Abstract—Computing and communication systems have been expanding and bringing a number of advancements to our way of life. However, this technological evolution has also contributed to the rise of the identity theft, mainly due to the advent of the digital identity. An alternative to overcome this problem is by the analysis of the user behavior, known as behavioral intrusion detection. Among the possible aspects to be analysed, this work focuses on the keystroke dynamics, which consists of recognizing users by their typing rhythm. This paper draws a comparison between some novelty detectors applied to keystroke dynamics: immune negative selection algorithms and auto-associative neural networks. Issues regarding the use of negative selection in high dimensional spaces are discussed and an alternative to deal with this problem is presented.

Keywords—keystroke dynamics; artificial immune systems; negative selection;

I. INTRODUCTION

It is clear that digital identities represent a key advancement in our society. However, the dissemination of these identities contributed for an increased data exposure and, consequently, for the identity theft [1]. Identity theft takes place when a person uses personal information of someone else as way to illegally pretend to be this person [2]. A promising alternative to curb this problem is by the use of behavioral intrusion detection systems [3], which detects anomalous behavior as potential intrusions.

Among the possible user aspects to be analysed, keystroke dynamics is studied here. This work shows the application of immune negative selection algorithms (NSAs) for recognizing users by their typing rhythm. These algorithms are novelty detectors, a class of classifiers that uses only samples from the positive class during the training phase. Afterwards, in the matching phase, these classifiers are able to differentiate between positive and negative data. As intruder samples are not always available, the approach of novelty detectors is more suitable for keystroke dynamics than binary classification, which requires positive and negative samples in the training phase. Novelty detectors are sometimes referred to as one-class classifiers [4].

A key issue when applying NSA is the lack of support for high-dimensional spaces [5], preventing its widespread use in some real-world problems. This paper proposes an alternative to overcome this issue by using cosine similarity. An auto-associative multilayer perceptron (AAMLP), a well-known novelty detector, is used as baseline to evaluate negative selection performance.

Throughout the paper, we present background information on keystroke dynamics and negative selection algorithms. In the end, we analyse the results obtained by the studied algorithms over a benchmark database. This work is organized as follows: in Section II, related work on keystroke dynamics is presented; Section III introduces negative selection algorithms and presents a NSA with high-dimensionality support for keystroke dynamics; Section IV details the experiments conducted here; Section V presents and discusses the results; and, finally, in Section VI, the conclusions are drawn.

II. KEYSTROKE DYNAMICS

Keystroke dynamics is considered to be a behavioral biometric technology and has several advantages over other technologies. Firstly, its implementation does not require any additional expenses with hardware, while other biometric technologies do (e.g. iris, fingerprint) [1]. Moreover, as the user does not need to perform actions specifically for the biometric system, the level of transparency of keystroke dynamics is enhanced, in contrast to a fingerprint or iris system, for instance, in which the user has to use a reader device. All these aspects contributes for an increased user acceptability when using this biometric technology [6].

The area of keystroke dynamics has been studied for more than 30 years and a number of works are available in the literature. One of the first works in the area is from 1980 [7]. Table I shows some of the researches carried out in keystroke dynamics. In this table, the number of users that took part in the experiments and the best performance reported is specified. This table is based on an adapted systematic review on keystroke dynamics we conducted [8]. There are two main forms of reporting results in keystroke dynamics:

- FAR and FRR: FAR (False Acceptance Rate) indicates the rate in which an intruder is misclassified as being a legitimate user and FRR (False Rejection Rate) indicates the rate in which a legitimate user is wrongly rejected by the system [6]. Usually, there is a trade off between FAR and FRR, so that when FAR increases, FRR tends to decrease and vice-versa.
- EER: EER (Equal Error Rate) represents the value when both FAR and FRR are equal [9].
Table I: Performance achieved by keystroke dynamics in previous works (adapted from [8]).

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Users</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gunetti and Picardi [10]</td>
<td>205</td>
<td>13% (ERR)</td>
</tr>
<tr>
<td>SVM [11]</td>
<td>100</td>
<td>6.95% (EER)</td>
</tr>
<tr>
<td>Nearest neighbour [12]</td>
<td>51</td>
<td>9.96% (EER)</td>
</tr>
<tr>
<td>Hidden Markov Model [13]</td>
<td>20</td>
<td>3.6% (EER)</td>
</tr>
<tr>
<td>Bleha (with equalization) [14]</td>
<td>47</td>
<td>6.2% (EER)</td>
</tr>
<tr>
<td>Manhattan Distance [15]</td>
<td>51</td>
<td>7.1% (EER)</td>
</tr>
<tr>
<td>Random Forests [16]</td>
<td>53</td>
<td>1% (FAR), 14% (FRR)</td>
</tr>
<tr>
<td>Tree-based [17]</td>
<td>12</td>
<td>0% (FAR), 3.47% (FRR)</td>
</tr>
<tr>
<td>R Measure [18]</td>
<td>205</td>
<td>0.005% (FAR), 5% (FRR)</td>
</tr>
<tr>
<td>AAML [4]</td>
<td>21</td>
<td>0% (FAR), 0.25% (FRR)</td>
</tr>
</tbody>
</table>

As it can be observed, the use of different performance measures make it difficult to compare previous works in the area. In addition to the works presented in Table I, we also highlight other researches which use immune algorithms to recognize users by keystroke dynamics: [19] and [20]. The first used a positive selection algorithm in a database of only five users, while the second applied a negative selection algorithm with a modified distance measure in a database of 20 users. Both works performed specific analysis and did not report their results in terms of FAR, FRR or EER.

III. NEGATIVE SELECTION ALGORITHMS

Negative selection algorithms are inspired by the self/non-self discrimination, a fundamental task of the biological immune system. This task is performed by the T-cells, which are cells with receptors in their surface. During the generation and maturation process of these cells, they undergo a process known as negative selection, in which T-cells whose receptors react with cells from the organism (self) are destroyed and, therefore, not released to the body. In computing, NSAs are usually applied to anomaly detection [21]. As shown by a recent intrusion detection review [5], NSAs are the immune algorithms most studied in this area.

The first NSA was proposed by Forrest in 1994 [22]. As shown in Figure 1, this algorithm has two phases: censoring (training) and detection (matching). In the first phase, detectors are generated randomly and each of them is tested against the training data. Those that match any self sample are rejected. This process is executed until a predefined number of valid detectors are generated. In the detection phase, each input sample received is checked by the detector set. If at least one detector matches it, the sample is classified as non-self and, otherwise, as self. In the context of keystroke dynamics, self represent legitimate users while non-self are the intruders.

In original NSA, all data are represented by a string of bits. Later on, these algorithms were adapted to support real-valued vectors [21]. In keystroke dynamics, extracted data is a vector of real values, making it more suitable to adopt the real-valued representation. Next subsection provides more details on two key variations based on real-valued representation: Constant sized detectors and V-Detector.

Figure 1: Negative Selection Algorithm.

A. Constant sized Detectors and V-Detector

In NSA with real-valued vectors, data are represented in the $d$-dimensional space $[0; 1]^d$. Detectors are $d$-dimensional vectors with coverage radius $r$, defining a hyper-sphere. A sample is matched by the detector if the Euclidean distance between the centre of the detector and the sample is equal or less than the radius $r$ of the detector [23].

One of the ways to generate detectors for a real-valued NSA is by defining a constant radius and generating random vectors as centers of these detectors. If the center of the random detector is not in the self space, this detector is added to the detectors set and, otherwise, it is rejected. This process of detector generation is performed until a predefined number of detectors is obtained. This approach is known as real-valued negative selection algorithm with constant sized detectors [24] and will be named CRNS in this paper.

Different versions of real-valued negative selection were proposed [5], but one that has showed promising results is the V-Detector [24]. Instead of using fixed values for the number of detectors and a fixed radius in all detectors, as CRNS does, V-Detector generates detectors adapted to each case, defining how many detectors are required and the radius of each detector in order to maximize coverage of non-self space. V-Detector has the advantage of usually generating less detectors and, as a consequence, decreasing the time needed for matching in the detection phase.

B. Scalability and High-dimensionality Problems

Although negative selection algorithms principles show to be suitable for anomaly detection, an issue prevents their widespread in the area: support for high-dimensional spaces [5]. Several studies have been conducted with negative selection and its variants showing detection rates higher than 90% in a number of experiments [24]. However, these studies used low dimensional spaces (up to five features). Earlier
works tested negative selection in higher dimensional spaces (with more than five features) and showed that they reached very low performances in such scenarios, with detection rates lower than 3% [23].

One factor that may contribute to this performance degradation in high-dimensional spaces is the use of Euclidean distance to compute distances in the negative selection algorithm. Euclidean distance is commonly used in low dimensional spaces, but it does not scale well in high dimensional scenarios [25]. For keystroke dynamics, this is a fundamental problem, as feature vectors usually have several attributes, implying in a high dimensional space.

C. Negative Selection with High-dimensional Support

In view of the mentioned high-dimensional problem, we propose the use of cosine similarity [25] instead of the Euclidean distance in NSA as way to overcome dimensionality problems in the recognition of users by keystroke dynamics. Cosine similarity is calculated according to (1), where \( \vec{x} \) and \( \vec{y} \) are real-valued vectors with the same dimensionality \( d \).

\[
sim(\vec{x}, \vec{y}) = \frac{\sum_{i=1}^{d} x_i y_i}{\sqrt{\sum_{i=1}^{d} x_i^2 \sum_{i=1}^{d} y_i^2}}
\]

This similarity is widely used in data mining algorithms as a way to address problems with Euclidean distance in spaces with high dimensions [25]. Another work used cosine similarity in negative selection, but for a different application and with the purpose of making the algorithm insensitive to the absolute amplitude of time series [26].

Cosine similarity values ranges from -1 (complete opposite) to 1 (exactly the same). As negative selection algorithms are based on distance measures, that is, how far a element is from another, the value calculated from the similarity must be modified to represent a distance. In this paper we adopted the calculation shown in (2), which is the same used by [26].

\[
dist(\vec{x}, \vec{y}) = 1 - \sim(\vec{x}, \vec{y})
\]

CRNS and V-Detector with the cosine similarity will be respectively named here as CRNS-C and V-Detector-C.

IV. EXPERIMENTAL SETUP

This section describes the test setup in terms of benchmark database, extracted features and parameters.

A. Benchmark database

The performance of the algorithms discussed here are assessed using the GREYC benchmark database from [27]. This database has data from 133 users, which typed the expression “greyc laboratory”. More than 7000 samples are available in this database and they were captured during a period of two months. In [11], only the users with at least 60 samples were used, resulting in 100 users. The same approach is adopted here.

B. Extracted Features

Previous works in keystroke dynamics extracted a number of different features from the typing data. However, the feature flight time is the single feature most used [8] and is, therefore, adopted in our experiments. Figure 2 shows this feature graphically, which is the time difference between the instants when a key is released and the next key is pressed. The use of this feature in the selected benchmark database results in a feature vector of dimension 15 for each sample.

![Figure 2: Features extracted from the keystroke timing data.](image)

During our experiments, the first samples are used for training, leaving the remaining ones for testing (matching). In the case of NSAs, these first samples are used to generate the detectors that are used for classifying the remaining data. Detector sets are generated per user. The approach of using the first samples for training was adopted here as this make the test closer to what would happen in practice. In a real-world application, users would start typing in a way and, through the time, might slightly change their typing rhythm, mainly due to learning. This is also known as concept drift in Machine Learning [28]. Therefore, in this case, the use of other techniques for defining training and testing sets, such as cross-validation, would result in inaccuracies.

C. Parameters

In this paper, five novelty detectors were tested: AAML, CRNS-C, CRNS, V-Detector-C and V-Detector, where NSAs with C at the end are their versions with cosine similarity. The values of the parameters adopted for each of the algorithms are shown in Table II.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Self radius</td>
<td>0.31 (Cosine) / 0.01 (Euclidean)</td>
</tr>
<tr>
<td>Maximum Number of Detectors</td>
<td>50</td>
</tr>
<tr>
<td>Detector radius (CRNS)</td>
<td>0.30 (Cosine) / 1.0 (Euclidean)</td>
</tr>
<tr>
<td>Estimated Coverage (V-Detector)</td>
<td>99%</td>
</tr>
<tr>
<td>Epochs (AAML)</td>
<td>500</td>
</tr>
<tr>
<td>Learning rate (AAML)</td>
<td>0.30</td>
</tr>
<tr>
<td>Alfa Momentum (AAML)</td>
<td>0.15</td>
</tr>
<tr>
<td>Threshold (AAML)</td>
<td>0.10</td>
</tr>
</tbody>
</table>

All parameters were defined in order to reach best accuracy rates in the experiments, while attempting to keep parameters that are common among the algorithms with the same values. This is illustrated by the maximum number of detectors, that assumed the same values for all negative selection algorithms. Our aim is to verify whether NSAs can achieve similar or better predictive performance when
compared to AAMLP, a known novelty detector from the literature. In order to define the AAMLP structure, we evaluated suggestions from a previous work that applied this classifier on keystroke dynamics [4]. The adopted structure uses 8 neurons in the hidden layer and 15 output neurons, which was the design with best accuracy rate here.

V. RESULTS

This section presents the results attained in the experiments conducted here.

A. Number of training samples

The performance of the novelty detectors are assessed here considering a static profile scenario, in which the user profile is established in a training phase and, after that, during the matching phase, it is not changed any more. Static profiles have two major advantages. Firstly, they demand less use of processing resources, as a static profile does not need to be updated several times. Secondly, they are not susceptible to incorporate behavior from intruders by small changes in the user behavior, in contrast to a dynamic profile, which is susceptible to this attack.

In the training phase, an important parameter is the required number of samples for the profile definition. In order to measure how this parameter would impact the performance, we conducted a series of experiments, ranging the number of training samples from 5 to 55. Due to the stochastic nature of the applied algorithms, we performed 30 executions for each case. Hence, the FAR and FRR results reported in the graph of Figure 3 are the average from 30 executions with all users for each of the algorithms. Standard deviation values ranged from 0.000 to 0.022 in all cases.

Both NSAs using Euclidean distance achieved a very low performance, with FAR higher than 50% in almost all tests. Due to that, their results are omitted from Figure 3 and these algorithms will not be further considered in our analysis.

As shown in Figure 3, for all algorithms, an increase in the number of training samples implied in a decrease of the FRR along with a slight increase of the FAR. User samples are less similar between them in the first captures but have a tendency of becoming more similar in the following captures, mainly due to user learning. As a result, when more samples are used for training, more variants of the user rhythm are trained as legitimate. This may lead to the increased FAR observed here.

AAMLP attained lower FAR in almost all cases, nevertheless, it also reached the worst FRR values. Better values of FRR were achieved by V-Detector-C and CRNS-C, which attained similar FRR performance, as shown by the closeness of the two dashed lines of these algorithms. However, in terms of accuracy, for almost all training configurations, CRNS-C reached best results.

B. Negative Selection x AAMLP

For comparing more directly the algorithms, we did a further performance comparison for 55 training samples, which was when all algorithms reached their best performances: CRNS-C x AAMLP (Figure 4a), CRNS-C x V-Detector-C (Figure 4b) and V-Detector-C x AAMLP (Figure 4c). In these graphs, the values plotted are the average accuracy rates for each of the 100 users in the database.

In Figure 4a, for example, $x$ axis is the accuracy rate for AAMLP and $y$ axis is the accuracy rate for CRNS-C. The graph is divided by a dashed line into two areas, one for each algorithm being compared. The algorithm with greater number of points in its area is the one with higher performance, as the graph is based on accuracy rates. Points close to the division line represent users that were classified with similar accuracy by both algorithms. In all cases, the closer to (1;1), the better is the performance.

By a visual analysis of the graphs in Figure 4, it is clear that both AAMLP and CRNS-C outperform V-Detector-C. According to our tests, in terms of accuracy, CRNS-C outperformed all algorithms for 15 training samples or more and this can also be observed in these graphs.

Apart from the analysis of the graphs in figures 3 and 4, a Friedman Test [29] was applied for comparing the performance of the algorithms. This analysis was done for each of the performance measures: FAR, FRR and accuracy. We found that there are significant differences among the performances for these measures. As the Friedman Test identified these differences, we performed the Nemenyi post-hoc test [29] to determine which algorithms caused the differences and we concluded that all algorithms were significantly different from each other. We considered $\alpha = 0.05$ in all statistical tests. Considering accuracy rates, CRNS-C reached best results, with an average performance 2.1% higher than AAMLP, which was the second algorithm in terms of accuracy.
The combined results show that NSAs are suitable for keystroke dynamics in a novelty detection scenario, achieving similar or superior predictive performance when compared to AAML. Moreover, the adoption of the cosine similarity allowed the NSA to obtain good results in the high-dimensional problem considered here, as opposed to the use of the standard Euclidean distance.

**C. Processing Time**

In addition to the evaluations in the last sections, we also compared the processing times of all algorithms. All tests were performed in a computer with Intel Core i7 3.40 Ghz processor. Figure 5 shows the box plots of training and matching times for all algorithms. From these graphs, it is clear that V-Detector-C spent more time in training and had a higher dispersion than other algorithms. Nonetheless, V-Detector-C was the fastest algorithm for matching. This result may be explained by the V-Detector-C design, which generates a reduced number of detectors. Still in terms of matching, AAML was the algorithm that spent more time for this task. Considering a balance between training and matching time, CRNS-C has shown superior results, as it was the fastest for training and achieved a matching time better than AAML and closer to V-Detector-C.

**VI. CONCLUSION**

In view of the need to curb identity theft, behavioral intrusion detection systems are a promising alternative. One of the aspects that can be analysed from the user behavior on the computer is the keystroke dynamics, which was tested here in a benchmark database. Five novelty detectors were used: one neural network and four negative selection algorithms. Due to the need to deal with a high-dimensional space in keystroke dynamics, we proposed the use of the cosine similarity in NSA, instead of the commonly used Euclidean distance. This proposal improved the algorithms performance considerably.

In terms of computer resources, immune-based negative selection algorithms performed better than the neural network. Moreover, immune CRNS-C attained best accuracy performance for most of the cases. V-Detector-C reached better FRR among the tested approaches, meaning it improves usability of the system as it decreases the false rejection of users, while it also provides security. However, in terms of FAR, AAML achieved best results. Consequently, this algorithm is more suitable for scenarios in which security is of paramount importance and false acceptance ought to be minimal. In summary, the choice of the algorithm should take into account the aim of the biometric system.

Several differences between previous works in keystroke dynamics, such as performance measures, database and configuration for training and testing, prevents us from drawing a direct comparison with the results reported here. For example, the majority of the works presented in Table I used far less users than the database adopted here. We must highlight that the results reported here depend on the benchmark database, sampling methods and normalization used. In future work, additional benchmark databases may be tested, although there are not many databases available in the area [27]. Moreover, we intend to apply other classifiers over the same database and with similar parameters in order to establish a more reliable comparison with previous works.
ACKNOWLEDGMENTS

The authors would like to thank FAPESP and CNPq for financial support. In addition, we would like to thank Professor Christophe Rosenberger for kindly providing us with the GREYC benchmark database for keystroke dynamics.

REFERENCES


