,05	0,23	54,80	6,83
0.5-70	2,43	2,94	0,69	1,11	3,12	3,69	1,25	0,31	53,68	6,65
0.5-80	2,76	3,40	0,45	0,60	3,21	3,77	1,46	0,43	53,53	6,19
0.5-90	2,26	2,86	0,44	0,54	2,70	3,26	1,60	0,48	54,62	6,78
0.5-100	**2,01**	**2,78**	**0,51**	**0,77**	**2,52**	**3,34**	**1,69**	**0,33**	**54,93**	**6,10**

TABLE VI
RESULTS OBTAINED WITH PSO 2.0 AND THRESHOLD 0.6.

Thr-Pop	FRR(%)		FAR(%)		class.error (%)		Time (s)		Reduction(%)	
	mean	s.dev	mean	s.dev	mean	s.dev	mean	s.dev	mean	s.dev
0.6-20	3,72	3,53	1,27	1,44	4,99	4,47	0,34	0,20	67,34	4,87
0.6-30	2,91	3,21	1,17	1,32	4,08	4,32	0,48	0,25	66,86	5,96
0.6-40	1,98	2,58	1,29	1,47	3,27	3,74	0,61	0,28	67,94	5,10
0.6-50	2,11	2,49	0,79	0,99	2,90	3,31	0,72	0,20	67,23	5,57
0.6-60	1,63	2,38	0,82	1,08	2,45	3,03	0,87	0,29	68,22	5,49
0.6-70	1,49	2,26	0,70	0,93	2,19	2,95	0,97	0,19	67,77	5,11
0.6-80	1,40	2,20	0,59	0,77	1,99	2,89	1,08	0,18	67,15	5,84
0.6-90	1,53	2,19	0,61	0,85	2,14	2,85	1,24	0,26	67,43	4,57
0.6-100	**1,21**	**1,86**	**0,77**	**1,18**	**1,98**	**2,77**	**1,36**	**0,23**	**67,16**	**6,04**

In Table VI, the lowest classification error was 1.98%, with an FRR of 1.21% and an FAR of 0.77%. The feature reduction rate was 67.16% and processing time was 1.36s, which was, once again, the worst processing time for this configuration.

Finally, Table VII displays the results obtained with PSO

2.0 and a threshold of 0.7. The lowest classification error was 1.77%, with an FRR of 0.97% and an FRR of 0.80%. Processing time was 1.12% and the reduction rate was 77.59%. Once again, the lowest classification error, lowest processing time and greatest reduction rate were obtained with the threshold set at 0.7.

TABLE VII
RESULTS OBTAINED WITH PSO 2.0 AND THRESHOLD 0.7.

Thr-Pop	FRR(%)		FAR(%)		class.error (%)		Time (s)		Reduction(%)	
	mean	s.dev	mean	s.dev	mean	s.dev	mean	s.dev	mean	s.dev
0.7-20	2,78	2,75	2,24	2,26	5,02	4,56	0,29	0,25	78,17	3,86
0.7-30	1,95	2,32	1,57	2,06	3,52	4,08	0,38	0,20	78,14	3,81
0.7-40	1,74	2,39	1,31	1,42	3,05	3,36	0,47	0,18	78,46	3,38
0.7-50	1,79	2,51	1,25	1,44	3,04	3,67	0,57	0,19	78,03	3,62
0.7-60	1,38	1,75	1,30	1,65	2,68	3,13	0,67	0,20	77,57	3,79
0.7-70	1,23	1,86	1,08	1,18	2,31	2,75	0,79	0,16	78,07	4,70
0.7-80	1,03	1,72	0,87	1,12	1,90	2,57	0,90	0,14	78,23	3,97
0.7-90	0,96	1,60	1,13	1,70	2,09	2,93	0,99	0,16	77,89	4,36
0.7-100	**0,97**	**1,70**	**0,80**	**1,03**	**1,77**	**2,48**	**1,12**	**0,27**	**77,59**	**4,10**

Figures 6, 7 and 8 present comparisons of the performances obtained with the best GA, PSO 1.5 and PSO 2.0 configurations for the three performance measures used in the present work. Figure 6 is a comparative graph of the classification errors. As we can see, the results were quite similar; PSO 1.5 with a threshold of 0.7 was slightly superior to the others. GA obtained the worse result, presenting some point far above its average.

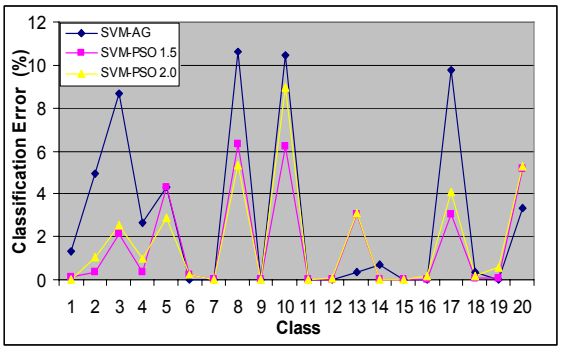

Figure 6 - Graph of classification errors obtained by GA, PSO 1.5 and PSO 2.0.

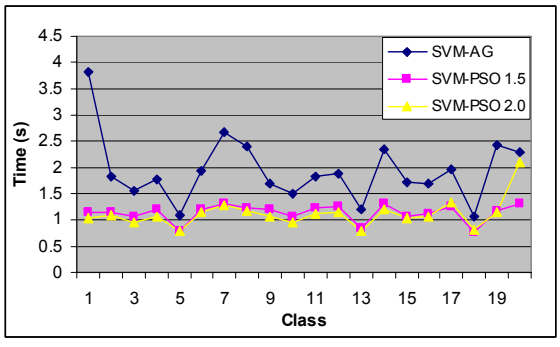

Figure 7 - Graph of processing times obtained by the GA, PSO 1.5 and PSO 2.0.

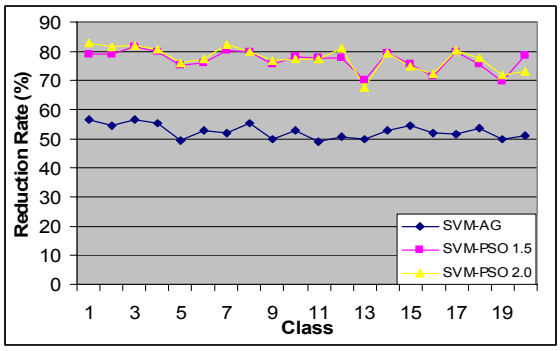

Figure 8 - Graph of feature reduction rates obtained by the GA, PSO 1.5 and PSO 2.0.

Figure 7 is a comparative graph regarding processing time. Contrary to what occurred in [6], the differences between the processing time with the GA and the two PSO configurations were quite small. Nevertheless, GA was unable to surpass PSO 1.5 or PSO 2.0 in any of the classes tested. We can also see that processing times achieved by PSO 1.5 and PSO 2.0 are practically equal.

Finally, Figure 8 is a comparative graph regarding the feature reduction rate. Contrary to what occurred in [6], GA had the worst performance in this measure, with a reduction rate of approximately 52.5%. PSO 1.5 and PSO 2.0 exhibited practically the same performance, in which the former obtained a rate of approximately 77% and the latter obtained a rate of 77.50%.

IX. CONCLUSION

From the previous section, we may conclude that the evolutionary PSO algorithm produced good results for the tasks of feature selection and personal identification. The algorithm presented here obtained a similar result to that in [6] with regard to classification error and a superior result with regard to the feature reduction rate. In [6], the lowest classification error obtained was 1.58%, with an FRR of 1.18% and an FAR of 0.41%; the feature reduction rate was 47.51% and the processing time was 0.69s. In the present work, the lowest classification error was 1.57%, with an FRR of 0.81% and an FAR of 0.76%; the feature reduction rate was far superior, achieving 77.04%, and the processing time was 1.13s. As we can see, the processing time here was practically twice that obtained in [6]. A negative point of the present work was having obtained a higher FAR; in most identification systems, it is more important to have a lower FAR than a lower FRR.

In the present work, the two PSO configurations tested were superior to the standard GA for all threshold values presented. The PSO with personal and global accelerations of 1.5 and a threshold of 0.7 was superior to the other configurations with regard to classification error, whereas the PSO with personal and global accelerations of 2.0 and a threshold of 0.7 was superior to GA with regard to classification error. Regarding the best GA, PSO 1.5 and

PSO 2.0 configurations, the lowest processing time was 1.13s, which was also achieved by PSO 1.5. The best feature reduction rate was 77.59%, which was obtained by PSO 2.0.

In future work, we intend to explore further the use of PSO as a feature selection algorithm, with the aim of better studying variations in the parameters and their influence on the results. We also intend to test techniques based on statistics, such as [10], and combine these with the method presented here and that presented in [6]. Yet another alternative would be to use auto-associative neural networks as a verifier, for, although [2] found these networks in conjunction with SVM to be quite slow for the feature selection task, we have seen that PSO exhibits a shorter processing time than GA.

ACKNOWLEDGMENT

The authors would like to acknowledge the financial support provided by the Brazilian National Research Council CNPq (Proc. 478534/2006-0).

REFERENCES

[1] E. Yu and S. Cho. Keystroke dynamics identity verification - its problems and practical solutions. Computers and Security, 23(5):428–440, 2004.

[2] S. Cho, C. Han and H. Kim. Web-based keystroke dynamics identity verification using neural network. J Organ Comput Electron Commerce, 4(10):295–307, 2000.

[3] F. Van Den Bergh and A. P. Engelbrecht. A Cooperative Approach to Particle Swarm Optimization. IEEE Transactions on Evolutionary Computation, Vol.8, No.3, June, 2004.

[4] J. Kennedy and R. Eberhart. Particle Swarm Optimization, in: Proc. IEEE Intl. Conf. on Neural Networks (Perth, Australia), IEEE Service Center,Piscataway, NJ, IV:1942-1948, 1995.

[5] J. Kennedy and R. Eberhart. Swarm Intelligence. Morgan Kaufmann Publishers, Inc, San Francisco, CA, 2001.

[6] G. L. F. B. G. Azevedo, G. D. C. Cavalcanti and E. C. B. Carvalho Filho. Hybrid Solution for the Feature Selection in Personal Identification Problems through Keystroke Dynamics (accept article). IEEE International Joint Conference on Neural Networks, 2007.

[7] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola and R.C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13, 1443-1471, 2001.

[8] B. Schölkopf, R. Williamson, A. Smola, J. Shawe-Taylor and J. Platt. Support vector method for novelty detection. In Advances in Neural Information Processing Systems 12, 2000.

[9] J. Yang and V. Honavar. Feature subset selection using a genetic algorithm. In: Motoda H, Liu H, editors. Feature extraction, in construction, and subset selection: a data mining perspective. New York: Kluwer; 1998.

[10] G. D. C. Cavalcanti, E. Pinheiro, E. C. B. Carvalho Filho. A System of Verification of Personal Identity through Keystroke Dynamics (in portuguese). In Proceeding of the V Brazilian National Artificial Intelligence Meeting, pages 791–800, 2005.

[11] H. Byun and S. Lee. Applications of support vector machines for pattern recognition: a survey. LNCS 2002; 2388:213-36.