Aplicación de GHSOM (Growing Hierarchical Self-Organizing Maps) a Sistemas de Detección de Intrusos (IDS)

Artículo de Revisión - Fecha de recepción: 18 de agosto de 2012 - Fecha de aceptación: 10 de septiembre de 2012

Eduardo Miguel De la Hoz
Magíster en Ingeniería de Computadores y Redes, Corporación Universidad de la Costa - CUC. Barranquilla, Colombia, edelahoz6@cuc.edu.co

Andrés Ortiz
Doctor en Tecnologías de la Información y las Comunicaciones, Universidad de Málaga. Madrid, España, aortiz@ic.uma.es

Julio Ortega
Doctor en Tecnologías de la Información y las Comunicaciones, Universidad de Granada. Granada, España, julio@atc.ugr.es

RESUMEN

Con el pasar de los años, en el ámbito de la seguridad informática el problema de la intrusión se desarrolla cada día más, incrementando la existencia de programas que buscan afectar a computadoras tanto a nivel local como a toda una red informática. Esta dinámica lleva a entender los ataques y la mejor manera de contrarrestarlos, ya sea previniéndolos o detectándolos a tiempo, procurando que su impacto sea menor al esperado por el atacante. En este artículo se presenta una revisión de los ataques a sistemas informáticos, ahondando en los Sistemas de Detección de Intrusos (IDS) y en la implementación de técnicas de agrupamiento de datos —como las redes neuronales—, con el fin de encontrar métodos con altas precisiones en la detección de anomalías. Esta propuesta presenta la aplicación de GHSOM en IDS, utilizando el conjunto de datos NSL-KDD, y mostrando las mejoras encontradas en la detección de ataques en el proceso de búsqueda.

Palabras clave

Seguridad informática, Sistemas de Detección de Intrusos (IDS), NSL-KDD, GHSOM, ataques.

Application of GHSOM (Growing Hierarchical Self-Organizing Maps) to Intrusion Detection Systems (IDS)

ABSTRACT

As time passes by, in the field of computer security, intrusion problems grow every day increasing the existence of programs that seek to affect computers both locally and across a network. This dynamic has led to an imminent need of understanding the attacks and finding the best way to counteract them either by preventing them or by detecting them on time, diminishing the impact expected by the attacker. This article presents a review of attacks on computer systems, delving into the Intrusion Detection System (IDS) and the implementation of data clustering techniques like neural networks in order to find high accuracy methods for anomaly detection. This proposal presents GHSOM for IDS using NSL-KDD dataset, and illustrates attack detection improvement in the search process.

Keywords

Computer security, Intrusion Detection Systems (IDS), NSL-KDD, GHSOM, attacks.

INTRODUCCIÓN

La seguridad es definida por la RAE [1] como: "cualidad de seguro", siendo "seguro" [2] el adjetivo que se define, según esta misma entidad, como: "Libre y exento de todo peligro, daño o riesgo"; de igual forma el término informática [3] es definido como: "Conjunto de conocimientos científicos y técnicas que hacen posible el tratamiento automático de la información por medio de ordenadores".

Asumiendo las precisiones anteriores, se logra inferir que la seguridad informática lo que pretende es escudar la información, siendo esta última la "agregación de datos que tiene un significado más allá de cada uno de éstos" [4], y tendrá un sentido particular según quién y cómo la procese.

El valor que puede tener la información es relativo y en muchos casos se le da una valoración subjetiva u objetiva debido a su alto grado de intangibilidad, algo que no suele ocurrir con las aplicaciones, los edificios, los equipos de cómputo o la documentación física.

La información puede ser pública o privada, además de cumplir con cualidades como son la integridad, el aislamiento, la confidencialidad, la disponibilidad, la autenticidad, la protección a la réplica, el no repudio, la consistencia, la privacidad, el control y la auditoría, la cual podemos definir como la capacidad que se tiene sobre el sistema en relación con procesos o acciones y determinar así cómo y quién las realiza.

Al hablar de seguridad informática e información también se deben mencionar las amenazas, siendo estas consideradas como cualquier elemento que comprometa al entorno del sistema. Las amenazas pueden ser catalogadas en tres espacios de tiempo: antes del ataque, durante y después del mismo. Lo anterior nos conduce a los mecanismos que garantizan la seguridad de un sistema informático en esas fases:

a) *La prevención (antes):* Son aquellos mecanismos que aumentan la fiabilidad o seguridad del sistema durante su funcionamiento normal.

b) *La detección (durante):* Aquí podemos nombrar aquellos mecanismos orientados a revelar violaciones de seguridad como lo son los sistemas de detección o prevención de intrusos, de los cuales se hablará más adelante.

c) *La recuperación (después):* Son aquellos mecanismos que se aplican cuando ya el sistema ha sido penetrado para poder llevarlo a su funcionamiento normal.

Es sabido por la comunidad informática que los encargados de comprometer la seguridad y tratar de burlar los mecanismos que garanticen un eficaz y eficiente desempeño de los sistemas son los intrusos, los cuales son clasificados con base en la magnitud del daño que causan al sistema atacado [5]: los primeros son los IC (Intrusos Curiosos), se les dice a las personas curiosas que no tienen un grado de conocimiento en informática pero que aun así intentan acceder a sistemas a los que oficialmente no tienen acceso y no causan mayor daño al sistema, solo lo espían. En segundo lugar están los crackers, que son aquellos usuarios que con cierto nivel de conocimiento usan un sistema de manera ilegítima para ejecutar programas

maliciosos o robar información para su beneficio. Por último están los IR (Intrusos Remunerados), los que son catalogados como los de más peligro ya que aunque no son comunes en pequeñas organizaciones cuentan con una gran experiencia en temas de seguridad y un amplio conocimiento del sistema, el cual utilizan para iniciar acciones maliciosas con fines delictivos.

En la actualidad las redes son más susceptibles a ataques que en años pasados, ya que los usuarios acceden a la red a través de conexiones de banda ancha y móvil, así como a través de Internet. Este aumento en el nivel de acceso introduce ataques maliciosos a los que se conoce como ataques remotos. En esta investigación se propone la utilización de técnicas de agrupamiento de datos así como la implementación de nuevas y novedosas prácticas computacionales para la detección de ataques y anomalías en las redes de datos causadas por cualquiera de los tres tipos de usuarios mencionados anteriormente. Así, se expondrá la temática referente a las técnicas de agrupamiento de datos, se presentará el tema de las redes neuronales y también se puntualizará sobre los métodos utilizados para la investigación desarrollada, describiendo en detalle la implementación de GHSOM, donde se mostrarán los resultados obtenidos en el estudio adelantado, en los cuales los autores ponen a disposición de la comunidad científica en general nuevos métodos con altos porcentajes de acierto en la detección de anomalías en las redes de datos actuales.

ATAQUES A SISTEMAS INFORMÁTICOS

Los ataques informáticos son intrusiones ilegales a la seguridad de un sistema, siendo una intrusión la materialización de una amenaza. Autores como Heady [6] definen intrusión como cualquier conjunto de acciones que tratan de comprometer la integridad, confidencialidad o disponibilidad de un recurso.

Gran parte de las intrusiones son realizadas por los puertos de las computadoras destino, utilizando programas de escaneo de puertos. Esta técnica de exploración pretende hallar qué servicios están siendo ofrecidos por una red o servidor, para realizar conexiones o intentos de conexión a diferentes puertos (TCP o UDP) en la víctima, esperando obtener —dependiendo de su estado de abierto, cerrado o bloqueado— respuestas de alguno o algunos de ellos e inferir qué aplicación o servicio está activo en dicho puerto. El estado más vulnerable a un ataque es cuando el puerto se encuentra abierto, esto significa que una aplicación servidor está escuchando por ese puerto las peticiones de los clientes que se conecten. Por ejemplo, si recibe una respuesta del puerto 22 supondrá que se trata del servicio de SSH, aunque probablemente sea un servidor web el que esté escuchando peticiones en aquel puerto si el administrador del host así lo ha configurado. Un puerto abierto puede brindar información sobre las vulnerabilidades de seguridad del sistema. Por esta razón, una de las primeras actividades que un atacante intentará realizar en contra de un sistema es sin duda una exploración de puertos, por lo que se recomienda tener todos los puertos cerrados a menos que sea estrictamente necesario utilizarlos ya que si se hace efectivo el escaneo de puertos el atacante obtendrá información básica acerca de los servicios ofrecidos y, adicionalmente, otros detalles del entorno.

EDUCOSTA
EDITORIAL UNIVERSITARIA DE LA COSTA

Existe una gran cantidad de tipos de escaneo [7] que pueden basarse en los protocolos TCP o UDP, diferenciándose básicamente por las banderas de protocolo utilizadas tales como SYN, ACK o RST aplicables solo a TCP, pero los dos tipos de escaneo más comunes son el TCP Connect y el TCP SYN, como se muestra en la Fig. 1 [8] suministrada por el Institute for Internet Security (IIS). Se observa la utilización del proceso de conexión convencional del protocolo TCP conocido como triple handshaking o saludo triple, llamado así por los tres mensajes que se intercambian al inicio de una nueva conexión (SYN, SYN_ACK y ACK). El escaneo TCP SYN o semi-abierto no establece una conexión por cada puerto, es más rápido y difícil de detectar.

Ataques a sistemas informáticos

En la actualidad existen múltiples y diversos ataques informáticos para vulnerar sistemas informáticos; entre los más usados podemos mencionar al Spoofing; este tipo de ataque se caracteriza por la elaboración de tramas TCP/IP utilizando una dirección IP falseada. El objeto [9] de dicho ataque es que el atacante simule la identidad de otra máquina en una red de datos con el propósito de adquirir acceso a recursos de un tercer sistema con el que se ha podido llegar a tener confianza basándose en el nombre o la dirección IP de la máquina suplantada. Este ataque es en la actualidad uno de los más usados por las personas que cuentan con un gran conocimiento del protocolo TCP/IP ya que muchos sistemas basan su funcionamiento en anillos de confianza, esquema que hace que los usuarios de un sistema aporten su clave pública al sistema y firmen las claves del resto de los usuarios, método que no garantiza que una clave es de quien dice ser, más que por la confianza que se pueda tener en el firmante de dicha clave.

Otro importante ataque que hoy día es implementado para estos propósitos es el llamado "Negación de servicio", también conocido por sus siglas en inglés como DoS (Denial of Service) que en su gran mayoría están dirigidos contra un recurso informático, ya sea una máquina o una red. En este ataque el o los atacantes tratan de limitar de

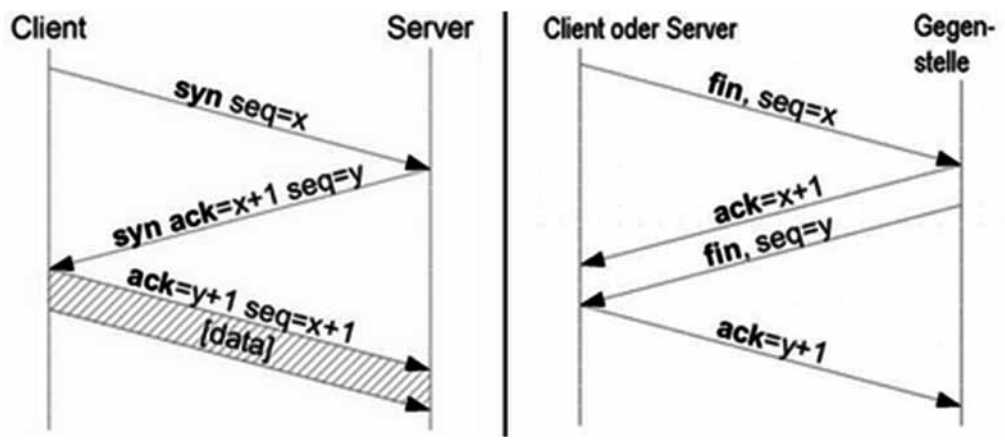

Fig. 1 Resumen de los tipos de escaneo más comunes: TCP Connect y TCP SYN

una manera parcial o total a usuarios legítimos el acceso a servicios prestados por un recurso informático. Esta característica hace que este ataque se convierta en uno de los ataques más sencillos y contundentes contra todo tipo de servicios, convirtiéndose en un serio problema, ya que un delincuente informático puede interrumpir constantemente un servicio sin necesidad de grandes conocimientos o recursos, utilizando programas sencillos o con la ayuda de una gran masa de usuarios ingenuos al ataque.

Existen dos tipos principales de ataques de negación de servicio [10], los primeros se encargan de explorar grietas o desperfectos del software para causar una falla en los procesos del sistema, agotando así sus recursos. A los segundos se les denomina "por inundación", que son los encargados de inundar a un sistema o a un componente del sistema con más información de la que estos pueden manejar. Con este tipo de ataques no existe ninguna grieta o desperfecto en el sistema que se deba reparar.

Aunque los ataques mencionados anteriormente logran dañar al sistema destino, se puede alcanzar un ataque aún más agresivo si se combinan dos o más técnicas de ataque, como es el caso reportado por ComputerWire [11] donde se utilizó la técnica de spoofing de direcciones IP y un ataque de negación de servicio.

Mecanismos de prevención

Debido al auge de los ataques se han desarrollado mecanismos para prevenir o tratar en lo posible que dichos ataques no produzcan el daño deseado por los atacantes o, en su defecto, no produzcan daño alguno. Entre estos podemos nombrar [12] la autenticación e identificación, los mecanismos de control de acceso, los cortafuegos que sirven para separar de la manera más segura posible una máquina o subred del resto y, por último, la criptografía [13], que garantiza la confiabilidad de las comunicaciones de datos por las redes privadas o públicas, valiéndose de cifrados de clave pública, privada o firmas digitales.

Mecanismos de detección

Utilizados para detectar violaciones de la seguridad o cualquier tipo de intento que pueda comprometer el óptimo desempeño del sistema. Los más representativos de estos mecanismos son los IDS, conocidos comúnmente como Sistemas de Detección de Intrusos, que se encargan de supervisar y registrar toda actividad de un sistema para su análisis en busca de alguna actividad maliciosa y dar así la respuesta más apropiada a dicha actividad. Entre los IDS disponibles se pueden nombrar: OSSEC [14], Tripwire [15], Snort [16], RealSecure [17], Prelude [18], entre otros.

Mecanismos de recuperación

Si ocurriese el caso en donde la prevención y la detección han fallado, surgen mecanismos que ayudan a recuperar aquellas anomalías presentadas después de un ataque o violación al sistema. La idea que se tiene con estos mecanismos es poder regresar al sistema informático a su funcionamiento normal, ya sea mediante elementos como los antivirus, hardware adicional o mecanismos de detección que incluyan software

para la recuperación de sistemas a su estado inicial.

SISTEMAS DE DETECCIÓN DE INTRUSOS (IDS)

Los IDS supervisan y registran los eventos que ocurren en una computadora o en una red de computadoras, para analizarlos y correlacionarlos en la búsqueda de señales de intrusión y dar así una respuesta que permita corregir esta situación [19], además avisan de la ocurrencia de malas prácticas, como en el caso de los usuarios autorizados que intentan sobrepasar sus límites de restricción de acceso a la información [20], con el ánimo de poder dar con los responsables del ataque y tomar acciones conducentes a superar la vulnerabilidad y castigar, si se puede, a el o a los responsables de dicho ataque. Además, un IDS dispone de los medios para manejar alertas cuando se detectan signos de intrusión, lo que permite tomar las medidas correspondientes en el menor tiempo posible.

Los IDS, independientemente del sistema que vigilen o de su forma de trabajo, generalmente propenden a minimizar el consumo de recursos, permiten o no utilizar una configuración según las políticas de seguridad que dicte la organización, se adaptan a los cambios acelerados que sufren los sistemas y usuarios, incluyendo un proceso rápido y sencillo de actualización; ejecutan sus funciones de forma continua, transparente y con un mínimo de supervisión; así mismo, toleran fallos y se recuperan de ellos aun si la red colapsa en algún momento. Por último, los IDS deben dar la posibilidad de recuperar componentes comprometidos y, en caso contrario, administrar un tipo de alerta.

Eficiencia de los IDS

Para medir la eficiencia de un IDS a la hora de detectar intrusiones en un sistema determinado se usan las siguientes características: precisión, rendimiento, completitud, tolerancia a fallos y tiempo de respuesta.

La *precisión* es la capacidad que tiene el sistema IDS para detectar ataques y distinguirlos del tráfico normal de una red. Para medir la precisión de un IDS se utilizan dos ratios: el porcentaje de falsos positivos; es decir, el número de veces que se detecta un ataque que no existe, y el porcentaje de falsos negativos; es decir, el número de ataques no detectados. Cuanto menores sean estos ratios mejor será el detector; sin embargo, por lo general, cuando disminuye uno aumenta el otro por lo que el ideal es encontrar un buen equilibrio entre los dos. Lo anterior se resume en:

$$\text{Precisión} = \frac{\text{Ataques\_reales\_detectados}}{\text{Ataques\_reales\_detectados} + \text{Falsos\_positivos}}$$

El *rendimiento* hace referencia al número de eventos que es capaz de analizar un sistema. Lo ideal en este caso es que se analice el tráfico de la red en tiempo real aunque esto es altamente difícil de hacer si se quiere analizar a fondo el contenido de los paquetes. El rendimiento de un sistema depende de la capacidad de procesamiento de hardware de la que se disponga. Cada tipo de IDS debe alcanzar un equilibrio entre la cantidad de paquetes que debe analizar y la profundidad de este análisis.

La *completitud* de un IDS se cumple cuando es capaz de detectar todos los tipos de ataques. Por lo general, los sistemas detectan solo algunos tipos y es necesario combinar

varias técnicas para conseguir la completitud. Esta característica debe equilibrarse con la *precisión*; es decir, poder detectar la mayor parte de los ataques sin que el número de falsas alarmas crezca demasiado. Su fórmula es la siguiente:

$$\text{Completitud} = \frac{\text{Ataques\_reales\_detectados}}{\text{Ataques\_reales\_detectados} + \text{Falsos\_negativos}}$$

La *tolerancia a fallos* da al IDS la propiedad de resistir a los ataques. Un IDS debe ser seguro y robusto para evitar que un ataque lo inutilice dejando todo el sistema vulnerable. Además de resistir a los ataques, esta propiedad determina la capacidad del IDS a resistir un fallo del sistema, por ejemplo un corte de luz o la destrucción del terminal donde se ejecuta. Es importante que un IDS guarde los patrones que utiliza para la detección y los "logs" que ha ido recogiendo con su ejecución.

El *tiempo de respuesta* es el tiempo que tarda en reaccionar ante un ataque, ya sea lanzando una alarma o poniendo medidas para detener o retardar la intrusión. Obviamente, cuanto menor sea el tiempo de respuesta más efectivo será el sistema, aunque debe asegurarse de que existe realmente un ataque antes de actuar.

Clasificación de los IDS

La categorización de los Sistemas de Detección de Intrusos ha sido tratada por varios autores, entre ellos podemos mencionar a Hervé Debar [21], Stefan Axelsson [22] de Chalmers University of Technology, así como a Wu y Banzhaf [20], quienes los clasificaron según la fuente de datos o información, el análisis y la estrategia de respuesta.

Los IDS por fuentes de información hacen referencia a la fuente u origen de los datos que se utilizan en el análisis para determinar si una intrusión se ha llevado a cabo. Aquellos que utilizan el análisis hacen mención al método de detección utilizado como estrategia. Y, por último, los que utilizan mecanismos de respuesta son los que una vez se ha determinado si ha sucedido alguna intrusión, pueden o bien responder de forma activa ante la misma, o bien registrar la detección y no realizar acción alguna.

Tendencias a futuro

Hoy día se trabaja en la implementación de IDS utilizando métodos avanzados, con lo que se ha logrado cierta eficiencia y eficacia en la detección de intrusiones; entre ellos podemos encontrar [23] la detección de patrones en ciertas actividades del sistema operativo o en cadenas de tráfico de red relacionadas con potenciales ataques. El problema del que adolecen estos sistemas de patrones es que las firmas que contienen los ataques conocidos, al estar almacenados en algún lugar del sistema, los imposibilita para detectar futuras intrusiones que sean desconocidas; es decir, que no correspondan con alguno de los patrones almacenados por el sistema. El ejemplo más claro que se puede mencionar de este tipo de sistemas es STAT [24], el cual basa su funcionamiento en la detección de patrones.

Otra de las nuevas tendencias muy utilizadas en la actualidad es la inteligencia artificial, la cual es capaz de identificar patrones de ataques o actividad anómala por medio de ciertos indicadores para un perfil, como es el caso de las redes neuronales, encarga-

das de elaborar modelos dinámicos para los perfiles de uso normal de un sistema, los cuales se van retroalimentando con los diferentes tipos de actividad, ya sea anómala o no, y así obtener perfiles que se adecúen al comportamiento habitual del sistema.

Por último, también son prometedores los métodos estadísticos que identifican desviaciones numéricas en ciertos indicadores, basándose en perfiles previamente definidos de lo que se constituye como un ataque o actividad legítima. Basándose en el análisis estadístico, los métodos de este tipo analizan el comportamiento previo de un evento para poder determinar de manera aproximada el valor que tendrían en futuras repeticiones.

TÉCNICAS DE AGRUPAMIENTO DE DATOS

Al hablar de agrupamiento de datos en necesario hablar de patrones, definidos como lo opuesto al caos, una entidad vagamente definida que puede ser nombrada [25].

Teniendo en cuenta lo anterior, para poder reconocer un patrón se deben tener en cuenta dos aspectos [26]: el primero de ellos es la clasificación supervisada donde se identifica el patrón dentro de una clase predefinida y el segundo es la no supervisada donde se asigna el patrón a una clase aún no conocida.

La clasificación no supervisada o agrupamiento mencionado fue el centro neurálgico del trabajo de investigación realizado y representa un problema complejo ya que los datos pueden originar grupos con diferentes formas y tamaños. Las propiedades que deben ser tenidas en cuenta al momento de agrupar las detalla Jain [27] y en ellas se expresa que para poder llevar a cabo agrupamientos óptimamente se debe tener en cuenta que los patrones dentro de un grupo son más similares entre sí que patrones que pertenecen a grupos diferentes, y que un grupo consiste en una relativa y alta densidad de puntos separados de otros grupos por una relativa y baja densidad de puntos.

Para poder hacer un buen uso del agrupamiento de datos y la implementación de técnicas que usen patrones, es necesario hacer una reducción significativa de la dimensionalidad, ya que esto se traduce en un bajo coste de medición y una mejor precisión en la clasificación. Para esto se debe hacer una reducción en las características que se van a evaluar, obteniendo un clasificador más rápido y con un consumo de memoria menor pero sacrificando la precisión del sistema al reducir el poder de discriminación. En relación con la reducción de la dimensionalidad se aplican algoritmos de selección y extracción de características como PCA, LDA o ICAFAST, formándose así nuevas características con base en transformaciones o combinaciones del conjunto original utilizado. Además de lo anterior, se debe validar el grupo determinando, su cohesión frente a otros elementos y el nivel de aislamiento que tenga con otros patrones que no pertenezcan a él. Validado el grupo se debe estimar el error del sistema de reconocimiento con base en todas aquellas muestras que se tengan, divididas en conjuntos de entrenado y prueba.

Etapas del proceso de agrupamiento

En la Fig. 2 se muestran los pasos que involucran el agrupamiento de datos.

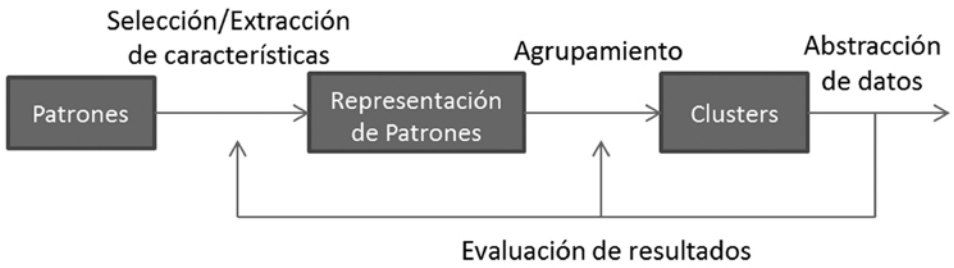

Fig. 2 Etapas del agrupamiento de datos

La representación de patrones es el primer paso en los procesos de agrupamiento. Los patrones muestran las características que utilizarán el o los algoritmos para hacer agrupamientos.

En la mayoría de los casos las características hacen referencia a la dimensión de vectores multidimensionales que tienen correlación directa con dichas características, las cuales pueden ser cualitativas o cuantitativas. Estas características en los patrones pueden ser determinadas mediante la selección y extracción. En la selección se identifica el conjunto de características originales más efectivo para la tarea de agrupamiento, mientras que en la extracción se aplican diversas transformaciones en las características existentes para producir nuevas características. Estos dos métodos son utilizados en la práctica del agrupamiento y selección de datos para mejorar el rendimiento de cómputo al implementar algoritmos de un alto grado de complejidad.

Otro de los puntos a tener en cuenta al momento de hacer agrupamiento de datos es saber definir la medida de proximidad entre los patrones, ya que la métrica con que se midan estos datos, bien sea por similitud o disimilitud entre ellos, dará resultados más óptimos, dependiendo del método escogido.

El agrupamiento, como se aprecia en la Fig. 2, es el punto del proceso donde se forman los grupos propiamente dichos; para este proceso pueden darse métodos probabilísticos, teoría de grafos, lógica difusa, entre otras técnicas. En esta figura se aprecia la abstracción de datos, donde se representan los datos de la manera más sencilla posible para su procesamiento en grupos compactos, por lo general en términos de patrones representativos o prototipos. Al brindar una descripción simple e intuitiva de los grupos, la abstracción de datos es una herramienta útil para el análisis exploratorio.

Por último, se evalúan los resultados obtenidos del proceso completo, en su mayoría de manera subjetiva, teniendo en cuenta el criterio específico de optimización utilizado para realizar el agrupamiento y los pasos anteriormente descritos. Con este análisis se busca determinar si el agrupamiento obtenido es significativo y así probar su validez [28].

Técnicas frecuentes para agrupamientos de datos

Las técnicas de agrupamiento de datos son variadas y entre las más comunes se pueden encontrar aquellas que utilizan las redes neuronales que, con su arquitectura de procesamiento paralela y distribuida, se han

ganado un puesto importante en las investigaciones de hoy día, brindándoles a los investigadores la oportunidad de "entrenar" sus propuestas. Otra técnica muy usada actualmente son los algoritmos evolutivos que con el uso de teoría evolutiva como son los cromosomas, el cruzamiento, la selección, mutación y fitness proveen la capacidad de mejorar la población de una manera considerable, aportando optimización en este tipo de procesos. Se encuentra también el agrupamiento difuso, en el cual se logra asociar cada patrón con todos los grupos usando una función de pertenencia, siendo esta función la clave del éxito en este tipo de métodos. Por último, se puede mencionar la técnica de vecinos cercanos que es una de las más usadas, ya que la proximidad juega un papel importante y sirve de base para procedimientos de agrupamiento. Un patrón usualmente pertenece al mismo grupo que su vecino más cercano, y dos patrones son considerados similares si comparten vecinos [29].

Técnicas de agrupamiento utilizadas en este trabajo

En este trabajo se mostrarán técnicas de clasificación no supervisadas como los Self-Organizing Maps (SOM) y los Growing Hierarchical Self-Organizing Maps (GHSOM). Teniendo en cuenta este tipo de clasificación, se dividirán los métodos en aquellos que producen una serie de particiones anidadas, basadas en un criterio para unir o dividir grupos basados en su similitud, a los cuales les llamaremos *jerárquicos* y los que identifican la partición que optimiza un criterio de agrupamiento, que denominaremos *particionales* [30].

La construcción de los métodos jerárquicos se hace de forma aglomerativa o divisiva. Los primeros implementan la jerarquía de abajo hacia arriba, con el fin de crear un grupo por objeto y posteriormente unirlos gradualmente hasta que todos los objetos pertenezcan al mismo grupo. La construcción que implementa el método divisivo trabaja a la inversa, de arriba a bajo, creando inicialmente un único grupo al que pertenecen todos los objetos, para luego ser dividido gradualmente.

REDES NEURONALES ARTIFICIALES

El valor de las redes neuronales empieza a tomar fuerza en los años ochenta, cuando se dan las primeras publicaciones que ratifican su uso y dan veracidad de sus resultados. Una de las publicaciones más influyentes fue *Neural Networks and Physical Systems With Emergent Properties* [31] donde se dieron las primeras razones formales para justificar el funcionamiento de las redes neuronales y, por lo tanto, dio legitimidad a su uso. Otro de los resultados que dieron validez a las redes neuronales es que problemas combinatorios de optimización, como el problema del agente viajero, fueron formulados y resueltos en términos de una función de energía de una red, la cual se minimiza cuando la red alcanza un estado estable.

De las características más importantes en la dinámica de una red neuronal son las llamadas propiedades emergentes. En palabras de Hopfield [31], "ellas (las propiedades emergentes) nacen de la interacción de un gran número de elementos, entonces son la consecuencia de relaciones microscópicas que parecen tener vida propia". Sin embargo,

ensamblar una colección de neuronas no es suficiente para producir pensamiento.

Actualmente las redes neuronales artificiales se pueden simular en software o implementarlas en hardware. La implementación en hardware en su totalidad y de manera eficaz y eficiente es hoy día el gran reto, ya que es la única manera de aprovechar al máximo todas las capacidades de las RNA (Redes Neuronales Artificiales). Al mismo tiempo, miles de investigadores de diversos campos, tales como neurociencias, psicología, medicina, matemáticas, física e ingeniería, han resuelto un número importante de problemas del mundo real utilizando redes neuronales.

Arquitectura de las redes neuronales

Las RNA pueden ser divididas en dos grandes categorías: las redes "hacia-delante" (feedforward), en las cuales no existen ciclos; y las redes de retroalimentación (feedback), en las cuales sí existen ciclos y por lo tanto la red se retroalimenta.

Las redes hacia-adelante son redes con varias capas donde solo existe conexión entre capas consecutivas y no tienen memoria ya que la respuesta de una de estas redes a un dato de entrada dado, es independiente de los estados previos de la red; mientras que las redes de retroalimentación son sistemas dinámicos donde por cada dato de entrada, las respuestas de las neuronas son computadas por medio de las conexiones de retroalimentación, ocasionando una modificación en los vectores de pesos de las neuronas. De esta manera, la red se encuentra en un nuevo estado, repitiendo el proceso hasta alcanzar algún tipo de equilibrio o convergencia.

El aprendizaje de las neuronas artificiales

Toda red neuronal debe tener la capacidad de aprender, independientemente de la arquitectura que la rija. El concepto de aprendizaje se relaciona con actualización de los pesos sinápticos de las neuronas con el fin específico de mejorar la forma en que realiza cierta tarea concreta. Para poder hacer efectivo este aprendizaje, se tiene en cuenta el llamado vector de pesos, vector que representa la actualización iterativa de los pesos sinápticos. Con este proceso de aprendizaje la red neuronal alcanza su desempeño óptimo mediante la modificación de los vectores de pesos y es precisamente por esta razón que se considera a las redes neuronales sistemas adaptables.

La actualización de los pesos en las redes neuronales se puede hacer de varias formas, algunas de ellas utilizan información previamente dada mientras que otras deben aprender a establecer los pesos correctos a partir del conjunto de datos.

Aplicando lo anterior, se puede decir que existen tres formas en las que una red neuronal puede aprender. La primera de ellas es mediante el aprendizaje supervisado; en este tipo de aprendizaje se provee a la red de las respuestas correctas para cada elemento del conjunto de entrenamiento. En la segunda se implementa aprendizaje no supervisado, donde no se requiere de ninguna respuesta correcta ya que es la red la encargada de explorar los datos y buscar correlación entre los posibles patrones que puedan existir. Por último, se encuentra un aprendizaje híbrido en el que se programa la neurona para que aprenda con una porción de los datos utilizando aprendizaje su-

EDUCOSTA
EDITORIAL UNIVERSITARIA DE LA COSTA

pervisado, mientras que los pesos restantes son obtenidos de manera no supervisada.

Redes neuronales auto-organizables

Las redes más utilizadas en tareas de agrupamiento de datos son aquellas que hacen uso del aprendizaje competitivo, siendo los Mapas Auto-organizables de Kohonen (SOM) [32] y las redes Adaptative Resonance Theory (ART) [33] (y las derivaciones de ambos) los modelos más difundidos.

Como ya se mencionó, en el trabajo realizado para esta investigación se implementaron las redes Self-Organizing Maps (SOM) [34] y la mejora a estas denominada Growing Hierarchical Self-Organizing Maps (HGSOM) [35] explicada en detalle más adelante.

Aparte de estas redes, existen otras técnicas que implementan modelos constructivos como son Growing Neural Gas [36], Incremental Grid-Growing [37], Growing Self-Organizing Maps [38], Density Based Growing Neural Gas [39].

SOM (SELF-ORGANIZING MAPS)

Estudios científicos han evidenciado que hay neuronas en el cerebro que se organizan en muchas zonas de forma tal que las informaciones captadas del entorno a través de los órganos sensoriales se representan internamente en forma de mapas bidimensionales, y aunque esta organización neuronal está predeterminada genéticamente, algunos autores coinciden en que parte de ella se origina mediante el aprendizaje, sugiriendo que el cerebro lograría tener la posibilidad de formar mapas topológicos de las informaciones recibidas del exterior. Es precisamente en estas ideas en las que se basa T. Kohonen para proponer un modelo de red neuronal con capacidad de formar mapas de características de manera similar como ocurre en el cerebro, donde su objetivo primordial fue demostrar que un estímulo externo o datos de entrada por sí solos, con estructura propia y descripción funcional del comportamiento de la red, eran suficientes para formar mapas topológicos de semejanza con los creados por el cerebro.

Funcionamiento de SOM

Una red SOM, como mapa auto-organizativo, trata de establecer una correlación entre los datos de entrada y un espacio bidimensional de salida, creando mapas topológicos de dos dimensiones, con miras a que si existen datos de entrada con características comunes, se activen neuronas situadas en zonas próximas de la capa de salida. Una representación visual de este procedimiento se muestra en la Fig. 3, arquitectura típica de un mapa auto-organizado, donde X cantidad de neuronas de salida se disponen en forma bidimensional para representar los mapas de características.

Se puede afirmar, por lo tanto, que el algoritmo SOM para la generación de mapas define una función del espacio de entrada a una red de neuronas en el plano, donde dicha función define una proyección del conjunto de datos multidimensionales (invisible) a un espacio visible (normalmente bidimensional). Este "espacio visible" del conjunto de datos permite que las relaciones de semejanza que se presentan entre los datos dentro del espacio multidimensional

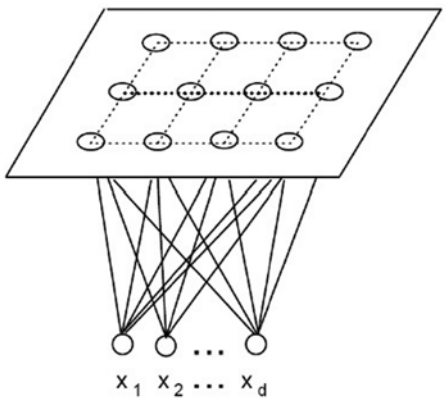

Fig. 3 Arquitectura típica de un mapa auto-organizativo [40]

puedan ser observadas en un despliegue bidimensional denominado "mapa". De ahí el término mapa para describir este tipo de redes.

Esta visualización del conjunto de datos es generada de forma automática y sin intervención humana, lo que categoriza al algoritmo SOM como un algoritmo de redes neuronales de aprendizaje no supervisado, de los que se habló en apartados anteriores y que promueve su evolución durante su entrenamiento, de manera que cada unidad de salida será sensible a reconocer y organizar porciones específicas en el espacio de entrada, acción que se realiza sin ningún tipo de retroalimentación con el medio externo y sin la utilización de información a priori. Este modelo es uno de los más populares que se utilizan en redes neuronales artificiales y pertenece a la categoría de redes con aprendizaje competitivo donde las células reciben de manera idéntica la información de entrada sobre la cual compiten, siendo esta competencia una lucha por determinar cuál de las neuronas es la que mejor representa a un estímulo de entrada dado. Como resultado de esta competencia solo una neurona es activada en cada momento, como se muestra en la Fig. 4.

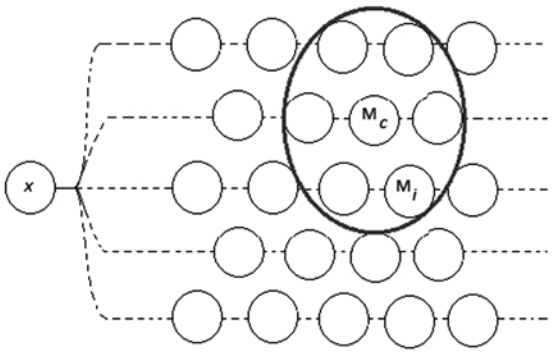

Fig. 4 Neurona ganadora = Mc y Neurona vecina = Mi [41]

El algoritmo de SOM

El algoritmo de SOM tiene un funcionamiento específico, dado por un algoritmo bien definido. A continuación se explica su funcionamiento:

- El SOM es un vector bidimensional de neuronas como sigue:

$$M = \{m_1,\ldots,m_{p\,x\,p}\}$$

- Una neurona es un vector que se le llama patrón y que se puede representar así:

$$M_i = \{m_{i1},\ldots,m_{in}\}$$

Lo anterior muestra que la neurona tiene las mismas dimensiones que los vectores de entrada por lo que se dice que es n-dimensional. La relación que existe entre una neurona y sus respectivas vecinas se da por una relación de vecinos la cual propone la topología o estructura del mapa. En la Fig. 5 se puede apreciar claramente las dos topo-

logías posibles usadas en SOM: la hexagonal y la rectangular.

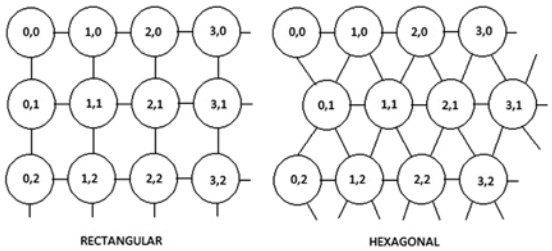

Fig. 5 Posibles topologías de SOM [42]

En el algoritmo SOM básico, las relaciones topológicas entre los nodos (hexagonal o rectangular) y el número K de neuronas se fijan desde el inicio, definiendo la escala o la granularidad del modelo resultante y las distancias entre las unidades del mapa, de acuerdo con la distancia euclidiana entre los vectores de localización; sin embargo, en ocasiones es más práctico usar otras funciones de distancia.

En la "ejecución" del algoritmo de SOM existen varias etapas como son: el pre-procesamiento de datos, la inicialización, el entrenamiento, la visualización y la validación. El pre-procesamiento de datos consiste en la etapa de alimentación de datos al sistema SOM. Estos datos son extraídos de la red y se busca, en lo posible, que los datos sean de calidad ya que si se cuenta con datos erróneos de entrada se corre el riesgo de que los resultados cumplan esa misma tendencia. Esto implica una etapa de eliminación de información no óptima para el procesamiento, por lo que se hace un análisis en primera instancia de un subconjunto de datos relevantes del modelo en cuestión. En la gran mayoría de los casos los datos de entrada para hacer este pre-procesamiento deben normalizarse para tener una escala de 0 a 1 y poder tomar a la distancia como una medida posible de similitud.

El segundo punto que se debe tener en cuenta en el algoritmo es la inicialización, que consiste en el uso de valores al azar o de las primeras muestras para poder darles vida a las neuronas (patrones). Si se usa una inicialización al azar se asignan valores aleatorios y es utilizada comúnmente cuando no se tiene la información suficiente sobre los datos de entrada en el inicio del entrenamiento. La segunda forma de inicializar es la de las primeras muestras, donde se utilizan los primeros datos de entrada asignándolos a los patrones, con la ventaja de que automáticamente son ubicados en la parte correspondiente del espacio de entrada.

Acto seguido se debe hacer un entrenamiento, siendo este un proceso iterativo en el tiempo donde se van ingresando a la red muestras del conjunto de datos de entrada para que ella misma las "aprenda". Este proceso suele ser lento y con un coste computacional alto ya que la razón del verdadero aprendizaje consiste en elegir una neurona ganadora utilizando el concepto de "medida de similitud" con el que se actualizan los valores de los patrones de los vecinos. Como el proceso es iterativo, se repite hasta acotar el error y acercar a las neuronas a una representación óptima de los datos de entrada.

La representación matemática del proceso planteado de entrenamiento se detalla a continuación:

En cada momento t del proceso de entrenamiento, un vector de entrada $x(t) \in \mathbb{R}^n$ es conectado a todas las neuronas en paralelo vía los vectores de referencia ω_i de cada

neurona. Se produce una competencia de las neuronas para saber cuál de ellas es capaz de representar de mejor manera al dato de entrada *x(t)*.

La competencia consiste en que dado cualquier $x \epsilon X$ encontrar una neurona tal que su vector de referencia ω_c cumpla con:

$$\|x - \omega_c\| = \min_{i=1} N \{\|x - \omega_i\|\} \quad (1)$$

a la neurona ganadora η_c se le define como el nodo que mejor representa al dato *x* o *Best Matching Unit (BMU)*. Nótese que el subíndice *c* es función de *x*; para cada *x* existe un $\eta_c(x)$.

Si se presentara el punto en que este índice no esté bien definido; es decir, cuando para un dato *x* existan dos $\eta_e, \eta_d \epsilon N$ tal que:

$$d(x,\eta_e) = mín\ \{d(x,\eta_i)!\ i = 1, \ldots, k\} = d(x,\eta_d),$$

la selección de un único *c(x)* debe hacerse de manera aleatoria. Se tiene, entonces, la siguiente notación:

$$x \sim \eta \iff \eta = \eta_{c(x)}$$

Para cada tiempo *t* se realiza la ecuación (1) de manera que se puede definir *c = c(t)* tal que $x(t) \sim \eta_{c(t)}$, las neuronas que se encuentran dentro de una vecindad de $\eta_{c(t)}$ en el arreglo bidimensional aprenderán de la misma entrada *x(t)* (Fig. 6).

La vecindad de $\eta_{c(t)}$ sobre la red se define a partir del vector de localización $r_{c(t)}$ de la siguiente manera:

$$N_{c(t)} = \{i \in N \mid \|r_{c(t)} - r_i\| \leq \rho(t)\} \quad (2)$$

donde $\rho(t)$ es el radio de la vecindad en el tiempo *t*.

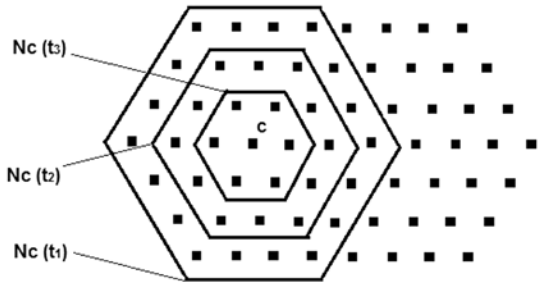

Fig. 6 Variación en el tiempo del radio de la vecindad, t1<t2<t3 [41]

Como se observa en (2), el radio de la vecindad varía en función de *t*. Para efectos de la convergencia del algoritmo, la variación del radio a través del tiempo debe cumplir las condiciones (1) y (2) siguientes (Fig. 6).

1. Si $t_i \leq t_j \Rightarrow \rho(t_i) \geq \rho(t_j)$
2. $\rho(t) \to 0$ cuando $t \to \alpha$

En este proceso hay que ser muy cautelosos cuando se le da el tamaño inicial a $\rho(0)$; si desde un comienzo la vecindad es muy pequeña, el mapa no se ordenará globalmente, lo cual implicará que el mapa generado se verá como un mosaico de partes entre las cuales el ordenamiento cambia discontinuamente. Para evitar esto se recomienda que $\rho(0)$ empiece siendo más grande que la mitad del diámetro de la red.

Para el proceso de aprendizaje inicial se utilizan valores aleatorios para los vectores de referencia $w_i(0)$. En las versiones más simples del SOM los valores sucesivos para los vectores de referencia se determinan recursivamente por el siguiente mapeo de iteraciones:

$$w_i(t+1) = w_i(t) + h_{ci}(t)\,[x(t) - w_i(t)]$$

La función $h_{ci}(t)$ desempeña un papel fundamental en este proceso. A esta función se le conoce como función de vecindad, y tiene la siguiente forma:

$$h_{ci}(t) = h\left(\left\|r_{c(t)} - r_i\right\|, t\right) \qquad (3)$$

lo cual implica que el valor de la función depende de la distancia entre la neurona η_i y la neurona ganadora $\eta_{c(t)}$ en el tiempo *t*.

El ancho promedio *ρ(t)* y la forma de $h_{ci}(t)$ definen la rigidez del mapa que será asociada a los datos. Independientemente de cuál sea la forma explícita de la función (3), debe ser tal que $h_{ci}(t) \rightarrow 0$ mientras $\left\|r_{c(t)} - r_i\right\|$ se incrementa.

Una de las definiciones más simples que se encuentran de la función vecindad es la siguiente:

$$h_{ci}(t) = \alpha(t), \text{ si } i \in N_c(t)$$
$$h_{ci}(t) = 0 \quad , \text{si } i \notin N_c(t)$$

el valor de *α(t)* se define como factor de aprendizaje, el cual cumple con la condición $0 < \alpha(t) < 1$ y usualmente *α(t)* es una función monótona decreciente.

Retomando las etapas en la ejecución del algoritmo SOM, viene ahora la cuarta, que se constituye en la visualización. El SOM puede representarse de una manera visual como una aproximación de la función de densidad de probabilidad de los datos de entrada [43]. La representación U-Matrix (Unified Distance Matrix) del SOM visualiza la distancia entre neuronas adyacentes, distancia que es calculada y representada con diferentes colores entre los nodos adyacentes. Un color oscuro representa una distancia grande mientras que uno de baja intensidad significa que los patrones están cerca unos de otros. Lo anterior infiere que en los resultados obtenidos en la visualización se pueda hablar de "clases" o áreas claras y "separadores" o áreas oscuras, siendo esta manera de representación de gran ayuda para "visualizar" los datos de entrada sin tener información a priori sobre las clases. La Fig. 7 ilustra el resultado de la etapa de visualización.

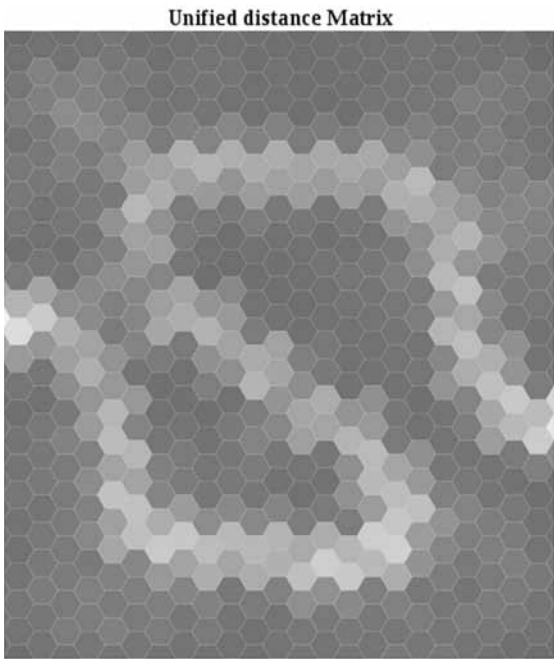

Fig. 7 U-Matrix [44]

Por último encontramos la etapa de validación, donde se prueba el modelo que se va a utilizar para tener la certeza de que los datos de salida serán razonables y ciertos. En la mayoría de los casos esta etapa se hace usando un conjunto independiente de datos; muy parecido al entrenamiento pero no parte de él, siendo esta porción como un caso representativo del caso general.

GHSOM (GROWING HIERARCHICAL SELF-ORGANIZING MAPS)

Los Mapas Auto-organizativos Jerárquicos, de las siglas en inglés GHSOM [34] (Growing Hierarchical Self Organizing Maps) se proponen como una extensión de los mapas auto-organizativos SOM y HSOM [45] con los dos siguientes propósitos:

1. SOM tiene una arquitectura de red fija, es decir el número de unidades de uso, así como la distribución de las unidades tiene que ser determinada antes del entrenamiento.
2. Los datos de entrada que son de naturaleza jerárquica deberían estar representados en una estructura jerárquica para mayor claridad de la representación. GHSOM utiliza una estructura jerárquica de varias capas, donde cada capa está formada por un número de SOM independiente. Solo un SOM se utiliza en la primera capa de la jerarquía.

Por cada unidad del mapa una SOM podría añadirse a la siguiente capa de la jerarquía. Este principio se repite con el tercer nivel del mapa y las demás capas de la GHSOM. Este proceso se describe en la Fig. 8.

El proceso de inserción de columnas o filas se rige por los siguientes pasos:

a. Los pesos de cada unidad se inicializan con valores aleatorios.
b. Se aplica el algoritmo estándar de SOM.
c. La unidad con la mayor desviación entre el vector de pesos y los vectores de entrada es elegida para representar la unidad de error.

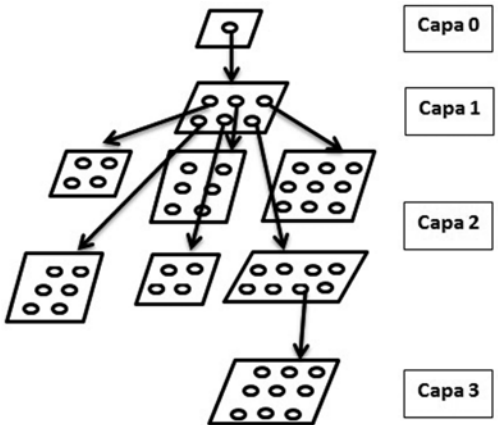

Fig. 8 Arquitectura de GHSOM [46]

d. Una fila o una columna se inserta entre la unidad de error y la unidad vecina más distinta en términos de espacio de entrada.
e. Los pasos b al d se repiten hasta que el Error de Cuantificación Medio (MQE) alcanza un determinado umbral, una fracción del error de cuantificación promedio de la Unidad i, en la capa de procedimiento de la jerarquía.

La representación gráfica de lo anterior se muestra en la Fig. 9. Allí se aprecia el Estado A, este es el diseño de SOM antes de la inserción de nuevas unidades. Donde "e" es la unidad de error y "d" es el vecino más distinto. En Estado B se muestra el diseño de SOM después de insertar una fila entre "e" y "d".

En cuanto a la profundización de la jerarquía de GHSOM, la idea general es mantener el control si el nivel más bajo de SOM ha logrado una cobertura suficiente para la entrada de datos subyacente.

Los pasos con más detalles son: comprobar el error medio de cuantificación de cada

EDUCOSTA
EDITORIAL UNIVERSITARIA DE LA COSTA

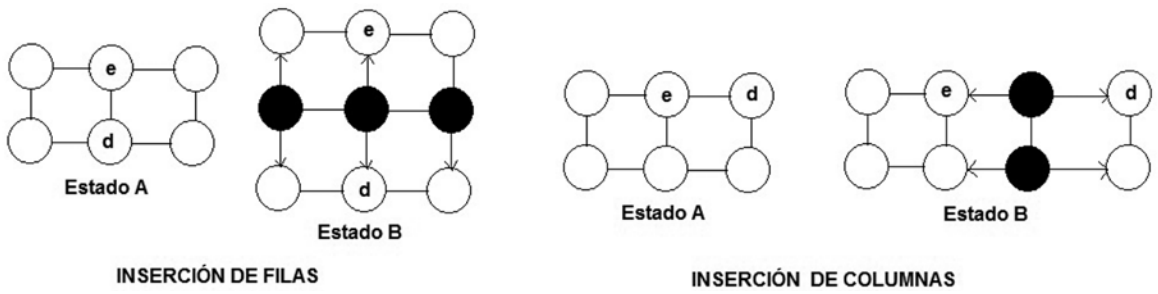

Fig. 9 Inserción de unidades en arquitectura GHSOM [34]

unidad para asegurarse de que está por encima de cierto umbral dado: indicando el nivel de granularidad deseado de una representación de datos como una fracción del error de cuantificación inicial en la capa 0.

En segundo lugar se asigna una capa SOM para cada unidad con un error medio de cuantificación superior a los umbrales indicados, y entrenar el SOM con los vectores de entrada asignados a esta unidad.

GHSOM ofrece una manera conveniente de organizar por sí mismo los datos jerárquicamente en capas y ofrece a los usuarios la posibilidad de elegir el nivel de detalle de la representación en los diferentes niveles de la jerarquía. Por otra parte, el algoritmo GHSOM determinará automáticamente la arquitectura de la SOM en diferentes niveles.

Los inconvenientes de este modelo incluyen la fuerte dependencia de los resultados en un número de parámetros que no son ajustados automáticamente. Umbrales altos por lo general dan como resultado una GHSOM plana con grandes SOM individuales, mientras que los bajos se traducen en jerarquías profundas con mapas más pequeños.

El algoritmo GHSOM

La etapa de entrenamiento del algoritmo de GHSOM empieza con la configuración inicial; en este paso se crea el "mapa" en el nivel 0 con una sola unidad. El vector m_0 de pesos de esta unidad es inicializado, la media de todos los vectores de entrada y también se calcula el *error medio de cuantificación* mqe_0.

Básicamente, el *error medio de cuantificación* mqe_i de una unidad *i* es la desviación entre el vector de pesos y los vectores de entrada asignados a esta unidad. Se calcula como el promedio de la distancia euclidiana entre el vector de pesos *mi* y los vectores de entrada *xj* que son elementos del conjunto de vectores de entrada *Ci* que se asignan a esta unidad *i*, como se ve en la ecuación (4), con | · | denotando la cardinalidad de un conjunto.

$$mqe_i = \frac{1}{n_{c_i}} \cdot \sum_{x_j \in c_i} \left\| m_i - x_j \right\| , n_{c_i} = |c_i| \quad (4)$$

Específicamente, el *error medio de cuantificación* mqe_0 de la unidad en la capa 0 se calcula como se detalla en la expresión (5), donde n_I es el número de todos los vectores entrada *x* del conjunto de datos de entrada *I*.

$$mqe_0 = \frac{1}{n_I} \cdot \sum_{x_j \in I} \|m_0 - x_j\| \, , n_I = |I| \quad (5)$$

El valor de mqe_0 puede ser considerado como una medida de la diferencia de los datos de entrada. Esta medida tendrá un papel fundamental durante el proceso de crecimiento de la red neuronal.

Seguidamente a este procedimiento de inicialización, viene el proceso de formación y crecimiento del mapa. Este nuevo proceso se inicia creando un nuevo SOM debajo de la capa 0 del mapa con un tamaño inicial de 2x2 unidades. Esta primera capa del mapa se entrena con el procedimiento estándar de SOM. Después de un número fijo λ de iteraciones de entrenamiento el *mqe* de todas las unidades es analizado. Un alto *mqe* muestra que para esta unidad en particular del espacio de entrada no se representa con suficiente precisión. Por lo tanto, se necesitarán nuevas unidades para incrementar la calidad de la representación del espacio de entrada. La unidad con el *mqe* más alto se seleccionará entonces como la unidad de error *e*. Una nueva fila o columna de unidades se inserta en medio de esta unidad de error y su vecino más opuesto. Los vectores de peso de las nuevas unidades se inicializan con el promedio de sus vecinos correspondientes.

El crecimiento de un mapa puede ser descrito matemáticamente de la siguiente manera: si C_i es el subconjunto de vectores x_j de los datos de entrada que se asigna a la unidad *i*, es decir, $C_i \subseteq I$, y m_i el vector de peso de la unidad *i*, entonces *la unidad de error e* se calcula como la unidad con *error medio de cuantificación* máximo:

$$e = \arg\max_i \left\{ \frac{1}{n_{ci}} \cdot \sum_{x_j \in C_i} \|m_i - x_j\| \right\}, \quad n_{ci} = |C_i| \quad (6)$$

La selección de sus *d* vecinos más disímiles se determina por la distancia máxima entre el vector de pesos de la unidad *e*, y los vectores de peso de las unidades vecinas. Una fila completa o columna de unidades se inserta entre *d* y *e,* como se mostró en la Fig. 11, donde las flechas indican la respectiva unidad vecina para inicializar el vector de pesos.

El proceso de crecimiento continúa hasta que el *error medio de cuantificación*, conocido como MQE, en mayúsculas, alcanza una cierta fracción τ_1 del mqe_u de la unidad correspondiente en la capa superior; es decir, la unidad que constituye la capa 0 del mapa para la primera capa de mapa. El MQE de un mapa se calcula como la media de todos los errores medios de cuantificación mqe_i (ver la expresión (4)) del subconjunto *U* del mapa de las unidades de los datos asignados:

$$MQE_m = \frac{1}{n_U} \cdot \sum_{i \in U} mqe_i \, , n_U = |U| \quad (7)$$

El criterio de parada para un mapa *m* en su etapa de entrenamiento está definido por:

$$MQE_m < \tau_1 \cdot mqe_u \quad (8)$$

donde mqe_u es el error medio de cuantificación de la unidad *u*, correspondiente a la capa superior. En efecto, cuanto menor sea el parámetro τ_1 que se elija, mayor será el tiempo de entrenamiento, y mayor el mapa resultante. Para el caso de la primera capa del mapa el criterio de parada es $MQE_1 < \tau_1 \cdot mqe_0$. El parámetro τ_1, por lo tanto,

sirve como medida de control para el tamaño final de cada mapa para definir el grado en que cada uno de ellos tiene que representar la información asignada a la unidad basada en un mayor detalle.

Cuando el entrenamiento del mapa termina, cada unidad tiene que ser revisada para su expansión, es decir, debe ser más "refinada" en un mapa en la siguiente capa. Esto significa que para las unidades que representan un conjunto de diversos vectores de entrada debe crearse un nuevo mapa en la siguiente capa. El límite para la expansión está determinado por un segundo parámetro τ_2, que define la granularidad de representación de datos, requisito que debe cumplirse por cada unidad. Por lo tanto, se constituye el criterio global de parada con un mínimo de calidad de representación de datos para todas las unidades como una fracción de la disparidad de todos los datos de entrada descritos por mqe_0.

$$mqe_i < \tau_2 \cdot mqe_0 \quad (9)$$

Si se presentara el caso de que (8) es falsa para la unidad i, es decir, mqe_i sea mayor o igual a $\tau_2 \cdot mqe_0$, se debe entonces crear un mapa pequeño en la siguiente capa, pero si el criterio de parada se cumple, no se requieren más expansiones. En este punto hay que tener en cuenta que, a diferencia del criterio de parada para el crecimiento horizontal del mapa determinada por τ_1 y el *mqe* de la unidad de la capa superior, este segundo criterio se basa únicamente en el mqe_0, es decir, el *mqe* de la capa de *0*, por cada unidad en todos los mapas.

En consecuencia, con la etapa de entrenamiento que tiene GHSOM, los vectores de entrada para realizar este proceso en el nuevo mapa son los asignados a la unidad que acaba de ser ampliada. Este nuevo mapa seguirá creciendo de forma indefinida hasta que cumpla con algunos de los criterios de parada descritos anteriormente. Todo el proceso se repite para las capas subsiguientes hasta que el criterio fijado en la expresión (8) se cumpla en todas las unidades en las capas más bajas.

El parámetro τ_2 define la calidad mínima de representación de todas las unidades en la capa más baja de cada ramificación, lo que garantiza que la calidad de la representación de datos cumple con un criterio mínimo para todas las partes del espacio de entrada. Con GHSOM se proporciona automáticamente el número necesario de unidades en las áreas respectivas.

Es necesario dejar claro que los parámetros τ_1 y τ_2 en el proceso de crecimiento de GHSOM merecen una importante consideración ya que son la base para la creación de los nuevos mapas.

El parámetro τ_1 controla el proceso de crecimiento real de la GHSOM. Básicamente, la jerarquía de datos puede ser representada de diferentes maneras, siendo el primer caso aquel que tiene jerarquías inferiores con detalladas mejoras que la capa subsiguiente, o el de las jerarquías más profundas, que proporcionan una separación estricta de los diferentes sub-grupos mediante la asignación de mapas por separado.

En el primer caso se prefiere a mapas más grandes en cada capa, lo que explica una representación plana de la mayor parte de los datos, permitiendo una estructura menos je-

rárquica. En el segundo caso, sin embargo, donde por lo general se prefieren los mapas más pequeños, cada uno de los cuales describe solo una pequeña parte de las características de los datos, con más énfasis en la detección y representación de la estructura jerárquica.

Por lo tanto, cuanto menor sea el parámetro τ_1, mayor será el grado de explicación de los datos en un solo mapa. Esto se traduce en mapas más grandes como el mapa del error medio de cuantificación (*MQE*) que será inferior a las unidades que están disponibles para la representación de los datos. Si τ_1 se establece en un valor bastante alto, no es necesario que el *MQE* quede muy por debajo de la *mqe* de la unidad que se basa en la capa superior. Así, un mapa más pequeño va a satisfacer el criterio de parada para el proceso de crecimiento horizontal, que requiere la representación más detallada de los datos que se realiza en las capas posteriores.

A diferencia del parámetro τ_1, el τ_2 se encarga de controlar el nivel de detalle mínimo de representación de datos, es decir, ninguna unidad puede representar datos con una granularidad más gruesa de la posible. Si los datos asignados a una sola unidad aún tiene una mayor variación, entonces ya sea una nueva fila o columna se añadirán al mismo mapa durante el proceso de crecimiento horizontal para hacer más unidades disponibles para la representación de los datos, o en su defecto, un nuevo mapa será añadido procedente de esta unidad, representando con más detalle los datos de esta unidad en una capa posterior.

Esta granularidad absoluta de representación de datos por lo general se especifica como una fracción de la diferencia inherente a la recolección de datos como tal, que se expresa en el error medio de cuantificación de la unidad en la capa 0.

En conclusión, se puede decir que cuanto menor es el valor del parámetro τ_1, más plana la jerarquía, y cuanto menor sea el valor del parámetro τ_2, mayor será el número de unidades resultante de la red GHSOM.

IMPLEMENTACIÓN DE GHSOM

Las características para la implementación de IDS o IPS del tráfico capturado en este trabajo, son las utilizadas por el conjunto de datos NSL-KDD [47]. Estas características se dividen en tres clases debido a que la detección e identificación de algunos de los ataques requiere el uso de más que una clase; estas características son: las básicas, las basadas en contenido y las características del tráfico.

El primer paso realizado en esta investigación fue analizar el conjunto de datos mencionado con el fin de extraer las características de los archivos de texto y la construcción de los vectores que forman el espacio de características.

Este espacio de características se compone de vectores pertenecientes a $\mathbb{R}^{41}$ y contienen las características de tráfico de conexión. Entre las características tomadas en cuenta se mencionan la duración de la conexión, el tipo de protocolo, o el número de intentos de user-to-root. Se deja claro que la selección de características basadas en red en los IDS no es tarea sencilla y que se pueden utilizar técnicas multivariantes [48]

EDUCOSTA
EDITORIAL UNIVERSITARIA DE LA COSTA

para tal acción, como es el caso de PCA [49] o LDA [50]. Aunque el uso de estas técnicas es relativamente bueno, en otros trabajos se ha demostrado que el conjunto completo de características supera a la función de selección con métodos como ACC/LDA [49]. En este trabajo se usó una estructura dinámica, como GHSOM y el conjunto completo de características.

Para poder aplicar GHSOM en IDS, es necesario codificar algunas de las características cualitativas como el protocolo, el servicio, la bandera y otras incluidas en el conjunto de datos NSL-KDD debido a la naturaleza numérica de los vectores de entrada de GHSOM. La codificación de estas características se realiza mediante la asignación de valores numéricos con el fin de mantener una distancia lo suficientemente alta entre ellos para la eficacia del clasificador SOM. Se han elegido el número 1 y otros números primos con distancias superiores a seis. Este código ha sido elegido con el fin de aumentar la distancia entre las diferentes características. Por ejemplo, en el protocolo de los componentes de los vectores de entrada, el TCP se codifica como 1, UDP como 7, e ICMP como 17. Los vectores de entrada se han normalizado con el único fin de evitar que algunas características tengan más influencia que otras, restando la media y dividiendo por la desviación, con lo que queda cada dimensión con valores entre 0 y 1, y el espacio de características pertenece a $\mathbb{R}^{41}$.

Cálculo de la BMU en GHSOM

Teniendo claras las características que se deben usar, se realizó el cálculo de la BMU o unidad ganadora utilizado un algoritmo iterativo, como lo muestra la Fig. 10, el cual recorre toda la jerarquía para determinar la unidad ganadora y el mapa a que pertenece. Después de calcular las distancias entre un patrón de entrada y los vectores de peso del mapa en la capa 0, se determina el mínimo de estas distancias. Una vez la neurona ganadora se encuentra en el mapa 1, se podría crecer desde dicha neurona, por lo que tenemos que comprobar si la neurona ganadora es una unidad padre. Esto se puede lograr con los vectores de los padres resultantes del proceso de entrenamiento del GHSOM. Si un nuevo mapa surge de la neurona ganadora, se calcula la BMU en este nuevo mapa. Este proceso se repite hasta llegar a un mapa donde la BMU no pueda crecer. Por lo tanto, la BMU en el GHSOM se identifica dentro de un mapa en una capa de la jerarquía como es el caso del mapa 5 y la unidad 4 (Fig. 10).

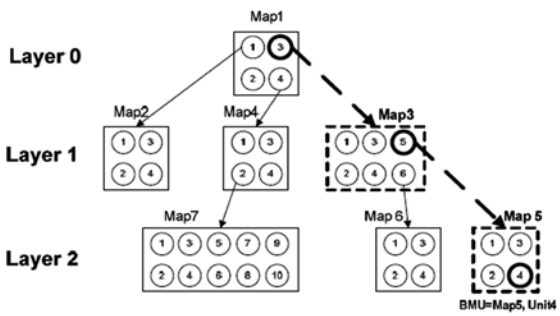

Fig. 10 Iteraciones para encontrar la BMU en GHSOM [51]

Entrenamiento y reetiquetado del GHSOM

El conjunto de datos NSLKDD cuenta con varios archivos para realizar pruebas, en nuestro caso se utilizó el "KDDTrain+_20Percent.TXT", para el en-

trenamiento de la estructura GHSOM. Este conjunto de datos contiene el 20% del total de todos los datos, siendo muestras únicas e irrepetibles de los demás conjuntos. Después del entrenamiento se usó *validación cruzada* para la obtención de los resultados probado el sistema con el 100% de los patrones de entrenamiento, utilizando el conjunto de datos "KDDTrain+.TXT". Como se comentó anteriormente, se utilizó el conjunto completo de características. Después de varias pruebas con diferentes valores de τ_1 para controlar la amplitud del mapa, y τ_2 para el control de la profundidad, se han seleccionado $\tau_1 = 0.6$ y $\tau_2 = 10^{-5}$. Esto podría hacer que el GHSOM crezca más de lo necesario, dejando algunas de las unidades sin etiquetar, lo que conlleva a que el número de neuronas en el mapa de salida alrededor de la neurona ganadora de datos de entrenamiento se incremente.

Cuando un patrón de entrada similar a uno de los patrones de entrenamiento se presenta a la GHSOM, la neurona ganadora puede etiquetarse o no. Si la neurona ganadora no es etiquetada, se utiliza un esquema basado en probabilidad para determinar la etiqueta de esa neurona. Más específicamente, la neurona ganadora es etiquetada o reetiquetada con la etiqueta más repetida en sus vecinos mediante el uso de una probabilidad calculada de acuerdo con:

$$P_u = \frac{M_{\sigma(u,\varepsilon)}(u)}{n} \qquad (10)$$

En esta expresión, P_u es la probabilidad de que la unidad ganadora *u* sea reetiquetada con éxito, donde $M_{\sigma(u,\varepsilon)}(u)$ es la etiqueta que aparece con más frecuencia en la vecindad gausiana $\sigma(u,\varepsilon)$, de la neurona ganadora *u*, y *n* es el número de neuronas que pertenecen a la vecindad de *u*, $\sigma(u,\varepsilon)$. En esta ecuación, el parámetro ε indica la anchura de la vecindad de la neurona ganadora.

En la Fig. 11 se muestra un ejemplo del proceso de reetiquetado. En esta figura, la BMU que no ha sido inicialmente etiquetada, se etiqueta ε=1 para establecer su vecindad. Para este ejemplo en particular se nota una vecindad con cuatro unidades etiquetadas con L1, una unidad con L2 y una unidad con L3. Entonces la probabilidad P_u de un reetiquetado exitoso para esta BMU será de 4/6 = 0,66 (66%) ($M_{\sigma(u,\varepsilon)}(u)$) = 4, n = 6).

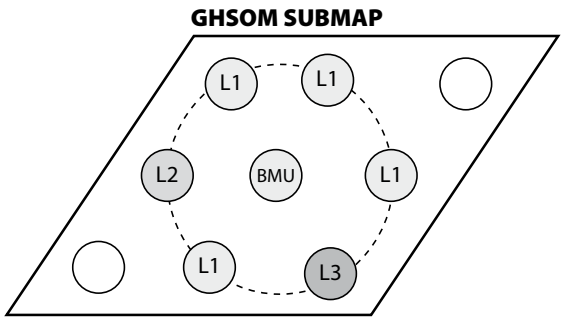

Fig. 11 Ejemplo del proceso de reetiquetado (unidades blancas sin etiqueta) [51]

Resultados del experimento

Para el experimento se utilizó el software matemático MATLAB, el clasificador GHSOM y el conjunto de datos NSL-KDD [47]. Los resultados obtenidos se muestran en la Fig. 12 donde se logra apreciar el éxito de detección para cada tipo de ataque incluido en el conjunto de datos NSL-KDD. La figura ilustra que el proceso de etiquetado realizado, basado en la probabilidad,

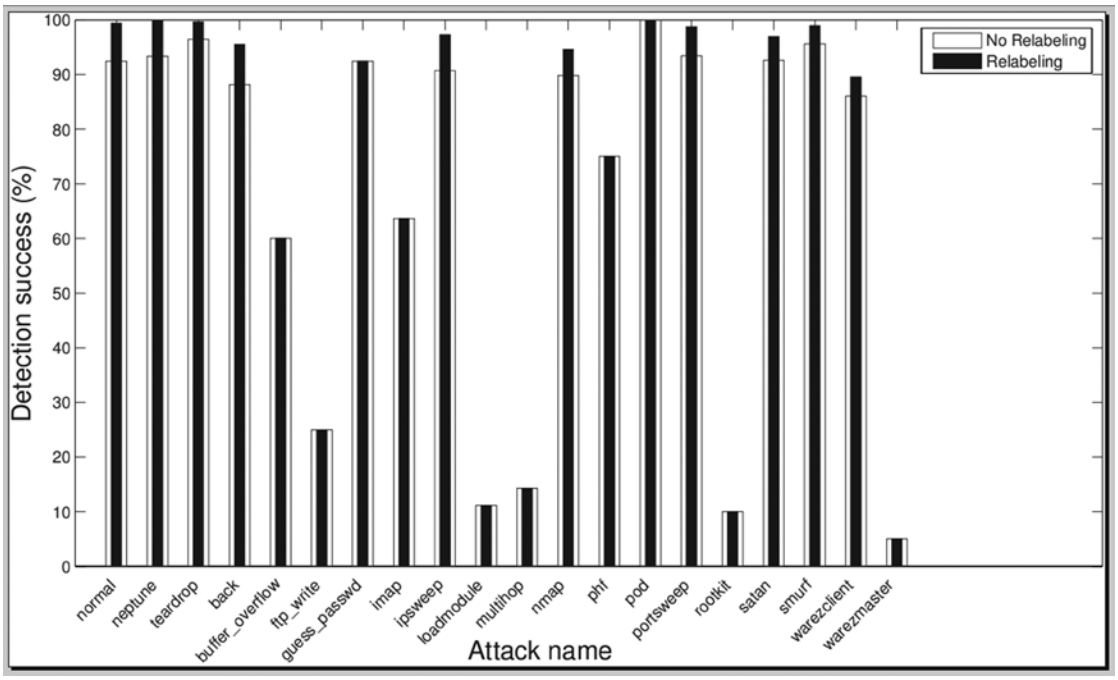

Fig. 12 Detección para cada tipo de ataque incluido en el conjunto de datos NSL-KDD

aumenta el éxito de detección para la mayoría de los ataques aunque algunos de ellos como el de multisaltos (multihop), no son detectados antes del proceso de reetiquetado de la unidad.

En la Fig. 13 se presenta la tasa de éxito de detección por tipo de ataque; como puede observarse en todos los casos, el método de reetiquetado aumenta el rendimiento de clasificación, así como la tasa de detección de éxito. En cuanto a los ataques User to Root (U2R), el rendimiento es peor que el obtenido para otros ataques, aunque se debe dejar claro que los patrones de entrenamiento de este tipo de ataque en el NSL-KDD son significativamente menores para este ataque que para los demás [47].

Los porcentajes estadísticos de la ejecución se aprecian en la Tabla I.

Tabla I
Porcentajes de normales/anómalos detectados

Porcentaje normales detectados correctamente	99,68%
Porcentaje anómalos detectados correctamente	99,42%

En la Tabla II se muestran los resultados de pruebas de otras propuestas de IDS basadas en SOM y GHSOM extraídas de [47]. En ella se observa cómo el método nuestro de GHSOM con probabilidades de reetiquetado (RL-GHSOM) alcanza un alto índice de ataques detectados y supera claramente la tasa de falsos positivos proporcionada por las otras propuestas similares.

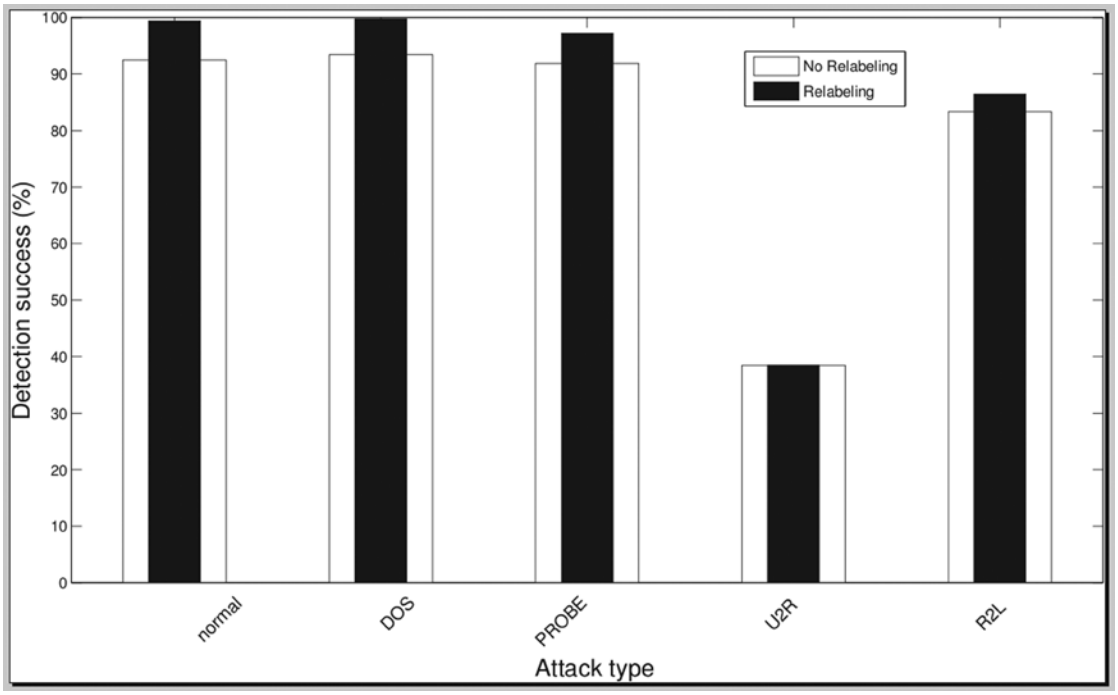

Fig. 13 Tasa de éxito de detección (Vertical) por tipo de ataque (Horizontal)

Tabla II
Comparativa del método utilizado con otros similares

Implementación de IDS	Ataques detectados (%)	Falsos positivos (%)
RL-GHSOM	99.68	0.02
GHSOM	99.99	3.72
K-Map	99.63	0.34
SOM	97.31	0.34

CONCLUSIONES

Las tipologías de organización de datos propia de los métodos de acceso y las propiedades de generalización y aprendizaje de las redes neuronales, convierten a las técnicas de SOM y sus derivados en procesos altamente eficientes en tareas de clasificación y recuperación de información. Por lo tanto, es posible aprovechar las características propias de dos estructuras al mismo tiempo, haciendo de la familia SOM y sus predecesores una herramienta versátil por su aplicación en casi cualquier dominio de datos.

En este trabajo se presenta una vista diferente de aplicación de los IDS o sistemas de detección de intrusos en redes de datos aprovechando las capacidades de la técnica usada por GHSOM. Se ha utilizado el conjunto completo de características del dataset NSLKDD para la consecución de resultados, por lo que se ha diseñado un sistema de etiquetado (o reetiquetado) de las BMU que usa la probabilidad para reetiquetar con éxito.

Los resultados obtenidos empleando la técnica mencionada son prometedores, ya que permite detectar el 99.68% de los patrones

de tráfico normal y el 99.42% de los anormales. Consecuentemente, si comportamientos normales y anormales son detectados con precisión, es posible realizar algunas otras acciones en tiempo real, como el bloqueo de IP, y las tareas de IDS/IPS podrían ser implementadas de manera eficiente en arquitecturas tales como los procesadores de red, haciendo de los ambientes informáticos compartidos un lugar más seguro.

APORTES

En este trabajo se ha hecho un programa en MATLAB que implementa el algoritmo GHSOM basándose en la Toolbox existente de SOM y aplicando funciones de entrenamiento y reetiquetado sobre los datos iniciales del dataset NSLKDD.

Los resultados obtenidos usando la técnica mencionada prometen trabajos futuros cargados de buenas noticias, ya que los porcentajes de acierto son lo suficientemente apreciables en comparación con otras técnicas de detección gracias a su capacidad de detección del 99.6824% de los patrones de tráfico normal y el 99.4235% de los anormales, ofreciendo así una luz verde para unir fuerzas en la investigación de este tema que tiene innumerables campos de acción en la vida diaria.

Otro de los aportes importantes que cabe mencionar es la capacidad que tiene el algoritmo realizado en detectar "Falsos Positivos" o tráfico normal detectado como anómalo (2.509541e-01) así como los "Falsos Negativos" o trafico anómalo detectado como normal (5.764967e-01). Por último, y no menos importante, hay que mencionar que la precisión del algoritmo implementado en este trabajo tiene un porcentaje del "99.597533%".

TRABAJOS FUTUROS

Como trabajo futuro, se considera la implementación real del IDS (Sistema de Detección de Intrusos), permitiendo hacer una aplicación eficaz y eficiente en la detección de conductas normales y anormales con la precisión del tiempo real así como mejoras a la implementación del mismo, como son el bloqueo de IP en tiempo real con miras a mejorar la calidad de la prevención en las intrusiones de la red.

De igual forma se piensa trabajar en el uso de la optimización multiobjetivo para permitir seleccionar las características más discriminatorias que generan pequeñas GHSOM y acelerar el paso de extracción de las características y el cálculo de la BMU en GHSOM, procedimiento que no solo acelerará la creación de los mapas sino que optimizará el proceso de detección. Pensando en esto, se plantea implementar alternativas para una ejecución del IDS mediante el uso de la optimización de los módulos del kernel en equipos de cómputo con varios procesadores y/o tarjetas de red programables.

Se considera que el aprovechamiento del actual paralelismo en los nodos de procesamiento actual mejorará el rendimiento de IPS (Intrusion Prevention System), lo que posibilitará la implementación de sistemas eficaces en la prevención activa de intrusos.

AGRADECIMIENTOS

La culminación de este artículo pudo desarrollarse gracias al apoyo de los doctores

Andrés Ortiz García, profesor PhD. de la Escuela Técnica Superior (E.T.S.) en Ingeniería de Telecomunicación de la Universidad de Málaga - España y Julio Ortega Lopera, Catedrático del Departamento de Arquitectura y Tecnología de Computadores de la Universidad de Granada - España; así mismo, agradezco el apoyo al doctorando Emiro De la Hoz Franco, docente tiempo completo de la Corporación Universidad de la Costa (CUC), de Barranquilla - Colombia, los cuales participaron activamente en la generación de significativos aportes crítico-constructivos que conllevaron a la culminación de la investigación presentada en este artículo. Se agradece también el apoyo brindado por el Departamento de Arquitectura y Tecnología de Computadores y a la Escuela Técnica Superior de Ingeniería Informática y de Telecomunicación de la Universidad de Granada - España y a diferentes estamentos de la Corporación Universidad de la Costa de Barranquilla - Colombia, como son la dirección del programa de Ingeniería de Sistemas, la Facultad de Ingeniería, el grupo de Investigación de Ingeniería del Software - Redes y el Centro de Investigaciones de la Facultad de Ingenierías.

REFERENCIAS

[1] Real Academia Española, Diccionario de la Lengua Española. [Online] Disponible en: http://lema.rae.es/drae/?val=seguridad

[2] Real Academia Española, Diccionario de la Lengua Española. [Online] Disponible en: http://lema.rae.es/drae/?val=seguro

[3] Real Academia Española, Diccionario de la Lengua Española: [Online] Disponible en: http://lema.rae.es/drae/?val=informat%C3%ADca

[4] Asociación de Técnicos de Informática - ATI. *Glosario básico inglés-español para usuarios de Internet*, [Online] Disponible en: http://www.ati.es/novatica/glosario/glosario_internet.txt

[5] A. Villalón Huerta, *Seguridad en Unix y redes*. [Online] Disponible en: http://www.rediris.es/cert/doc/unixsec/unixsec.pdf, pp. 6-7. 2002.

[6] R. Heady, G. Luger, A. Maccabe and M. Servilla, *The Architecture of a Network Level Intrusion Detection System*. Technical report, Department of Computer Science, University of New Mexico, August 1990.

[7] Fyodor, *Network Mapping Tool* [Online]. Disponible en: http://www.insecure.org/nmap

[8] Institute for Internet Security [Online]. Disponible en: http://www.internet-sicherheit.de/en/research/recent-projects/internet-early-warning-systems/internet-analysis-system/recent-results/

[9] Guo, Fanglu, Jiawu Chen and Tzi cker Chiueh: *Spoof Detection for Preventing DoSv Attacks against DNS Servers*. In: 26th IEEE International Conference, pp. 37-39. 2006.

[10] S. Kumar, *Classification and Detection of Computer Intrusions*. Tesis de Doctorado, Purdue University, 1995, citeseer.ist.psu.edu/kumar95classification.html

[11] ComputerWire, *DDoS Really, Really Tested UltraDNS.* Informe técnico, [Online]. Disponible en: http://www.theregister.co.uk/2002/12/14/ddos_attack_really_really_tested/ attack really really tested, December 2002.

[12] T. Olovsson, *A Structured Approach to Computer Security.* Informe técnico, Chalmers University of Technology, pp. 37-73. 1992.

[13] A. Villalón Huerta, *Seguridad en Unix y redes*. [Online] Disponible en: http://www.rediris.es/cert/doc/unixsec/unixsec.pdf, pp. 11 - 12. 2002.

[14] B. Daniel, OSSEC. [Online]. Disponible en: www.ossec.net, 2006.

[15] Ch. Hosmer and M. Duren, "Detecting Subtle System Changes Using Digital Signatures". En *Information Technology Conference, IEEE*. Laboratory at Purdue University, pp. 125-128, 1998.

[16] M. Roesch, *Lightweight Intrusion Detection for Networks.* [Online]. Disponible en: http://www.snort.org, 2005.

[17] O. Dain and R. Cunningham, Fusing Heterogeneous Alert Streams into Scenarios. Massachusetts Institute of Technology, September 2001. citeseer.ist.psu.edu/dain-01fusing.html

[18] L. Girardin, "An Eye on Network Intruder-administrator Shootouts". En *Proceedings of the Workshop on Intrusion Detection and Network Monitoring (ID'99)*, Berkeley, CA, USA, 1999. USENIX Association. citeseer.ist.psu.edu/girardin99eye.html. pp. 19-28.

[19] A. Siraj, R. B. Vaughn and S. M. Bridges, "Intrusion Sensor Data Fusion in an Intelligent Intrusion Detection System Architecture". En *Proceedings of the 37th Annual Hawaii International Conference*, p. 10, 2004.

[20] S. X. Wu and W. Banzhaf, "The use of computational intelligence in intrusion detection systems: A review". *Applied Soft Computing,* 10(1), 1-35. doi: 10.1016/j.asoc.2009.06.019. 2010.

[21] H. Debar, M. Dacier and A. Wespi, "A revised taxonomy for intrusion-detection systems". *IBM Research Technical Report,* October 1999.

[22] S. Axelsson, *Intrusion Detection Systems: A Taxonomy and Survey.* Technical Report 99-15, Dept. of Computer Engineering, Chalmers University of Technology, Goteborg, Sweden. 2000.

[23] Networks, Enterasys, *Intrusion Detection Methodologies Demystified.* [Online]. Disponible en: http://www.enterasys.com/products/ids/whitepapers/, 2005. Ver también: S. Northcutt, *Inside Network Perimeter Security: An Analyst Handbook.* Ed. New Riders edición, 2003. pp. 125-127. Ver también: R. Bace, *ICSA: An Introduction to Intrusion Detection and Assessment.* [Online]. Disponible en: http://www.icsalabs.com/html/communities/ids/whitepaper/Intrusion1.pdf, 2005. Ver también: *SANS: Intrusion Detectopm FAQ.* [Online]. Disponible en: http://www.sans.org/resources/idfaq/, 2005., y H. F. Tipton and E. Auerbach: *Information Security Management Handbook.* 1999.

[24] S. Kumar and E. H. Spafford, "Software Architecture to Support Misuse Intrusion Detection". En *Proceedings of the 18th National Information Security Conference,* pp. 194-204. 1995.

[25] S. Watanabe, *Pattern recognition: human and mechanical.* John Wiley & Sons, Inc., New York, NY, USA. 1985.

[26] A. K. Jain, M. Ñ. Murty and P. J. Flynn, "Data clustering: a review". *ACM Computing Surveys,* 31(3). pp. 264-323, 1999.

[27] A. K. Jain, M. Ñ. Murty and P. J. Flynn, "Data clustering: a review". *ACM Computing Surveys,* 31(3). p. 30, 1999.

[28] R. C. Dubes, *Cluster analysis and related issues*. 1993.

[29] A. K. Jain and R. C. Dubes, *Algorithms for clustering data.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA. 1988.

[30] A. K. Jain, M. Ñ. Murty and P. J. Flynn, "Data clustering: a review". *ACM Computing Surveys,* 31(3). p. 278, 1999.

[31] J. J. Hopfield, "Neural networks and physical systems with emergent properties", *Proceedingns of the National Academy of Sciences* 79. pp. 2554-2558, 1982.

[32] T. Kohonen, "Self-organized formation of topologically correct feature maps". *Biological Cybernetics,* 43. pp. 59-69, 1982.

[33] G. A. Carpenter and S. Grossberg, "The art of adaptive pattern recognition by a self-organizing neural network". *Computer*, 21(3). pp. 77-78, 1988.

[34] M. Dittenbach, D. Merkl and A. Rauber, "The Growing Hierarchical Self-Organizing Map". In S: Amari, C. L. Giles, M. Gori and V. Puri (editors), *Proc of the International Joint Conference on Neural Networks (IJCNN 2000)*, volume VI, Como, Italy. IEEE Computer Society. pp. 15-19, 2000.

[35] B. Fritzke, "A growing neural gas network learns topologies". In G. Tesauro, D. S. Touretzky and T. K. Leen (editors), *Advances in Neural Information Processing Systems 7*. MIT Press, Cambridge MA. pp. 625-632, 1995.

[36] J. Blackmore and R. Miikkulainen, "Incremental grid growing: Encoding high-dimensional structure into a two-dimensional feature map". In *Proceedings of the International Conference on Neural Networks ICNN93,* volume I. Piscataway, NJ. IEEE Service Center. pp. 450-455, 1993.

[37] D. Alahakoon, S. K. Halgamuge and B. Srinivasan, "A structure adapting feature map for optimal cluster representation". In *International Conference on Neural Information Processing ICONIP98*. pp. 809-812, 1998.

[38] A. Ocsa, C. Bedregal and E. Cuadros-Vargas, "DB-GNG: A constructive self-organizing map based on density". In *Proceedings of the International Joint Conference on Neural Networks (IJCNN07)*. IEEE, 2007.

[39] A. K. Jain, J. Mao and K. M. Mohiuddin, *Artificial neural networks: A tutorial.* IEEE Computer, 29(3):31-44, 1996.

[40] T. Kohonen, *Self-Organizing Maps*, 3ra Edición, Springer-Verlag, p. 86, 2001.

[41] T. Kohonen, *Self-Organizing Maps*, 3ra Edición, Springer-Verlag, 2001.

[42] T. Kohonen, "The Self-Organizing Maps". *Proceedings of the IEE*, Vol. 78, No. 9, September 1990, p. 1467.

[43] T. Kohonen, *Self-Organizing Maps*. Springer, Berlin, 1995.

[44] Imagen disponible en Internet: http://www.peltarion.com/doc/images/SOM_Unified_distance_matrix.gif

[45] M. Dittenbach, D. Merkl and A. Rauber, "The Growing Hierarchical Self-Organizing Map". In S. Amari, C. L. Giles, M. Gori and V. Puri, (editors), *Proc of the International Joint Conference on Neural Networks (IJCNN 2000),* volume VI, Como, Italy. IEEE Computer Society. pp. 199-216, 2000.

[46] S. P. Luttrell, "Hierarchical self-organizing networks". In *Proceedings of the International Conference on Neural Networks (ICANN'89)*. London, U.K. pp. 2-6, 1989.

[47] G. R. Zargar and P. Kabiri, "Selection of Effective Network Parameters in Attacks for Intrussion Detection". In: *IEEE International Conference on Data Mining.* 2010.

EDUCOSTA
EDITORIAL UNIVERSITARIA DE LA COSTA

[48] E. J. Palomo, E. Domínguez, R. M. Luque And J. Muñoz, "Network security using growing hierarchical self-organizing maps". In: M. Kolehmainen, P. Toivanen, and B. Beliczynski (eds.) *ICANNGA 2009*. LNCS, vol. 5495. Springer, Heidelberg. pp. 130-139, 2009.

[49] R. Datti and B. Verma, "Feature Reduction for Intrusion Detection Using Linear Discriminant Analysis". *International Journal on Engineering Science and Technology* 2(4). pp. 1072-1078, 2010.

[50] S. Mukkamala and A. H. Sung, "Feature Ranking and Selection for Intrusion Detection Systems Using Support Vector Machines". In: *Proceedings of the Second Digital Forensic Research Workshop*. 2002.

[51] A. Ortiz, J. Ortega, A. Martínez and A. Prieto, "Intrusion detection and prevention by using Hierarchical Selforganizing Maps and Multiobjective-based feature selection". *International Journal on Neural System*. pp. 232-239, 2011.