PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA INFORMÁTICA

CLASIFICACIÓN DE DATOS DE IDS BASADA EN LS-SVM CON PSO

EDUARDO MAURICIO LEÓN GARCÍA

INFORME FINAL DE PROYECTO PARA
OPTAR AL TÍTULO PROFESIONAL DE
INGENIERO CIVIL EN INFORMÁTICA

DICIEMBRE DEL 2010

Pontificia Universidad Católica de Valparaíso
Facultad de Ingeniería
Escuela de Ingeniería Informática

CLASIFICACIÓN DE DATOS DE IDS BASADA EN LS-SVM CON PSO

EDUARDO MAURICIO LEÓN GARCÍA

Profesor Guía: **Nibaldo Rodríguez Agurto**

Carrera: **Ingeniería Civil en Informática**

Diciembre del 2010

A mi madre, que sin ella
no sería lo que soy hoy.

Índice de contenido

Resumen

Muchas instituciones manejan enormes cantidades de información que circula por sus redes como paquetes de datos, que son vulnerables frente a ataques informáticos de diversa índole. Una opción para mitigar el riesgo de estos ataques es usar sistemas de detección de intrusiones (IDS), los que monitorean las redes constantemente en busca de anomalías. Sin embargo, los ataques son diversos y cambiantes, en consecuencia siempre existirá la necesidad de mejorar los procesos de clasificación.

En este contexto, el objetivo del presente proyecto de título es desarrollar e implementar la técnica LS-SVM con PSO para la detección de intrusiones. Para llevar a cabo la construcción y evaluación de cada modelo de clasificación, cada proceso es dividido en dos etapas: entrenamiento y pruebas. En la primera, el algoritmo de clasificación es optimizado por PSO, de tal manera, que una vez construido el modelo, cuente con la capacidad de responder bien frente a cualquier dato, incluyendo los desconocidos. En la segunda etapa, se realizan pruebas para medir la bondad del modelo en base a una serie de índices. Un punto fundamental en este desarrollo fue que los conjuntos de datos, de las dos etapas, tuvieran la menor cantidad de datos en común. De esta manera se pudo evaluar efectivamente la capacidad de generalización de cada modelo.

Los resultados obtenidos por el mejor modelo indican que se alcanzó un 83,49% de exactitud promedio, del cual, al rededor de un 6% fue aportado por PSO. Este aporte podría parecer algo insustancial, sin embargo, se debe considerar que entre mayor sea el desempeño del clasificador por si solo (sin optimización), menores serán las mejoras que hacer. Además, se debe considerar que los modelos se entrenaron con 2.000 datos, para, posteriormente, ser puestos a prueba contra 22.544 en la fase de pruebas.

Es posible afirmar que LS-SVM es una excelente técnica para clasificación, que además, demostró un alto poder de generalización, ya que demostró un gran desempeño tras clasificar más de 20.000 datos absolutamente desconocidos. Respecto a PSO, se pudo comprobar las ventajas de su uso, puesto que aumentó la efectividad de LS-SVM en relación al problema de IDS.

Palabras-clave: Intrusion Detection System, Support Vector Machine, Least Squares, Particle Swarm Optimization

Abstract

Many institutions handle huge amounts of information that flows through their networks as data packets, which are vulnerable to attacks from various kinds. One option to mitigate the risk of these attacks is to use intrusion detection systems (IDS), which constantly monitor the network for anomalies. However, the attacks are varied and changing, therefore there will always be the need to improve the classification processes.

In this context, the objective of this project is to develop and implement LS-SVM technique with PSO for intrusion detection. To carry out the construction and evaluation of each classification model, each process is divided into two stages: training and testing. In the first, the classification algorithm is optimized by PSO, so that once built the model, has the ability to respond well against any data, including unknown. In the second stage, tests are done to measure the accuracy of the model based on a number of rates. A key point in this development was that the data sets of the two stages, had the least amount of data in common. In this way it could effectively evaluate the generalization ability of the models.

The results obtained indicate that the best model was reached on 83.49% average accuracy, which, around 6% was contributed by PSO. This contribution might seem somewhat small, however, consider that the higher the performance of the classifier by itself (without optimization), the smaller the improvements to do. Also, consider that training processes were used about 2.000 data, to then be tested against 22.544 in the testing phase.

Arguably, LS-SVM is an excellent technique for classification, which also showed a high power of generalization, because it showed a great performance after classify more than 20.000 absolutly unknown data. Regarding PSO, it was observed the advantages of its use, because it could increase the effectiveness of LS-SVM in relation to the IDS problem.

Keywords: Intrusion Detection System, Support Vector Machine, Least Squares, Particle Swarm Optimization

Lista de Abreviaturas o Siglas

AUC : Area Under an ROC Curve
DAPSO : Particle Swarm Optimizer with Dynamic Adaptation
DOS : Denial of Service
FN : False Negative
FP : False Positive
IDS : Intrusion Detection System
IPSO : Improved Particle Swarm Optimization
KDD : Knowledge Discovery and Data Mining
LS : Least Squares
PSO : Particle Swarm Optimization
QA : Quadratic Programming
R2L : Remote To Local attacks
RBF : Radial Basis Function
ROC : Reciver Operating Characteristics
SVM : Support Vector Machine
TN : True Negative
TP : True Positive
U2R : User To Root attacks

Lista de Ilustraciones

Lista de Tablas

1. Introducción

Existe cierta información muy relevante para las personas y las organizaciones debido al gran valor que significa para ellas, sobre todo cuando es de carácter sensible o privada, y más aún si los volúmenes manejados de esta son prácticamente inmensurables. Agregando el factor "interconectividad" en nuestras formas de vida, es que el riesgo de vulnerar esta información se ve aumentado considerablemente. Por lo tanto, y al igual que las personas, es que cada organización a lo largo de su vida se ve en la necesidad de contar, dentro de lo que su alcance le permita, con la mejor tecnología posible, garantizando la buena salud de su estado y sus operaciones.

Hoy en día la necesidad por aumentar el nivel de seguridad de las organizaciones, en términos virtuales, crece con mayor velocidad en proporción a las nuevas amenazas que ponen en riesgo su preciada información. Ya sea por entretenimiento, curiosidad, interés o con fines maliciosos los hackers ponen a prueba cada tipo de barrera existente en busca de alguna vulnerabilidad explotable de la cual aprovecharse y así lograr sus objetivos. Es por esto, que constantemente se están creando o mejorando mecanismos para fortalecer la gran diversidad de sistemas que se generan. Algunos ejemplos comunes que podrían evitar o reducir ciertos tipos de amenazas son: el uso de políticas de seguridad, encriptación, cortafuegos, filtros de paquetes y procesos de detección/prevención de intrusiones.

El uso de procesos de detección de intrusiones es una buena práctica a seguir cuando se quiere mejorar la seguridad de un red, ya que son capaces de monitorear cada evento que ocurre en ella y analizarlos en busca de conductas anormales. Vale decir que no solo se preocupan de los eventos generados desde el exterior de la red sino que también monitorea los eventos que son producidos por las mismas entidades que poseen una participación formal o reconocida en la red interna. Los sistemas que automatizan dicho proceso son conocidos como Intrusion Detection System [1] (IDS).

En la actualidad, los IDS tiene como primera prioridad la efectividad con que responden a eventuales amenazas [2]. El desafío fundamental de estos es "*saber*" discernir con fundamentos entre cualquier comportamiento considerado como normal y el resto, para alertar a tiempo en caso de que sea necesario. Para poder realizar dicha distinción estos sistemas deben contar con ciertos conocimientos útiles que les permitan realizar una buena clasificación. Cabe destacar que existen diversos modos de "*capacitar*" a un IDS y que continuamente se están realizando mejoras y nuevas técnicas con el fin de disminuir los riesgos inherentes a los datos que atraviesan una red.

Este proyecto expone una alternativa para mejorar la efectividad en la clasificación de intrusiones, lo que podría conllevar a disminuir el riesgo virtual al cual se ven ligadas las organizaciones. Se aplicará una técnica de clasificación conocida como Least Square Support Vector Machine (LS-SVM) [3], la que será potenciada con Particle Swarm Optimization (PSO) [4], una metaheurística de optimización combinatorial renombrada por su excelencia. La tarea de PSO será mejorar el desempeño de LS-SVM, a través de la búsqueda de parámetros, que combinados, conformen un modelo optimizado capaz de predecir a qué

categoría pertenece cada nuevo evento.

El presente informe se encuentra organizado de la siguiente manera. En la primera parte del documento se exponen los objetivos del proyecto. En segundo lugar, se mostrará, en base a una serie de conceptos relacionados, los elementos relevantes del tema, como por ejemplo las metodologías de detección usadas en IDS, el trabajo paso a paso que realiza LS-SVM y el funcionamiento en detalle de PSO, es decir, el Marco Teórico en que se basó el desarrollo del proyecto. En tercer lugar, se presentará el Estado del Arte en base a investigaciones recientes, con el objetivo de afinar el rumbo y establecer el alcance del presente trabajo, incluyendo cierta orientación para la lectura de terceros ajenos a la problemática. En cuarto lugar, se exponen las principales tareas de carácter práctico llevadas a cabo a lo largo del proyecto: se describe el proceso utilizado para la preparación de los datos de IDS, la implementación de las técnicas mencionadas y las pruebas realizadas, exponiendo, en última instancia, la evaluación e interpretación de los resultados alcanzados. Finalmente, se exponen las conclusiones del proyecto.

2. Objetivos

En cuanto a los objetivos del proyecto de título, a continuación se presenta el objetivo general y el listado de objetivos específicos correspondiente.

2.1 Objetivo General

- Desarrollar un modelo de clasificación de detección de intrusiones utilizando la versión de mínimos cuadrados de las máquinas de vectores de soporte con algoritmos de optimización de enjambre de partículas.

2.2 Objetivos Específicos

- Explicar el funcionamiento de las máquinas de vectores de soporte y optimización de enjambre de partículas.
- Diseñar y estimar los parámetros para la versión de mínimos cuadrados de las máquinas de vectores de soporte mediante el uso de la optimización de enjambre de partículas y algunas de sus variantes.
- Evaluar y contrastar los modelos de clasificación propuestos.

3. Marco Teórico

El objetivo principal de este capítulo es dar a conocer los conceptos que permitan abordar, de manera general, el problema de clasificación de datos de IDS con LS-SVM y PSO, y además, profundizar sobre cada una de estas componentes. Lo que se busca con el marco teórico es establecer o fundar las bases para el desarrollo del proyecto y conseguir situarse en el contexto de los temas involucrados, los términos utilizados, acontecimientos importantes, los algoritmos, su evolución, etc.

En primera instancia se abordará el tema desde el punto de vista de los IDS, sus características, las metodologías de detección actuales, los criterios de evaluación al realizar las clasificaciones y qué conjunto de datos será utilizado para experimentar y por qué. Posteriormente, se profundizará en SVM como técnica de clasificación para la problemática en cuestión, exponiendo sus fundamentos teóricos y el modo de funcionamiento, haciendo énfasis en la versión LS-SVM con las respectivas justificaciones o motivos de su elección. Finalmente, se hablará de la metaheurística para optimización combinatorial PSO, se describirá su comportamiento, el modo de operación detallado del algoritmo, cómo ha evolucionado desde su aparición y cuál será la función objetivo utilizada para el problema de clasificación.

3.1 Itrusion Detection System

Un Sistema de Detección de Intrusiones, o simplemente IDS, es un sistema automatizado que examina constantemente el tráfico de datos que fluye a través de una red de computadores, con el fin de detectar eventos anormales. En el caso de descubrir comportamientos de tal naturaleza, el IDS debe reaccionar y responder de alguna manera predeterminada, como, por ejemplo, realizar un registro detallado del evento ocurrido o ejecutar alguna alerta.

De manera análoga, se podría decir que el funcionamiento y los objetivos de un IDS se asemejan al trabajo que debe realizar un guardia de seguridad en un recinto privado. El guardia debe estar pendiente de todos los lugares de acceso para aquellas personas que entran o salen del perímetro que le corresponde vigilar. Además, dicho personaje, debiese estar continuamente observando las imágenes captadas por las cámaras de vigilancia, con el fin de ir chequeando que todo se encuentre en orden. En caso de detectar alguna persona desconocida, sospechosa o algún comportamiento irregular, el guardia debiera proceder sistemáticamente a realizar una determinada acción. Por ejemplo, interrogar a un visitante desconocido que acaba de entrar o dar aviso inmediato a carabineros frente al caso de un delito que le es insostenible o difícil de manejar.

Para que el recinto privado y todo lo que este alberga se encuentre seguro frente a los tipos de comportamiento mencionados, se debe contar con una persona altamente capacitada para realizar dicha labor, de modo que, frente a cualquier eventualidad, todo resulte de la mejor manera posible. Es decir, el guardia de seguridad debe tener la cualidad de distinguir

entre las actividades normales y las que no lo son, además, tiene que reaccionar de manera eficiente y eficaz para minimizar los daños que, eventualmente, se podrían provocar. De esta manera es como debiera funcionar un IDS que monitorea una determinada red. Para conseguirlo, son necesarios procesos robustos que cuenten con una alta exactitud a la hora de chequear o analizar un nuevo evento para de determinar, con plena seguridad, a qué clase de actividad se está enfrentando.

El concepto de IDS se fue acuñando en la década de los 80'. A partir de aquel entonces ha evolucionado desde el uso de estadísticas para detección de anomalías, a través de sistemas expertos basados en reglas, hasta el uso minería de datos para encontrar patrones característicos. Hoy en día, **la detección de intrusiones** es el proceso de monitoreo de los eventos que ocurren en un sistema computacional o red, con el objetivo de analizar y buscar signos de posibles *incidentes:* amenazas de violaciones (inminentes) a las políticas de seguridad computacional, políticas de uso aceptable, o prácticas de seguridad estándar. Un IDS es, técnicamente, software que automatiza el proceso de detección de intrusiones previamente mencionado [5].

3.1.1 Modelo de Procesos para Detección de Intrusiones

Según Rebecca Bace y Peter Mell [2], la mayoría de los IDS se pueden describir en términos de tres componentes funcionales fundamentales, los cuales son presentados a continuación.

- **Fuentes de información**
  Corresponden a las distintas fuentes de información de eventos utilizadas para determinar cuándo se está llevando a cabo una intrusión. Estas fuentes pueden ser obtenidas de diferentes niveles del sistema, siendo las más comunes las de monitoreo de la red, host y aplicaciones.

- **Análisis**
  Es la sección de los IDS que actualmente organizan y dan sentido a los eventos derivados de las fuentes de información, decidiendo cuándo estos eventos indicarán que se está llevando a cabo una intrusión o si ya ha tenido lugar. Los enfoques de análisis más comunes son "*misuse detection"* y "*anomaly detection"*. Misuse Detection se basa en comportamientos anómalos conocidos para detectar anomalías según ciertos patrones o signos, y Anomaly Detection, en las actividades que son carácter normal o regular para la red, con objeto de determinar o clasificar como anomalías a todos aquellos comportamientos que sean distintos de los ya conocidos como comunes o normales.

- **Respuesta**
  Es el conjunto de acciones que toma el sistema cuando una intrusión es detectada. Típicamente están agrupadas en medidas activas y pasivas. Con medidas activas se realizan algunas intervenciones automatizadas por parte del sistema, y con las pasivas se reportan los resultados del IDS a los humanos, quienes, finalmente, son los que tendrán lo opción de tomar alguna decisión.

3.1.2 Metodologías de Detección Comunes

Los IDS suelen utilizar una gran variedad de metodologías para el proceso de detección, cuya finalidad es ampliar su alcance y exactitud. Existen tres metodologías que son las más utilizadas y cada una de ellas posee ventajas y desventajas respecto a las otras. Sin embargo, estas se pueden integrar logrando complementarse entre si para beneficiar sus objetivos. A continuación se otorga una breve descripción de cada una de estas metodologías en base a lo establecido en [5].

- **Signature-Based Detection** (Detección en base a signos)
  Esta metodología se basa en ataques conocidos y usa comparaciones sencillas para concluir si lo que está siendo detectando coincide con un ataque o no. En definitiva, es una técnica considerablemente útil frente a ataques conocidos, no obstante, su desempeño cambia cuando se enfrenta a variaciones de estos y al analizar ataques que no se encuentran en su conjunto de datos conocido.

- **Anomaly-Based Detection** (Detección en base a anomalías)
  Esta detección se basa en el comportamiento de toda aquella actividad considerada como normal dentro del sistema que se esté auditando y determinará que un evento es anómalo cuando la conducta de este tenga alguna desviación de la considerada como normal. Para tomar resolución sobre las actividades normales se crean perfiles de usuarios, host, redes, etc. La desventaja de esta metodología es que en el caso en que los perfiles sean creados de forma automática, se puede dar la situación en que, mientras se crean los perfiles, un ataque que se esté llevando a cabo de manera sutil y periódica, podría fácilmente caer en alguno de los perfiles. Además, si el ataque es posteriormente intensificado es probable que no sea detectando dada la poca desviación del comportamiento observado. Por otra parte, la gran ventaja de la metodología es que es muy eficaz frente a ataques nuevos.

- **Stateful Protocol Analysis** (Análisis en completitud del estado del protocolo)
  Es el proceso de comparar perfiles predeterminados de definiciones, generalmente aceptadas, de la actividad de protocolo benigna para cada estado del protocolo, contra los eventos observados para identificar desviaciones. Con "stateful" se refiere a que el IDS es capaz de entender y rastrear el estado de los protocolos de red, transporte y aplicación que tienen alguna noción de estado, por ejemplo, los estados de una sesión al intentar autenticarse contra un sistema.

Entre otras metodologías para modelar y reconocer comportamientos normales e irregulares se puede encontrar modelos estadísticos, enfoques de sistemas inmunes, chequeo de archivos y de corrupciones, redes neuronales, listas blancas, igualación de expresiones, análisis de transición de estados, lenguajes dedicados, algoritmos genéticos, etc. [6]. En el Capítulo 4 se podrán encontrar con algunas de las metodologías usadas en la actualidad y que, además, fueron vinculadas por distintos autores al mismo problema de IDS tratado en este proyecto.

3.1.3 Criterios de Evaluación

Para medir la bondad del proceso de detección de un IDS se utiliza un conjunto de criterios complementarios, los cuales dependerán de qué clase de evento (real) se esté llevando a cabo y cómo es clasificado. Es por esto, que para realizar dicha evaluación, primero es necesario conocer a qué categorías corresponden los eventos, para luego, con las predicciones obtenidas por dicho sistema, realizar las comparaciones correspondientes.

Cabe destacar que generalmente, y en el caso de este trabajo, los IDS realizan clasificación binaria, es decir, que los eventos son considerados solamente como normales o anómalos, no se consideran categorías adicionales. En base a esto la identificación o decisión tomada por el sistema es evaluada considerando los cuatro casos posibles de la clasificación binaria. Dichos casos son generalizados en la Tabla 3.1 donde "+" significa que sí hay una intrusión o respuesta y "–" lo contrario.

Tabla 3.1 Cuantificadores de los IDS

	Intrusión +	Intrusión –
Respuesta +	**TP**	FP
Respuesta –	FN	**TN**

- **TP** (*True Positive* o Verdaderos Positivos): Conocido como "*acierto*", es cuando realmente hay una intrusión y es detectada correctamente.
- **TN** (*True Negative* o Verdaderos Negativos): Cuando no se ha llevado a cabo una intrusión y no se ha dado respuesta como si fuese una.
- **FP** (*False Positive* o Falsos Positivos): Intrusión inexistente, pero algún evento ha señalado al IDS a dar una respuesta, también conocida como "*falsa alarma*".
- **FN** (*False Negative* o Falsos Negativos): Conocido como *"fallo"*, es cuando existe una intrusión real, pero el IDS no tuvo la capacidad para detectarla.

En base a los cuantificadores anteriores, a continuación, se exponen los índices más utilizados en clasificación binaria.

- **Sensibilidad**
  Conocida como Tasa de Verdaderos Positivos, mide la proporción de TP que han sido identificados correctamente como tal. Es decir, el porcentaje de intrusiones que son identificadas por tener realmente tal condición. En el caso de IDS, la sensibilidad también es conocida como "Tasa de Detección" y se calcula de la siguiente manera:

$$Sensibilidad = \frac{TP}{TP + FN} \quad (1)$$

- **Especificidad**
  Conocida como Tasa de Verdaderos Negativos, mide cuán específico es el clasificador. Es decir, mide la proporción de TN que son identificados correctamente. Esto es, el porcentaje de eventos normales que son clasificados como tal o por no tener la condición de intruso. La Especificidad se calcula de la siguiente manera:

$$Especifidad = \frac{TN}{TN + FP} \quad (2)$$

- **Exactitud**
  Es el grado de cercanía de las mediciones de una cantidad a su valor real. Es decir, la proporción de resultados verdaderos (TP y TN) en la población. Es el criterio que más es utilizado en la fase de pruebas y se calcula de la siguiente manera:

$$Exactitud = \frac{TP + TN}{TP + TN + FP + FN} \quad (3)$$

- **Precisión**
  Conocida como repetitibilidad, es el grado para el cual mediciones repetidas bajo condiciones sin cambios muestran los mismos resultados. Es decir, la proporción de TP contra todos los resultados positivos (TP y FP). La precisión se calcula de la siguiente manera:

$$Precisión = \frac{TP}{TP + FP} \quad (4)$$

El desempeño ideal para un modelo de clasificación es acertar a todos los tipos de eventos analizados, es decir, obtener un 100% de TP y TN, o dicho de otro modo, 0% en FP y FN lo que lograría que todos lo indices recién expuestos coincidieran en alcanzar un 100% en sus resultados. Sin embargo, al enfrentarse a problemas de clasificación reales y de alta complejidad, como el problema de IDS, se dificulta el camino para alcanzar la esperada perfección. Se debe considerar que el comportamiento de la actividad de una red es dinámico y caótico, lo cual implica un gran desafío a la hora de detectar y diferenciar patrones de manera correcta.

Cuando los datos que se transmiten en una red son valiosos y, además, se encuentran en juego, es preferible que la naturaleza de un IDS sea más conservadora, en otras palabras, es más conveniente que se genere una alerta por un evento normal, que inclinarse por ignorar actividad anómala y ser victima de algún tipo de amenaza. No obstante, no se debe caer en los extremos, un IDS podría, eventualmente, tener una tasa de detección del 100%, lo que sería

ideal, pero esto no tendría sentido si al mismo tiempo el sistema mantuviera una tasa de falsas alarmas extremadamente alta. Bajo estas condiciones no se justificaría el uso de tal sistema, ya que al clasificar toda actividad como una intrusión se generaría tal cantidad de alertas que el comportamiento normal en la red se podría ver totalmente alterado o interrumpido. En un caso como este, el IDS podría congestionar la red con mensajes de alerta o, si este fuese más preventivo aun, podría llevar a un corte continuo del tráfico para todos los tipos de eventos que ocurran, independiente de cuál fuese su naturaleza.

En base a lo anterior es que se hace necesario que un IDS sea efectivo, pero sin caer en los extremos. Por lo tanto, a la hora de evaluar un clasificador, este debe poseer una pequeña tendencia a disminuir su Especificidad, pero manteniendo una alta Sensibilidad para que su comportamiento sea más conservador. Una forma de conocer la tendencia que tiene el modelo de clasificación es basarse en la tasa de detección y la tasa de falsas alarmas.

Usualmente existe un equilibrio o *"trade-off"* entre las medidas de Sensibilidad y Especificidad, o dicho de otro modo, entre la **tasa de detección** y la **tasa de falsas alarmas** (*1-Especificidad*). Este equilibrio puede ser transformado en una herramienta complementaria para ayudar a medir la bondad el modelo de clasificación. A fin de poder representar gráficamente dicha relación, es que se utiliza un gráfico llamado Receiver operating characteristics o, simplemente, curva ROC.

La curva ROC es un gráfico bidimensional que representa trade-offs relativos entre beneficios (TP) y costos (FP). Generalmente, es usado para comparar distintos modelos de clasificación. La mejor predicción posible de un método, producirá un punto en la esquina superior izquierda o en la coordenada (0,1) del espacio de la curva ROC, representando 100% de sensibilidad (sin FN) y 100% de especificidad (sin FP). El punto (0,1) es llamado *"la clasificación perfecta"*.

Según los autores de [3], la curva ROC muestra las habilidades separativas de un modelo de clasificación binario. Si área bajo la curva ROC es igual a 1 quiere decir que el modelo fue capaz de separar correctamente cada uno de los puntos de datos involucrados en su proceso de entrenamiento. En el caso de que el área bajo la curva sea igual a 0,5 quiere decir que el clasificador construido no posee ningún poder de discriminación. Si ni siquiera fue capaz de separar los puntos de datos, menos será su capacidad de clasificar datos de manera correcta en una fase de pruebas.

Según la interpretación de Fawcett [7], el AUC (Area Under an ROC Curve) es equivalente a la probabilidad de que un clasificador determine una instancia positiva de manera más alta que la de una negativa, esto considerando que ambas sean escogidas aleatoriamente. Esto es, desde la perspectiva de los IDS, que la probabilidad de clasificar correctamente un evento escogido al azar es mayor cuando se trata de un evento anómalo que de un evento normal o no intrusión.

En definitiva, es mucho más deseable que un IDS clasifique correctamente a los eventos anómalos que a los normales. Sin embargo, se debe tener en cuenta que esto es solo una probabilidad y que siempre habrá espacio para el error.

A continuación se muestran dos curvas ROC a modo de ejemplo. En la **Ilustración 3.1** se expone una curva con un área de 0,93578 y en la **Ilustración 3.2** otra con un área equivalente a 1, la que corresponde a un caso de separación perfecta.

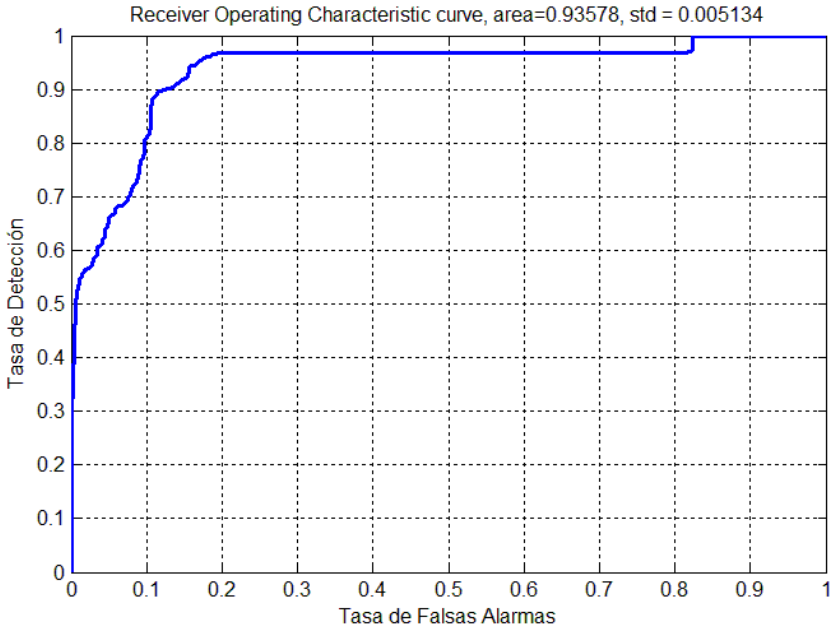

Ilustración 3.1. Curva ROC de un entrenamiento regular con 2.500dp

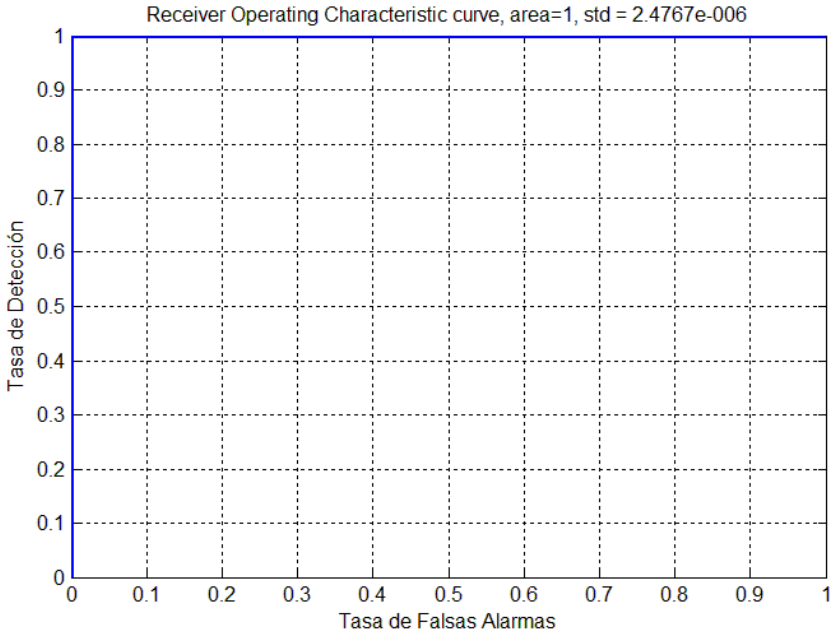

Ilustración 3.2.Curva ROC de un entrenamiento bueno con 2.500dp

3.1.4 Conjunto de Datos KDD'99

Para llevar a cabo evaluaciones fiables de modelos de clasificación para el problema de IDS es necesario el uso de un conjunto de datos apropiado. Es decir, que el conjunto de datos haya sido originado, en lo posible, a partir de datos reales o sino de simulaciones complejas del comportamiento común de los eventos en una red. Luego de tener los datos base, es conveniente que estos sean refinados, eliminando redundancia y distribuciones poco normales. De manera adicional es recomendable utilizar datos que hayan sido puestos a prueba o analizados en otras publicaciones puesto que serán útiles para realizar comparaciones entre resultados, seguir sugerencias y buenas prácticas que guíen el buen desarrollo del proyecto.

En lo que respecta a los datos auditados en este trabajo, se encontró disponible un conjunto de datos extensamente utilizado por la mayoría de los trabajos realizados sobre IDS en la última década. Por lo tanto, desde un comienzo se posee la ventaja de que existe una amplia disponibilidad de información útil relacionada con el conjunto de datos, que además, ha ido aumentando o mejorando a través del tiempo y también, los antes mencionados resultados que prestan mucha utilidad a la hora de realizar comparaciones con otros modelos de clasificación, incluso cuando han sido desarrollados con otras metodologías.

El conjunto de datos de IDS mencionado fue dado a conocer en la competencia KDD'99 y es conocido como el conjunto de datos KDD'99 [8]. Estos datos son una versión de un conjunto estándar compuesto de una gran variedad de intrusiones simuladas en un entorno de red militar. Estos datos son el producto de un programa preparado y gestionado por MIT Lincoln Labs llamado *"1998 DARPA Intrusion Detection Evaluation Program",* con el fin de identificar y evaluar una investigación sobre detección de intrusiones. Esta versión corresponde al conjunto de datos utilizado en *"The Third International Knowledge Discovery and Data Mining Tools Competition"* en 1999, cuyo objetivo era construir un detector de intrusiones de red, un modelo predictivo que fuese capaz de distinguir entre conexiones "malas", llamadas intrusiones o ataques, y conexiones "buenas" o normales.

En el sitio web [8] de la Universidad California, Irvine se encuentran disponible un listado de archivos que contienen los datos de la competencia en distintos porcentajes, versiones corregidas, entre otras cosas. Vale decir, que los archivos que poseen el 100% de los datos para el entrenamiento y las pruebas contienen 4.898.431 y 311,027 puntos de datos respectivamente. Además, cada punto de datos esta compuesto por 41 características *(ver Apéndice 1)* y una etiqueta que lo categoriza, esta puede ser "normal" o alguna de las cuatro subcategorías de ataques preestablecidas *(ver Apéndice 2)*. En resumen esto se traduce en una matriz de casi 5 millones de filas por 41 columnas a libre disposición de investigadores con el fin de que puedan entrenar y evaluar el desempeño de sus modelos de clasificación.

Respecto a las clases de datos del conjunto KDD'99, como se ha mencionado con anterioridad, se encuentran las de datos normales y datos anómalos. Los datos normales corresponden a los eventos normales realizados por los usuarios propios de la red, en cambio, los datos anómalos, o simplemente ataques, son los eventos causados deliberadamente por atacantes hacia la red. Además, esta última categoría se puede dividir en cuatro subcategorías: Denial of Service (DOS), User to Superuser o Root Attacks (U2R), Remote to User Attacks

(R2L) y Probing (Probe). Sin embargo, generalmente se utiliza solo una clase que engloba a los cuatro tipos mencionados bajo el nombre de “anómalo” o “ataque”, esto con el fin de realizar una clasificación binaria, en la que se contemplan dos categorías o clases.

A continuación se muestra una breve descripción de cada una de las “categorías malignas” incorporadas en KDD'99.

- **Denial of Service**
  Es una clase de ataque en donde el atacante hace que algún recurso de memoria o cómputo se sobreutilise o se llene logrando que no se puedan manejar las solicitudes legítimas, o “denegando” el acceso a usuarios legítimos. Algunos ejemplos son Apache2, Back, Land, Mail bomb, SYS Flood, Ping of death, Process table, Smurf, Syslogd, Teardrop y Udpstorm.

- **User to Superuser o Root Attacks**
  Es una clase de ataque en la que el atacante comienza con accediendo a una cuenta de usuario normal en el sistema y es capaz de explotar alguna vulnerabilidad para obtener acceso de root al sistema. Algunos ejemplos son Eject, Ffbconfig, Fdformat, Loadmodule, Perl, Ps y Xterm.

- **Remote to User Attacks**
  Es una clase de ataque en la que un atacante envía paquetes a una máquina sobre una red sin tener una cuenta de usuario y explota alguna vulnerabilidad para ganar acceso local tal como si fuese un usuario. Algunos ejemplos son Dictionary, Ftp_write, Guest, Imap, Named, Phf, Sendmail, Xlock y Xsnoop.

- **Probing**
  Es una clase de ataque donde el atacante escanea una red de computadores para obtener información o para encontrar vulnerabilidades conocidas. Una atacante con un mapa de las máquinas y servicios que están disponibles en una red puede usar esta información para buscar vulnerabilidades. Algunos ejemplos son Ipsweep, Mscan, Nmap, Saint y Satan.

A pesar de contar con el conjunto de datos KDD'99 original, este no será utilizado en dicho estado, esto debido a que posee ciertas características que influyen directamente los resultados obtenidos por el clasificador, lo que por ende, altera su desempeño de manera adversa o favorablemente. Dichos problemas fueron establecidos y demostrados por [9] el año 2009 tras un intenso análisis del conjunto de datos. Estos problemas se vieron reflejados al evaluar y comparar los datos originales con los mejorados frente a una serie de diversas técnicas de clasificación.

Uno de los problemas del conjunto original es que posee gran redundancia de datos, es decir, un registro se encuentra repetido una o más veces a largo del conjunto de datos al que pertenece. Este problema podría hacer que el aprendizaje de cada modelo se enfoque en los datos más frecuentes, y por lo tanto, le quite importancia a aquellos registros que por naturaleza no poseen una alta frecuencia, lo que podría ser un grave problema para un detector de intrusiones porque, eventualmente, se podría estar ignorando a un peligroso ataque. Por otra

parte, este problema podría beneficiar los resultados de aquellos clasificadores que tengan la capacidad de determinar correctamente datos de pruebas que se encuentran repetidos.

Un segundo problema, muy relacionado con el anterior, es que las cinco clases o categorías que dividen a los puntos de datos no se encuentran distribuidas equitativamente, es decir, que hay casos en que el número de datos de cierta categoría es muy desproporcional en comparación al resto. Esta característica podría beneficiar a aquellos modelos de clasificación que tengan la capacidad de clasificar correctamente a las categorías que tengan un mayor porcentaje de participación en el conjunto de datos de pruebas.

Ambas características podrían provocar algún tipo de tendencia en aquellos modelos que sean creados y puestos a evaluación bajo el uso de los conjuntos de datos con problemas. Además, cabe destacar que es muy probable que el uso de estos datos pueda influir de tal manera que cualquier resultado obtenido sea de poca fiabilidad, incluso llegando a niveles donde el desempeño de algunos modelos de clasificación se vea "favorecido".

Como respuesta a los problemas de evaluación que causan los datos originales, es que se utilizará una versión llamada NSL-KDD [10], la que no contiene redundancia en absoluto, se encuentra bien distribuida y, además, consta de algunas correcciones. Por lo tanto, se usará un conjunto de datos reducido que gracias a sus características producirá resultados de mucha mayor confianza. De manera complementaria, se utilizará el conjunto de datos de pruebas original, con el fin de hacer comparaciones con investigaciones que han utilizado dicho conjunto como base en sus desarrollos y, además, para realizar comparaciones de desempeño de un modelo de clasificación forjado con el conjunto de datos de entrenamiento en cuestión, frente a su versión mejorada, sin los problemas señalados.

Cabe destacar, que todos los logros alcanzados por el trabajo realizado en [9] serán de gran ayuda a todo el desarrollo de este proyecto, y posiblemente a todos las investigaciones relacionadas con el problema de clasificación con datos de IDS, ya que brindan un excelente punto de referencia para evaluar cualquier técnica que se utilice, posibilitando la realización de comparaciones fiables bajo las mismas condiciones y características.

A continuación se presentan algunas comparaciones entre las distintas versiones del conjunto de datos KDD'99. En la tabla Tabla 3.2 se muestran algunas estadísticas de los datos de entrenamiento y en la Tabla 3.3 de los datos de pruebas. En las dos tablas se logra exponer los problemas mencionados con anterioridad. A través de las columnas se puede percibir la amplia redundancia de datos y, a través de las filas, las grandes diferencias entre las cantidades de registros que posee cada categoría. Los datos fueron extraídos directamente del trabajado realizado por [9].

Tabla 3.2 Estadísticas de registros redundantes en el conjunto de datos de entrenamiento de KDD'99

	Registros Originales	Registros Distintos	Ratio de Reducción
Ataques	3.925.650	262.178	93,32%
Normal	972.781	812.814	16,44%
Total	4.898.431	1.074.992	**78,05%**

Tabla 3.3 Estadísticas de registros redundantes en el conjunto de datos de pruebas de KDD'99

	Registros Originales	Registros Distintos	Ratio de Reducción
Ataques	250.436	29.378	88,26%
Normal	60.591	47.911	20,92%
Total	311.027	77.289	**75,15%**

En ambas tablas (datos de entrenamiento y pruebas), se puede apreciar la gran cantidad de registros repetidos, reduciendo los conjuntos de datos originales al rededor de un impresionante **76,5%**. Respecto a las diferencias en distribución de los datos, se puede observar que es mucho mayor que la redundancia, sobre todo en los datos originales. Vale decir que en los conjuntos de entrenamiento y pruebas se puede apreciar que los ataques superan, aproximadamente, en un excesivo **400%** a los datos de la clase normal.

Es muy importante destacar que al reducir la redundancia o equilibrar la distribución de lo datos, no se busca simular el comportamiento real de una red ya que haciéndolo se logra todo lo contrario. Sin embargo, dicho procesamiento de datos es útil para conocer el desempeño real del clasificador, ya que ha clasificado de cierta manera un determinado registro, es muy probable que a la hora de evaluar un dato igual o con características similares, este tome la misma decisión, lo que podría llegar a influenciar los resultados obtenidos de manera positiva. En pocas palabras, lo que realmente se busca al preprocesar los datos de esta manera es asegurar que al momento de evaluar el desempeño del clasificador sus resultados sean lo más fiables posible.

Por otra parte, los creadores del conjunto de datos NSL-KDD quisieron añadir mayor dificultad a sus experimentos proporcionando los puntos de datos más difíciles de clasificar a las distintas técnicas de clasificación. Para esto, crearon aleatoriamente tres subconjuntos de datos de entrenamiento pertenecientes a KDD, con 15.000 registros cada uno, con el fin de evaluar el desempeño de las distintas técnicas frente a los conjuntos de datos sin redundancia NSL-KDD (1.074.992 para entrenamiento y 77.289 para pruebas) asociando a cada registro el "nivel de dificultad" de ser clasificado correctamente. El objetivo de esto fue crear subconjuntos de entrenamiento y pruebas dando prioridad (de manera inversamente proporcional) a los registros más difíciles de clasificar por las técnicas usadas.

En la Tabla 3.4 y Tabla 3.5 se muestra un listado indicando la cantidad de registros bien clasificados respecto a los 21 procesos de entrenamiento realizados (7 técnicas * 3 conjuntos de datos aleatorios de KDD), en donde la cantidad de procesos de la primera columna indica, aproximadamente, la cantidad de procesos que realizaron correctamente la clasificación de cierto registro. En la Tabla 3.4 se muestran los resultados obtenidos por la selección de registros respecto al conjunto de datos de entrenamiento y, posteriormente, en la Tabla 3.5, los que corresponde a los de pruebas.

Tabla 3.4 Estadísticas de registros seleccionados aleatoriamente de KDD de entrenamiento

Clasificaciones	Registros Distintos	Porcentaje	Registros Seleccionados
0-5	407	0,04	407
6-10	768	0,07	767
11-15	6.525	0,61	6.485
16-20	58.995	5,49	55.757
21	1.008.297	93,80	62.557
Total	1.074.992	100,00	**125.973**

Tabla 3.5 Estadísticas de registros seleccionados aleatoriamente de KDD de pruebas

Clasificaciones	Registros Distintos	Porcentaje	Registros Seleccionados
0-5	589	0,76	585
6-10	847	1,10	838
11-15	3.540	4,58	3.378
16-20	7.845	10,15	7.049
21	64.468	83,41	10.694
Total	77,289	100,00	**22.544**

Como se puede constatar en las tablas, los conjuntos de datos de entrenamiento y pruebas quedan reducidos a 125.973 y 22.544 respectivamente. Sin embargo, a pesar haber conseguido tal significativa reducción, el conjunto de datos de entrenamiento sigue siendo grande, es por esto que los autores decidieron considerar solamente el primer 20% de dichos datos como el conjunto de datos de entrenamiento base a utilizar, de esta manera en sus procesos de entrenamiento con las distintas técnicas se acabó utilizando solamente **25.192 registros**.

En el Capítulo 5 se muestra con mayor detalle los pasos realizados para preprocesar estos conjuntos de datos y, además, cuáles son los resultados obtenidos en base a los datos preprocesados luego de ser utilizados en los procesos de las etapas de entrenamiento y pruebas con las técnicas de clasificación y optimización contempladas en el presente proyecto.

En la siguiente sección se hará una reseña de la metodología utilizada para construir el modelo clasificación de datos, SVM. Se explicará el funcionamiento de la técnica y cómo es mejorada con la versión de mínimos cuadrados. Además, se dará a conocer la manera en que SVM trabaja con problemas no lineales, como es el caso del conjunto de datos mencionado en esta sección, dado el comportamiento caótico de los datos en una red, y cómo es posible lograr una separación efectiva de las distintas categorías en la fase de entrenamiento para, posteriormente, llevar a cabo una clasificación de datos que funcione correctamente sobre datos con patrones similares o totalmente desconocidos en la fase de pruebas.

3.2 Support Vector Machine

Support Vector Machine (SVM) es un conjunto de métodos relacionados de aprendizaje supervisado, que sirven para realizar clasificación y regresión de datos. Sin embargo, por la naturaleza y los intereses de este proyecto, SVM solo será utilizada para realizar clasificación binaria sobre el conjunto de datos de IDS mencionado en la sección anterior y no para realizar regresión con ellos.

SVM pertenece a la familia de algoritmos de clasificación lineal, no obstante, esta técnica puede ser modificada para resolver problemas no lineales como el caso de los datos de IDS. Su finalidad es separar y categorizar los datos de entrenamiento en las clases "normal" y "anómalo", de tal manera que cuando reciba un nuevo dato a partir de un supuesto evento, pueda ser capaz de determinar con exactitud a qué clase corresponde.

La forma en que SVM trabaja para separar las distintas clases o realizar un "entrenamiento", es a través de un hiperplano separador, de dimensión inferior a la del conjunto de datos. Pueden existir muchos hiperplanos, pero no se escoge cualquiera, sino que aquel que establezca la mayor separación entre las clases. Por ejemplo, dos conjuntos de datos pueden estar separados por infinitos hiperplanos (como se muestra de forma simplificada en la Ilustración 3.3), sin embargo SVM buscará aquel que posea un **margen máximo**, es decir, el que genere la mayor separación posible entre ambas clases.

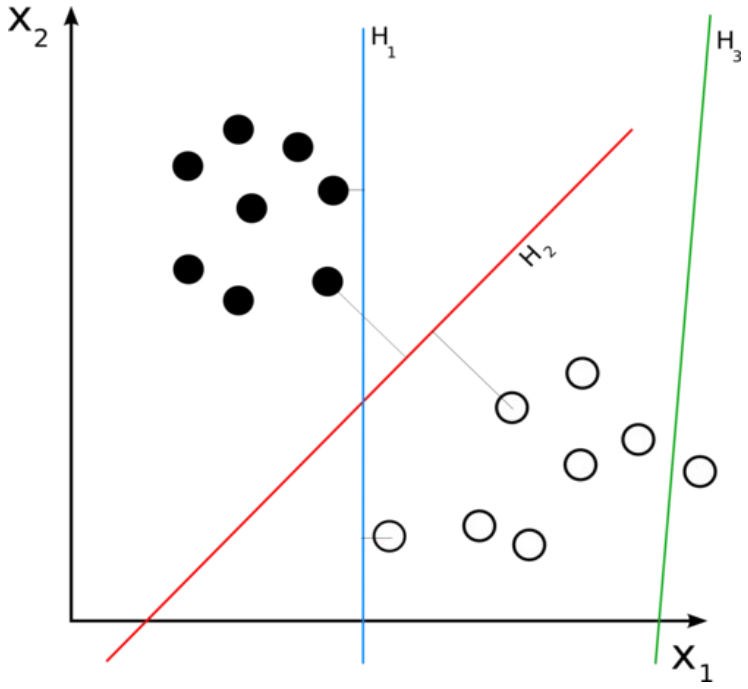

Ilustración 3.3.Hiperplanos separadores

Para poder clarificar el concepto mencionado anteriormente es que se utiliza como ejemplo el típico caso de un problema "linealmente separable", donde se tienen dos conjuntos de datos, descritos por dos características (bidimensional) que además, se pueden separar fácilmente con una linea recta. Dicho problema es equivalente a un hiperplano en un espacio bidimensional como se puede apreciar en la Ilustración 3.4.

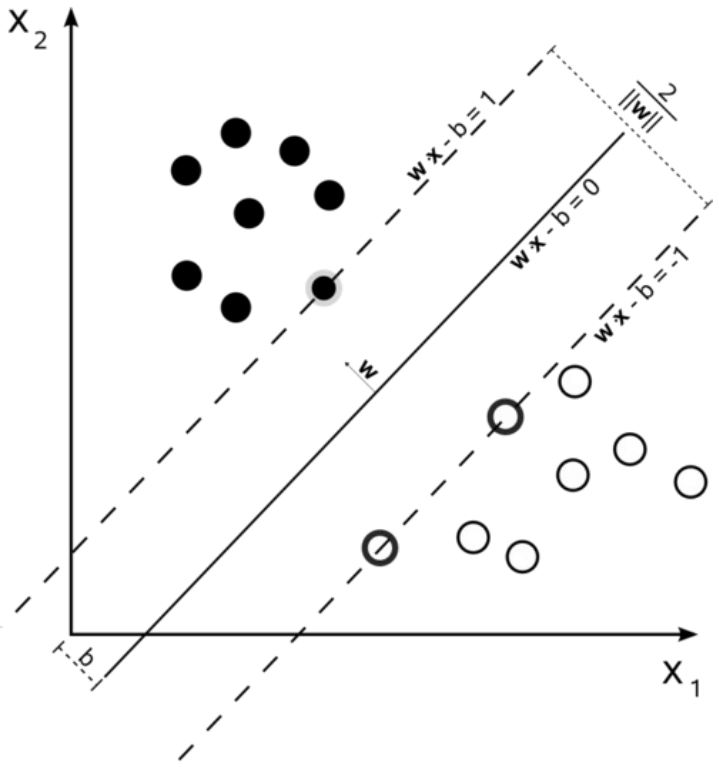

Ilustración 3.4.Hiperplano de margen máximo

Además, en la Ilustración 3.4 se puede apreciar que en paralelo al hiperplano de margen máximo $(\omega * x - b = 0)$ se encuentran dos vectores. Estos vectores son los llamados *"vectores de soporte"* ya que son los que "soportan" a cada clase de datos y, por lo tanto, son los que marcan el limite del margen máximo $(2/\|\omega\|)$.

En el caso de IDS, y como se ha mencionado con anterioridad, se puede clasificar los datos en dos grandes grupos, los datos normales, pertenecientes a los comportamientos computacionales comunes de una determinada organización frente a la red y los anómalos, que serían todos los restantes, en los cuales se pueden encontrar los diversos tipos de ataques conocidos como los que fueron presentados en distintas categorías en la Sección 3.1.4. Puede que esta descripción parezca simple, pero no lo es. Los datos de una red poseen un comportamiento estocástico y además, cada paquete consta de muchas características que lo describen, lo que hace que el problema de IDS sea de naturaleza no lineal.

SVM puede trabajar con datos no lineales y para lograr este objetivo realiza la separación de las categorías en un espacio de dimensión superior. Esta transformación es conocida como *"el truco del kernel"[11] y* consiste en mapear las observaciones no lineales originales en un espacio de dimensión superior *n*, donde en seguida es utilizado un hiperplano de dimensión *n*-1 para realizar la separación de las clases.

A modo de ejemplo, se puede suponer el caso de un problema de clasificación bidimensional donde la separación lineal de las categorías de datos no fuese posible en el espacio de origen. Al trasladar cada punto de datos desde el espacio de origen a un espacio de dimensión superior (*n*=3), eventualmente se podría realizar una clasificación lineal con un hiperplano de dimensión *n*=2 en aquel espacio nuevo, lo que a fin de cuentas, resultaría completamente equivalente a realizar una **clasificación no lineal** del problema en el espacio de origen. En la Ilustración 3.5 se representa el comportamiento del truco del kernel de dicho ejemplo.

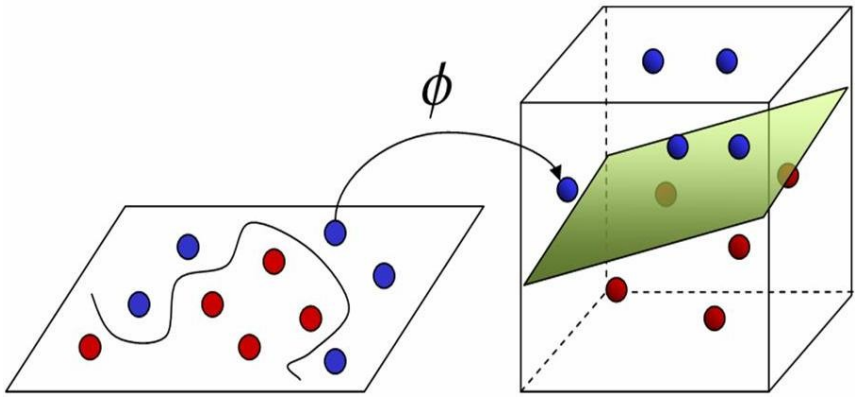

Ilustración 3.5.Truco del Kernel

Por consiguiente, luego de finalizada la separación de los datos o fase se entrenamiento, cuando se reciba un nuevo dato para ser analizado por SVM (no necesariamente en una fase de pruebas), este será mapeado directamente en el espacio de dimensión superior "*cayendo*" en alguna de las divisiones espaciales preestablecidas, o dicho de otro modo, realizando una clasificación como tal.

Nuevamente, se debe destacar que puede existir una gran cantidad de hiperplanos de dimensión *n-1* entre las clases (independiente de la dimensión del problema), sin embargo, se debe escoger aquel que establezca la separación máxima entre ambas categorías. Es en este punto donde se puede vislumbrar la necesidad de un proceso de optimización que sea capaz de dar solución a tal problema. Es decir, que pueda encontrar el hiperplano separador de margen **máximo** que permita la mayor disociación de las clases de datos no lineales que posee el problema de IDS.

A continuación se expone el proceso que realiza SVM para resolver problemas de clasificación. Cada detalle está basado en el trabajo de Suykens y Vandewalle [3].

Dado un conjunto de entrenamiento de N puntos de datos $\{y_k, x_k\}_{k=1}^N$, donde $x_k \in \Re^n$ es el k-ésimo parámetro de entrada e $y_k \in \Re$ es el k-ésimo parámetro de salida, la aproximación del método de vectores de soporte tiene como objeto construir un **clasificador** de la forma:

$$y(x) = sign[\sum_{k=1}^{N} \alpha_k y_k \psi(x, x_k) + b] \tag{5}$$

donde α_k son constantes reales positivas y b es una constante real. Para $\psi(\cdot,\cdot)$ se tienen las siguientes opciones:

- $\psi(x, x_k) = x_k^T x$ SVM lineal
- $\psi(x, x_k) = (x_k^T x + 1)^d$ SVM polinomial de grado d
- $\psi(x, x_k) = \exp(-\|x - x_k\|_2^2 / \sigma^2)$ SVM Radial Basis Function
- $\psi(x, x_k) = \tanh(k x_k^T x + \theta)$ SVM neuronal de doble capa

donde σ, k y θ son constantes

El clasificador se construye de la siguiente manera. Se asume que

$$\begin{cases} w^T \varphi(x_k) + b \geqslant 1, & si\ y_k = +1 \\ w^T \varphi(x_k) + b \leqslant -1, & si\ y_k = -1 \end{cases} \tag{6}$$

lo que es equivalente a

$$y_k [w^T \varphi(x_k) + b] \geqslant 1, \quad k = 1, \ldots, N \tag{7}$$

donde $\varphi(\cdot)$ es una función no lineal, la que mapea el espacio de entrada en un espacio de dimensión superior. Sin embargo, esta función no está construida explicitamente. Con el fin de tener la posibilidad de violar (7), en el caso de que un hiperplano separador **no exista** en este espacio de dimensión superior, se introducen las variables ξ_k de manera que

$$\begin{cases} y_k [w^T \varphi(x_k) + b] \geqslant 1 - \xi_k, & k = 1, \ldots, N \\ \xi_k \geqslant 0, & k = 1, \ldots, N \end{cases} \tag{8}$$

De acuerdo con el principio de minimización de riesgo estructural, el riesgo ligado es minimizado formulando el siguiente problema de optimización:

$$\min_{w,\xi_k} J_1(w,\xi_k)=\frac{1}{2}w^T w+c\sum_{k=1}^{N}\xi_k \tag{9}$$

sujeto a (8). Por consiguiente, se construye el Lagrariano

$$\mathscr{L}_1(w,b,\xi_k;\alpha_k,v_k)=J_1(w,\xi_k)-\sum_{k=1}^{N}\alpha_k\{y_k[w^T\varphi(x_K)+b]-1+\xi_k\}-\sum_{k=1}^{N}v_k\xi_k \tag{10}$$

introduciendo los multiplicadores de Lagrange $\alpha_k\geqslant 0,\ v_k\geqslant 0\ \ (k=1,\ldots,N)$. La solución se da por el punto de silla de montar, computando

$$\max_{\alpha_k,v_k}\ \min_{w,b,\xi_k}\ \mathscr{L}_1(w,b,\xi_k;\alpha_k,v_k) \tag{11}$$

se obtiene

$$\begin{cases}\dfrac{\partial\mathscr{L}_1}{\partial w}=0\rightarrow w=\sum_{k=1}^{N}\alpha_k y_k\varphi(x_k)\\ \dfrac{\partial\mathscr{L}_1}{\partial b}=0\rightarrow\sum_{k=1}^{N}\alpha_k y_k=0\\ \dfrac{\partial\mathscr{L}_1}{\partial\xi_k}=0\rightarrow 0\leqslant\alpha_k\leqslant c,k=1,\ldots N\end{cases} \tag{12}$$

lo que deja a la solución al siguiente problema de **programación cuadrática** (QA)

$$\max_{\alpha_k}Q_1(\alpha;\varphi(x_k))=\frac{-1}{2}\sum_{k,l=1}^{N}y_k y_l\varphi(x_k)^T\varphi(x_l)\alpha_k\alpha_l+\sum_{k=1}^{N}\alpha_k \tag{13}$$

de tal manera que

$$\begin{cases}\sum_{k=1}^{N}\alpha_k y_k=0\\ 0\leqslant\alpha_k\leqslant c,k=1,\ldots N\end{cases}$$

La función $\varphi(x_k)$ en (13) se relaciona a continuación a $\psi(x,x_k)$ mediante la imposición

$$\varphi(x)^T\varphi(x_k)=\psi(x,x_k) \tag{14}$$

la que es motivada por el teorema de Mercer. Notese que para la SVM neuronal de doble capa, la condición de Mercer sólo es válida para ciertos valores de los parámetros k y θ.

El clasificador (5) está diseñado por la solución de

$$\max_{\alpha_k} Q_1(\alpha_k; \psi(x_k, x_l)) = \frac{-1}{2} \sum_{k,l=1}^{N} y_k y_l \psi(x_k, x_l) \alpha_k \alpha_l + \sum_{k=1}^{N} \alpha_k \tag{15}$$

sujeto a las restricciones de (13). No es necesario calcular w o $\varphi(x_k)$ a fin de determinar la superficie de decisión. Debido a que la matriz asociada con este problema de programación cuadrática no es indefinida, la solución para (15) será global.

Por otra parte, se puede demostrar que los hiperplanos (7) que satisfagan la restricción $\|w\|_2 \leq a$ tienen una dimensión-VC h que está limitada por

$$h \leq min([r^2 a^2], n) + 1 \tag{16}$$

donde $[.]$ denota la parte entera y r es el radio de la bola más pequeña conteniendo los puntos $\varphi(x_1), \ldots, \varphi(x_N)$. El hallazgo de esta bola se hace mediante la definición de la función de Lagrange

$$\mathscr{L}_2(r, q, \lambda_k) = r^2 - \sum_{k=1}^{N} \lambda_k (r^2 - \|\varphi(x_k) - q\|_2^2) \tag{17}$$

donde q es el centro de la bola y λ_k son los multiplicadores de Lagrange positivos. En una forma similar como para (9) se encuentra que el centro es igual a $q = \sum_k \lambda \varphi(x_k)$, donde los multiplicadores de Lagrange derivan de

$$\max_{\lambda_k} Q_2(\lambda_k; \varphi(x_k)) = -\sum_{k,l=1}^{N} \varphi(x_k)^T \varphi(x_l) \lambda_k \lambda_l + \sum_{k=1}^{N} \lambda_k \varphi(x_k)^T \varphi(x_k) \tag{18}$$

tal que

$$\sum_{k=1}^{N} \lambda_k = 1$$
$$\lambda_k \geq 0, \; k = 1, \ldots N$$

Basándose en (14), Q_2 también puede ser expresado en términos de $\psi(x_k, x_l)$. Finalmente, se selecciona un SVM con dimensión VC mínima mediante la solución de (15) y el computo de (16) de (18).

3.2.1 Least Squares Support Vector Machine

Ahora, y continuando con el trabajo de Suykens y Vandewalle [3], se introduce la versión de mínimos cuadrados al clasificador SVM mediante la formulación del problema de clasificación de la siguiente forma:

$$\min_{w,b,e} J_3(w,b,e)=\frac{1}{2}w^T w+\gamma\frac{1}{2}\sum_{k=1}^{N} e_k^2 \tag{19}$$

sujeto a las restricciones de igualdad

$$y_k[w^T\varphi(x_k)+b]=1-e_k, \quad k=1,\ldots,N \tag{20}$$

Las diferencias importantes con SVMs son las restricciones de igualdad y la suma del término de **error cuadrático** e_k, el que simplifica el problema extremadamente.

La solución se obtiene luego de construir el Lagrariano:

$$\mathscr{L}_3(w,b,e;\alpha)=J_3(w,b,e)-\sum_{k=1}^{N}\alpha_k\{y_k[w^T\varphi(x_k)+b]-1+e_k\} \tag{21}$$

Donde α_k son los multiplicadores de Lagrange (que ahora pueden ser negativos o positivos debido a las restricciones de igualdad).

Las condiciones para optimizar

$$\begin{cases} \frac{\partial \mathscr{L}_3}{\partial w}=0 \quad \rightarrow \quad w=\sum_{k=1}^{N}\alpha_k y_k\varphi(x_k) \\ \frac{\partial \mathscr{L}_3}{\partial b}=0 \quad \rightarrow \quad \sum_{k=1}^{N}\alpha_k y_k=0 \\ \frac{\partial \mathscr{L}_3}{\partial e_k}=0 \quad \rightarrow \quad \alpha_k=\gamma e_K, \qquad k=1,\ldots N \\ \frac{\partial \mathscr{L}_3}{\partial \alpha_k}=0 \quad \rightarrow \quad y_k[w^T\varphi(x_k)+b]-1+e_k=0, \quad k=1,\ldots N \end{cases} \tag{22}$$

pueden ser inmediatamente escritas como la solución del siguiente conjunto de ecuaciones lineales

$$\left[\begin{array}{ccc|c} I & 0 & 0 & -Z^T \\ 0 & 0 & 0 & -Y^T \\ 0 & 0 & \gamma I & -I \\ \hline Z & Y & I & 0 \end{array}\right]\left[\begin{array}{c} w \\ b \\ e \\ \hline \alpha \end{array}\right]=\left[\begin{array}{c} 0 \\ 0 \\ 0 \\ \hline \vec{1} \end{array}\right] \tag{23}$$

Donde $Z=[\varphi(x_1)^T y_1;\ldots;\varphi(x_N)^T y_N]$, $Y=[y_1;\ldots;y_N]$, $\vec{1}=[1;\ldots;1]$, $e=[e_1;\ldots;e_N]$, $\alpha=[\alpha_1;\ldots;\alpha_N]$. Al eliminar ω, e, la solución también está dada por

$$\left[\begin{array}{c|c} 0 & -Y^T \\ \hline Y & ZZ^T+\gamma^{-1}I \end{array}\right]\left[\begin{array}{c} b \\ \hline \alpha \end{array}\right]=\left[\begin{array}{c} 0 \\ \hline \vec{1} \end{array}\right] \tag{24}$$

La condición de Mercer se puede volver a aplicar a la matriz $\Omega=ZZ^T$ donde

$$\begin{aligned}\Omega_{kl}&=y_k y_l \varphi(x_k)^T\varphi(x_l)\\&=y_k y_l \psi(x_k,x_l)\end{aligned} \tag{25}$$

Por lo tanto, el clasificador (5) es encontrado mediante la solución de un conjunto de **ecuaciones lineales** (24)-(25) en vez de **programación cuadrática**. Los parámetros del kernel como σ (para el kernel RBF) pueden ser optimizados escogiendo de acuerdo a (16). Los valores de soporte α_k son proporcionales a los errores en los puntos de datos (22), mientras que en el caso de (18) la mayoría de los valores son iguales a cero. Por consiguiente, se puede hablar con mayor precisión de un espectro de valores de soporte en el caso de la versión de mínimos cuadrados de SVM.

En conclusión a lo anterior, el método de mínimos cuadrados es útil para simplificar significativamente el trabajo que realiza SVM para elaborar un modelo de clasificación. Esto se traduce en un mejor desempeño del proceso de clasificación y por ende, en un menor costo computacional. Además, vale decir que gracias a lo anterior, se puede aportar en gran medida a una tarea tan crítica como la construcción de un clasificador para detección de intrusiones.

Como se mencionó recientemente hay ciertos parámetros que se pueden optimizar de manera que el clasificador pueda aproximarse a una clasificación de mayor exactitud al momento de enfrentarse a datos nuevos (con patrones desconocidos o poco similares a los aprendidos en el entrenamiento). El objeto de la optimización de dichos parámetros es lograr incrementar la **capacidad de generalización** propia del modelo de clasificación.

Los parámetros involucrados en el proceso de clasificación son el de compensación o penalidad y los parámetros propios del kernel a usar. El parámetro de compensación se encarga de penalizar a aquellos puntos de datos que no se pueden separar fácilmente y los lleva a su clase correspondiente asignándoles un costo que determina el esfuerzo realizado para ello. Por otra parte, los parámetros del kernel que se utilice influyen en la complejidad y holgura que tome la superficie separadora en el espacio en base a cada punto de dato.

Cabe mencionar que la elección de estos parámetros podría conllevar al **sobreajuste del modelo**, disminuyendo la efectividad de este al momento de clasificar datos desconocidos, ya que al sobreajustarse se limitaría solamente a clasificar bien los datos conocidos. Es por esto que se recomienda ser cuidadoso en los procedimientos de ajustes de parámetros.

En la siguiente sección se expone una metaheurística llamada PSO, la que a través de su peculiar comportamiento ayudará en la obtención de dicha combinación de parámetros, con el fin de mejorar la efectividad del modelo de clasificación basado en LS-SVM.

3.3 Particle Swarm Optimization

Particle Swarm Optimization (PSO), es una metaheurística desarrollada por Eberhart y Kennedy [4], inspirada por el comportamiento social de las bandadas de pájaros y cardúmenes de peces. En PSO un número de entidades simples (las partículas) son puestas en el espacio de búsqueda de algún problema o función, y con cada una de las locaciones actuales de ellas se evalúa la **función objetivo**. Luego, cada partícula determina su movimiento a través del espacio de búsqueda, combinando algún aspecto de la historia de su locación actual con la mejor locación de uno o más miembros del enjambre. A esto se le agrega alguna perturbación al azar.

La siguiente iteración toma lugar después de que todas las partículas han sido movidas. Eventualmente, el enjambre en su totalidad, como una bandada de pájaros en busca de alimentos, probablemente se acerque a un óptimo de la función objetivo.

Cada individuo en el enjambre de partículas se compone por tres vectores de dimensión D, donde D es la dimensionalidad del espacio de búsqueda. Estos son la posición actual $\vec{x}_i$, la mejor posición anterior, $\vec{p}_i$ y la velocidad $\vec{v}_i$. En este proyecto la dimensionalidad del tamaño de búsqueda corresponderá al número de parámetros que requiera LS-SVM, es decir, uno, por el parámetro de compensación, más la cantidad de parámetros dependientes del kernel a utilizar.

La posición actual $\vec{x}_i$ puede ser considerada como el conjunto de coordenadas que describe a un punto en el espacio. En cada iteración del algoritmo, la posición actual es evaluada como una solución del problema. Si la posición es mejor que cualquiera que se haya encontrado hasta ahora, entonces las coordenadas son guardadas en el segundo vector, $\vec{p}_i$. El **valor del mejor resultado** evaluado hasta ahora en la función es guardado en una variable que puede ser llamada $pbest_i$ (por "previous best") para comparaciones en iteraciones posteriores. El objetivo, es seguir buscando mejores posiciones y actualizando $\vec{p}_i$ y $pbest_i$. Luego, se escogen nuevos puntos en el espacio añadiendo las coordenadas de $\vec{v}_i$ a $\vec{x}_i$. El algoritmo opera ajustando $\vec{v}_i$, el cual puede ser efectivamente visto como el largo del paso.

El enjambre de partículas es más que sólo una colección de partículas. Una partícula por si misma casi no tiene poder para resolver cualquier problema; el progreso ocurre sólo cuando las partículas interactúan [12]. La resolución de problemas es un fenómeno de la población en general, emergiendo del comportamiento individual de las partículas a través de sus interacciones. En ningún caso, las poblaciones son organizadas de acuerdo a algún orden de estructura comunicacional o topología, como las redes sociales. La topología consiste típicamente en esquinas bidireccionales conectando pares de partículas, por lo tanto si j está en la vecindad de i, i también está en la de j. Cada partícula se comunica con alguna otra partícula y es afectada por el mejor punto encontrado por cualquier miembro de su vecindad topológica. Esto es sólo el vector $\vec{p}_i$ por el mejor vecino, el cual se denotará con $\vec{p}_g$. Los posibles tipos de poblaciones "redes sociales" son muy variadas, pero en la práctica se han usado ciertos tipos más frecuentemente.

En el proceso de optimización de enjambre de partículas, la velocidad de cada partícula se ajusta iterativamente de manera que la partícula oscila estocásticamente alrededor de las locaciones $\vec{p}_i$ y $\vec{p}_g$.

Existe una serie de variantes del algoritmo PSO [12][13] (Adaptativo, Híbrido, etc.), sin embargo, para esta investigación serán puestas a prueba solamente las principales. A continuación se expone el algoritmo original y además, las principales variantes que han evolucionado desde su aparición.

3.3.1 Algoritmo Original

i. Inicializar una población de partículas con posiciones y velocidades aleatorias en las *D* dimensiones del espacio de búsqueda.

ii. Para cada partícula, evaluar la función fitness en *D* variables.

iii. Comparar el valor fitness de las partícula con su mejor posición anterior $(\vec{p}_i)$. Si el valor actual es mejor que su $pbest_i$, entonces se debe asignar el valor actual al $pbest_i$, y a su mejor posición anterior $(\vec{p}_i)$ la locación actual $(\vec{x}_i)$ en el espacio de búsqueda.

iv. Identificar la partícula en la vecindad con el mejor éxito hasta el momento, y asignar su índice a la variable $\vec{p}_g$.

v. Cambiar la velocidad y posición de la partícula de acuerdo a las siguientes ecuaciones:

$$\vec{v}_i \leftarrow \vec{v}_i + \vec{U}(0,\phi_1)\otimes(\vec{p}_i-\vec{x}_i) + \vec{U}(0,\phi_2)\otimes(\vec{p}_g-\vec{x}_i) \quad (26)$$

$$\vec{x}_i \leftarrow \vec{x}_i + \vec{v}_i \quad (27)$$

vi. En el caso de que se cumpla cierto criterio de detención (usualmente un buen fitness o un número máximo de iteraciones), entonces finalizar. Sino, volver al 2° paso.

Observación:

- $\vec{U}(0,\phi_i)$ representa un vector formado por números aleatorios uniformemente distribuidos en $[0,\phi_i]$, el que es generado aleatoriamente en cada iteración y para cada partícula. ϕ_1 y ϕ_2 son conocidos como coeficientes de aceleración.
- $\otimes$ multiplicación por componentes
- En la versión original de PSO, cada componente de v_i se mantiene dentro del rango $[-V_{max},+V_{max}]$

En PSO se habla de **exploración** cuando las partículas se distribuyen ampliamente en el espacio en busca de posibles valores óptimos, en cambio, cuando se habla sobre **explotación** se refiere a una concentración de la búsqueda en una determinada región del espacio. En el algoritmo original, el valor del parámetro V_{max} fue agregado con el fin de influenciar el balance entre la exploración y la explotación. Sin embargo, esto condujo a algunos problemas, ya que al ser implementado, la trayectoria de las partículas falló en converger. Donde se esperaba pasar de los pasos de gran escala, que caracterizan la búsqueda exploratoria, hacia una más fina o búsqueda enfocada en la explotación, V_{max} simplemente cortó las oscilaciones de las partículas.

3.3.2 Peso de Inercia

Motivados por el deseo de controlar mejor el alcance de la búsqueda, reducir la importancia de V_{max}, y tal vez eliminarla por completo, en [14] se propuso la siguiente modificación en las ecuaciones con el fin de actualizar el comportamiento o la evolución de las partículas de PSO:

$$\vec{v}_i \leftarrow \omega \vec{v}_i + \vec{U}(0,\phi_1) \otimes (\vec{p}_i - \vec{x}_i) + \vec{U}(0,\phi_2) \otimes (\vec{p}_g - \vec{x}_i) \tag{28}$$

donde ω fue llamado como *"peso de inercia"*. Al interpretar $\vec{U}(0,\phi_1) \otimes (\vec{p}_i - \vec{x}_i) + \vec{U}(0,\phi_2) \otimes (\vec{p}_g - \vec{x}_i)$ como una fuerza externa, $\vec{f}_i$, actuando sobre una partícula, luego la aceleración de la partícula puede ser escrita como $\Delta \vec{v}_i = \vec{f}_i - (1-\omega)\vec{v}_i$. Es decir, la constante $(1-\omega)$ actúa eficazmente como coeficiente de fricción, y por lo tanto, ω se puede interpretar como la fluidez del medio en el cual se mueve una partícula. Esto quizás explica por qué los investigadores han encontrado que el mejor funcionamiento podría ser obtenido, inicialmente, fijando ω a algún valor relativamente alto (ej., 0,9), el que corresponde a un sistema donde las partículas se mueven en un medio de viscosidad baja realizando una exploración extensiva, para luego, ir reduciendo gradualmente ω a un valor mucho más bajo (ej., 0,4), donde se representaría un medio de carácter más disipante y explotativo, lo que sería mejor al dirigirse a un óptimo local. Cabe destacar que es posible iniciar el algoritmo con valores superiores ($\omega > 1$), los que podrían hacer el enjambre inestable, siempre y cuando dicho valor se reduzca lo suficiente para llevar al enjambre a una región estable.

Con una elección apropiada de ω y de los coeficientes de aceleración ϕ_1 y ϕ_2, el comportamiento del algoritmo PSO se puede hacer mucho más estable, tanto así que se puede realizar sin la necesidad del valor V_{max} o establecerlo a un valor mucho más alto, tal como el valor del rango de cada variable (en cada dimensión). En este caso, V_{max} podría llegar a mejorar el rendimiento, aunque con el uso de la técnica que incluye el peso de inercia, ya no es necesario el uso de esta variable para lograr amortiguar la dinámica de las partículas del enjambre.

3.3.3 Coeficientes de Constricción

Aunque los primeros investigadores reconocieron que era necesario el uso de una forma de amortiguar la dinámica de las partículas, no se tenía absoluta claridad de los motivos. Sin embargo, al no utilizar estos amortiguadores (por ej. V_{max}) se provocaba inestabilidad en el enjambre al cabo de pocas iteraciones. Posteriormente, en un análisis realizado por [15] se estableció una estrategia para la colocación de unos "coeficiente de constricción" en términos de las fórmulas; estos coeficientes controlaron la convergencia de la partícula y permitieron un método elegante y bien explicado para prevenir la explosión, asegurar la convergencia, y eliminar el uso arbitrario del parámetro V_{max}. Dichos métodos fueron nombrados como "Clerc's Constrictions".

Una de las maneras más simples para incorporar los coeficientes de constricción es la siguiente:

$$\vec{v}_i \leftarrow \chi(\vec{v}_i + \vec{U}(0,\phi_1)\otimes(\vec{p}_i-\vec{x}_i) + \vec{U}(0,\phi_2)\otimes(\vec{p}_g-\vec{x}_i)) \tag{29}$$

donde $\phi = \phi_1+\phi_2 > 4$ y

$$\chi = \frac{2}{\phi-2+\sqrt{(\phi^2-4\phi)}} \tag{30}$$

Cuando se usa el método de constricción de Clerc, el valor de ϕ (la suma de ϕ_1 con ϕ_2) es establecido, comúnmente, a 4,1 y, así, la constante de multiplicación χ es aproximadamente 0,7298. Por lo tanto, la velocidad anterior terminará siendo multiplicada por 0,7298 y cada uno de los términos, o componentes, $(\vec{p}-\vec{x})$ multiplicados por un número aleatorio limitado por $0{,}7298*2{,}05 \approx 1{,}49618$.

Las partículas restringidas convergerán sin la necesidad de usar algún valor V_{max}. Sin embargo, experimentos y aplicaciones realizadas posteriormente [16], han concluido que un mejor enfoque, como regla de oro prudente para el uso de la metaheurística, es limitar el valor de V_{max} a X_{max}, el rango dinámico de cada variable en cada dimensión, en conjunto con (29) y (30). El resultado de esto es un algoritmo PSO sin parámetros con problemas específicos. En base a estas cualidades, dicho algoritmo se convertiría en lo que hoy es conocido como el PSO canónico.

Cabe mencionar que un algoritmo PSO con coeficientes de constricción posee una equivalencia algebraica respecto a un PSO con peso de inercia. De hecho, las ecuaciones (28) y (29) se pueden transformar de una a la otra mediante el mapeo de los parámetros $\omega \Leftrightarrow \chi$ y $\phi_i \Leftrightarrow \chi\phi_i$. Por lo tanto, los ajustes óptimos sugeridos por Clerc [15], para lograr dicho propósito, en el PSO con peso de inercia, son establecer el valor $\omega = 0{,}7298$ y $\phi_1 = \phi_2 = 1{,}49618$.

3.4 Cross-validation: la interfaz entre PSO y LS-SVM

Tras conocer las técnicas para conformar el modelo de clasificación, ahora es el momento para exponer el modo de interacción entre ellas o, bajo la perspectiva de PSO, la **función objetivo o fitness**. La idea es utilizar una función que proporcione resultados de alta confianza para asegurar la calidad o bondad del modelo al ser puesto en marcha.

Luego de una extensa investigación sobre métodos para seleccionar modelos de clasificación se decidió profundizar en cross-validation y alguna variantes, ya que, a pesar de ser técnicas costosas, se caracterizan por la fiabilidad de sus estimaciones.

El objetivo de cross-validation es entregar una estimación de alta confianza del desempeño de un modelo de clasificación. Para lograrlo, esta se basa, simplemente, en un proceso iterativo cuyo principio fundamental es mantener la **representatividad de la muestra**. Es decir, que a partir de un conjunto de datos determinado y un proceso de estimación de calidad, se obtengan evaluaciones fiables sobre el desempeño de los modelos.

Cross-validation [17] es un método estadístico de evaluación y comparación de algoritmos de aprendizaje, opera dividiendo un determinado conjunto de datos en 2 segmentos, uno usado para entrenar el modelo y el otro para medir su desempeño (estimación del costo) o **validarlo**. En esta técnica, los conjuntos de entrenamiento y validación son cruzados en rondas sucesivas, de tal modo, que cada punto de datos tenga la oportunidad de ser validado en contra.

La forma básica de cross-validation es conocida como *"k-fold cross-validation"*, donde el conjunto de datos a evaluar se divide en *k* partes, idealmente iguales, llamadas pliegues o "*folds*". Posteriormente, se realizan *k* iteraciones que se componen de un entrenamiento y una validación, de manera tal que en cada iteración se tome un segmento distinto para ser validado. Luego de concluir las *k* validaciones cruzadas basta con promediar los costos obtenidos. El procedimiento se ejemplifica a continuación en la Ilustración 3.6.

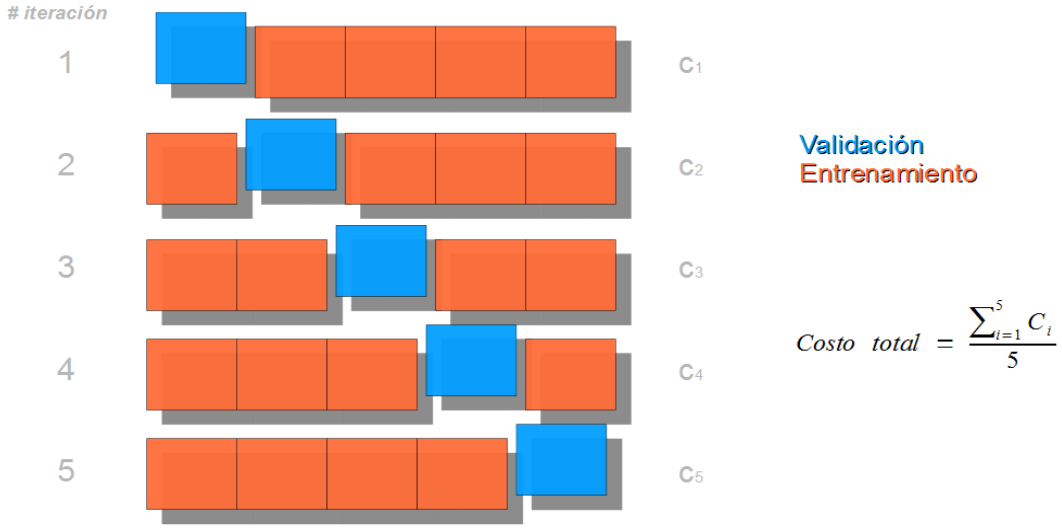

Ilustración 3.6. "5-fold Cross-validation"

Según Kohavi [18], "*stratified ten-fold cross-validation"* es el mejor método para seleccionar un modelo de clasificación de un determinado conjunto de datos. La cualidad de **estratificado**, se refiere a que cada uno de los segmentos es proporcional, en número de categorías, al conjunto base. Por ejemplo, si el conjunto de datos usado tiene 60% de datos de la clase A y 40% de la clase B, los subconjuntos (folds) idealmente deberán tener las mismas proporciones de clases de registros a disposición. Dicho autor recomienda utilizar la técnica mencionada, ya que es mejor considerada que el cross-validation común en términos de sesgo y varianza estadística.

En el siguiente capítulo se expondrán algunos trabajos que se encuentran directamente relacionados con el tema tratado en este proyecto, la mayoría realizados recientemente. Varios de ellos aplican SVM y algoritmos de PSO, como el convencional o la variante con peso de inercia para distintos problemas, inclusive para el de IDS. Sin embargo, en ninguno de estos trabajos se da énfasis sobre los métodos de estimación de desempeño en los modelos de clasificación construidos, lo que no proporciona resultados de gran confianza ni posibilita la realización de comparaciones detalladas con otros modelos.

4. Estado del Arte

En este capítulo son consideradas algunas investigaciones o trabajos que se han realizado últimamente y se encuentran relacionados fuertemente con el tema del presente proyecto. Esto es, con el fin de conocer los avances que se han alcanzado en la actualidad, respecto al problema de IDS, a través del uso de algoritmos de aprendizaje supervisado. Además, será de utilidad para establecer referencias, seguir buenas prácticas y también, para poner en clara evidencia qué es lo que se debe o no hacer.

En primera instancia se expondrá la investigación que ha proporcionado mayor utilidad en el desarrollo de este proyecto, ya que de ella se obtuvo la serie de conjuntos de datos utilizados y, además, un gran punto de referencia para realizar comparaciones. Además, se detallará parte del análisis y los resultados obtenidos con dichos conjunto de datos. Posteriormente, se mostrarán trabajos sobre SVM en conjunto con IDS que fueron considerados por su utilidad. Esto basándose en las prácticas observadas, algunas consideradas como buenas o beneficiosas y otras, claramente, como malas o incluso contraproducentes. Finalmente, se expondrán distintas variantes de la metaheurística PSO, que a la fecha, han sido consideradas como las más efectivas a la hora de conseguir tu objetivo.

4.1 Distintas técnicas de clasificación frente a KDD'99

En el trabajo realizado por [9] en 2009 (mencionado en la Sección 3.1.4 como fuente del conjunto de datos a utilizar), se pusieron a prueba siete técnicas conocidas de machine learning sobre el conjunto de datos KDD'99, estas fueron: J48 decision tree learning, Naive Bayes, NBTree, Random Forest, Random Tree, Multilayer Perceptron y Support Vector Machine (todas estas técnicas se pueden encontrar y utilizar abiertamente en la colección del software Weka [19]). Lo que merece ser destacado es que tras un extenso análisis y previo a la ejecución de dichas técnicas, los autores elaboraron conjuntos de datos de entrenamiento y pruebas con características especiales con el fin de conocer las cualidades reales de cada una.

Los resultados de los modelos “representantes” de cada técnica utilizada en dicha investigación serán de gran utilidad en la etapa de análisis de resultados, ya que, como se mencionó en el capítulo anterior, en este proyecto se utilizarán los mismos conjuntos de datos en el proceso de construcción y puesta en marcha de los modelos, y por lo tanto, se podrán realizar comparaciones con un alto grado de confianza entre los resultados exhibidos por cada modelo representante y todos aquellos modelos que sean construidos en base a dichos datos.

Para evaluar las técnicas seleccionadas por los autores para resolver el problema de IDS, se conformaron y utilizaron tres conjuntos de pruebas distintos. El primer conjunto, nombrado KDDTest, corresponde a un segmento del conjunto de datos de pruebas de KDD99 (el conjunto de datos original). El segundo conjunto, llamado $KDDTest^{+}$ es una versión reducida o mejorada del anterior (como se vio en la Sección 3.1.4). Finalmente, y no menos importante, el tercer conjunto de datos empleado, nombrado $KDDTest^{-21}$, fue elaborado inteligentemente en base a los registros más difíciles de clasificar por los 21 procesos de entrenamiento llevados

a cabo por las 7 técnicas de aprendizaje sobre 3 distintos subconjuntos de datos de entrenamiento. Es decir, el segmento de datos donde los registros no fueron correctamente clasificados por los 21 procesos de entrenamiento sino que por 20 de ellos o menos.

En las siguientes imágenes, Ilustración 4.1 e Ilustración 4.2, se muestran los resultados obtenidos producto de la evaluación de cada técnica frente a los dos primeros conjuntos de datos de pruebas mencionados en el párrafo anterior, KDDTest y KDDTest$^+$ respectivamente.

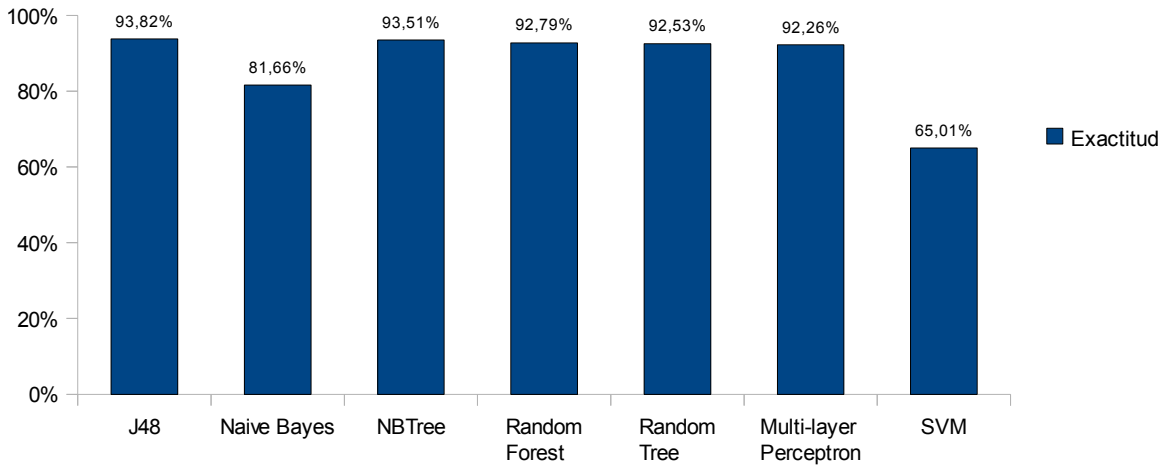

Ilustración 4.1. Desempeño de los algoritmos de aprendizaje seleccionadas sobre KDDTest

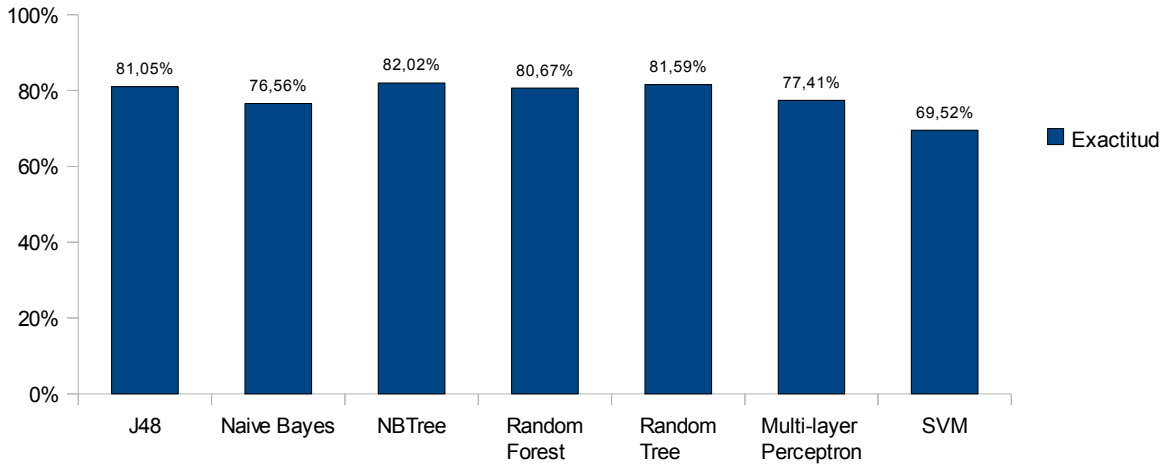

Ilustración 4.2. Desempeño de los algoritmos de aprendizaje seleccionadas sobre KDDTest$^+$

En las ilustraciones se puede observar que en los resultados de SVM, en términos de exactitud, son los menores alcanzados entre las siete técnicas utilizadas. Sin embargo, es la única que tuvo una **variación positiva** (4,51%) respecto de los datos originales a los

modificados, puesto que las otras técnicas se ven afectadas "favorablemente" con el alto porcentaje de puntos de datos repetidos en las pruebas, o dicho de otro modo y según lo concluido por los autores, que SVM fue la única técnica en clasificar incorrectamente en varias oportunidades uno o varios registros que se encontraban repetidos en el conjunto de pruebas, para luego, frente al conjunto KDDTest^{+}, volver a clasificarlos mal, pero en esta nueva oportunidad, solo una única vez dada la no redundancia de registros. Por otra parte, absolutamente todas las técnicas restantes presentaron una **variación negativa** entre 5,10% y 14,85%, lo cual, para algunos casos, es extremadamente significativo. Por último, y a pesar de la variación adversa en sus resultados, cabe destacar que el algoritmo NBTree tuvo un alto desempeño en las pruebas realizadas con KDDTest^{+}.

El último conjunto de datos puesto a prueba KDDTest^{-21} fue evidentemente el más complejo de clasificar. En la Ilustración 4.3 se pueden observar los resultados obtenidos por las distintas técnicas, donde se puede apreciar una disminución del desempeño a nivel general. Además, hay que destacar que nuevamente NBTree es el algoritmo que alcanza la mayor exactitud con un 66,16%.

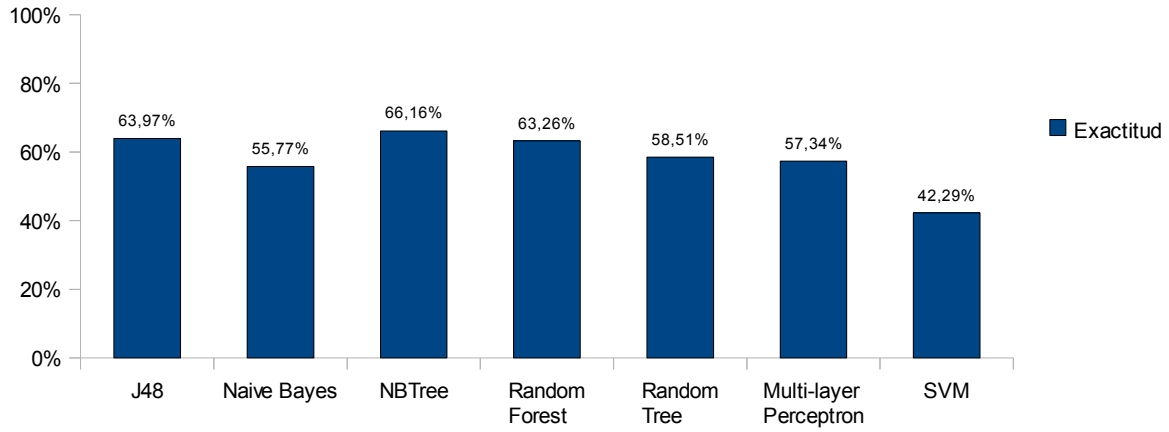

Ilustración 4.3. Desempeño de los algoritmos de aprendizaje seleccionadas sobre KDDTest^{-21}

Un punto importante a destacar respecto al trabajo citado es que en los distintos métodos de clasificación empleados, se utilizaron (en los casos posibles) **parámetros por defecto** provistos por el software utilizado, Weka. Como ya se ha comentado con anterioridad, estos influyen directamente en el comportamiento y, por ende, en los resultados que producen las técnicas. Se debe hacer énfasis que en este proyecto se desea realizar un modelo de clasificación bastante distinto y mejor, en comparación al SVM del trabajo aludido. Los resultados expuestos no poseen relación alguna con los que se puedan obtener bajo la construcción de otros modelos ya que uno de los objetivos principales que se persigue aquí es la optimización de los parámetros y, por lo tanto, mejorar el desempeño del algoritmo de aprendizaje supervisado LS-SVM.

4.2 C4.5 v/s SVM

En el año 2009, se hizo una comparación entre los resultados del ganador de la competencia KDD'99, árboles de clasificación (o arboles de decisión) y las máquinas de vectores de soporte [20]. En la investigación se realizaron comparaciones en términos de exactitud, tasa de detección, tasa de falsas alarmas y la exactitud de otros ataques en diferentes proporciones de datos de carácter normal. Los datos utilizados para los procesos de entrenamiento y pruebas fueros los del conjunto de datos original KDD'99, ya que en base a dicho fundamento se podrían realizar comparaciones con la técnica ganadora de la competencia.

En tanto a los resultados obtenidos, se puede destacar que árboles de clasificación (con el algoritmo C4.5) fue superior a SVM respecto a la exactitud, sus promedios fueron 64,94% y 62,70% respectivamente y en tanto a la tasa de detección C4.5 vuelve a superar a SVM, sus promedios fueron 70,62% y 58,68% respectivamente. No obstante, SVM obtuvo los mejores resultados respecto de la tasa de falsas alarmas. En la Tabla 4.1 se expone una comparación en detalle entre C4.5 y SVM respecto a la tasa de falsas alarmas.

Tabla 4.1 Comparación de la tasa de falsas alarmas entre C4.5 y SVM

Porcentaje de data normal (%)	C4.5 (%)	SVM (%)
10	3,50	2,60
20	2,25	2,10
30	2,13	2,57
40	1,33	0,65
50	1,14	1,44
60	0,95	1,60
70	1,17	0,47
80	1,78	0,26
90	1,23	0,63
Promedio	**1,44**	**1,00**

Los resultados indican que la posición que adoptó SVM, en comparación a la del algoritmo C4.5, fue, en gran parte, menos conservadora a través de las distintas cantidades de datos normales involucrados. En base a esto y su baja tasa de detección se puede concluir que, al momento de analizar los datos, SVM clasificó con una tendencia hacia los eventos normales. Es decir, que se clasificaron muchos eventos normales como tal. SVM tuvo mejores tasas de falsas alarmas que C4.5, esto se hace más notorio en los casos en donde los porcentajes de data normales fueron menores.

Cabe destacar que en el trabajo mencionado se utilizó una librería llamada Libsvm [21] para implementar SVM y que se utilizó una función llamada grid para encontrar sus parámetros (similares a los de LS-SVM). Un proceso de baja complejidad y poco eficaz, desde el punto de vista de la optimización, no obstante, esto no implica que al utilizar la versión de mínimos cuadrados de SVM o cambiando la técnica para la obtención de parámetros a algo más elaborado o avanzado como PSO u otro algoritmo de optimización, necesariamente se

obtendrán mejores resultados, esto no es una regla y, por ende, no siempre se cumple. Además, que un algoritmo mejore un determinado proceso no quiere decir que también sirva para mejorar procesos de otra naturaleza.

Hay que tener en cuenta que LS-SVM es sólo otra versión de SVM que en teoría vuelve más eficiente su trabajo, no su efectividad. Además, hay que destacar que agregar o utilizar nuevas técnicas no implica necesariamente mejor desempeño, ya que eventualmente se pueden tener las mejores herramientas, pero sin el correcto uso de ellas jamás se lograrán los resultados esperados.

Finalmente se puede decir que SVM pierde otra vez frente a una técnica de clasificación, en esta oportunidad no por mucho. El algoritmo C4,5 superó a SVM en 2,24% en términos de exactitud. Tal vez los parámetros de SVM no fueron escogidos correctamente o, probablemente, el algoritmo C4.5 se vio afectado positivamente por la gran redundancia de registros inherente al conjunto de datos original.

4.3 SVM con PSO para IDS

En una investigación llamada *"A Real-time Intrusion Detection System Based on PSO-SVM"* realizada por [22] en 2009, se utilizó un PSO híbrido, el cual es una mezcla entre el PSO convencional usado para optimizar, en este caso los parámetros de SVM y además, un PSO binario usado para la selección de características representativas de los conjuntos de datos. Con el solo hecho de leer el título ha de parecer el trabajo más cercano al de este proyecto, no obstante no presentó mayores aportes para esta ya que cuenta con un problema gravísimo. La forma o los criterios en que se basaron para realizar las mediciones en la fase de pruebas no son lo suficientemente relevantes para conocer con claridad la bondad real del modelo de clasificación presentado.

Los autores utilizaron el conjunto de datos KDD'99 y, por lo visto, el único procesamiento que les hicieron antes de comenzar la etapa de entrenamiento, fue separar los datos etiquetados como normales de los anómalos para, posteriormente, realizar la reducción de características con PSO binario. Respecto a los aspectos básicos de preprocesamiento de datos, no se habla de redundancia, distribución y al parecer ni siquiera se llevo a cabo un escalamiento. En teoría se utilizó el conjunto de datos completo (casi 5 millones de registros) o quizás un subconjunto de datos seleccionados aleatoriamente, no hay manera de saberlo con la información presentada por los autores.

En los resultados obtenidos por los investigadores se puede apreciar que, en base al uso de ciertas características relevantes obtenidas por PSO binario, se logró mejorar significativamente la **Tasa de Detección** desde un 82,6387% a un impresionante 99,8438%, no obstante, en ninguna sección del documento se puede apreciar el indicador de exactitud o al menos la tasa de falsas alarmas del modelo de clasificación propuesto, por lo tanto, se podría estar presenciando un hecho más de lo establecido por [9], donde se comenta que no todos los investigadores informan explicitamente los mismos criterios para evaluar el desempeño de los algoritmos en las distintas etapas del proceso de clasificación. En este caso, ni siquiera los criterios básicos necesarios para sacar conclusiones sobre los modelos.

En el documento en cuestión se muestran los parámetros para SVM, que fueron obtenidos mediante el uso de PSO, y además, se indica que las tasas de detección obtenidas están en términos de ataques "conocidos", de lo cual se puede inferir que, aparentemente, las pruebas realizadas sobre su modelo fueron llevadas a cabo deliberadamente con el conjunto de datos de entrenamiento en vez de uno distinto de pruebas o que simplemente probaron con un subconjunto de datos poblado completamente de datos normales. También existe la probabilidad de que se trate de errores en la traducción del trabajo al inglés, sin embargo son cosas que no se pueden descuidar.

A modo de destacar los detalles de la investigación, se puede decir que no es fiable basarse en un solamente en un único indicador, sobre todo si no se es capaz de exponer nitidamente la realidad detrás de la construcción del modelo clasificación. ¿Qué pasaría si se obtiene un 100% en la tasa de detección, pero al mismo tiempo un alto porcentaje en la tasa de falsas alarmas (la mayoría de los datos normales considerados como ataques)? Se podría hablar de un modelo que fue mal entrenado y que posee una tendencia a ciertos tipos de datos (en este caso quizás fue un subconjunto de los datos normales) y por lo tanto, el modelo estaría considerando prácticamente todo evento como si fuese un ataque incluso en el caso de no serlo y por ende tendría una excelente tasa de detección de intrusiones, pero al mismo tiempo una alta tasa de falsas alarmas, lo que finalmente se traduciría en una pésima exactitud. Al parecer es muy probable que el trabajo aludido en esta sección coincida con un caso de semejante naturaleza.

4.4 Variantes de PSO

En esta sección se presentan algunas variantes de la metaheurística PSO que han emergido en los últimos tiempos con el fin de mejorar la técnica convencional. Los aspectos que se suponen mejorados están relacionados con la capacidad de convergencia del algoritmo, es decir, que las partículas tengan un comportamiento más eficiente o eficaz a través de sus iteraciones en busca de un óptimo. Todas estas variantes serán puestas a prueba analizando su comportamiento, desempeño, precisión y otras cosas, con el fin de, posteriormente, seleccionar a aquella que haya otorgado los mejores resultados para el modelo de clasificación LS-SVM frente al conjunto de datos de IDS.

4.4.1 Peso de Inercia Dinámico

En el trabajo de [23] se presenta una variante de PSO llamada IPSO (Improved Particle Swarm Optimization), la cual supone mejoras de eficiencia al converger, es decir, menor cantidad de iteraciones para alcanzar el óptimo. Esta variante trabaja modificando la forma en que evoluciona el peso de inercia a través de las iteraciones del algoritmo. En vez de poseer una variación lineal como fue introducida y establecida por [14], en esta versión, el peso de inercia varía de forma exponencial basándose en la siguiente ecuación:

$$\omega' = \omega * u^{-k}; \quad \omega \in [0,1], \quad u \in [1.0001, 1.005] \tag{31}$$

Se pudo constatar que cuando el valor de u tiende a 1,005 (límite superior establecido por los autores), la variación del peso de inercia a través de las iteraciones, es casi lineal. Ver Ilustración 4.4 (función de la curva con coeficientes aproximados).

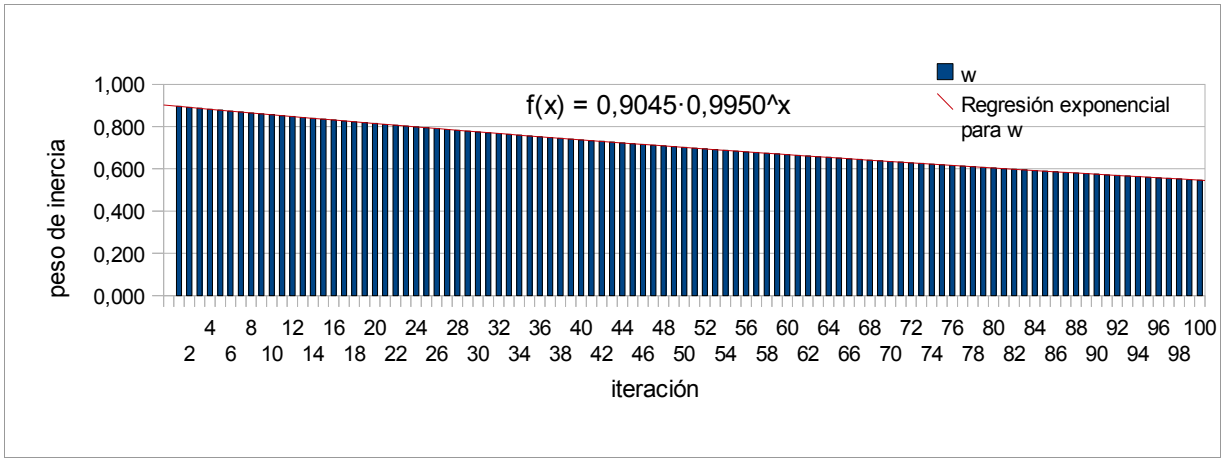

Ilustración 4.4 Peso de inercia con variación exponencial

Por otra parte, cuando el valor de u tiende al mínimo valor establecido 1,0001, el comportamiento del peso de inercia es prácticamente lineal como se puede apreciar en Ilustración 4.5 (función de la curva con coeficientes aproximados).

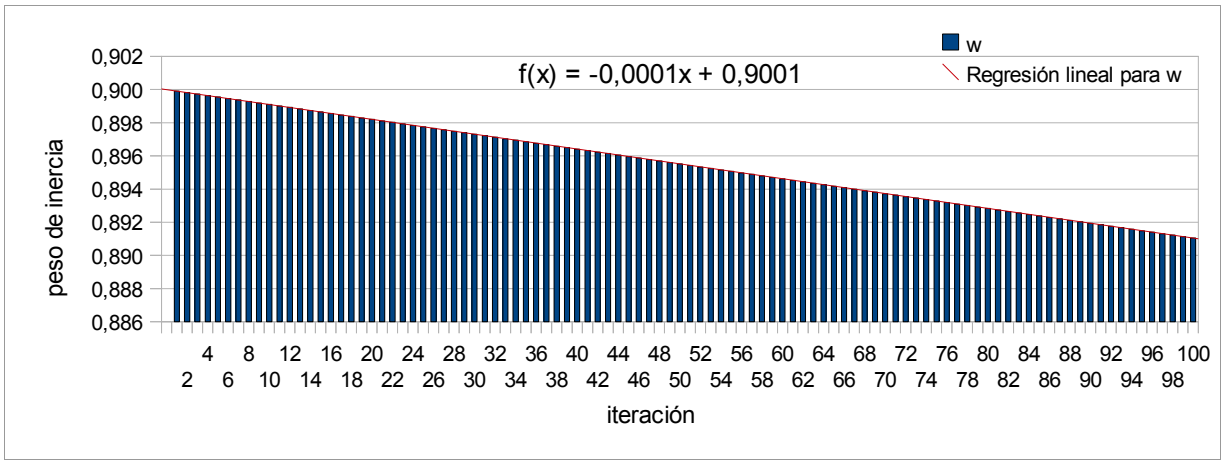

Ilustración 4.5 Peso de inercia con variación lineal decreciente

Respecto a esta variante, se pudo comprobar que posee una pequeña diferencia en la evolución del peso de inercia respecto del algoritmo convencional (peso de inercia con variación lineal), sin embargo, IPSO será puesto a prueba en esta investigación ya que según los autores tiene un comportamiento más eficaz y eficiente que el PSO convencional a la hora realizar el proceso de optimización.

4.4.2 PSO con Adaptación Dinámica

En la investigación de [24] se propone una variante de PSO nombrada DAPSO (particle swarm optimizer with dynamic adaptation) que modifica su comportamiento mediante adaptación dinámica evitando que el algoritmo converja rápidamente a un mínimo local. Para lograr esto, se utiliza una fórmula modificada para la actualización de la velocidad, donde la aleatoridad en el curso de la actualización de la velocidad de la partícula es reducida razonablemente y por otra parte, lo más interesante es que el peso de inercia de cada partícula es diferente, estos hace que las partículas se comporten de una manera un poco más independiente a lo largo de las iteraciones.

En DAPSO se agregan dos parámetros que describen el estado evolutivo del algoritmo, uno es el factor de velocidad evolutiva de cada partícula que dependerá de la evolución del valor fitness, o el costo de cada partícula, y el otro es el factor del grado de agregación del enjambre, el que dependerá del promedio de los costos del enjambre y del mejor costo obtenido en cierta iteración.

Los autores definen el peso de inercia de cada partícula, en base a los parámetros señalados, de la siguiente manera:

$$\omega_i^t = f(h_i^t, s) \tag{32}$$

donde h es el factor de velocidad evolutiva propio de cada partícula y s el grado de agregación del enjambre como un todo. A continuación se especifica cómo son calculados dichos parámetros.

Factor de velocidad evolutiva:

$$h_i^t = \left| \frac{min(F(pbest_i^{(t-1)}), F(pbest_i^t))}{max(F(pbest_i^{(t-1)}), F(pbest_i^t))} \right| \tag{33}$$

donde $F(.)$ corresponde a la función fitness. En base a la función h se puede deducir que su dominio es $0 < h \leq 1$. Este parámetro toma en cuenta la historia de la iteraciones de cada partícula y refleja la velocidad evolutiva de cada partícula, es decir, cuando una partícula mejore su fitness de una iteración a otra ($h \neq 1$), su peso de inercia propio disminuirá haciendo lo mismo con su velocidad. Con esto lo que se busca es que aquella partícula en particular, tienda a concentrar su búsqueda.

Grado de agregación:

$$s = \left| \frac{min(F_{tbest}, \overline{F_t})}{max(F_{tbest}, \overline{F_t})} \right| \tag{34}$$

donde $\overline{F_t}$ es la media de valores fitness de todas las partículas del enjambre en la iteración t y F_{tbest} representa el valor óptimo global encontrado en dicha iteración. Este es el parámetro que, en teoría, hace que el enjambre evite converger a óptimos locales, aumentando la velocidad de sus partículas cuando sus fitness sean similares en una iteración.

Hasta este punto el problema es cómo hacer variar el ω con los factores establecidos previamente. El propósito de la variación es darle al algoritmo mayor habilidad para buscar rápidamente y, al mismo tiempo, evitar converger a óptimos locales.

Los autores de DAPSO definen el peso de inercia como sigue:

$$\omega_i^t = \omega_{ini} - \alpha(1 - h_i^t) + \beta s \tag{35}$$

donde ω_{ini} es el valor inicial de ω. En base a que $0 < h \leq 1$ y $0 < s \leq 1$, se puede obtener que $1 - \alpha \leq \omega \leq 1 + \beta$. Los valores de α y β a escoger se encuentran típicamente en el rango $[0,1]$.

Con las modificaciones propuestas se logra que el peso de inercia varíe dinámicamente en base a las iteraciones y al estado evolutivo de cada partícula. El peso de inercia se reducirá proporcionalmente en relación al incremento de la velocidad evolutiva, o dicho de otro modo, cuanto menor sea el valor de h, menor será la velocidad adquirida por la partícula. Por otra parte, el peso de inercia de todas las partículas se verá incrementado cuando los fitness de ellas, en cierta iteración, sean similares entre si.

4.4.3 Otros

En el trabajo de [25] se define una función en términos del valor fitness, el tamaño del enjambre y la dimensión del espacio de búsqueda para ajustar **adaptativamente** el peso de inercia.

$$\omega = [3 - \exp(-S/200) + (R/8 * D)^2]^{-1} \tag{36}$$

donde S es el número total de partículas en el enjambre, D es el tamaño de la dimensión del espacio de solución, y R denota el rango del valor fitness de cada partícula.

Cuando una partícula obtenga un buen fitness, entonces se le reducirá el peso de inercia para acelerar su ratio de convergencia, por el contrario, este será aumentado para que tenga la oportunidad de continuar explorando. En caso de que el tamaño del enjambre sea grande, se utilizará un peso de inercia pequeño para potenciar la búsqueda local aumentando el ratio de convergencia, de lo contrario, se empleará un gran peso de inercia para mejorar la búsqueda global con el fin de encontrar el óptimo global. En caso de un problema de optimización sobre un espacio de búsqueda multidimensional, se usará un gran peso de inercia para fortalecer la habilidad para escapar de los óptimos locales, por el contrario, se empleará un peso pequeño para reforzar la capacidad de búsqueda local.

Por términos de ambigüedad, el último trabajo mencionado no será considerado para ser aplicado en el presente proyecto. Sin embargo, los anteriores serán comparados junto con el algoritmos de PSO convencional.

Tomando en cuenta completamente lo bueno y lo malo del estado del arte, del tema que trata este proyecto, se pone fin al establecimiento de las bases teóricas que determinarán el proceso de implementación a seguir.

5. Implementación

En este capítulo se darán a conocer cuáles son las actividades que fueron necesarias para llevar a la práctica los objetivos propios del presente proyecto de título y cómo fueron implementadas. El modelo de procesos a seguir obedece, básicamente, a una serie simple de pasos que se ha adecuado al estado del arte de la problemática en cuestión. Principalmente se pueden distinguir tres grandes actividades que describen dicho modelo: la preparación de datos, el data mining y la interpretación y evaluación de resultados, de las cuales ninguna es más o menos importante que la otra pues todas juegan un rol fundamental y son imprescindibles para esta clase de trabajos.

En primer lugar, se expondrá sobre la preparación de datos tomando en cuenta sus ventajas y desventajas. En segundo lugar, se explicará cómo fue realizada la construcción y puesta en marcha del modelo de clasificación propuesto, haciendo énfasis en cómo se logra la conexión e interacción correcta de las técnicas PSO y LS-SVM. Finalmente, serán expuestos los resultados obtenidos por los modelos de clasificación conseguidos con las distintas variantes de PSO aplicadas, en conjunto con algunas comparaciones con trabajos realizados por terceros. Todo lo anterior será presentado con un análisis en detalle.

Antes se comenzar de lleno con los principales temas de esta sección, cabe señalar que cada una de las actividades de implementación mencionadas se realizaron bajo el uso del entorno de computación numérica MATLAB 7.10.0 (R2010a) con las Toolboxes: LS-SVMLab (versiones 1.5 a la 1.8) [26] y Particle Swarm optimization [27].

5.1 Preparación de los Datos

El primer paso a seguir denominado “Preparación de datos” es concebido, en este proyecto, como una actividad simplificada que se basa solo en algunas de las tareas previas a la minería de datos, ya que engloba actividades comunes como la selección, preprocesamiento y transformación de datos, todo esto obteniendo provecho de los avances relacionados con el conjunto de datos KDD'99 que han sido encontrados hasta la fecha.

Con el simple hecho de utilizar un conjunto de datos elaborado por terceros, se está ahorrando gran parte del proceso de selección, es decir, para el problema tratado aquí no ha sido necesario intersectar una red para capturar los paquetes de datos que son transmitidos a través de ella a fin de establecer los conjuntos de datos base para los procesos de entrenamiento y pruebas. El único “trabajo de selección” realizado fue buscar y escoger un conjunto de datos ampliamente conocido y, posteriormente, una versión mejorada y mucho más beneficiosa, para los investigadores, del mismo.

El preprocesamiento o preparación de datos como tal, trata básicamente sobre aumentar la calidad que poseen los datos y hacerlos más valiosos. Es decir, que tras un cierto procesamiento, los datos generen mayor valor a la hora de ser analizados y procesados. Esto se podría traducir en diversas mejoras para cualquiera de las etapas del proceso de análisis, como

por ejemplo, disminuir la complejidad del modelamiento del problema, incrementar la eficiencia de los procesos de entrenamiento y pruebas, aumentar el nivel de fiabilidad de los resultados, etc.

De lo anterior se desprende que el preprocesamiento de datos es una etapa independiente del proceso de clasificación, ya que no se mezcla ni interactúa con las etapas de entrenamiento y pruebas. Por ende, lo hace un proceso totalmente independiente de la método de clasificación a utilizar y con mayor razón, en el caso de este proyecto, de la técnica de optimización asociada.

Eventualmente, el procesamiento de datos se podría volver un proceso de alta complejidad cuando se agregan distintas actividades de tratamiento de datos y más aun cuando la naturaleza de las metodologías utilizadas para cada una de esas actividades ya es compleja de por si. Algunas actividades comúnmente relacionadas a dicha preparación de datos son la limpieza de datos, eliminación de ruido, reducción de los datos (registros o puntos de datos) y reducción de la dimensionalidad (características de los datos). A pesar de todo lo anterior, lo más probable es que cualquier esfuerzo que sea realizado correctamente sobre los datos en bruto se traducirá en un beneficio para cualquier procesos posterior que se desarrolle sobre ellos.

5.1.1 Datos utilizados

Como el problema de IDS es tan popular, no es difícil encontrar trabajos parcial o completamente dedicadas al análisis y/o procesamiento del conjunto de datos KDD'99. Es por esto, que se ha decidido aprovechar los avances realizados en reducción de redundancia y distribución normal de los datos, establecido por [9] y los de reducción de características realizados por [28][29], como base y apoyo para seguir adelante con la implementación. En consecuencia, se ocuparán los conjuntos de datos establecidos por los primeros autores señalados y posteriormente filtrados en base a lo logrado por los segundos, lo que en el fondo se traduce en el uso de datos que poseen un claro avance en términos de preprocesamiento.

Cabe destacar que la reducción de características (en este caso, en base al descubrimiento de las más relevantes) puede ser “un arma de doble filo” ya que se produce un trade-off entre la efectividad de los resultados y la eficiencia para alcanzarlos. Además, eventualmente se podría afectar los resultados finales al eliminar alguna característica que este correlacionada con otras. Por ejemplo, si se eliminan muchas características que probadas por si solas tienen una influencia ínfima en los resultados, ya sea directa o indirectamente, existe la probabilidad de que la suma de algunas de ellas pueda contribuir negativamente al resultado final, ya que estas podrían aportar mayor valor al participar en conjunto con cualquiera de las características restantes, incluyendo a las que no han sido eliminadas, que al hacerlo por si solas.

En resumen, actividades tan importantes y de gran impacto como las de preparación de datos siempre contribuirán de alguna manera u otra los resultados de la clasificación de datos e incluso la eficiencia para obtenerlos al realizarse de manera correcta, sobre todo cuando se trabaja con datos tan complejos como los de IDS.

5.1.2 Actividades Realizadas

Como consecuencia de la elección de la toolbox LS-SVMLab para implementar el modelo de clasificación propuesto, es que surge la necesidad de aplicar una serie de transformaciones al conjunto de datos previamente generado con el fin de adaptarse a la forma de trabajo requerida por dicha herramienta.

La toolbox LS-SVMlab requiere que los datos de entrenamiento y pruebas estén en matrices de datos numéricos. Sin embargo, de las 41 características del conjunto de datos utilizado, tres de ellas son cualitativas (tipo string) y entre ellas suman un total de 77 categorías distintas. Por lo tanto, se realizó una representación numérica para cada uno de los datos pertenecientes a dichas características. En la Ilustración 5.1 se muestran los pasos realizados para completar dicho proceso de adaptación.

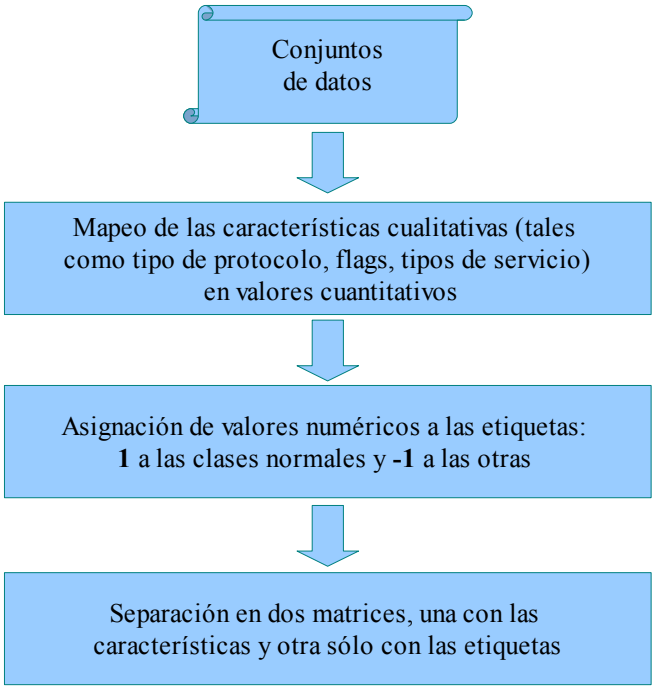

Ilustración 5.1: Adaptación del conjunto de datos KDD'99

Una última, pero no menos importante tarea de tratamiento de los datos realizada fue la de **escalamiento**, función incorporada MATLAB bajo el nombre de *"zscore"* y también en LS-SVMlab para su uso opcional en las fases de entrenamiento y pruebas a través del parámetro *"preprocess"*. Es una tarea de transformación de datos que consiste en hacer coincidir el dominio de cada una de las características para que la clasificación no se vea afectada o influenciada por aquellas características que poseen los datos de mayor magnitud entre sus registros. A pesar de ser una actividad de complejidad baja y extremadamente sencilla de realizar, esta influye considerablemente en la efectividad obtenida por los modelos

de clasificación.

La función utilizada para escalamiento es la siguiente

$$z = \frac{x-\mu}{\sigma} \tag{37}$$

donde x corresponde al vector (en este caso bidimensional) que se quiere escalar, μ a la media de la población y σ a su desviación estándar.

En MATLAB, es una operación extremadamente simple que al ser ejecutada no debiera quitar mucho tiempo. Por ejemplo, al ser ejecutada sobre el conjunto de datos de entrenamiento KDDTrain+ (de NSL-KDD) tardó al rededor de 0,14 segundos y con el 20% inicial del mismo, que es el conjunto de entrenamiento base de esta implementación, tardó aproximadamente 0,035 segundos.

Con objeto de poner en manifiesto la importancia del proceso de escalamiento es que se realizaron una serie de clasificaciones con fines comparativos. Para llevar a cabo esta simple prueba, se utilizaron 1.000dp del conjunto de datos KDDTrain+, el kernel RBF y combinaciones de parámetros ingresadas aleatoriamente. En la Tabla 5.1 se podrá apreciar la exactitud obtenida en una serie de clasificaciones con los datos sin escalar y otra con los mismos parámetros, kernel y puntos de datos, pero con los datos previamente escalados.

Tabla 5.1 Diferencias de la Exactitud respecto al escalamiento de datos

	c	σ	Exactitud		Diferencia
			Sin escalamiento	Con escalamiento	
LS-SVM c/ kernel RBF	100	1	60,1934%	75,9714%	15,7780%
	1.000	0,5	55,4028%	75,3903%	19,9875%
	10.000	0,5	**55,4028%**	**75,9093%**	**20,5065%**
	10.000	0,0001	52,9054%	59,2974%	6,3920%
Media			**55,9761%**	**71,6421%**	**15,6660%**

Como se puede observar en la tabla se alcanzó una diferencia promedio de 15,6% y una diferencia máxima de 20,5% aproximadamente (en términos de exactitud) lo cual representa el gran beneficio que otorga la actividad de preprocesamiento señalada.

En resumen, las actividades de preprocesamiento llevadas a cabo para el conjunto de datos NSL-KDD (conjunto previamente preprocesado respecto a reducción y distribución de puntos de datos) fueron, en primera instancia, la reducción de características (de 41 a 13 en base a la literatura mencionada), y posteriormente, las de transformación: adaptación a la herramienta usada y de escalamiento mediante MATLAB. De esta manera culmina el primer paso de la implementación, estableciendo el conjunto de datos base para cualquier operación posterior.

5.2 Data Mining

Sin duda esta es la actividad de mayor complejidad y crítica en la implementación y por lo tanto, la que requiere mayor esfuerzo y dedicación. Esto es, dado que en esta etapa es donde se ponen en práctica las técnicas de clasificación y optimización que deberán conformar el modelo que, posteriormente, será responsable de entregar la mejor solución posible al problema en cuestión. Para conseguir esta solución es de vital importancia comprender el funcionamiento por separado de cada técnica para contar con las capacidades y experiencias necesarias para explicar las causas de cualquier comportamiento y resultado obtenido, ya sea esperado o no, al momento de ejecutar las técnicas en conjunto. Es por esto que se ha dado énfasis en estudiar los principales aspectos teóricos relacionados a este trabajo en vez de solo aplicar un conjunto de herramientas de alto prestigio que probablemente otorguen una solución de calidad media o bastante aceptable.

El objetivo principal de esta etapa es hallar un modelo de clasificación que sea capaz de determinar de manera correcta a qué clase corresponde cada observación o evento que se quiera analizar. El principal aspecto a considerar, es que el modelo tenga la capacidad de clasificar bien cualquier dato que le sea entregado, no solo los datos que ya conoce sino que también aquellos datos que poseen patrones desconocidos. En este lugar es donde hace ingreso la técnica de optimización, la que se requiere para aumentar el **poder de generalización** del modelo de clasificación evitando que se **sobreajuste**. Es decir, que el clasificador tenga la capacidad para desempeñarse de la mejor forma posible frente a datos que no hayan estado presentes en su fase de entrenamiento o, dicho desde el otro punto de vista, que no se comporte bien únicamente con los datos que en un principio forjaron sus criterios de decisión.

Respecto al uso adecuado de los datos, hay que destacar que luego de definir los conjuntos de entrenamiento y pruebas, se deben utilizar exclusivamente para sus etapas correspondientes. Es decir, que el conjunto designado como “conjunto de datos de pruebas” no sea usado en el proceso de entrenamiento o viceversa. Por otro lado, no se debe entrenar con los dos conjuntos de datos unidos y luego probar el modelo obtenido con uno o los dos conjuntos, y tampoco se debe entrenar con uno y luego probar el modelo con los mismos datos. No tiene ningún sentido realizar evaluaciones para estos casos ya que como el modelo haya logrado un buen entrenamiento y cuente con conocimientos de los datos que se vayan a probar, también tendrá la facilidad de clasificar de manera correcta a la gran mayoría de estos.

En base a lo mencionado en los párrafos anteriores, se puede deducir que el proceso de optimización se encuentra ligado única y exclusivamente al conjunto de datos de entrenamiento, no obstante surge la siguiente interrogante: ¿cómo se logra medir la capacidad de generalización de un modelo sin utilizar los datos de pruebas?, la respuesta es simple, para optimizar el modelo manteniendo el uso único de los datos de entrenamiento se deben realizar **procesos de clasificación a menor escala** con subconjuntos de dichos datos. Es decir, el conjunto de datos de entrenamiento es subdividido en nuevos conjuntos que cumplan los roles de entrenamiento y pruebas. Al tener estos subconjuntos se deben realizar clasificaciones que determinarán la capacidad de generalización del conjunto padre. Dentro de la siguiente sección se podrá apreciar en detalle cómo opera la evaluación descrita.

5.2.1 Procedimiento general

En primer lugar, se explicará desde un punto de vista abstracto y de la manera más didáctica posible cómo debe ser el comportamiento general y la comunicación entre las técnicas de clasificación y optimización utilizadas. Posteriormente, se profundizará en los aspectos que explican la **interacción** correcta entre estas técnicas haciendo hincapié en sus principios.

La secuencia de pasos que describe el comportamiento de la combinación entre PSO y LS-SVM es la siguiente:

i. Inicialmente, PSO dará forma a cada partícula en base a una combinación de parámetros aleatorios (la que dependerá de ciertos rangos previamente determinados). Una vez que todas las partículas estén constituidas, serán enviadas a LS-SVM para ser puestas a evaluación.

ii. Cuando LS-SVM recibe las partículas, estas son evaluadas individualmente en un proceso de clasificación. A cada una se le asignará un costo que es calculado en base al porcentaje de datos mal clasificados. Entre mayor sea el error, *más costosa* será la partícula para el modelo.

iii. LS-SVM envía de vuelta los resultados de las partículas a PSO, el cual, basándose en los costos estimados hará viajar a las partículas en el espacio de búsqueda generando nuevos valores para cada una de ellas.

iv. Se finaliza el ciclo cuando se obtiene un óptimo (la partícula obtiene un costo que cumple un determinado criterio de detención) o cuando se alcanza un número de interaciones previamente señalado para PSO. Un ejemplo del caso ideal (utópico para el problema de IDS) sería cuando una partícula proporcione cero error de clasificación al ser evaluada en LS-SVM gracias a la composición de sus parámetros.

v. En caso de no cumplir con los criterios señalados en el paso anterior, las partículas son reenviadas con sus nuevas posiciones (combinación de parámetros) a LS-SVM (2° paso).

vi. Finalmente, cuando se ha terminado con el proceso de optimización, se llevan a cabo las etapas de entrenamiento y pruebas con la mejor partícula (conjunto de parámetros), es decir, la que posea el menor costo asociado.

En el segundo paso se menciona a grandes rasgos la forma en que LS-SVM estima el costo de cada partícula, no obstante, esta sencilla tarea se podría convertir en un gran problema, ya que por cada una de las partícula habría que ejecutar un proceso de clasificación usando el conjunto de datos establecido. En un escenario común, con 20 partículas, 200 iteraciones de PSO y suponiendo que no se cumple con los criterios de detención establecidos, se terminaría realizando un total de **4.000 clasificaciones**, lo que podría convertirse en un procedimiento muy costoso en caso de no contar con el hardware adecuado o si el conjunto de

datos usado es demasiado grande.

Para llevar a cabo cada una de las miles de clasificaciones mencionadas, es necesario realizar un proceso de entrenamiento y uno de prueba. Puede parecer bastante lógico, pero a la hora de evaluar un conjunto de parámetros, hacer solamente un entrenamiento y una prueba es una alternativa tremendamente insuficiente. Esto se debe a que al realizar solamente estas dos operaciones, un conjunto de datos poseerá el rol de entrenamiento y el otro, en contra parte, el de pruebas, lo que implica que el costo que se obtenga no será un valor representativo del conjunto **como un todo** sino que representará solo a aquel subconjunto que se encuentre cumpliendo con el rol de entrenamiento. Para evitar este problema y lograr que los costos de las partículas y futuros resultados sean fiables, es que se ha optado por utilizar una técnica que se basa en el **intercambio de roles** entre los distintos subconjuntos existentes, cross-validation.

La función fitness utilizada es, específicamente, "*stratified ten-fold cross-validation*", la cual, según lo visto en la Sección 3.4, estima los costos de manera tal, que estos se caracterizan por poseer un grado de representatividad mayor de la muestra en comparación con el cross-validation común. Además, como al realizar clasificaciones de datos se busca disminuir los errores de estas, habrá que minimizar la función fitness para que esta se dedique a **minimizar los costos** de las partículas.

Se debe recordar que entre menor sea el costo de una partícula, mayor será la capacidad del clasificador en términos de generalización, lo cual aumenta directamente la probabilidad de que el modelo de clasificación sea más efectivo sobre aquellos puntos de datos que no hayan estado presentes en el proceso de entrenamiento, o dicho de otro modo, sobre aquellos registros que tengan patrones distintos o totalmente desconocidos a los tratados en aquella fase.

Para el proceso de optimización de este trabajo, el conjunto de datos de entrenamiento se dividirá en diez segmentos, como se señaló más arriba, utilizando. Por cada movimiento que haga una partícula, esta será evaluada a través de la ejecución de la función fitness, la que a su vez, estará produciendo y ejecutando diez clasificaciones binarias de menor escala. Considerando el escenario mencionado más arriba, en donde se concluía la realización de 4.000 clasificaciones estas se incrementarían en 40.000, debido al uso de la función fitness mencionada.

A pesar de ser un técnica sencilla de entender, fácil de implementar y de vital importancia para lograr la correcta participación mutua entre los métodos de clasificación y optimización, cross-validation tiene la gran desventaja de ser una técnica extremadamente costosa, más aun cuando los conjuntos de datos utilizados son de gran envergadura o, para el caso de LS-SVM en particular, cuando los rangos de los parámetros a optimizar son muy amplios.

5.2.2 Construcción de Modelos

En esta sección, se explicará en detalle cómo fueron llevados a cabo cada uno de los diversos procesos de clasificación implementados en este proyecto. Además, se expondrán los resultados de las pruebas en base a los procesos de entrenamiento con distintos kernels, cantidades de puntos de datos, cantidades de características y combinaciones de parámetros. Cabe destacar, que la mayoría de las combinaciones de parámetros, el de compensación y los que requiere cada kernel, se obtendrán con las variantes de PSO escogidas, incluyendo el algoritmo original introducido en 1995 [4].

Para realizar la mayoría de las pruebas se podrá apreciar el uso deliberado del conjunto de datos KDD'99, a pesar de los problemas que conlleva. Esto se debe principalmente a que será de utilidad para demostrar qué es lo que sucede cuando los datos usados no son del todo representativos y, además, para posteriormente realizar pseudo-comparaciones con los resultados de las investigaciones encontradas en la literatura ya que hasta la actualidad, la mayoría de los trabajos sobre clasificación de datos de IDS contemplaba el uso de los datos KDD'99. Por otra parte, estos datos fueron usados en los inicios del presente proyecto, no así el conjunto de datos NSL-KDD, ya que éste fue encontrado un par de meses después.

Para construir cada modelo se tomó una muestra aleatoria del conjunto de datos NSL-KDD, la que corresponde, aproximadamente, al 0,02% del conjunto de datos “KDD'99 train” (definitivamente es un conjunto muy grande). El fin de esto es realizar evaluaciones fiables de clasificadores construidos en base a los datos procesados por [9] con su conjunto “KDDTrain+”.

En la Tabla 5.2 se expone un resumen de los elementos base utilizados en los distintos procesos de clasificación.

Tabla 5.2 Elementos comunes usados para la construcción de modelos de clasificación

Conjunto de datos de entrenamiento	Puntos de datos
3,969% NSL-KDD (3,969% KDDTrain+)	1.000
Conjuntos de datos de pruebas (nombre del archivo)	
NSL-KDD (KDDTest+)	22.544
KDD'99 test (corrected)	**311.027**

La evaluación de los distintos modelos de clasificación construidos, son denominadas *Escenarios* y su orden de aparición en el documento está establecida conforme iban evolucionando los modelos en términos de desempeño.

Los primeros dos escenarios son los menos complejos, ya que su finalidad, en conjunto, es exhibir las diferencias entre el uso o no de un proceso de optimización al construir un modelo de clasificación con LS-SVM. Por otra parte, los escenarios posteriores aumentan el grado de complejidad utilizando otros kernels de LS-SVM y distintas variantes de PSO.

5.2.2.1 Escenario 1: Sin optimización

Se utilizó el kernel Radial Basis Function en LS-SVM y se usaron parámetros ingresados intuitivamente. Luego de conformar los modelos, se realizaron las pruebas utilizando los conjuntos de datos NSL-KDD y KDD'99. Los valores elegidos para los parámetros y los resultados producidos en base a estos, se pueden apreciar en la Tabla 5.3.

Tabla 5.3 Resultados de la clasificación sin optimización

	c	σ	Exactitud	
			NSL-KDD	KDD'99
LS-SVM c/ kernel RBF	1	1	65.2923%	75,58%
	10	1	58,7500%	65,61%
	1	10	60,2843%	75,81%
	1.000	0,0001	54,1400%	66,82%

Se hace notar la diferencia de resultados entre los dos conjuntos de datos expuestos, KDD'99 y NSL-KDD. Por otra parte, se logra apreciar la necesidad por algún proceso que ayude a obtener una mejor combinación de parámetros para aumentar el desempeño del modelo.

5.2.2.2 Escenario 2: PSO convencional

Se utilizaron parámetros obtenidos por PSO convencional para crear el modelo de clasificación, posteriormente se realizaron pruebas utilizando los conjuntos de datos NSL-KDD y KDD'99. La cantidad de iteraciones fijadas para PSO fue solamente de 50 unidades. Además, nuevamente se utilizó solamente el kernel Radial Basis Function en LS-SVM.

Tabla 5.4 Resultados con PSO convencional

	c	σ	Exactitud	
			NSL-KDD	KDD'99
LS-SVM c/ kernel RBF	1.203,900	0,00900	57,8924%	63,3045%
	2.424,420	7,25195	67,6059%	75,9091%
	2.331,700	12,4910	61,0200%	65,9441%
	2.468,624	2,08520	69,5600%	75,7897%

Los resultados mostrados en la Tabla 5.4 no presentan mayores diferencias a los de la Tabla 5.3. Probablemente sea necesaria la ejecución de un proceso de optimización con mayor prolongación para encontrar un conjunto de parámetros indicado. Además, queda pendiente el uso de otros kernels para la creación de modelos.

5.2.2.3 Escenario 3: Reducción de características

Este escenario está dedicado únicamente a poner a prueba la reducción de características en los conjuntos de datos que fue mencionada en la sección 5.1.1.

Se utilizaron parámetros obtenidos por el algoritmo PSO convencional en un total de 100 iteraciones. Además se ocupó el kernel RBF para la construcción de los modelos y cada unos de los clasificadores basados en PSO fue probado utilizando los conjuntos de datos NSL-KDD. En la tabla Tabla 5.5 se exponen los resultados de tres modelos que utilizaron todas las características de los conjuntos de datos. Posteriormente, en la Tabla 5.6 se muestran otros tres modelos que ocuparon los conjuntos de datos reducidos.

Tabla 5.5 Resultados LS-SVM basado en PSO con 41 características

Modelo	c	σ	Exactitud	Tiempo optimización	Tiempo training	Tiempo testing
#1	1833,4012	40,34856	77,6792855468505%	00:08:06,908	00:00:00,150	00:00:02,320
#2	1867,2360	21,09461	79,9148332150461%	00:07:48,328	00:00:00,154	00:00:02,245
#3	3376,5890	19,78109	75,7008516678495%	00:09:47,116	00:00:00,157	00:00:02,190
media			**77,7649901432487%**	**00:08:34,117**	**00:00:00,1536**	**00:00:02,2523**

Tabla 5.6 Resultados LS-SVM basado en PSO con 13 características

Modelo	c	σ	Exactitud	Tiempo optimización	Tiempo training	Tiempo testing
#1	1197,4610	12,09805	75,8206174591909%	00:07:38,141	00:00:00,145	00:00:02,122
#2	933,6222	23,00887	76,6244106862034%	00:06:58,702	00:00:00,155	00:00:02,203
#3	230,7811	21,61876	76,1045067423705%	00:07:35,743	00:00:00,148	00:00:02,082
media			**76,1831782959216%**	00:07:24,195	00:00:00,149	00:00:02,136

Al comprar la Tabla 5.5 con la Tabla 5.6 se puede apreciar que la complejidad en la construcción y evaluación de los modelos se ve disminuida al utilizar menos características en los conjuntos de datos. Sin embargo, como consecuencia de la ausencia de algunas características, se reduce considerablemente la efectividad de los modelos. Por lo tanto, no es conveniente perder tanto desempeño, al rededor de un 2%, por ahorrar unos cuantos segundos al clasificar.

Por otra parte, se puede apreciar que los tiempos alcanzados en los procesos de optimización son bastante variables respecto a la media. Esto se puede explicar al considerar que las partículas se desplazan por distintos lugares del espacio de búsqueda y, por lo tanto, es muy probable que algunos enjambres terminen posicionándose en ciertos sectores (combinaciones de parámetros) que dada su naturaleza provoquen que la evaluación de la función fitness sea mucho más compleja y, por ende, tome mucho más tiempo en ser calculada.

5.2.2.4 Escenario 4: Distintos kernels para LS-SVM

Se utilizó una serie de parámetros obtenidos por el algoritmo PSO convencional en un total de 100 iteraciones y se ocuparon distintos tipos de kernels para el proceso de entrenamiento, esto a excepción del kernel lineal, ya que no se utilizó para ser optimiazado por PSO debido a que claramente, por su naturaleza, no cuenta con parámetros adicionales. Posteriormente, estos parámetros se probaron utilizando los conjuntos de datos NSL-KDD y KDD'99, como se expone en la Tabla 5.7.

Tabla 5.7 Resultados LS-SVM con distintos kernels y PSO convencional

Kernel	Parámetro c	Parámetros Kernel	Exactitud	
			NSL	KDD'99
Lineal	1	-	75,6742%	90,9925%
	1000	-	76,1089%	91,0005%
Polinomial,	344,27	[25; 65,2604]	77,6703%	92,1432%
parámetros: (d, p)	2,727341644635282e+02	[25;85,638787755325140]	77,9276%	92,1824%
RBF,	3,053992927861814e+02	25,018988017671530	79,3925%	92,1589%
parámetro: σ	1,251996931309970e+02	21,546594570501670	78,7172%	92,0612%

Respecto a los resultados de la Tabla 5.7, se puede decir que las diferencias entre los distintos kernels no fueron significativas. La principal causa de esto se debe a que el rango de los valores de los parámetros no fueron (deliberadamente) los indicados. Por ejemplo, el kernel RBF y el Polinomial tomaron valores muy pequeños para el parámetro *C*, esto quiere decir que la penalidad que se le otorgó a los errores de clasificación fue ínfima y por ende, no lo suficientemente adecuada. En tanto a los parámetros del kernel RBF, se puede inferir que, en ambos casos, a pesar de que tomaron valores no muy grandes, probablemente, no fueron lo necesariamente pequeños como para entregar mayor flexibilidad u holgura a la superficie del clasificador.

Como se puede apreciar en los últimos escenarios, la variación entre los modelos de clasificación sin optimizar y los optimizados no es significativa, incluso cuando la cantidad de iteraciones de PSO fue aumentada y se utilizaron variantes que suponían grandes mejoras.

Se pudo corroborar efectivamente que el conjunto de datos original KDD'99 necesita una gran actividad destinada solamente al preproceso, con el fin de evitar que los modelos tengan tendencia o sean influenciados por puntos de datos ampliamente repetidos. Cabe destacar, que por un lado se obtuvo resultados relativamente bajos con el conjunto de datos NSL-KDD, en cambio los resultados obtenidos en las pruebas realizadas con los datos KDD'99 fueron sumamente superiores. Esto último, demuestra de manera fiel, el caos provocado en dicho conjunto de datos por motivos de mala distribución (sesgo) y redundancia de datos.

5.2.2.5 Escenario 5: Variantes de PSO

En la Tabla 5.8 se encuentran los parámetros obtenidos por algunas variantes de PSO; su costo correspondiente, dado por la función objetivo *"cross-validation"*, y la exactitud frente a pruebas con el conjunto de datos NSL-KDD. En cada proceso de optimización se realizaron en 100 iteraciones y se utilizó el kernel RBF en LS-SVM para todos los entrenamientos.

Tabla 5.8 Resultados LS-SVM con el kernel RBF y variantes de PSO

Variante PSO	c	σ	Fitness	Exactitud
PSO con coeficientes de restricción	1305,58420000000	47,9158707000000	0,0014600000	**79,8837828246984%**
PSO con peso de inercia dinámico (exponencial)	1961,04195895021	4,97450733221489	0,0039900000	73,9251242015614%
PSO con adaptación dinámica	43,48433000000	7558,84100000000	0,0003200000	72,8003903477644%

Como se puede apreciar, los costos (columna *Fitness*) otorgados por la función fitness son extremadamente bajos. Es decir, que frente a muchos puntos de datos desconocidos (conjunto de validación), el modelo fue capaz de clasificarlos prácticamente superando el 99% de exactitud. Sin embargo, el comportamiento que tuvo el modelo frente a los datos de pruebas no fue tan exitoso como se suponía debía ser, o por lo menos, como se esperaba. Aparentemente esto podría ser obra de algún error de implementación y efectivamente lo era. En este escenario, es de gran importancia ya que en él se expone un gran error de implementación encontrado hacia el final del presente proyecto.

El problema principal de los costos expuestos, es que lamentablemente, no son lo suficientemente fiables o representativos. El motivo de esto, es que se estaba utilizando de manera incorrecta la función cross-validation. Dicha función necesita permutar los puntos de datos, con el fin de obtener una buena distribución de estos a la hora de realizar entrenamientos y validaciones. Es el mismo principio utilizado a gran escala a la hora de realizar entrenamientos y pruebas. Un conjunto, tanto de entrenamiento como de pruebas, no puede estar sesgado ya que puede influenciar los resultados de la clasificación.

El problema específico, fue que en cada ocasión que se utilizaba cross-validation, se generaba una permutación aleatoria de los datos del conjunto de datos de entrenamiento. Es decir, que para una determinada partícula (combinación de parámetros para LS-SVM), la función objetivo, eventualmente, podría arrojar distintos resultados al ser llamada varias veces. Es por esto, que luego de muchas iteraciones, era bastante probable encontrar un resultado o valor fitness cercano al óptimo. Por lo tanto, la probabilidad de que los valores generados sean poco fiables aumenta considerablemente.

Esta última información ha sido deliberadamente incluida con el fin de prestar utilidad a futuras investigaciones relacionadas con el área de la clasificación. Luego de superar dicho problema, a continuación se exponen nuevos escenarios con resultados fiables y más realistas.

5.2.2.6 Escenario 6: PSO canónico

En la Tabla 5.9 se encuentran los parámetros obtenidos por PSO canónico o PSO con coeficientes de restricción, su respectivo costo (valor fitness) y los resultados de las pruebas con los datos NSL-KDD y KDD'99. Además, se debe considerar que los datos de entrenamiento fueron los mismos para los cinco modelos; se realizaron 100 iteraciones en el proceso de optimización y se utilizó el kernel RBF para LS-SVM en la fase de entrenamiento.

Tabla 5.9 Resultados de PSO Canónico con LS-SVM usando el kernel RBF

			Exactitud	
c	σ	**Fitness**	NSL-KDD	KDD'99
1038,296	88,32373	0,10900	82,2835344215756%	92,7096830199113%
2007,904	91,26073	0,10700	82,6472675656494%	92,7315459330159%
2295,481	83,46674	0,11400	82,8823633782825%	92,7469785775603%
1190,233	89,11664	0,10900	82,3678140525195%	92,7138626944754%
1326,309	88,16671	0,10900	82,4875798438609%	92,7209359898916%
MEDIA		**0,10960**	82,5337118523776%	92,7246012429709%
STD		**0,00261**	0,238060798758804	0,0150023140409127

Es posible notar que el algoritmo PSO no posee una alta precisión al momento de buscar óptimos, dentro de un rango de valores determinado, para los datos de IDS. Esto se puede inferir, ya que ninguna de las combinaciones de parámetros (c , σ) de la Tabla 5.9 se repite. Sin embargo, a pesar de lo anterior, algunos costos obtenidos por la función fitness si se repiten, 0,10900. Por lo tanto, se puede determinar que el algoritmo fue capaz de llegar a un mismo resultado en más de una ocasión. Este alcance de los costos debe estar influenciado por el tamaño de la muestra.

Respecto a la exactitud alcanzada en la Tabla 5.9 frente a los distintos tipos de conjuntos de pruebas, se puede decir que en relación a los escenarios anteriores, se han obtenido mejores resultados. Esto es, debido a que se ha adquirido mayor experiencia sobre el uso de ambos algoritmos (LS-SVM y PSO) y de la función fitness, cross-validation. Cada uno tiene cosas buenas y malas que hacen muy compleja su interacción.

Por ejemplo, por la alta complejidad de cross-validation, no es factible utilizar grandes conjuntos de datos en sus operaciones. Por lo tanto, al obtener resultados de PSO con conjuntos de datos reducidos, se está generando parámetros dedicados solamente a la pequeña cantidad de datos determinada inicialmente. Posteriormente, si estos mismos parámetros obtenidos se quieren utilizar en un entrenamiento con un mayor número de datos, su uso no tendrá sentido alguno, ya que la separación de las categorías es más compleja y, por lo tanto, es necesaria una mayor penalidad o una superficie con más flexibilidad u holgura. Esto también puede ocurrir en modo inverso, de parámetros extremos a conjuntos de datos pequeños.

En una serie de pruebas finales, se realizaron modelos con 2.000 puntos de datos. El proceso de optimización fue extremadamente costoso, no obstante, los resultados obtenidos por el clasificador desarrollado fueron alentadores.

En el proceso de optimización el costo alcanzado por la mejor partícula fue de 0,08533. Este costo fue calculado al utilizar los parámetros c con valor igual a 3.172,5088 y σ con valor igual a 43,7601 en la función fitness. Luego de construir el modelo de clasificación con estos parámetros y probarlo con el conjunto de datos KDDTest+ y KDDTest se alcanzó una exactitud de 83,49% y 92,69% respectivamente.

Lamentablemente la cantidad de datos utilizada en la construcción de los modelos, específicamente en el proceso de optimización, se encontraba al límite de la factibilidad. Al intentar usar unos cientos de datos adicionales el proceso se tornaba total y absolutamente ineficiente.

Cabe destacar que el proceso más costoso fue solamente el de optimización, no así el de entrenamiento o el de pruebas.

5.3 Análisis de Resultados

En este sección se expone el análisis detallado de los resultados de los modelos más relevantes que fueron obtenidos en la sección anterior. Además, se proporciona la ganancia alcanzada entre el mejor y el peor modelo construido. Finalmente, se compara el mejor modelo de clasificación con los modelos encontrados en la literatura.

5.3.1 Modelos Construidos

A través de los distintos criterios de evaluación, se presentará la bondad de cada modelo producido por las distintas variantes de PSO. Las variantes a considerar son:

- PSO con coeficientes de constricción, en adelante *"PSO"*
- PSO con peso de inercia dinámico (exponencial), en adelante *"IPSO"*
- PSO con adaptación dinámica, en adelante *"DAPSO"*

Además, se debe recalcar la configuración común utilizada en las variantes de PSO:

- Número de partículas: 20
- Número de iteraciones: 150

En los gráficos de la Ilustración 5.2, Ilustración 5.3 y Ilustración 5.4, se exponen las distintas variantes considerando la exactitud, la tasa de detección y la tasa de falsas alarmas respectivamente.

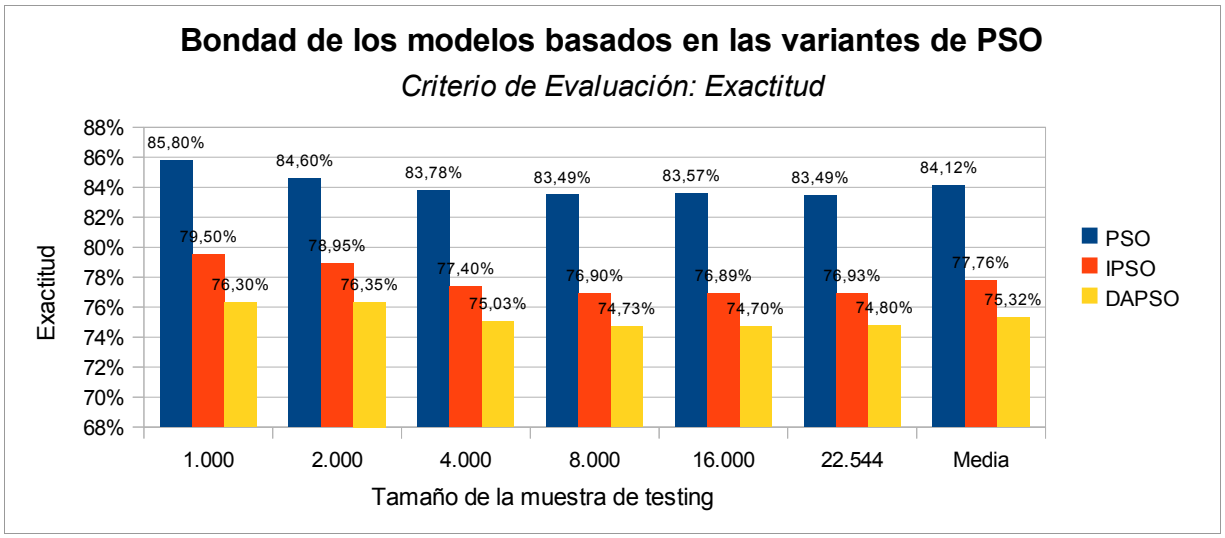

Ilustración 5.2 Exactitud de los modelos creados por las variantes de PSO

Como se puede apreciar en el gráfico de la Ilustración 5.2, el modelo construido por PSO con coeficientes de constricción es el que posee los mejores resultados cuando se trata de exactitud. Superando a IPSO y DAPSO en un 6,36% y 8,80% respectivamente. Es decir, que el algoritmo utilizado tuvo la capacidad de clasificar correctamente la mayor cantidad de eventos analizados. Hasta aquí no se tiene certeza de que el modelo haya clasificado mejor a los datos normales que lo anómalos o viceversa. A pesar de poseer la mayor exactitud, ¿será este modelo el que posea la mayor Tasa de Detección y la menor Tasa de Falsas Alarmas? Para verificar esto se deben examinar los siguientes gráficos. Ver Ilustración 5.3 y Ilustración 5.4.

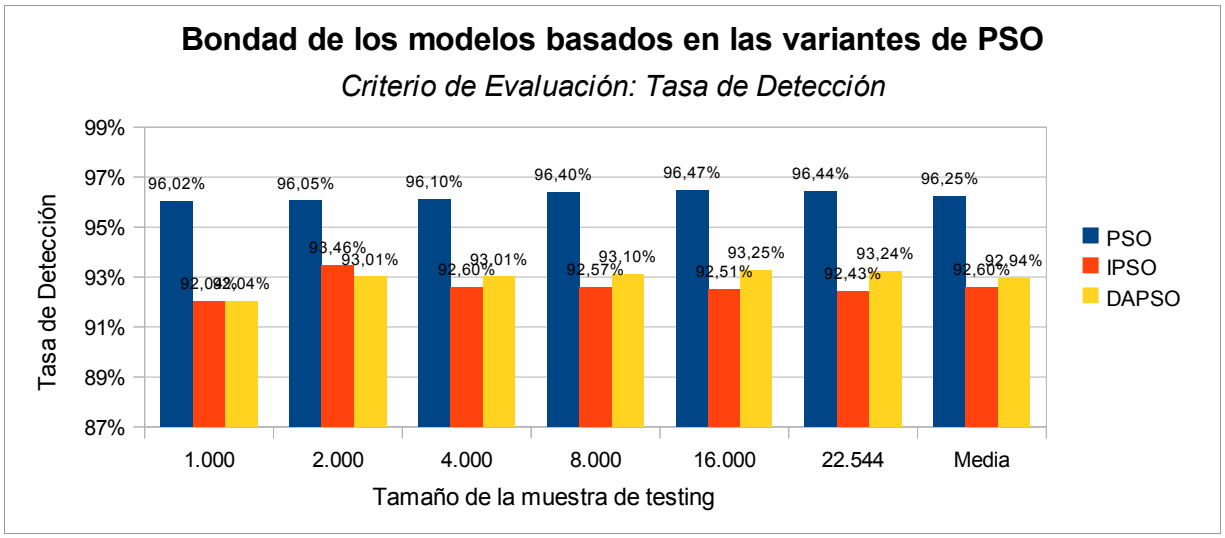

Ilustración 5.3 Tasa de Detección de los modelos creados por las variantes de PSO

Nuevamente el modelo provisto por PSO con coeficientes de constricción es quien supera al resto, pero ahora, respecto a la Tasa de Detección, lo que quiere decir que este modelo tiene una mayor capacidad para detectar datos anómalos que el resto de las variantes en juego. En un IDS es primordial contar con esta característica ya que de esta manera gran parte de las anomalías que amenacen la red podrán ser efectivamente alertadas.

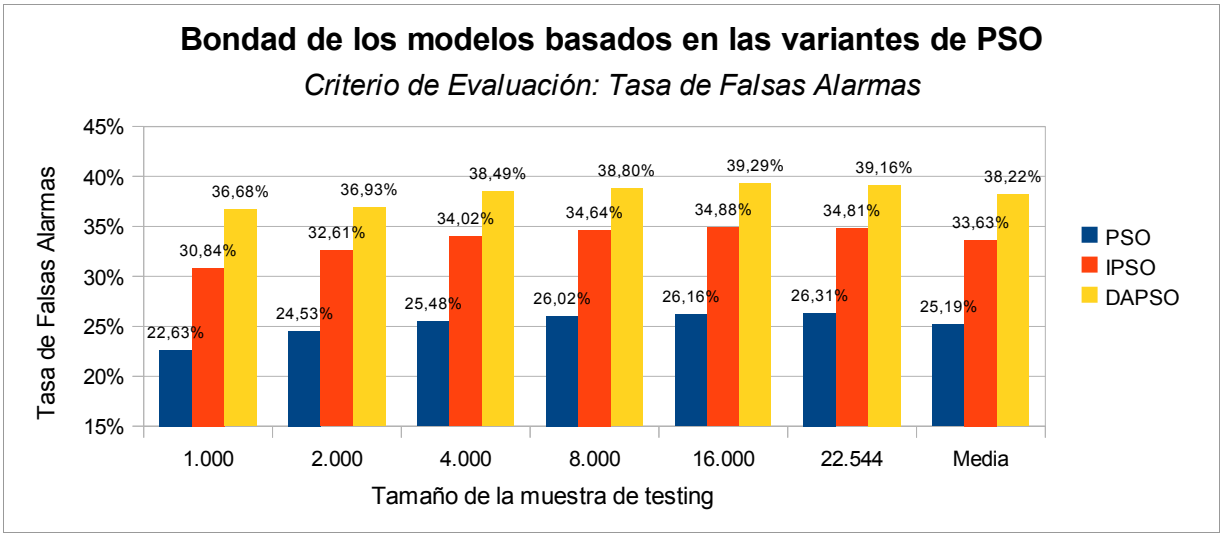

Ilustración 5.4 Tasa de Falsas Alarmas de los modelos creados por las variantes de PSO

En base a la Tasas de Falsas Alarmas de cada modelo, expuestas en la Ilustración 5.4, se puede decir que el modelo creado con la variante PSO con coeficientes de constricción fue el que menos se equivocó al tratarse de eventos considerados como anómalos, siendo que realmente de trataba de eventos normales. Desde otro puntos de vista, se puede decir que los otros modelos tienen una posición un poco más conservadora, lo que en el ámbito de seguridad es una característica clave. Sin embargo, como la exactitud del mejor modelo fue muy alta, es muy difícil que cometa errores de clasificación, independiente de la clase analizada, por eso su tasa de falsas alarmas es menor que el resto.

Un caso muy útil a considerar sería la tarea de escoger entre los modelos restantes. Como se puede apreciar en el gráfico de la Ilustración 5.2, IPSO supera a DAPSO por un poco más de 2 puntos porcentuales en exactitud. Sin embargo, DAPSO obtuvo una mayor tasa de detección, es decir, que IPSO ignoró más casos de anomalías que DAPSO. Definitivamente la tendencia de este último es de carácter conservador, alertando gran parte de los eventos como anomalías, como se muestra en la Ilustración 5.3, lo que explica su baja exactitud. No obstante, si lo que prima es la seguridad, DAPSO sería la alternativa correcta.

Para finalizar este proceso de comparación de modelos se debe decir que tanto la tasa de detección como la tasa de falsas alarmas son de utilidad para comparar a aquellos modelos que posean una exactitud similar, pues, dependiendo del tipo de problema, se debe escoger en base a alguna tendencia. La que para el caso de IDS es claramente la tasa de detección.

5.3.2 Modelo de Clasificación Considerado

El mejor modelo fue construido gracias al PSO con coeficientes de constricción y en adelante se utilizará como punto de referencia en las comparaciones. Primero, se comparará con el peor modelo obtenido y, posteriormente, con los modelos obtenidos desde la literatura.

5.3.2.1 Comparaciones con el Peor Modelo Construido

Para destacar la diferencia entre el modelo escogido y el resto, es que se calcula la ganancia respecto al peor modelo. El porcentaje de ganancia es calculado a través de la siguiente ecuación:

$$G = \frac{A-B}{A} * 100 \tag{38}$$

donde G representa la ganancia de A sobre B.

Considerando que el peor modelo fue PSO con adaptación dinámica, la ganancia obtenida del mejor por sobre este, en términos de exactitud sobre el conjunto de datos KDDTest+, es de un **10,41%**.

5.3.2.2 Comparaciones con Trabajos Relacionados

A continuación se muestran los resultados que fueron obtenidos en el trabajo de [9] más el desempeño del modelo de clasificación logrado en este proyecto.

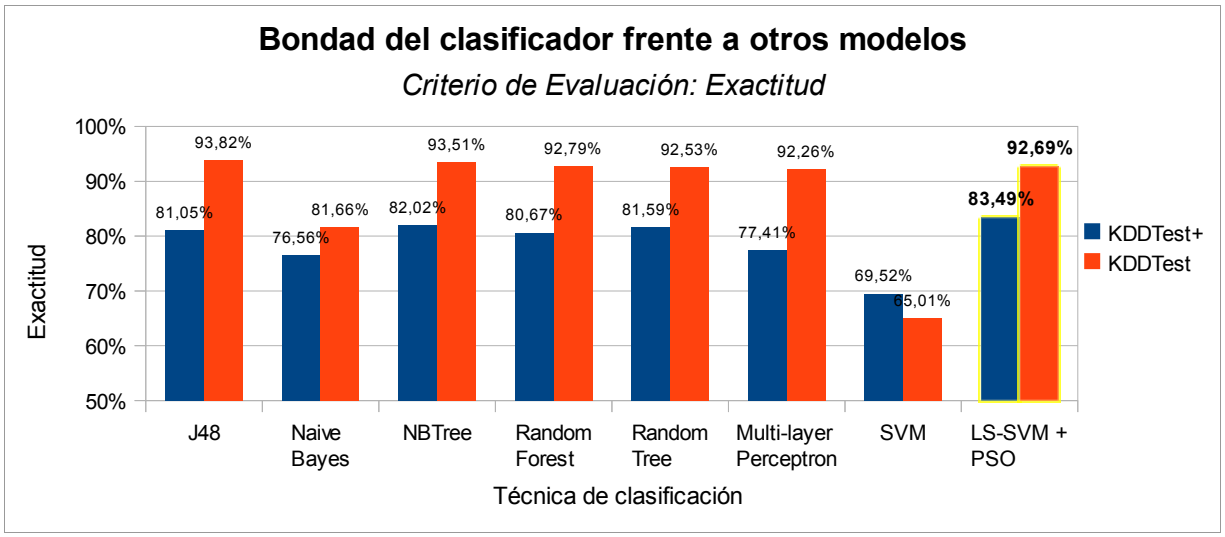

Ilustración 5.5 Exactitud del clasificador frente a otros modelos

Todas las técnicas mostradas en el gráfico de la Ilustración 5.5 se pusieron a prueba con los mismos datos. No obstante, el modelo de este proyecto fue construido con un menor

porcentaje de los mismos datos en la etapa de entrenamiento debido a restricciones de hardware, específicamente 2.000 puntos de datos contra los 25.192 del conjunto de datos KDDTrain+ con que se entrenó al resto de las técnicas del gráfico. Además, se debe destacar que solo se comparó en términos de exactitud debido a que los resultados de [9], lamentablemente, no presentaban otros criterios de evaluación. En cuanto a las tasas que faltan en el último gráfico, la de detección y la de falsas alarmas, correspondientes al mejor modelo sobre los datos originales (KDDTest), alcanzaron un 99,25% y un 8,90% respectivamente.

6. Conclusiones

A lo largo del desarrollo del presente proyecto de título se logró constatar lo extremadamente amplio que es el mundo de la clasificación de datos. Desde la detección de anomalías en una red de computadores, hasta el reconocimiento de la voz humana, son algunos de tantos problemas que, constantemente, requieren de soluciones como las que ofrece el aprendizaje supervisado. En consecuencia, se han desarrollado muchas técnicas que permanecen en constante evolución para enfrentar de mejor forma estos problemas.

Respecto a los objetivos del proyecto, se puede decir con toda seguridad que se han conseguido de una manera bastante satisfactoria. Se generaron modelos de clasificación, sin el uso de optimización, contrastándolos positivamente con los modelos construidos en base a PSO, sin embargo, gracias al afán por mejorar los distintos aspectos del trabajo se culminó con la obtención de resultados que definitivamente superaron las expectativas iniciales, incluso sobrepasando varios de los logros alcanzados en investigaciones realizadas por terceros respecto al problema de IDS.

En cuanto a PSO, es posible señalar que es un algoritmo muy eficaz y que cada vez se hace más eficiente gracias a la frecuente aparición de mejoras. Sin embargo, su comportamiento no es el mismo cuando se pone a prueba frente a problemas donde la matemática no se encuentra muy vinculada, como lo son los problemas de clasificación de datos de gran complejidad que poseen características no lineales y estocásticas. Esto se pudo demostrar ya que a pesar de que PSO logró mejorar la exactitud de LS-SVM, no lo hizo de manera significativa y eficaz, como lo haría al maximizar o minimizar una función matemática. En la mayoría de las oportunidades en que se realizaron procesos de optimización, el enjambre jamás pudo llegar a situarse en un mismo punto en el espacio, no obstante, algunos de los costos alcanzados lograron coincidir en más de una ocación. Otra posible razón pudo ser el bajo número de puntos de datos utilizados en la función fitness *"cross-validation"*, debido a su alto costo computacional.

En tanto a los resultados finales obtenidos, se pudo comprobar que la gran capacidad de generalización propia de LS-SVM proporcionó grandes ventajas en la clasificación de los datos de IDS, demostrando un excelente nivel de efectividad a pesar de que el proceso de optimización de cada modelo, conformado por un enjambre de 20 partículas, se utilizó una baja cantidad de puntos de datos (2.000) y un bajo número de iteraciones (150).

Respecto a los puntos de datos y características relevantes, se puso en evidencia que gran parte de los modelos de clasificación tienden a tomar ventaja de los registros redundantes y también de los conjuntos sesgados, lo que en consecuencia se refleja en la generación de resultados poco fiables. La reducción de 41 a 13 características, en los distintos conjuntos de datos, disminuyó la complejidad del modelo de clasificación, sin embargo, también se vio reducida la efectividad de los modelos y, además, las mejoras en los tiempos de procesamiento totales no fueron significativas como se esperaba. En promedio se calculó una diferencia del 2,065%. Dada la adversidad provocada por dichos efectos, no se continuó haciendo uso de la reducción de características.

Un punto importante a destacar fue el hecho de disminuir los tiempos de procesamiento o construcción de modelos a través de la reducción de los conjuntos de datos y, al mismo tiempo, conservar el buen desempeño de los modelos al momento de llevar a cabo las pruebas. En definitiva, obtener buenos resultados al clasificar 22.544 o 311.027 puntos de datos con un modelo que fue entrenando con un par de miles, proporciona una gran satisfacción y al mismo tiempo corrobora las virtudes que poseen las técnicas y procedimientos aplicados a lo largo del proyecto.

Como trabajo futuro, se recomienda la evaluación y uso de mejores funciones fitness para problemas de clasificación. Lo ideal sería utilizar una función que no sea tan costosa como lo es cross-validation o sus variantes, ya que por sus altos costos de procesamiento hace imposible (en una arquitectura de hardware básica) realizar operaciones de manera razonable, con muestras representativas y en poco tiempo. Además, se sugiere investigar sobre actividades de preprocesamiento de datos más avanzadas, pues incrementar la calidad de los datos incide considerable y directamente en los resultados que producen los modelos de clasificación, independiente de la técnica empleada para conformarlos.

7. Referencias

[1] Anderson J. P., *Computer Security Threat Monitoring and Surveillance*, 1980. James P. Anderson Co., Fort Washington, Pa.

[2] Bace R. and Mell P., *NIST Special Publication on Intrusion Detection Systems*, 2002. Disponible vía web en: www.21cfrpart11.com/files/library/.../nist_intrusiondetectionsys.pdf Revisada por última vez el 01 de Julio de 2010 National Institute of Standards and Technology

[3] Suykens J.A.K. and Vandewalle J., *Least Squares Support Vector Machine Classifiers*, 1999. Neural Processing Letters, vol. 9, no. 3, pp. 293–300.

[4] Kennedy J. and Eberhart R., *Particle swarm optimization*, 1995. inProc. IEEE Int. Conf. Neural Netw. (ICNN) , vol. 4, pp.1942–1948

[5] Scarfone K. and Mell P., *Guide to Intrusion Detection and Prevention Systems (IDPS)*, 2007. Disponible vía web en: csrc.nist.gov/publications/nistpubs/800-94/SP800-94.pdf. Revisada por última vez el 01 de Julio de 2010. Special Publication 800-94, Recommendations of the National Institute of Standards and Technology

[6] Verwoerd T. and Hunt R., *Intrusion Detection Techniques and Approaches*, 2002. Disponible vía web en: http://sureserv.com/technic/datum/pdf/ids/english/Intrusion%20Detection%20Techniques%20and%20Approaches.pdf Revisada por última vez el 01 de Julio de 2010 Department of Computer ScienceUniversity of Canterbury, New Zealand

[7] Fawcett T., *An introduction to ROC analysis*, 2006. Pattern Recognition Letters 27 (2006) 861–874

[8] KDD, *Knowledge Discovery and Data mining Cup 1999 Dataset*, 1999. Disponible vía web en: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, Revisada por última vez el 28 de Junio de 2010

[9] Tavallaee M., Bagheri E., Lu W. and Ghorbani A., *A Detailed Analysis of the KDD CUP 99 Data Set*, 2009. Proceedings of the 2009 IEEE Symposium on Computational Intelligence in Security and Defense Applications (CISDA 2009)

[10] NSL-KDD, *The NSL-KDD Data Set*, 2009. Disponible vía web en: http://nsl.cs.unb.ca/NSL-KDD Revisada por última vez el 29 de Junio de 2010

[11] Aizerman M., Braverman E. and Rozonoer L., *Theoretical foundations of the potential function method in pattern recognition learning*, 1964. Automation and Remote Control 25:821--837.

[12] Poli R., Kennedy J. and Blackwell T., *Particle swarm optimization. An overview*, 2007. Swarm Intelligence, 1(1):33-57.

[13] Del Valle Y., Venayagamoorthy G., Mohagheghi S., Hernandez J. and Harley R., *Particle Swarm Optimization: Basic Concepts, Variants and Applications in Power Systems*, 2008. IEEE Transactions on Evolutionary Computation, vol. 12, no. 2, pp. 171–195.

[14] Shi Y. and Eberhart R., *A modified particle swarm optimizer*, 1998b. In Proceedings of the IEEE internationalconference on evolutionary computation (pp. 69–73).

[15] Clerc, M. and Kennedy, J., *The particle swarm--explosion, stability, and convergence in a multidimensional complex space*, 2002. IEEE Transaction on Evolutionary Computation, 6(1), 58-73

[16] Eberhart R. and Shi Y., *Comparing inertia weights and constriction factores in particle swarm optimization*, 2000. In Proceedings of the IEEE congress on evolutionary computation (CEC) (pp. 84-88), San Diego, CA. Piscataway: IEEE

[17] Refaeilzadeh P., Tang L. and Liu H., *Cross-Validation*, 2008. Arizona State University

[18] Kohavi R., *A study of Cross-validation and Bootstrap for Accuracy Estimation and Model selection*, 1995. 1995 International Joint Conference on Artificial Intelligence (IJCAI)

[19] WEKA, *Waikato environment for knowledge analysis (weka) version 3.5.7*. Disponible vía web en: http://www.cs.waikato.ac.nz/ml/weka/, Revisada por última vez el 30 de Junio de 2010

[20] Wu S. and Yen E., *Data mining-based intrusion detectors*, 2009. Expert Systems with Application`s, Vol. 36, Issue 3, Part 1, pp. 5605-5612.

[21] Chang C. and Lin C., *LIBSVM -- A Library for Support Vector Machines*. Disponible vía web en: http://www.csie.ntu.edu.tw/~cjlin/libsvm/ Revisada por última vez el 30 de Junio de 2010

[22] Wang J., Hong X., Ren R. and Li T., *A Real-time Intrusion Detection System Based on PSO-SVM*, 2009. Proceedings of the 2009 International Workshop on Information Security and Application (IWISA 2009)Qingdao, China

[23] Jiao B., Lian Z. and Gu X., *A dynamic inertia weight particle swarm optimization algorithm*, 2008. Chaos, Solitons and Fractals 37, 698-705

[24] Yang X., Yuan Jinsha, Yuan Jiangye and Mao H., *A modified particle swarm optimizer with dynamic adaptation*, 2007. Applied Mathematics and Computation 189, 1205-1213

[25] Dong C., Wang G., Chen Z. and Yu Z., *A Method Of Self-Adaptive Inertia Weight For PSO*, 2008. 2008 International Conference on Computer Science and Software Engineering

[26] LS-SVMlab, *Least Squares - Support Vector Machines Toolbox*. Disponible vía web en: http://www.esat.kuleuven.be/sista/lssvmlab/. Revisada por última vez el 5 de Mayo de 2010.

[27] PSO, *Particle Swarm Optimization Toolbox*, 2006. Disponible vía web en: http://www.mathworks.com/matlabcentral/fileexchange/7506. Revisada por última vez el 17 de Octubre de 2010.

[28] Mahmud W., N.Agiza H. and Radwan E., *Intrusion Detection Using Rough Sets basedParallel Genetic Algorithm Hybrid Model*, 2009. Proceedings of the World Congress on Engineering and Computer Science 2009 Vol IIWCECS 2009, October 20-22, 2009, San Francisco, USA

[29] Kayacik H., Zincir-Heywood A. and Heywood M., *Selecting Features for Intrusion Detection: A Feature Relevance Analysis on KDD 99 Intrusion Detection Datasets*, 2005. Proceedings of the Third Annual Conference on Privacy, Security and Trust, October 2005, St. Andrews, Canada.

Apéndices

Apéndice 1. Características del conjunto de datos KDD'99.

Tabla A1.1 Listado de las características definidas para los registros de conexiones [8].

#	Característica	Descripción	Tipo
1	Duration	Duración de la conexión	Continuo
2	Protocol type	Protocolo de conexión (ej: tcp, udp)	Discontinuo
3	Service	Servicio de destino (ej: telnet,ftp)	Discontinuo
4	Flag	Flag de estado de la conexión	Discontinuo
5	Source bytes	Bytes enviados de la fuente al dentino	Continuo
6	Destination	Bytes enviados desde el destino hacia la fuente	Continuo
7	Land	1 si la conexión es de/para el mismo host/port; sino 0	Discontinuo
8	Wrong fragment	Número de fragmentos incorrectos	Continuo
9	Urgent	Número de paquetes urgentes	Continuo
10	Hot	Número de indicadores “hot”	Continuo
11	Failed logins	Numero de logins fallidos	Continuo
12	Logged in	1 si el logged in es correcto; sino 0	Discontinuo
13	# compromised	Número de condiciones “comprometidas”	Continuo
14	Root shell	1 si root shell es obtenido; sino 0	Continuo
15	Su attempted	1 si el comando “su root” es intentado; sino 0	Continuo
16	# root	Número de accesos de “root”	Continuo
17	# file creations	Número de operaciones de creación de archivos	Continuo
18	# shells	Número de shell prompts	Continuo
19	# access files	Número de operaciones sobre acceso a archivos de control	Continuo
20	# outbound cmds	Número de comandos de salida en una sesión ftp	Continuo
21	Is hot login	1 si el login pertenece a la lista “hot”; sino 0	Discontinuo
22	Is guest login	1 si el login es un “guest”; sino “0”	Discontinuo
23	Count	Número de conexiones al mismo host como la conexión actual en los 2 segundos anteriores	Continuo
24	Srv count	Número de conexiones al mismo service como la conexión actual en los 2 segundos anteriores	Continuo
25	Serror rate	% de conexiones que tienen errores “SYN”	Continuo
26	Srv serror rate	% de conexiones que tienen errores “SYN”	Continuo
27	Rerror rate	% de conexiones que tienen errores “REJ”	Continuo
28	Srv rerror rate	% de conexiones que tienen errores “REJ”	Continuo
29	Same srv rate	% de conexiones al mismo servicio	Continuo

#	Característica	Descripción	Tipo
30	Diff srv rate	% de conexiones a servicios distintos	Continuo
31	Srv diff host rate	% de conexiones a hosts distintos	Continuo
32	Dst host count	Cantidad de conexiones con el mismo host de destino	Continuo
33	Dst host srv count	Cantidad de conexiones con el mismo host de destino y bajo el uso del mismo servicio	Continuo
34	Dst host same srv rate	% de conexiones con el mismo host de destino y bajo el uso del mismo servicio	Continuo
35	Dst host diff srv rate	% de servicios distintos en el host actual	Continuo
36	Dst host same src port rate	% de conexiones al host actual con el mismo src port	Continuo
37	Dst host srv diff host rate	% de conexiones al mismo servicio provenientes de distintos hosts	Continuo
38	Dst host serror rate	% de conexiones al host actual que tienen un error S0	Continuo
39	Dst host srv serror rate	% de conexiones al host actual y servicio específico que tienen un error S0	Continuo
40	Dst host rerror rate	% de conexiones al host actual que tienen un error RST	Continuo
41	Dst host srv rerror rate	% de conexiones al host actual y servicio específico que tienen un error RST	Continuo

Apéndice 2. Clases del conjunto de datos KDD'99.

Tabla A2.1 Listado de las clases que aparecen en el 10% del conjunto de datos KDD'99 [29].

Ataque	# Muestras	Categoría
Smurf	280790	dos
Neptune	107201	dos
back	2203	dos
teardrop	979	dos
pod	264	dos
land	21	dos
normal	97277	normal
satan	1589	probe
ipsweep	1247	probe
portsweep	1040	probe
namp	231	probe
warezclient	1020	r2l
guess_passwd	53	r2l
warezmaster	20	r2l
imap	12	r2l
ftp_write	8	r2l
multihop	7	r2l
phf	4	r2l
spy	2	r2l
buffer_overflow	30	u2r
rootkit	10	u2r
loadmodule	9	u2r
perl	3	u2r