Detección de intrusiones en redes de datos mediante redes neuronales

Gianfranco Fagioli y Nicolás Arato
Trabajo práctico final de "Inteligencia Computacional", Ing. Inf.-FICH-UNL.

***Resumen*— En el presente trabajo se aborda la detección de intrusiones en redes de datos locales, la cual forma parte de las herramientas de la ciberseguridad, centrándose en la protección de activos de información a través del tratamiento de amenazas que ponen en riesgo la información procesada, almacenada y transportada por los sistemas interconectados. Para la detección de intrusiones se utilizaron redes neuronales multicapa interconectadas de diversas formas y por último se midió el desempeño de las arquitecturas propuestas con el objetivo de poder compararlas entre sí y saber para qué caso una determinada arquitectura es la más apropiada.**

***Palabras clave*— instrusiones en redes, redes neuronales multicapa, arquitecturas, clasificación.**

I. Introducción

La integración cada vez más profunda de Internet en la vida social está modificando la forma en que las personas se comunican, aprenden y trabajan, aunque al mismo tiempo también las expone a amenazas de seguridad cada vez más serias. Proyectos y negocios hacen que su éxito dependa del acceso a sus servicios, y hacer que sus infraestructuras sean más resistentes a las condiciones actuales y futuras de internet y menos dependiente de la participación humana requiere de una mejor intervención automatizada.

La ciberseguridad (*Cibersecurity* en inglés) es un conjunto de tecnologías y procesos diseñados para proteger computadoras, redes, programas y datos, de ataques y accesos no autorizados, alteración o destrucción. Dentro de los sistemas de seguridad, los sistemas de detección de intrusos (IDS) ayudan a descubrir, determinar e identificar comportamientos no autorizados del sistema, como el uso, la copia, la modificación y la destrucción de los datos. Su objetivo es detectar que algo sospechoso sucede en la red y actuar en función de esto.

Este problema de seguridad en las redes fue abordado por Amira Sayed A. Aziz en [1], quien plantea utilizar algoritmos genéticos para detectar y descartar los paquetes normales utilizando una función de *fitness* que calcula cuán normal es el paquete, utilizando luego distintas capas de clasificadores para determinar el tipo de ataque de los paquetes. Otro trabajo que aborda el tema es el desarrollado por Xueqin Zhang y Jiahao Chen [2], donde utilizan redes neuronales profundas para la realizar la clasificación.

En este trabajo se propone, a partir de paquetes capturados de la red, desarrollar un clasificador basado en redes neuronales para que funcione como IDS, utilizando como datos de entrada los encabezados de los paquetes capturados. Mediante estos encabezados se pueden identificar diferentes tipos de ataque, como son denegación de servicio, Remote to local, user to root y probe, entre otros.

II. Dataset

Los datos utilizados fueron obtenidos a partir del NSL-KDD [3], el cual es un conjunto mejorado del original KDD cup99 [4]. Este es el conjunto de datos utilizado para el Tercer Concurso Internacional de Descubrimiento de Conocimientos y Herramientas de Minería de Datos, que se realizó en conjunto con KDD-99 La Quinta Conferencia Internacional sobre Descubrimiento de Conocimientos y Minería de Datos.

La base de datos cuenta con 39300 paquetes donde cada uno tiene 39 valores que corresponden a los diferentes campos como por ejemplo: protocolo, tipo de servicio, longitud del paquete, tiempo de vida del paquete, bits de banderas, etc. que corresponden a las diferentes capas de red que van agregando valores a medida que se genera el paquete en el emisor.

En este trabajo se realizó un preprocesamiento de estos datos, con el fin de acotar el alcance del mismo. Estos cambios comprenden limitación del número de ataques a clasificar, normalización de los datos para que puedan ser leídos por nuestros algoritmos y utilización del conjunto de NSL-KDD train para todo el trabajo.

Los distintos tipos de ataques tenidos en cuenta son: normal (paquete no malicioso), neptune, smurf, warezclient, ipsweep, nmap, portsweep, satan. Con esta discriminación intentamos evitar el tratamiento del principal problema que se tiene con este conjunto de datos que es el gran desbalance entre clases, donde hay un número muy inferior de ciertos ataques con respecto a los demás.

A continuación se muestra la cantidad de paquetes para cada clase:

Clase	Cantidad de paquetes
normal	21264
neptune	13207
smurf	811
warezclient	308
ipsweep	1143
nmap	473
portsweep	947
satan	1147

TABLA I
CANTIDAD DE PAQUETES PARA CADA CLASE.

Dado que los valores de algunos campos pueden tener valores mucho mayores que los que se utilizan para banderas (las cuales toman valores 0 o 1) se deben normalizar los datos. Para esto se utiliza la normalización *mean range* desarrollada por Wei Wang en [5].

$$x_i = \frac{v_i - min(v_i)}{max(v_i) - min(v_i)}$$

donde v_i es el valor del atributo actual, y $min(v_i)$ y $max(v_i)$ son calculados teniendo en cuenta todos los valores para ese atributo.

III. Modelos propuestos

Para resolver el problema se utilizan 2 tipos de arquitecturas de redes neuronales. Haciendo uso de la validación cruzada con la variante hold out, manteniendo el 20 % de los datos para las pruebas en cada partición.

A. *MLP (Perceptrón multicapa)*

En esta primer propuesta de solución se utiliza un MLP para clasificar los ataques. Este cuenta con 39 neuronas en la capa de entrada, una capa oculta y 3 neuronas en la capa de salida que permiten identificar la clase predicha y actualizar los pesos de la red según el error cometido.

B. *MLP jerárquico*

En esta propuesta se utilizan todos los datos de entrenamiento para entrenar un primer MLP el cual solamente es entrenado para clasificar entre paquetes: *normales* y *ataques*. Un segundo MLP se conecta a continuación del anterior con el fin de clasificar tipos de paquetes: *neptune* y *otros ataques*. Esta red es entrenada con la misma base de datos anterior pero sin los paquetes que tienen etiquetas *normal*. Seguido a esta red se conecta otro MLP para clasificar entre: *smurf* y *otros ataques*, el cual es entrenado con los mismos datos pero excluyendo los datos de tipo *normal* y *neptune*. Se continúa con esta secuencia de conexiones de MLP entrenadas para clasificar 2 tipos de datos hasta llegar a contemplar todas las clases que contiene la base de datos. Esta arquitectura puede verse en la Fig. 1. Cada MLP contiene parámetros personalizables, lo que permite encontrar de manera ad hoc los valores de la red que mejoran los resultados obtenidos previamente. A continuación se presenta el pseudocódigo de la implementación propuesta.

```
inicio
inicializar parametros de particion
validacion_cruzada(Bdatos)
%-------------------------------------------------
for i=1 to cantidad de particiones
        inicializar estructura1
        inicializar parametros de MLP1
        [w1]= entrenamientoMLP1(BD)
        [BD]= actualizacion de
            data_train(BD, tipo1)
        .
        .
        .
        inicializar estructura7
        inicializar parametros de MLP7
        [w7]= entrenamientoMLP7(BD)
%-------------------------------------------------
        [t_acierto, matriz_confusion] =
            pruebaMLPjerarquico(w1,
            parametrosMLP1,. . .,w7,
            parametrosMLP7)
        Matriz=Matriz+matriz_confusion;
end
MC=Matriz/cant_part;
[mconf, recall, precision, miss_rate,
    accuracy]=medidas(MC);
fin
```

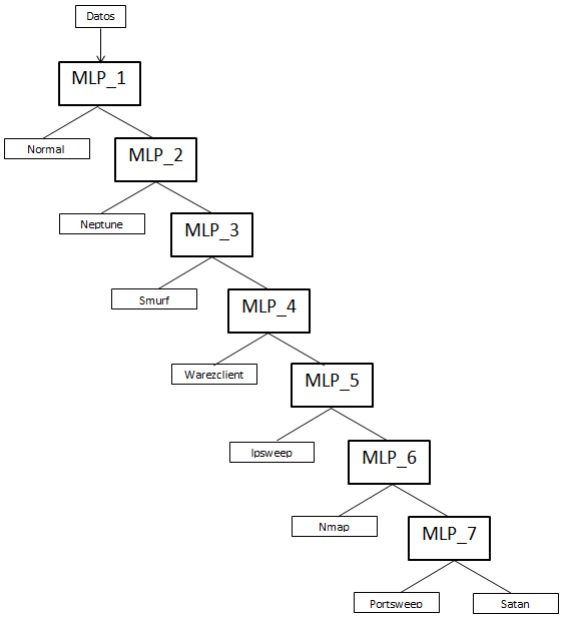

Fig. 1. Arquitectura del MLP jerárquico.

IV. Resultados

Para visualizar los resultados se muestran las matrices de confusión promedio con todas las etiquetas y las matrices de confusión promedio de ataques vs normales para las dos propuestas. Estas matrices se calculan mediante la suma de las matrices de confusión arrojada por cada partición de prueba dividido la cantidad de particiones. Una vez obtenidas estas matrices se calculan las siguientes métricas de performance: recall, precision, missrate, accuracy.

Para el MLP jerárquico se obtuvo la siguiente matriz de confusión promedio

	nor	nep	smu	war	ips	nmap	por	sat
nor	4238.2	1.4	3.2	6.2	9.6	2.6	3.4	4.6
nep	2.6	2624	0	0	0	0.4	0.8	0
smu	2.4	0	155.2	0	0	0.2	0	0
war	0	0.2	0	65.2	0	0	0	0
ips	3.2	0	0	0	219.2	5	0	0.2
nmap	0.2	0	0	0	1.2	90.2	0	2
por	1.8	0	0	0	0.4	0	187.4	1.2
sat	1.2	0	0	0	0	1.6	0.8	224.2

TABLA II
MATRIZ DE CONFUSIÓN PROMEDIO DEL MLP JERÁRQUICO.

	Ataque	No ataque
Ataque	3565.4	31
No ataque	11.4	4238.2

TABLA III
MATRIZ DE CONFUSIÓN PROMEDIO DEL MLP JERÁRQUICO TENIENDO EN CUENTA ATAQUES Y NO ATAQUES.

Los resultados obtenidos también están condicionados con la cantidad de pruebas realizadas y los parámetros

	recall	precision	miss rate	accuracy
MLP 1	0.9971	0.9914	0.0029	0.9947
MLP 2	0.9979	0.991	0.0021	0.9949
MLP 3	0.9974	0.989	0.0026	0.9937
MLP jerárquico 1	0.9955	0.9912	0.0045	0.9939
MLP jerárquico 2	0.9967	0.9904	0.0033	0.9942
MLP jerárquico 3	0.9968	0.9914	0.0032	0.9946

TABLA IV

MÉTRICAS OBTENIDAS CON LAS ARQUITECTURAS PROPUESTAS.

personalizados en cada MLP. En ambos clasificadores se dejaron libres sólo la cantidad de capas ocultas y las neuronas en cada una de ellas. Diferentes modificaciones de estos parámetros arrojaron las métricas expuestas en la tabla IV.

V. CONCLUSIÓN

A pesar de obtener muy buenos resultados con ambas propuestas no se pueden comparar con otras soluciones del estado del arte, debido a las simplificaciones que hicimos de la base de datos.

Se esperaba que con el MLP se obtuvieran resultados menos favorables que utilizando MLPs especializados en cada tipo de datos. Los resultados obtenidos contrastan con la suposición inicial.

Puede observarse que a pesar del desbalance de las clases, la propuesta de utilizar un solo MLP como clasificador para todas las clases arroja similares resultados que el clasificador jerárquico.

Al no conocer la forma de la superficie de error en el espacio R^{39} donde se sitúan los datos, no podemos afirmar con las pruebas aquí realizadas, que estos resultado no se pueden mejorar, ya que no sabemos si la superficie cuenta con un mínimo local o varios mínimos locales de similares características que estancan la convergencia de nuestros algoritmos. De manera que para encontrar mejores resultados se necesitan realizar más pruebas.

REFERENCIAS

[1] AMIRA SAYED A. AZIZ, ABOUL ELLA HASSANIEN, SANAA EL-OLA HANAFY y M.F.TOLBA. Multi-layer hybrid machine learning techniques for anomalies detection and classification approach. In: 13th International Conference on Hybrid Intelligent Systems (HIS). (2013)

[2] XUEQIN ZHANG y JIAHAO CHEN Deep Learning Based Intelligent Intrusion Detection. In: 9th IEEE International Conference on Communication Software and Networks. (2017)

[3] "https://github.com/defcom17/NSL-KDD

[4] "http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[5] WEI WANG, XIANGLIANG ZHANG, SYLVAIN GOMBAULT y SVEIN J. KNAPSKOG. Attribute Normalization in Network Intrusion Detection. In: 10th International Symposium on Pervasive Systems, Algorithms, and Networks. (2009)