Keystroke Dynamics Aplicado a la Clasificación de Intrusos

Gissel Zamonsky Pedernera
Universidad FASTA,
Mar del Plata, Argentina, 7600
gissel@gmail.com

Sebastian Sznur
Universidad FASTA,
Mar del Plata, Argentina, 7600
sz.sebas@gmail.com

Sebastián García
Universidad FASTA,
Mar del Plata, Argentina, 7600
eldraco@gmail.com

y

Gustavo Javier Meschino
Universidad FASTA, Universidad Nacional de Mar del Plata
Mar del Plata, Argentina, 7600
gustavo.meschino@gmail.com

Resumen

La dinámica de tipeo ha demostrado ser una técnica biométrica de comportamiento eficiente como parte de las técnicas de detección de intrusos. En este trabajo se busca elaborar algoritmos para clasificar intrusos por medio del análisis de sus patrones de tipeo a la hora de realizar acciones dentro de un servidor. Para la selección de características de los patrones se recurrió a algoritmos evolutivos utilizando los enfoques de filtrado (*filter*) y de envoltorio (*wrapper*). Se cuenta con datos provenientes de la captura de teclas presionadas por intrusos reales en diversos *honeypots*. Las principales características de este estudio consisten en la caracterización y reconocimiento de intrusos reales y no de usuarios, considerándose la detección de las personas y no las acciones. Se muestran los resultados obtenidos para la clasificación experimentando con redes autoorganizadas basadas en la Teoría de la Resonancia Adaptativa (ART) y mapas autoorganizados de Kohonen (SOM), y el método estadístico *k-means*. Se obtuvo un modelo que se desempeñó satisfactoriamente.

Palabras claves: Dinámica de tipeo, Seguridad, Algoritmos evolutivos, Intrusos, Clasificación.

1. Introducción

Biometría es un término utilizado para indicar un conjunto de características que se cree que son únicas tanto de la fisiología cómo del comportamiento de un individuo, y por esa razón, difíciles de duplicar.

Keystroke dynamics es una rama de la biometría que se dedica al estudio del reconocimiento del patrón de tecleo de una persona. El objetivo del presente trabajo es desarrollar y evaluar el desempeño de diversos métodos de clasificación de intrusos por medio de *keystroke dynamics* empleando técnicas específicas de inteligencia computacional y estadística.

Se parte de la suposición de que el comportamiento de una persona se modifica al estar atacando un servidor ajeno. La intencionalidad del ataque determina la posibilidad de poder perfilar a dicha persona como un intruso. Así, forma parte del objetivo identificar y clasificar intrusos y no usuarios. Para atraer a los intrusos se utilizaron sistemas denominados *honeypots* que son vulnerables a los ataques. Dichos sistemas no tienen objetivos de producción, con lo cual, todo lo que se haga en ellos es considerado un ataque y toda persona que realice dichos ataques es, debido a la intencionalidad, un intruso. En el *honeypot* se registran todas las teclas que se ejecutan en la máquina. Junto con cada tecla también se almacena la fecha, la hora exacta con una precisión de millonésimas de segundo, la terminal desde la cual se ejecutó y un código identificador del usuario.

Al no contar con datos preclasificados, no se pudieron utilizar métodos supervisados. Se exploraron exhaustivamente varios métodos con el fin de determinar el más adecuado para el fin planteado.

2. Estado del Arte

2.1. *Keystroke Dynamics*

Este trabajo presenta una manera novel de clasificar a los intrusos mediante *Keystroke Dynamics*. La mayoría de los trabajos hallados en la bibliografía parten de la premisa de que lo que se van a clasificar son usuarios.

En líneas generales las investigaciones han probado que cada persona tiene su patrón de *Keystroke Dynamics* y que en mayor o menor medida éste puede ser utilizado para diferenciarlas.

Algunas investigaciones se centran en la extracción de características mientras que otras lo hacen directamente en la clasificación o identificación de los usuarios. En cuanto a la extracción de características, se ha analizado qué patrones deben tenerse en cuenta en la extracción y los posibles problemas que surgen [7]. También se han utilizado algoritmos genéticos para seleccionar las características más relevantes, logrando así reducir drásticamente los errores de clasificación [28].

Se ha reportado una excelente performance utilizando un clasificador de Bayes [6], pero considerando cuidadosamente el tamaño del conjunto de muestras. En caso de ser reducido, se aconseja el método *K-NearestNeighbor*. Las redes neuronales, aunque han probado su buen rendimiento, requieren una gran cantidad de muestras para su correcto funcionamiento [16].

Pese a que son varios los factores que alteran el patrón de tipeo (condiciones físicas, mentales y del ambiente), una vez que el usuario está familiarizado con lo que escribe, las características se vuelven más estables [15]. Se ha investigado la relación que hay con el haberse habituado a lo que se está escribiendo [1].

Un criterio adecuado para clasificar a las investigaciones considera el análisis dinámico y estático [10]. En el análisis estático todos los usuarios tipean las muestras utilizando el mismo texto, mientras que el análisis dinámico implica un monitoreo continuo de lo que van escribiendo. Para evitar la confusión que estos términos pueden

acarrear, estas categorías también se pueden denominar como “texto fijo” y “texto libre” respectivamente. La mayoría de los trabajos se encuadran dentro de la primera categoría y sólo algunos tratan la temática correspondiente al segundo caso. La presente investigación presenta mayor afinidad con el análisis de texto libre, pues si bien el lenguaje de comandos de Linux es mucho más acotado que el lenguaje natural, no se ejecutan siempre los mismos comandos y tampoco se conserva el orden.

2.2. Selección de características y algoritmos de clasificación

Dado que se cuenta con datos reales y sin información extra de los mismos, se deben utilizar procedimientos no supervisados, que permiten una exploración de los datos [7]. Se requiere gran cuidado en la preparación y análisis de los datos, así como en la selección e implementación de los métodos apropiados. Pueden producirse resultados indeseados si no se han seleccionado correctamente las características discriminantes que se utilizarán en la etapa de clasificación. Es preciso utilizar la técnica de selección de características eliminando aquellas que no sean determinantes y/o obteniendo nuevas mediante la combinación de las anteriores, de tal forma que, sin perder información relevante, se reduzca el espacio dimensional y así sea más liviano computacionalmente [9].

2.2.1. Selección de características

Generalmente las características que se utilizan en la clasificación son extraídas subjetivamente y no tienen ninguna medida que indique si son las más adecuadas. Son varias las fuentes que indican la importancia de una buena selección de características teniendo en cuenta la calidad de los resultados finales [9] [18] [7] [11] [29].
El número de características (n) puede ser muy alto. Por esto, la exploración para obtener un subconjunto de esas características realizando una búsqueda completa puede demandar tiempos inaceptablemente largos, dado que hay 2^n posibles subconjuntos (porque cada característica puede estar habilitada o deshabilitada). Hay diferentes tipos de búsqueda que tienen por objetivo reducir ese tiempo y tratar de llegar a una solución que se acerque a la óptima. Los más efectivos son los basados en los principios evolutivos, los cuales se dedican a mantener una “población” de soluciones que va evolucionando hasta estabilizarse en la/las mejor/es solución/es. Dentro de ellos se encuentran los Algoritmos Genéticos (GA, *Genetic Algorithms*) y el Algoritmo Evolutivo de Selección Local (ELSA, *Evolutionary Local Selection Algorithm*).
Existen diversas medidas para determinar la performance. En el enfoque *wrapper* lo ideal, si se lo utiliza con métodos supervisados, es usar la tasa de clasificaciones erróneas como medida a minimizar. En el presente trabajo, al no contar con muestras etiquetadas, se deben utilizar otras medidas que tengan en cuenta la calidad de los conjuntos formados y su distribución.

2.2.2. Métodos de clasificación

Existe un tipo de redes neuronales, comúnmente utilizado, que suelen ser simples, monocapa y con algoritmos sencillos y rápidos que se utilizan para realizar el agrupamiento de patrones, conocidas como redes autoorganizadas. Durante el proceso de aprendizaje la red debe descubrir por sí misma rasgos comunes, regularidades, correlaciones, o categorías de datos de entrada, e incorporarlos a su estructura interna de conexiones [4].
Cuando el problema a resolver no tiene límites claros, como en el caso del presente trabajo, surge lo que se llama el dilema de la *estabilidad y plasticidad del aprendizaje.* Este dilema plantea cómo una red podría aprender nuevos patrones y cómo podría retener los previamente aprendidos. Como respuesta al *dilema* surgen las Redes basadas en la Teoría de la Resonancia Adaptativa (ART, A*daptive Resonance Theory*) que constan de dos capas entre las que se establecen conexiones hacia delante y hacia atrás. Dichas redes utilizan un aprendizaje no supervisado de tipo competitivo, donde se presenta cierta información de entrada y sólo una de las neuronas de salida de la red se activa.
El modelo de Kohonen (SOM, *Self Organizing Map*) es otro tipo de red autoorganizada de tipo competitivo, con capacidad de formar mapas de características de manera similar como ocurre en el cerebro. A diferencia de ART no existen conexiones hacia atrás y el aprendizaje es de tipo *off-line*, por lo que se distingue una *etapa de aprendizaje* (donde se presentan los datos de entrada y por semejanza entre ellos se establecen las diferentes categorías) y otra de *funcionamiento o consulta* (donde se clasifican los nuevos datos presentados a la red, en base a las categorías descubiertas en la etapa de aprendizaje).
La limitación de algunas redes neuronales es su costo computacional durante el entrenamiento. Un método de agrupamiento que no presenta esta limitación es el *k-means*. Este método intenta encontrar los centros de grupos o *clusters* naturales en los datos y requiere conocer el número de *clusters.* El algoritmo *k-means* es utilizado ampliamente debido a su sencillez [26] [25].
Dado que en el enfoque de este trabajo no se conocen el número de *clusters*, a primera vista se debería descartar el método de *k-means*. Sin embargo se ha presentado una forma de trabajar con algoritmos de este tipo donde no se

conoce el número de *clusters* [7]. La idea es repetir el proceso de *clustering* incrementando en cada paso el número de clases (n=1, n=2, etc.) y analizando los resultados hasta encontrar un número de *clusters* óptimo. Se han reportado buenos resultados buscando el número de *clusters* de esta manera [27].

3. Materiales y Métodos

Los datos con los que se cuenta están divididos en archivos. Cada uno de estos contiene información de las teclas que fueron presionadas en el *honeypot*. Todas las teclas fueron ingresadas dentro de terminales. Una terminal (también conocida como consola) es un programa informático que actúa como Interfaz de usuario para comunicarlo con el sistema operativo mediante una ventana que espera *órdenes* escritas por el usuario en el teclado, los interpreta y los entrega al sistema operativo para su ejecución.

Dentro del archivo hay una línea por cada tecla presionada. Cada línea se encuentra dividida en 4 columnas (cada una separada por dos puntos). La primeras dos columnas contienen una representación de la tecla presionada, la primera en código hexadecimal y la segunda en código ASCII o su interpretación. La tercera columna indica la terminal desde la cual se presionó la tecla. La última columna representa el tiempo (en segundos, con una precisión de una millonésima de segundo) transcurrido desde que se presionó la tecla anterior hasta que se presionó esa tecla, es decir, el tiempo entre dos teclas sucesivas.

Debido a que el tiempo de una tecla está directamente relacionado con la tecla anterior, se arman dígrafos antes de continuar, así no se pierde la relación entre ellas. Un dígrafo consiste en una tecla inicial, una tecla final y en el tiempo transcurrido entre que se presionó la primer tecla y se presionó la segunda. Para poder obtener más información, también se guardaron trigrafos, es decir, tres teclas consecutivas y dos tiempos: entre que se presiona la primera y la segunda tecla, y entre la segunda y la tercera.

Una sesión comienza cuando el intruso ingresa al *honeypot* y finaliza cuando se va. Si ingresa nuevamente no se puede estar seguro de que sea el mismo. Por esta razón en este trabajo se han tomado diferentes medidas para marcar la división entre sesiones: cambio de userid, cambio de terminal y si se encuentra un dígrafo con tiempo mayor a 600 segundos.

Luego de estudiar los datos, ver los digrafos y trigrafos más comunes, se elaboró una lista de 50 características. Se tuvo en cuenta en la selección, que un número elevado de sesiones las tuviesen. Se buscaron distintas relaciones entre los digrafos basándose en la bibliografía, así como relaciones entre trigrafos. Algunas de las características obtenidas de las sesiones fueron las siguientes: promedio del tiempo del digrafo LS (o sea el formado por las teclas 'L' y 'S'), promedio de tiempo del digrafo CD, relación entre el promedio de tiempo del digrafo LS y CD, diferencia entre el tiempo del digrafo CD y el digrafo S<Espacio>, tiempo promedio de todos los digrafos, desviación estándar, tiempo promedio de todos los digrafos terminados en <Enter>, promedio de la distancia del trigrafo CD<Espacio>.

Debido a que no se puede conocer *a priori* cuáles son las características y las combinaciones de éstas que identifican mejor a las sesiones, se optó por extraer manualmente una gran cantidad; y de acuerdo a la alta dimensión del espacio de soluciones posibles que eso generó y a la imposibilidad (por el elevado costo computacional) de hacer una búsqueda exhaustiva (requeriría hacer 2^{50} pruebas), se optó por usar un método de búsqueda dentro del espacio de soluciones a fin de encontrar una solución aceptable. Se eligió el algoritmo ELSA, debido a la buena relación desempeño/tiempo que han probado otras investigaciones.

4. Desarrollo del Método Evolutivo

4.1. Evolutionary Local Selection Algorithm (ELSA)

Para la implementación del ELSA se considera un grupo de agentes, donde cada uno tiene una cierta cantidad de energía y una solución (la selección de determinadas características) asociada. Se siguen los siguientes pasos:

- Se inicializan todos los agentes con la misma cantidad de energía y en la primera iteración todos ellos son ingresados al clasificador.
- Con la clasificación obtenida se determina la bondad de la solución de acuerdo a varias funciones de desempeño. Luego se reparten la energía en relación a los puntajes obtenidos por las funciones de desempeño, donde cada función tiene su propio reservorio de energía para repartir.
- A cada agente se le quita una cantidad fija (e igual para todos) de energía (se considera que es la energía que gastan en "vivir"). Si el agente quedó sin energía, es descartado ("muere"), pero si superó cierto umbral, se duplica (quedando cada clon con la mitad de la energía) y muta.

Luego comienza otra iteración. Así se obtiene una población de agentes que va evolucionando hacia varios objetivos, los cuales pueden ser variados e intercambiados fácilmente, pudiendo incrementar o decrementar su cantidad [26].

4.1.1. Implementación del algoritmo ELSA

- Se inicializan las variables. Éstas son: *p* (cantidad inicial de agentes), *tita* (umbral de reproducción), *Ecost* (costo de energía necesario para vivir), *T* (cantidad de iteraciones). Para establecer los valores de inicio, la idea general que se consideró fue procurar que "mueran" la misma cantidad de agentes que se reproducen, para tener una población estable, pues si crece demasiado el algoritmo se hace muy lento y si "mueren" demasiados concluye rápidamente sin alcanzar a evolucionar.
- Como no se conoce la cantidad de *clusters* óptima, esta cantidad *K* pasa a ser parte la solución y también va evolucionando. Es decir, cuando un agente muta, puede cambiar el número de características seleccionadas y el número de *clusters* con el que va a clasificar.
- La solución que tiene cada agente es un vector binario. La cantidad de bits es la suma de la cantidad de características más el máximo número de *clusters*. Dado que los casos con 0 y 1 *cluster* son irrelevantes, se descuentan dos bits. En los bits que representan a las características, un 1 significa que la misma está seleccionada y un 0 que no lo está. Los bits que representan *clusters* directamente se suman para obtener la cantidad de éstos.
- Se inicializan todos los agentes con la mitad de la energía necesaria para reproducirse.
- Se inicializan los reservorios de energía. Para mantenerlo estable se eligió que la cantidad de energía que entra al sistema sea la misma que la que sale por el gasto de los agentes.
- Comienza la iteración: se ejecuta el clasificador y se obtiene el desempeño del agente en cada función.
- Se reparte la energía entre los agentes. Hay un reservorio por cada función de desempeño. Por cada reservorio la idea es la siguiente: al peor agente no se le asigna nada y al resto se le reparte de acuerdo a su relación con los demás. Por ejemplo, si son tres agentes y en determinada función obtienen 0.8, 0.5 y 0.2, al primero no se le asigna nada, y al último se le da el doble (2/3) de lo que se le da al segundo (1/3). Se implementó de esta manera debido a que la evolución era lenta computacionalmente y si se seguía la implementación original los agentes morían antes de progresar.
- Si la energía resultante supera o iguala el valor de *tita*, el agente se duplica y ese duplicado muta un bit (se elige uno al azar y se invierte, puede ser correspondiente a una característica seleccionada o a la cantidad de *clusters*).
- Si la energía resultante queda en cero o menos, muere.
- Toma los dos mejores agentes y los reproduce con el método *Half Uniform Crossover* (HUX). El "hijo" mantiene los bits de la solución que coinciden en ambos "padres" y para los demás usa bits aleatorios. Esto mejora el desempeño del algoritmo y acelera la evolución. Se llenan de energía los repositorios, nuevamente.
- Comienza otra iteración.
- Al final se obtiene el mejor agente (ver sec. 4.3) y también la última "camada" de agentes que dejó el algoritmo.

4.2. Evaluación del desempeño

Las funciones de desempeño son la clave del éxito del algoritmo ELSA, pues los agentes van a evolucionar tratando de optimizar sus valores. Se han contemplado las siguientes:

*4.2.1. **Within:*** procura hallar *clusters* compactos. Calcula la distancia entre cada punto y su centro de *cluster*. Se busca minimizarlo. Se define como:

$$F_{within} = 1/n \sum_{k=1}^{K} \sum_{i=1}^{n} \alpha_{ik} \sum_{j \in J} \left(\left(x_{ij} - \gamma_{kj} \right) \right) / \lambda_k ,$$

donde $\mathbf{x_i}$: i-ésima sesión; $\mathbf{x_{ij}}$: j-ésima característica de la i-ésima sesión; $\alpha_{i,k}$: membresía de xi al *cluster* k; λ_k:cantidad de sesiones en el *cluster* k; K: cantidad de *clusters*; n: cantidad de sesiones y

$$\gamma_{kj} = \sum_{i=1}^{n} \alpha_{ik} x_{ij} / \sum_{i=1}^{n} \alpha_{ik} , j \in J .$$

4.2.2. Between: procura hallar ***clusters*** bien separados entre ellos. Calcula la distancia entre cada punto y el resto de los puntos de otros *clusters*. Se busca maximizarlo. Su expresión es:

$$F_{between} = 1/n \sum_{k=1}^{K} \sum_{i=1}^{n} \left(1 - \alpha_{ik}\right) \sum_{j \in J} \left(\left(x_{ij} - \gamma_{kj} \right) \right) / \left(K - \lambda_k \right).$$

4.2.3. DunnIndex: considera la relación entre los dos elementos más cercanos pertenecientes a distintos *clusters* sobre los dos elementos más lejanos pertenecientes al mismo *cluster*. Se busca minimizarlo.

4.2.4. Complexity: favorece la elección de bajas dimensiones. Cuanto menos características se eligieron, mejor es el resultado (no depende de cómo quedaron agrupados los datos). Se busca minimizarlo. Llamando **d** a la dimensión del conjunto seleccionado de características y D a la dimensión de todo el conjunto de características, se define:

$$F_{complexity} = \left(d - 1\right) / \left(D - 1\right).$$

4.2.5. Sesiones Partidas: favorece la selección de *clusters* coherentes. Las sesiones más largas fueron divididas, y esta función busca que las distintas partes sean clasificadas en el mismo *cluster*. Su rango va de 1 (el mejor de los casos) a 15 (el peor de los casos).

4.3. Mejor Agente

Se considera como mejor agente a aquel que en una iteración dada presenta una mejor solución, evaluada mediante la función de desempeño, que el resto. Por lo tanto se tendrán tantos de estos agentes como funciones activadas haya. Además entra en esta categoría el agente que obtuvo la mayor cantidad de energía en esa iteración.

5. Experimentos y Resultados

Es difícil evaluar la calidad de un algoritmo de aprendizaje no supervisado, y la selección de características agrega la dificultad de que la cantidad de *clusters* dependen de la dimensionalidad de las características seleccionadas. Además, cualquier subconjunto de características dado puede tener sus propios *clusters*, que pueden ser incompatibles con aquellos formados por diferentes subconjuntos.

Para la presente aplicación, debido a que no hay forma de saber cuáles sesiones pertenecen al mismo intruso, se buscaron diferentes maneras de verificar que la clasificación se realice correctamente. Con dicho fin, se tomaron aquellas sesiones conformadas por más de 1000 dígrafos y se dividieron creando dos nuevas sesiones. De esta forma se puede verificar la clasificación comprobando que las tres sesiones queden en el mismo *cluster* (se sabe que son del mismo intruso).

De todas las sesiones disponibles, se tomaron solamente aquéllas que tenían valores para todas las características. Al tomar las 50 características sólo se contaba con 24 sesiones que cumplieran dicha propiedad, por lo tanto se comenzaron a eliminar aquéllas que se encontraban en menos sesiones. De esta forma, se trabajó con un *cluster* de 106 sesiones que compartían 40 características.

El algoritmo ELSA se ejecutó repetidas veces variando en cada una las funciones de desempeño activadas y el método de clasificación. Los mismos fueron *k-means*, SOM y ART. Los resultados pueden observarse en las tablas 1, 2 y 3 respectivamente. Los valores de las funciones que no estaban activadas figuran entre corchetes, indicados para poder comparar las distintas soluciones. Hay un agente por cada ejecución realizada. Se seleccionó al que obtuvo mayor cantidad de energía en la última iteración.

k-means	Cant de clusters	Cant de caract	dunnIndex	between	within	complejidad dimensional	sesiones partidas
agenteKmeans1	32	1	137,5673	7,8256	0,1511	0	11
agenteKmeans2	34	1	[137,5673]	8,6621	0,1508	[0]	[11]
agenteKmeans3	17	21	3,8761	[2,8907]	[1,0080]	[0,5128]	1

Tabla 1: Resultados del clasificador k-means

SOM	Cant de *clusters*	Cant de caract	dunnIndex	between	within	complejidad dimensional	sesiones partidas
agenteSOM1	33	1	105,12	6,8578	0,1849	0	9
agenteSOM2	25	17	[11,4095]	3,5731	0,8792	[0,4103]	[11]
agenteSOM3	29	25	5,3543	[4,2644]	[1,3274]	[0,6154]	5

Tabla 2: Resultados del clasificador SOM

ART	Cant de *clusters*	Cant de caract	dunnIndex	between	within	complejidad dimensional	sesiones partidas
agenteART1	9	2	32,444	2,47	0,3079	0,0256	4
agenteART2	11	100	[6,0547]	17,6045	[0.3121]	[0,2308]	13
agenteART3	82	19	4,1380	[13,1514]	[0,8831]	[0,4615]	4

Tabla 3: Resultados del clasificador ART

De la última iteración de cada ejecución, además del agente que obtuvo mayor cantidad de energía, se conservaron los agentes que desempeñaron mejor en cada una de las funciones activadas. De esta forma, si había tres funciones activadas, se conservaron cuatro agentes (soluciones). Una vez calculados los valores para todas las funciones de desempeño, los agentes compitieron por la obtención de energía. El que obtuvo el mayor valor fue el agente de la tabla 4.

ART	Cant de *clusters*	Cant de caract	dunnIndex	between	within	complejidad dimensional	sesiones partidas
mejorAgente	90	7	6,6280	15,5748	0,3757	0,1538	8

Tabla 4: Agente con mayor energía

Se decidió considerar las sesiones partidas para verificar que la clasificación se realizó correctamente, pero el mejor agente de la tabla 4 preponderó las demás funciones. Por esa razón se eligió darle más importancia a la función de desempeño que realiza esta verificación que al resto. Entonces se tuvieron en cuenta sólo sesiones partidas. Puede llegarse al caso de que tome un agente cuyo valor de la función sea el mínimo pero que no tenga la mejor distribución de *clusters*. Se hizo competir nuevamente a los agentes seleccionados pero esta vez se tomó en cuenta el desempeño de la función *dunnIndex* y sesiones partidas, para lograr equilibrio entre la calidad de los *clusters* y la clasificación deseada. El resultado obtenido se muestra en la tabla 5.

k-means	Cant de *clusters*	Cant de caract	dunnIndex	between	within	complejidad dimensional	sesiones partidas
agenteK3	17	21	3,8761	[2,8907]	[1,0080]	[0,5128]	1

Tabla 5: Agente considerado como la mejor solución

En el último caso se encontró una solución en la cual todas las sesiones partidas se clasificaron correctamente, pese a que la cantidad de *clusters* no era trivial, y la función *dunnIndex* devolvió un valor adecuado. Este agente cuenta con la desventaja de que la cantidad de características seleccionadas es elevada.

6. Conclusiones

En este trabajo se presentó un enfoque novedoso en la clasificación de intrusos. Se trabajó con un algoritmo evolutivo con múltiples funciones objetivo, integrado con los clasificadores *k-means*, SOM y ART. El algoritmo ELSA mostró un desempeño satisfactorio debido a que permitió una rápida búsqueda en un espacio dimensional complejo y admitió el fácil intercambio de objetivos a los que se quería hacer tender la solución.

Se experimentó con diversas combinaciones de funciones de desempeño y algoritmos de clasificación no supervisados, obteniendo para cada uno de ellos diferentes soluciones que en mayor o menor medida se aproximaron a los resultados esperados. La mejor solución se obtuvo combinando la función que favorece *clusters* compactos y separados, y la función “sesiones partidas” que busca una correcta clasificación de sesiones conocidas.

El clasificador que presentó el mejor desempeño fue *k-means*, pese a ser el más sencillo, que a su vez presentó la ventaja de ser el menos costoso computacionalmente.
En trabajos futuros se buscará analizar la interacción entre los diversos criterios de optimización para lograr minimizar el número de características seleccionado, sin perder la cantidad ni la calidad de los *clusters* formados. También se proyecta incorporar una nueva función para considerar la distancia de Mahalanobis entre las características, ya que dicha distancia ha demostrado ser muy adecuada para este tipo de problemas.

Referencias

[1] Araújo L. C. F., Lizárraga M. G., et al. (2005). "Autentificación personal por dinámica de tecleo basada en lógica difusa." IEEE COMPUTER SOCIETY.
[2] Azevedo G.L.F. , Cavalcanti G.D.C. , et al. (2007). "An approach to feature selection for keystroke dynamics systems based on PSO and feature weighting." IEEE COMPUTER SOCIETY.
[3] Bergadano Francesco, Gunetti Daniele, et al. (2002). "User Authentication through Keystroke Dynamics." ACM.
[4] Brio Bonifacio Martín del y Molina Alfredo Sanz (2001). Redes Neuronales y Sistemas Difusos. Bogotá, Colombia, Alfaomega Grupo Editor.
[5] Cheng-Huang Jiang, Shiuhpyng Shieh, et al. (2007). Keystroke statistical learning model for web authentication. Proceedings of the 2nd ACM symposium on Information, computer and communications security. Singapore, ACM.
[6] Cho Tai-Hoon (2006). "Pattern Classification Methods for Keystroke Analysis." SICE-ICASE International Joint Conference.
[7] Duda Richard O., Hart Peter E., et al. (2001). Pattern Classification, Wiler InterScience.
[8] Dy Jennifer G. y Brodley Carla E. (2004). "Feature Selection for Unsupervised Learning." J. Mach. Learn. Res. 5: 845-889.
[9] Escudero Laureano F. (1977). Reconocimiento de Patrones. Madrid, España, Paraninfo.
[10] Gunetti Daniele y Picardi Claudia (2005). "Keystroke Analysis of Free Text." ACM.
[11] Guyon Isabelle (2003). "An introduction to variable and feature selection." J. Mach. Learn. Res. 3: 1157-1182.
[12] Hilera José R. y Martínez Víctor J. (1995). Redes Neuronales Artificiales. Fundamentos, modelos y aplicaciones. Madrid, España, Rama.
[13] Kacholia Varun y Pandit Shashank (2004). "Biometric Authentication using Random Distributions (BioART)." shashankpandit.com.
[14] Lee Cheng Jae-Wook, Choi Sung-Soon, et al. (2007). "An Evolutionary Keystroke Authentication Based on Ellipsoidal Hypothesis Space." ACM.
[15] Mroczkowski Piotr (2004). "Identity Verification using Keyboard Statistics." Linkoping University, Electronic Press.
[16] Obaidat M. S. y Sadoun Balqies (1997). "Verification of Computer Users Using Keystroke Dynamics.":IEEE COMPUTER SOCIETY.
[17] Paeckock Alen, Ke Xian, et al. (2004). "Typing Patterns: A Key to User Identification." IEEE COMPUTER SOCIETY.
[18] Rhee F.C.-H. y Lee Young Je (1999). "Unsupervised feature selection using a fuzzy-genetic algorithm." IEEE COMPUTER SOCIETY.
[19] Robinson John, Liang Vicky, et al. (1998). "Computer user verification using login string Keystroke Dynamics." IEEE COMPUTER SOCIETY.
[20] Shepherd S J (1995). "Continuous Authentication by analysis of keyboard typing characteristics." IEEE COMPUTER SOCIETY.
[21] Sim Terence y Janakiraman Rajkumar (2007). "Are Digraphs Good for Free-Text Keystroke Dynamics?" IEEE COMPUTER SOCIETY.
[22] Sondberg-Madsen Nicolaj, Thomsen Casper, et al. (2003). "Unsupervised feature subset selection." ECML/PKDD.
[23] Sung Ki-Seok y Cho Sungzoon (2006). "GA SVM wrapper ensemble for keystroke dynamics authentication." Lecture notes in computer science.
[24] Villani Mary, Tappert Charles, et al. (2006). "Keystroke Biometric Recognition Studies on Long-Text Input under Ideal and Application-Oriented Conditions." SCIS.
[25] Yijuan Lu, Ira Cohen, et al. (2007). Feature selection using principal feature analysis. Proceedings of the 15th international conference on Multimedia. Augsburg, Germany, ACM.
[26] YongSeog Kim, Street W. Nick, et al. (2000). Feature selection in unsupervised learning via evolutionary search. Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining. Boston, Massachusetts, United States, ACM.
[27] YongSeog Kim, Street W. Nick, et al. (2003). Feature selection in data mining. Data mining: opportunities and challenges, IGI Publishing: 80-105.
[28] Yu Enzhe y Cho Sungzoon (2003). "Biometrics-based Password Identity Verification: Some Practical Issues and Solutions." Http://dmlab.snu.ac.kr.
[29] Yu Enzhe y Cho Sungzoon (2004). "Keystroke dynamics identity verification—its problems and practical solutions." Elsevier Ltd.
[30] Zheng Zhao y Huan Liu (2007). Spectral feature selection for supervised and unsupervised learning. Proceedings of the 24th international conference on Machine learning. Corvalis, Oregon, ACM.
[31] Zhu Zexuan, Ong Yew-Soon, et al. (2007). "Wrapper-filter feature selection algorithm using a memetic framework." IEEE COMPUTER SOCIETY.