Grado en Matemática Computacional

Trabajo Final de Grado

Redes neuronales. Un modelo de clasificación para la detección de dominios DNS maliciosos.

Autor:
Yolanda Juan Payá

Supervisor:
Antonio Villalón
Tutor académico:
Pablo Gregori Huerta

Fecha de lectura: 27 de noviembre de 2015
Curso académico 2014/2015

Resumen

El presente Trabajo de Final de Grado (TFG) incluye una descripción detallada del trabajo realizado durante mi estancia en prácticas en el área de Seguridad de S2 Grupo y un estudio sobre las redes neuronales artificiales.

Desde S2 Grupo se propuso trabajar en un proyecto cuyo objetivo es el análisis de tráfico DNS de un sistema para la identificación de APT mediante una herramienta de detección de nombres de dominio DNS maliciosos. Dicha herramienta debía basarse en un modelo de clasificación. Uno de los requisitos más importantes exigidos por la empresa fue que la herramienta pudiese aprender por si sola. Por ello, finalmente decidí utilizar una red neuronal

Durante mi estancia, realicé un análisis sobre la información necesaria para poder clasificar el nombre de dominio y desarrollé gran parte de la herramienta. Más concretamente, implemente una serie de módulos en Python capaces de extraer dicha información y normalizarla para poder ser procesada por una red neuronal.

Por otra parte, en el estudio sobre redes neuronales se define esta familia de modelos inspirada en la red neuronal biológica, así como también, se indican técnicas de ajuste y entrenamiento de las mismas.

Palabras clave

DNS (Sistemas de nombres de dominio), seguridad informática, Python, redes neuronales, método de clasificación

Keywords

DNS (Domain Name System), computer security, Python, neural network, método de clasificación

Índice general

1. Introducción **9**

1.1. Un proyecto en seguridad informática para una matemática computacional 11

1.1.1. Objetivo de negocio 11

1.1.2. Objetivos técnicos 12

1.1.3. Objetivos formativos 13

2. Chica en prácticas contra los malos, un granito de arena **15**

2.1. S2 Grupo 15

2.2. Metodología y definición de tareas 17

2.3. Planificación temporal 18

2.4. Recursos utilizados 21

2.5. Resultados obtenidos 21

3. Redes Neuronales **23**

3.1. Introducción 24

3.2. Introducción al modelo PPR (Projection Pursuit Regression) 24

3.2.1. PPR en función del parámetro M 25

3.2.2. Ajuste del modelo PPR 25

3.2.3. PPR y redes neuronales 26

3.3. Redes Neuronales 26

3.3.1. Similitudes con el modelo PPR 28

3.3.2. Un poco de historia 28

3.4. Ajuste de una red neuronal 29

3.5. Entrenando una red neuronal 31

3.5.1. Definiendo los valores de inicio: los pesos 31

3.5.2. Combatiendo el sobreajuste 32

3.5.3. Escalado de entradas 32

3.5.4. Número de unidades ocultas y capas 34

3.5.5. Múltiples mínimos 34

3.6. Consideraciones computacionales 34

4. Análisis sobre la información necesaria para la clasificación 35

4.1. DNS: Una puerta entreabierta para las ciberamenazas 35

4.1.1. Sistema de Nombres de Dominio 35

4.1.2. Nombre de dominio DNS 36

4.1.3. Seguridad en el DNS 36

5. Preparación de los datos para ser procesados por la red neuronal 39

5.1. Datos iniciales 39

5.2. Métodos de obtención de información sobre los nombres de dominio DNS y normalización de los datos 40

5.2.1. Cotejo del FQDN del dominio en diferentes BlackLists 40

5.2.2. Obtención de la resolución del FQDN y cotejo de la misma en diferentes BlackLists de direcciones IP infectadas 40

5.2.3. Obtención de los servidores de nombres asociados al FQDN y cotejo de la misma en diferentes BlackLists de servidores de nombres infectados 41

5.2.4. Obtención de la información de registro del FQDN y cotejo de la misma en la base de datos de información disponible sobre los FQDN infectados . 42

5.2.5. Detección de ataques typosquatting 42

5.2.6. Detección de FQDN creados mediante un DGA 43

5.2.7. Comprobación de que no se trata de un NXDOMAIN 43

5.2.8. Comprobación de que no pertenece a una red Fast-Flux 44

5.2.9. Detección de contenido malicioso 44

6. Conclusiones **45**

Glosario **47**

Índice de figuras

1.1. Noticias recientes en el diario El Mundo sobre ciberataques. 9

1.2. Incidentes gestionados por el CCN-CERT (en número y criticidad).[3] 10

2.1. Lineas de negocio de S2 Grupo 16

2.2. Monitorización de infraestructuras TIC, especialmente en lo relativo a seguridad de la información. Más información: `https://www.youtube.com/watch?v=7eEyC8_eY38` 17

2.3. Métodología CRISP-DM. [12] 17

3.1. Gráficos sobre dos ejemplos de funciones de canto. 24

3.2. Esquema de una red neuronal red neuronal feed-forward con una única capa oculta. 26

3.3. En rojo, representación gráfica de la función sigmoide $\sigma(v)$, usada comúnmente en la capa oculta de la red neuronal. En azul, representación gráfica de $\sigma(sv)$ con $s = \frac{1}{2}$, y en morado, con $s = 10$. La escala del parámetro s controla el ratio de activación. Notemos que un valor de s mayor produce una activación más radical en $v = 0$. 27

3.4. Respresentación de los resultados de una red neuronal que usa una función de activación *softmax* y *error de entropía cruzada*. A la izquierda: Sin usar decadencia de peso. A la derecha: Usando decadencia de peso y logra aproximarse a el ratio de error de Bayes (morado). 33

3.5. Mapas de calor de los pesos estimados a partir de entrenamiento de las redes neuronales de la figura 3.4. La gama de colores varía de verde brillante (negativo) a rojo brillante (positivo). 33

4.1. Sistema de nomenclatura jerárquico del DNS. [4] 36

4.2. Técnicas de detección de bootnets. 38

Capítulo 1

Introducción

Hoy en día, es un hecho que vivimos conectados a la red. Los avances tecnológicos de los últimos años han hecho posible que tecnologías como internet estén al alcance de todos. Esto nos abre un gran mundo de posibilidades, pero también, nos hace vulnerables si no somos conscientes del riesgo que corremos.

Si trasladamos este tema al ámbito empresarial, nos damos cuenta de que, actualmente, el uso de plataformas tecnológicas y redes es más que habitual, es prácticamente imprescindible. El problema es que con ello, queda expuesto uno de los activos más valiosos de una empresa, la información. Por esto, cada vez son más frecuentes las noticias de ataques contra TIC[1] de gobiernos, administraciones públicas y empresas con alto valor estratégico.

Figura 1.1: Noticias recientes en el diario El Mundo sobre ciberataques.

Existe una gran variedad de amenazas ligadas a las vulnerabilidades a las que un sistema está expuesto. Podemos clasificarlas, según el efecto causado en el sistema, del siguiente modo:

[1] Tecnologías de la Información y las Comunicaciones

- Interceptación. Acceso no autorizado a la información.
- Modificación. Acceso y modificación no autorizadas de la información.
- Interrupción. Interrupción del funcionamiento del sistema.
- Generación. Adición de elementos en el sistema de información.

Sin ir más lejos, según el Informe de Ciberamenazas 2014 y Tendencias 2015 (CCN-CERT IA-9/15) elaborado por el CCN-CERT [2] del Centro Criptológico Nacional (CCN) [3] , 2014 fue un año especialmente significativo en materia de ciberamenazas.

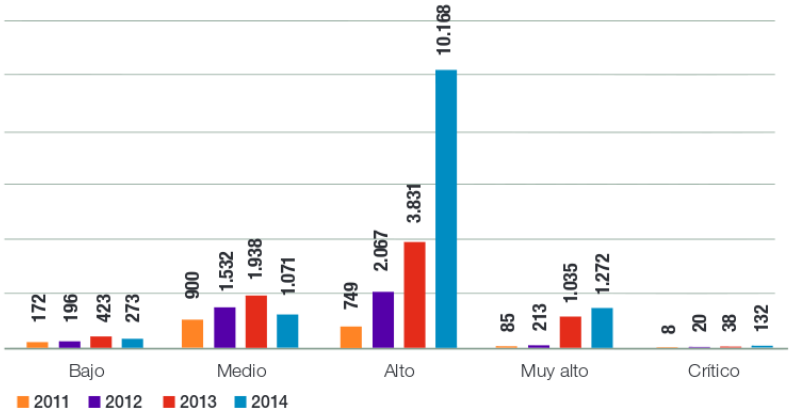

Figura 1.2: Incidentes gestionados por el CCN-CERT (en número y criticidad).[3]

Justamente, como se detalla en dicho informe, las amenazas especialmente significativas durante los últimos años están siendo el ciberespionaje, las Amenazas Persistentes Avanzadas (APT), código dañino y ransomware. [3]

Pero como si de una historia de héroes y villanos se tratase, la seguridad informática, es la disciplina que trata de combatir este tipo de ataques, garantizando cuatro importantes principios en todo sistema informático:

- Confidencialidad, es decir, la información solo es accesible para agentes autorizados.
- Disponibilidad de la información.
- Integridad, control de la modificación de la información.
- Autenticidad, que permite asegurar el origen de la información. [10]

[2]Capacidad de Respuesta a incidentes de Seguridad de la Información del Centro Criptológico Nacional
[3]Organismo adscrito al Centro Nacional de Inteligencia (CNI). `https://www.ccn.cni.es/`

Gracias a la concienciación que, poco a poco, va adquiriendo la población sobre los riesgos a los que estamos expuestos, la seguridad informática crece actualmente como una necesidad para ayudar a las empresas a hacer de su sistema informático un sistema seguro.

1.1. Un proyecto en seguridad informática para una matemática computacional

Cuando realicé la búsqueda de la empresa en la que iba a realizar mi estancia en prácticas, me aseguré de que ésta tuviese un gran interés en los proyectos de I+D+i dentro del ámbito de la seguridad. Fue entonces, cuando encontré S2 Grupo, una empresa valencia que colabora con el Centro Criptológico Nacional (CCN) y que en los últimos años ha recibido un importante número de reconocimientos por su éxito y carácter innovador.

Para muchos, la única relación posible entre la seguridad informática y las matemáticas es el arte de la criptología. Pero lo cierto es que un perfil de matemático computacional en este ámbito, puede dar mucho más de si.

Precisamente, fue una propuesta de S2 Grupo el realizar un Trabajo de Fin de Grado (TFG) que aunara la seguridad informática y las matemáticas por medio de el análisis de datos. El proyecto en el que trabajé consiste en el desarrollo de una herramienta capaz de detectar nombres de dominio DNS maliciosos a través de un modelo de clasificación.

1.1.1. Objetivo de negocio

El objetivo del proyecto propuesto es el análisis de tráfico DNS de un sistema para la identificación de amenazas internas, malware y APT. Para ello, se propone desarrollar una herramienta de detección de nombres de dominio DNS maliciosos.

Por otro lado, el propósito de S2 Grupo es poder utilizar esta herramienta, o al menos parte de ella, para mejorar CARMEN en un futuro.

1.1.1.1. CARMEN (Centro de Análisis de Registros y Minería de EveNtos)

CARMEN fue desarrollado por el CCN y S2 Grupo para la identificación de APT en sistemas informáticos.

Se trata de una herramienta que adquiere, procesa y analiza información de los tráficos de una red, para que posteriormente un analista sea capaz de tomar decisiones a partir de los

resultados proporcionados. Su cometido es la detección de usos indebidos de los tráficos de red y anomalías significativas, y la incorporación de nuevos conocimientos sobre dichas anomalías.

Las fuentes de datos de las que CARMEN obtiene información son: HTTP, DNS, SMTP y IPC.

De este modo, identifica movimientos externos (sistema C&C y servidor de exfiltración) y movimientos laterales de una APT, cubriendo también todas las vías de comunicación del sistema. [2]

1.1.1.2. ¿Cómo encaja el proyecto en CARMEN?

Con este proyecto se pretende llevar a cabo el procesamiento y análisis de las peticiones DNS de un sistema informático, centrándose únicamente en este tipo de comunicación. Más concretamente, la herramienta debe ser capaz de recibir un nombre de dominio, extraer información del mismo, y clasificarlo como legítimo o malicioso.

Para poder tener la posibilidad de adaptar dicha herramienta a CARMEN, se pide que sea un sistema modular en Python, es decir, que esté formada por módulos (o programas) independientes en Python para que posteriormente sea más fácil realizar futuras modificaciones.

1.1.2. Objetivos técnicos

Para poder llevar a cabo mi trabajo en el proyecto se deben cumplir los siguientes objetivos:

1. Definición de la información necesaria para realizar la clasificación de un nombre de dominio DNS.

   A priori, no tenemos información alguna sobre el nombre de dominio a clasificar, por ello, necesitamos definir qué tipo de información puede ser útil para llevar a cabo su clasificación.

2. Desarrollo de programas en Python para la obtención de dicha información.

3. Selección del modelo de clasificación a utilizar.

4. Desarrollo de programas en Python para la preparación de los datos.

   Dada la gran variedad de información que podemos obtener de un dominio, es necesario prepararla para que pueda ser procesada por un modelo de clasificación. Los datos deben estar normalizados.

5. Desarrollo de un programa en Python capaz de clasificar un nombre de dominio en función de la información disponible.

6. Realización de una batería de pruebas para la evaluación del modelo de clasificación.

1.1.3. Objetivos formativos

Mediante la realización de este TFG se pretende adquirir nuevos conocimientos y profundizar sobre las siguientes áreas:

- Modelo de clasificación: Redes Neuronales.

  De entre los modelos de clasificación estudiados en el Grado en Matemática Computacional, escogí las redes neuronales. A mi entender, este modelo era el que mejor se adaptaba a los requisitos de S2 Grupo. Con este TFG, no solo se aprende a implementarlas y/o utilizarlas, sino también se debe estudiar cómo funcionan y en qué se basan.

- Análisis DNS.

  Análisis del tráfico DNS de un sistema infectado por un malware o APT, para poder detectar este tipo de acciones.

- Conocer más sobre las técnicas de evasión de malware para no ser detectados en sus conexiones mediante DNS.

Capítulo 2

Chica en prácticas contra los malos, un granito de arena

Cierto es, que el campo de la seguridad informática me apasiona, y la posibilidad de realizar un TFG relacionado con esta disciplina fue más que grata. Con este trabajo esperaba aportar mi granito de arena a la empresa y adentrarme un poco más en este mundo.

Además, tuve la oportunidad de realizar mi estancia en prácticas mientras disfrutaba de una beca SICUE[1] que me permitió finalizar mis estudios en la UPM[2]. Esto, juntamente con el poder trabajar y ver cómo funciona una empresa de seguridad informática desde dentro, hizo de mi último año una gran experiencia.

2.1. S2 Grupo

Gran parte del desarrollo de la herramienta tuvo lugar durante mi estancia en prácticas. Dicha estancia la realicé en una empresa puntera, al menos a nivel nacional, en el ámbito de la seguridad informática, S2 Grupo, más concretamente en su sede de Madrid.

Como ellos mismos se definen en su dossier corporativo, S2 Grupo *es una empresa especializada en la seguridad de los procesos de negocio que ayuda a sus clientes a proteger su activo más valioso: la información, por lo que su cometido se centra muchas veces en la seguridad de la información desde cualquiera de sus diferentes puntos de vista.*

Se trata de una empresa de prestación de servicios especializados para la Monitorización de la Actividad de Negocio mediante la aplicación de Sistemas de Gestión en Tiempo Real, espe-

[1] Sistema de Intercambio entre Centros Universitarios Españoles
[2] Universidad Politécnica de Madrid

cialmente en el ámbito de la Gestión de Sistemas de Seguridad de los Sistemas de Información.

La empresa sigue cinco grandes líneas de negocio: Consultoría y auditoría de seguridad, Servicios de seguridad gestionada, Explotación segura de sistemas, Implantación de producto y Desarrollo seguro de aplicaciones.

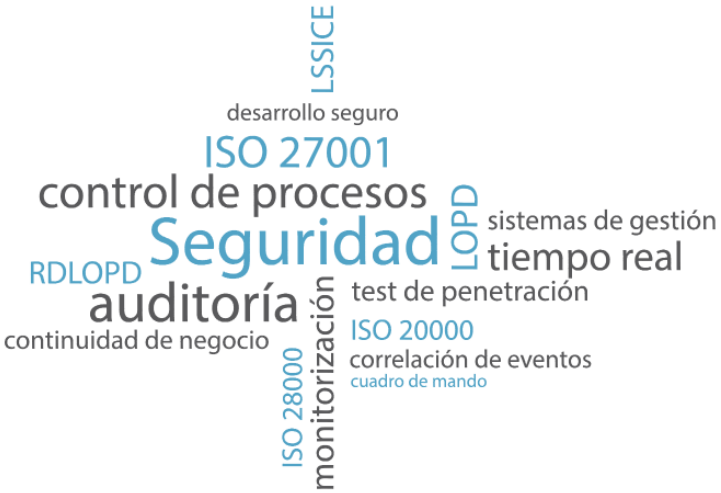

Figura 2.1: Lineas de negocio de S2 Grupo

De entre las distintas áreas que comprende S2 Grupo, realicé mi estancia en prácticas en el área de seguridad, cuyo director, Antonio Villalón, fue mi supervisor junto a Antonio Sanz, uno de los consultores en Madrid, al cual le era mucho más fácil supervisar la evolución del trabajo puesto que ambos trabajábamos en la misma ciudad.

Uno de los cometidos del equipo de seguridad de Madrid es la gestión de incidencias de seguridad por medio de herramientas como la que se muestra en la figura 2.2.

Otra de sus funciones es la explotación de sistemas críticos y realización de auditorías.

Cabe decir, que dado que S2 Grupo es una empresa que ofrece soluciones y servicios a clientes, muchos expertos de seguridad tienen objetivos específicos hechos a medida para cada cliente. [7]

Sin embargo, mi cometido en la empresa era bien distinto, puesto que mi objetivo era el desarrollo de una herramienta de seguridad. Aún así, durante mi estancia, tuve un gran apoyo por parte de mis compañeros del área de seguridad, puesto que no dudaron en intentar ayudarme a solucionar las dudas o problemas que iban surgiendo a medida que avanzaba con el trabajo.

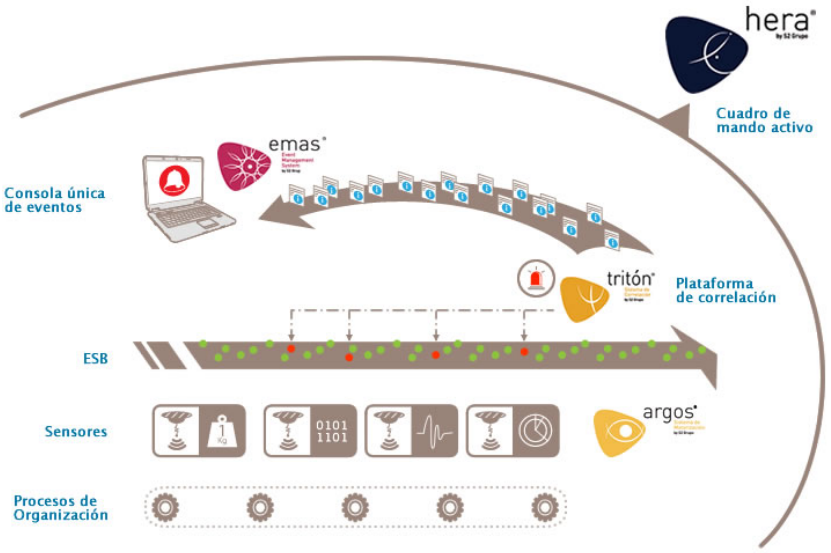

Figura 2.2: Monitorización de infraestructuras TIC, especialmente en lo relativo a seguridad de la información. Más información: `https://www.youtube.com/watch?v=7eEyC8_eY38`

2.2. Metodología y definición de tareas

Durante mi formación en la UPM en la asignatura de *Data Analytics* y de la mano de Ernestina Menasalvas, simulamos un concurso en el que una empresa de odontología buscaba la extracción de conocimiento de datos para la predicción del valor de éxito de un implante dental. Para ello, presentamos por equipos diferentes planes de proyecto que fuesen atractivos para la empresa para luego llevarlo a cabo.

Figura 2.3: Métodología CRISP-DM. [12]

Para dicho proyecto de minería de datos seguimos una metodología CRISP-DM (Cross Industry Standard Process for Data Mining), y puesto que también se adaptaba bastante bien al TFG, decidí basarme en ella.

CRISP-DM divide la evolución de un proyecto de minería de datos en seis fases:

- Comprensión del negocio en el que se desenvuelve el proyecto.

  En nuestro caso, tan solo fue necesario profundizar en algunos aspectos de la seguridad informática para comprender mejor el objetivo del proyecto en el que trabajé.

- Comprensión de los datos de los que se disponen.

  A priori no se dispone de información sobre los nombres de dominio a clasificar. Luego, se necesitó averiguar cómo conseguir dicha información y comprender su valor y significado.

- Preparación de los datos.

  Esta fase puede repetirse múltiples veces durante la realización del proyecto. Consiste en transformar, derivar o eliminar atributos de cada una de las entradas del conjunto de datos que se dispone.

- Modelado.

  Selección del modelo de clasificación y calibración de sus parámetros para la obtención de resultados óptimos.

- Evaluación.

- Implantación.

Se debe notar que la relación entre las fases es como la que se indica en el diagrama 2.3, es decir, para obtener mejores resultados en el modelado suele ser necesario volver a iniciar parte del proceso de compresión y preparación de los datos. [6]

2.3. Planificación temporal

Era la hora de la verdad, había que ponerse a trabajar, y lo primero fue diseñar una planificación temporal de las tareas que se debían realizar. Para ello, me base en el plan de proyecto de minería de datos que diseñé junto a otros dos compañeros en la asignatura de *Data Analytics* de la UPM.

Aún así, fue muy difícil elaborar una planificación inicial realista, puesto que desconocía los posibles problemas o imprevistos que podían surgir y su coste temporal, tanto en la fase de búsqueda de información como de implementación.

Finalmente, se definió a grandes rasgos la siguiente planificación inicial:

- Primeras quince semanas:

- Preparación del entorno de trabajo.
- Asimilación de los nuevos conceptos sobre seguridad para la comprensión del objetivo de la herramienta.
- Búsqueda de información sobre los diferentes ciberataques que utilizan conexiones DNS.
- Búsqueda de información sobre las técnicas de recolección de información en Python sobre un nombre de dominio DNS.
- Implementación de funciones en Python para la realización la obtención de dicha información.

- De la décimo sexta a la vigésimo segunda semana:
  - Preparación del conjunto de datos para poder ser procesados por el modelo.
  - Implementación del modelo de clasificación.
- Últimas cuatro semanas:
  - Ajuste de los parámetros del modelo de clasificación.
  - Evaluación del modelo.

A medida que avanzaba en mi trabajo, eramos conscientes de que los plazos definidos no iban a poder cumplirse.

Esto se debió a la compleja búsqueda de información sobre ciberataques clara y concisa, que además profundice en la relación que un determinado malware mantiene con el DNS. Otro de los problemas que tuve, fue el poder encontrar librerías de Python diseñadas para trabajar con peticiones DNS, que fueran capaces de proporcionar la mayor cantidad de información sobre un dominio sin errores.

Otro de los problemas que surgieron fue la lentitud con que las libreria de MySQL de Python trataba las inserciones y consultas en la base de datos. Por ello, decidí pasar a utilizar archivos JSON para el almacenamiento de datos. A pesar de que familiarizarme con ellos no me resultó difícil, el realizar los cambios en todos los programas y el crear la base de datos desde cero, sí llevo bastante tiempo.

Así pues, decidí diseñar una nueva planificación temporal frente a los problemas que habían aparecido hasta el momento.

- Primeras dos semanas:
  - Preparación del entorno de trabajo.
  - Asimilación de los nuevos conceptos sobre seguridad para la comprensión del objetivo de la herramienta.

    - Búsqueda de información sobre el modelo de clasificación a desarrollar.

- De la tercera a la séptima semana:

    - Búsqueda de información sobre los diferentes ciberataques que utilizan conexiones DNS.

- De la octava a la onceava semana:

    - Búsqueda de información sobre las técnicas de recolección de información en Python sobre un nombre de dominio DNS.

- De la décimo segunda a la vigésima semana:

    - Implementación de funciones en Python para la realización la obtención de dicha información.
    - Llevara a cabo las modificaciones correspondientes al almacenamiento de datos.

- La vigésimo primera semana:

    - Reforzar el código y corregir errores de implementación.

- De la vigésimo segunda a la vigésimo quinta semana:

    - Preparación del conjunto de datos para poder ser procesados por el modelo.
    - Implementación de funciones para la preparación de datos.

- La vigésimo sexta semana:

    - Reforzar el código y corregir errores de implementación.

De este modo, quedaban fuera de plazo:

- Implementación del modelo de clasificación.
- Ajuste de los parámetros del modelo de clasificación.
- Evaluación del modelo.

A causa de esto, la documentación sobre mi trabajo debía ser completa y detallada para poder ofrecer la posibilidad de que otra persona dentro de la empresa pudiese finalizarlo.

2.4. Recursos utilizados

A mi entrada, S2 Grupo me facilitó un equipo con conexión a la red de la empresa por cable y me asignaron una mesa de trabajo junto con los técnicos del área de seguridad.

Por el contrato de confidencialidad y las normas de seguridad de la empresa, el disco duro del equipo debía estar cifrado. Sin embargo, me dieron total libertad para instalar el sistema operativo que deseara, e instalé Linux Mint[3] y Kali Linux[4].

A nivel de software, al principio utilicé entornos de desarrollo interactivo para Python como Spyder y Eclipse[5]. Al final, debido a pequeñas complicaciones a la hora de la importación de módulos, decidí usar Sublime Text, editor de texto desarrollado originalmente como una extensión de Vim[6].

Por otra parte, se recurrió a múltiples librerías de Python que más adelante detallaremos, las cuales se consiguieron tanto por medio de los repositorios de Linux, como utilizando Python Package Index (PyPI). PyPI es el repositorio oficial para aplicaciones de Python en código abierto implementadas por terceros.

Además de esto, se necesitaron gran cantidad de nombres de dominio sobre los que poder realizar pruebas y utilizarlos después para el entrenamiento de la red neuronal. Este tipo de datos se consiguieron a través de diversas páginas web.

2.5. Resultados obtenidos

La verdad es que trabajar en un proyecto de estas características fue todo un reto, puesto que no pertenecía a un equipo de trabajo, sino que el avance del proyecto recaía únicamente sobre mi.

No obstante, pude lograr gran parte de los objetivos técnicos del trabajo. En concreto, los resultados obtenidos fueron los siguientes:

- Realización de un estudio sobre la información necesaria para realizar la clasificación de un nombre de dominio DNS. En el informe que redacté para S2 Grupo al concluir mi estancia prácticas, se hizo una descripción detallada dicha información.
- Desarrollo de un conjunto de programas en Python para la obtención de dicha información.

[3]Distribución del sistema operativo GNU/Linux
[4]Distribución basada en Debian GNU/Linux diseñada principalmente para la auditoría y seguridad informática
[5]IDE originalmente diseñado para JAVA, pero con plugins que permiten trabajar con Python
[6]Editor de texto propio de los sistemas UNIX

- Realización de un estudio y análisis de las redes neuronales como modelo de clasificación.
- Realización de un estudio de los criterios de normalización escogidos para cada uno de los atributos de los nombres de dominio.
- Desarrollo de un conjunto de programas en Python capaz de normalizar los datos.

Además, antes de la finalización de mi estancia, se estudiaron las diferentes librerías de las que dispone Python para la implementación de la red neuronal, y al final se decidió utilizar http://scikit-learn.org/stable/tutorial/index.html.

A pesar de no poder implementar la herramienta al completo durante mi estancia, con las recomendaciones que se describían en el informe que entregué a la empresa y las que en esta misma memoria se detallan, desde S2 Grupo no tendrán ningún impedimento para hacerlo.

Capítulo 3

Redes Neuronales

Índice

3.1. Introducción 24

3.2. Introducción al modelo PPR (Projection Pursuit Regression) . . . 24

3.2.1. PPR en función del parámetro M 25

3.2.2. Ajuste del modelo PPR 25

3.2.3. PPR y redes neuronales 26

3.3. Redes Neuronales 26

3.3.1. Similitudes con el modelo PPR 28

3.3.2. Un poco de historia 28

3.4. Ajuste de una red neuronal 29

3.5. Entrenando una red neuronal 31

3.5.1. Definiendo los valores de inicio: los pesos 31

3.5.2. Combatiendo el sobreajuste 32

3.5.3. Escalado de entradas 32

3.5.4. Número de unidades ocultas y capas 34

3.5.5. Múltiples mínimos 34

3.6. Consideraciones computacionales 34

Toda la información que se detalla en este capítulo es del libro: *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* por Trevor Hastie, Robert Tibshirani y Jerome Friedman [9].

3.1. Introducción

Durante mi formación en el grado, el concepto de *redes neuronales* ha estado presentes en dos disciplinas, la inteligencia artificial y la estadística. Esto se debe a que esta clase de métodos de aprendizaje ha sido desarrollada en ambas vertientes, pero siendo siempre un mismo modelo.

Un modelo de clasificación recibe ciertos atributos de una entrada y la clasifica en función de éstos. El propósito de este modelo es la extracción de combinaciones lineales de los atributos derivados de las entradas y luego modelar el resultado como una función no lineal de dichos atributos. De ese modo, conseguimos un poderoso método de aprendizaje con gran variedad de aplicaciones en diversos campos.

3.2. Introducción al modelo PPR (Projection Pursuit Regression)

Como en todo problema de aprendizaje supervisado disponemos de una entrada $X \in \mathbb{R}^p$ y una salida $Y \in \mathbb{R}$.

Definición 3.2.0.1. *Sea w_m, $m = 1, 2, \cdots, M$, p-vectores (vectores de p componentes) unitarios de los parámetros desconocidos. El **modelo PPR** es un modelo aditivo que se define como:*

$$f(X) = \sum_{m=1}^{M} g_m(w_m^T X) \tag{3.1}$$

Siendo $V_m = w_m^T X \in \mathbb{R}$ la derivación de los atributos de la entrada, en concreto, es la proyección de X sobre w_m. En el modelo, trataremos de buscar w_m para que el modelo se ajuste bien.
*La función $g_m(w_m^T X) \in \mathbb{R}^p$, es conocida como **función de canto** (ridge function), y como podemos notar, solo varia en la dirección de w_n. Las funciones g_m no están definidas, sino que se estiman a lo largo de las direcciones de w_m por medio de algún método de suavizado flexible.*

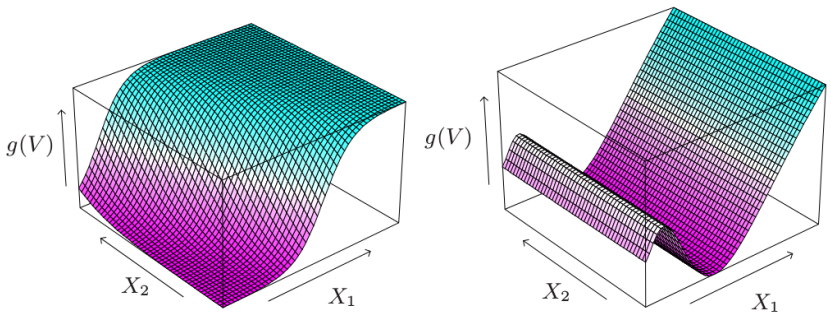

Figura 3.1: Gráficos sobre dos ejemplos de funciones de canto.

En la figura 3.1, podemos ver ejemplos de funciones de canto. A la izquierda: $g(V) = \frac{1}{1+exp(-5(V-0{,}5))}$ con $V = \frac{X_1+X_2}{\sqrt{2}}$. A la derecha: $g(V) = (V + 0{,}1)\sin\left(\frac{1}{V/3+0{,}1}\right)$ con $V = X_1$.

3.2.1. PPR en función del parámetro M

Si M es arbitrariamente grande, por la elección apropiada de g_m, el modelo PPR podría aproximar bastante bien cualquier función en $\mathbb{R}^p$, convirtiéndose así en un *modelo de aproximación universal*. Sin embargo, la interpretación del modelo suele ser dificil, puesto que las entradas se introducen en el modelo de forma compleja.

Por otro lado, con $M = 1$ tenemos un *modelo de índice único* propio de la econometría, es una excepción. Este modelo es ligeramente más general que el modelo de regresión lineal y ofrece una interpretación similar.

3.2.2. Ajuste del modelo PPR

Dado un conjunto de datos de entrenamiento (x_i, y_i) con $i = 1, 2, \cdots, N$, buscamos minimizar la **función de error**:

$$\sum_{i=1}^{N} \left[y_i - \sum_{m=1}^{M} g_m(w_m^T x_i) \right]^2 \tag{3.2}$$

Como en todo problema de suavizado necesitamos imponer límites de complejidad, bien explicita o implícitamente para prevenir el sobreajuste.

Consideremos $M = 1$.

Dado el vector director w, formamos las variables derivadas $v_i = w^T x_i$. De este modo tenemos un problema de suavizado unidimensional y podemos aplicar *scatterplot smoother*, o bien, *smoothing spline*, para obtener una estimación de g.

Por otro lado, dada g, debemos minimizar 3.2 sobre w. Para ello podemos utilizar la búsqueda Gauss-Newton. Este tipo de búsqueda es un método de quasi-Newton en el que la parte del hessiano que tiene la segunda derivada de g se descarta. Se puede derivar sencillamente como sigue: *Sea w_{old} la actual estimación de w, tenemos que:*

$$g(w^T x_i) \approx g(w_{old}^T x_i) + g'(w_{old}^T x_i)(w - w_{old})^T x_i \tag{3.3}$$

dando lugar a:
Sea w_{old} la actual estimación de w, tenemos que:

$$\sum_{i=1}^{N} \left[y_i - g(w^T x_i) \right]^2 \approx \sum_{i=1}^{N} g'(w_{old}^T x_i)^2 \left[\left(w_{old}^T x_i + \frac{y_i - g(w_{old}^T x_i)}{g'(w_{old}^T x_i)^2} \right) - w^T x_i \right]^2 \tag{3.4}$$

Para minimizar el lado derecho, se realiza una regresión por mínimos cuadrados con la salida $w_{old}^T x_i + \frac{y_i - g(w_{old}^T x_i)}{g'(w_{old}^T x_i)^2}$ de la entrada x_i con pesos $g(w_{old}^T x_i)^2$. Esto nos proporciona el vector w_{old} actualizado, w_{new}.

Estos dos pasos, fijar w y actualizar g y viceversa, deben iterarse hasta converger.

3.2.3. PPR y redes neuronales

El modelo PPR no ha sido ampliamente utilizado en el campo de la estadística, tal vez porque en el momento de su introducción (1981), sus demandas computacionales superaron las capacidades de las computadoras más fácilmente disponibles. A pesar de ello, representa un importante avance intelectual, que ha florecido en su reencarnación en el ámbito de las redes neuronales, el tema del resto de este capítulo.

3.3. Redes Neuronales

El término de red neuronal abarca una amplia variedad de modelos y métodos de aprendizaje. En este capítulo, describiremos la red neuronal más utilizada, la red de retropropagación con una única capa oculta.

El funcionamiento interno de las redes neuronales siempre ha estado envuelto en un ambiente de magia y misterio, pero aquí mostraremos que simplemente son modelos estadísticos no lineales, al igual que el modelo PPR visto en la sección anterior.

Una red neuronal es un modelo de regresión o clasificación de dos etapas y se representa como se muestra en la figura 3.2. Para la regresión este tipo de red, típicamente $K = 1$ y solo hay una unidad de salida Y_1.

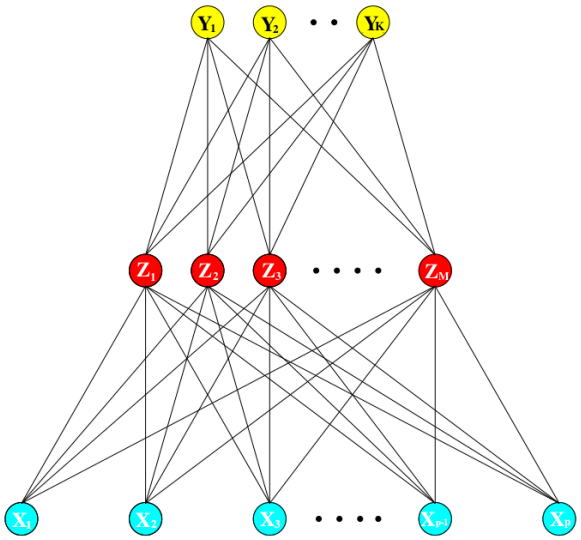

Figura 3.2: Esquema de una red neuronal red neuronal feed-forward con una única capa oculta.

Para la clasificación de K-clases, tenemos como entrada $X = (X_1, X_2, \cdots, X_P)$ y como salida

$Y_1, Y_2, \cdots, Y_K$ en el que el k-ésimo elemento es la probabilidad de que sea de la clase k. Una salida ideal de una entrada clasificada como clase k sería: $Y_k = 1 \quad \wedge \quad Y_i = 0 \quad \forall i \neq k$.

Por otro lado, los atributos derivados $Z_1, Z_2, \cdots, Z_M$ son creados a partir de combinaciones lineales de las X_p unidades de entrada, y la salida $Y_1, Y_2, \cdots, Y_K$ es modelada como una función de combinaciones lineales de los Z_m.

$$Z_m = \sigma(\alpha_{0m} + \alpha_m^T X) \quad con \quad m = 1, \cdots, M$$

$$T_k = \beta_{0k} + \beta_k^T Z \quad con \quad k = 1, \cdots, K$$

$$Y_k = f_k(X) = g_k(T) \quad con \quad k = 1, \cdots, K \tag{3.5}$$

siendo $Z = (Z_1, Z_2, \cdots, Z_M)$, $T = (T_1, T_2, \cdots, T_K)$, $\alpha_m \in \mathbb{R}^P$ y $\beta_k \in \mathbb{R}^M$.

La **función de activación** σ de un nodo define la salida de un nodo dada una entrada o un conjunto de entradas. Como función de activación de los nodos de la capa oculta se suele usar la función sigmoide:

$$\sigma(v) = \frac{1}{1 + e^{-v}} \tag{3.6}$$

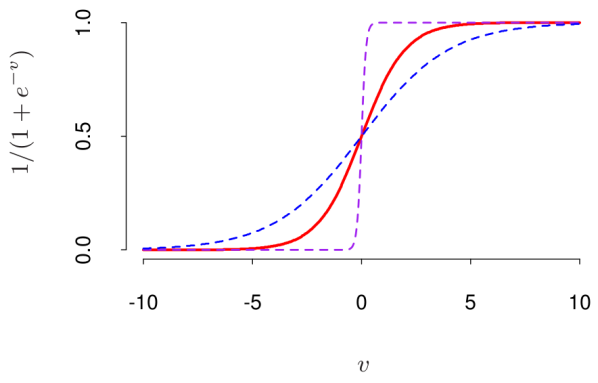

Figura 3.3: En rojo, representación gráfica de la función sigmoide $\sigma(v)$, usada comúnmente en la capa oculta de la red neuronal. En azul, representación gráfica de $\sigma(sv)$ con $s = \frac{1}{2}$, y en morado, con $s = 10$. La escala del parámetro s controla el ratio de activación. Notemos que un valor de s mayor produce una activación más radical en $v = 0$.

La **función de salida** $g_k(T)$ permite la transformación final del vector de salidas T. Para la regresión, se suele escoger la función identidad $g_k(T) = T_k$. Los primeros trabajos en clasificación de K-clases también utilizaban la función identidad, pero posteriormente se ha pasado a utilizar la función *softmax*:

$$g_k(T) = \frac{e^{T_k}}{\sum_{l=1}^{K} e^{T_l}} \tag{3.7}$$

Las unidades de la capa intermedia de la red, computan los atributos derivados Z_m, llamadas unidades ocultas puesto que los valores de Z_m no se muestran de forma explícita. Podemos pensar en Z_m como en una expansión de la entrada original X, la red neuronal sería entonces, un modelo lineal que usa dichas transformaciones como entradas. Notar que en este tipo de redes neuronales, los parámetros de las funciones base son aprendidos de los datos.

Ahora bien, si σ es la función identidad, entonces tendríamos un modelo lineal puesto que Z_m sería combinación lineal de las entradas. Podemos considerar una **red neuronal como una generalización no lineal de un modelo lineal**, tanto para los problemas de regresión como de clasificación.

En la figura 3.3 vemos que el ratio de activación de σ depende de la norma de α_m, y si $||\alpha_m||$ es muy pequeño, la unidad opera en la parte lineal de su función de activación.

3.3.1. Similitudes con el modelo PPR

El modelo de red neuronal con una capa oculta tiene exactamente la misma forma que el modelo PPR introducido en la sección anterior. La diferencia es que el modelo PPR usa funciones no paramétricas $g_m(v)$, mientras que las redes neuronales usan una función mucho más simple basada en σ, con tres variables como argumentos. En concreto, viendo una red neuronal como un modelo PPR, podemos identificar:

$$\begin{aligned} g_m(w_m^T X) &= \beta_m \sigma(\alpha_{0m} + \alpha_m^T X) \\ &= \beta_m \sigma(\alpha_{0m} + ||\alpha_m||(w_m^T X)) \end{aligned} \tag{3.8}$$

con $w_m = \frac{\alpha_m}{||\alpha_m||}$ es la m-ésima componente del vector unidad. Ya que $\sigma_{\beta,\alpha_0,s}(v) = \beta\sigma(\alpha_0 + sv)$ tiene una menor complejidad que $g(v)$, la redes neuronales usan un mayor número de funciones de las que se usan en el modelo PPR.

3.3.2. Un poco de historia

Nombrar que las redes neuronales están basadas en el funcionamiento del cerebro humano. Cada unidad representa una neurona, y cada conexión una sinapsis. En los primeros modelos, las neuronas se activaban cuando la señal que pasaba superaba cierto umbral. En el modelo que hemos descrito, esto se corresponde con el uso de una función escalón para $\sigma(Z)$ y $g_m(T)$.

Más tarde, las redes neuronales fueron reconocidas como una herramienta útil para la modelización estadística no lineal, y por esto, la función escalón no es suficientemente suave para la optimización. De aquí que la función escalón fuese reemplazada por una función de umbral suave como la sigmoide (3.3).

3.4. Ajuste de una red neuronal

El modelo de red neuronal descrito tiene parámetros desconocidos, conocidos como *pesos*. El objetivo es buscar los valores de los pesos apropiados para que el modelo se ajuste al conjunto de datos de entrenamiento. Definimos el conjunto de pesos θ de la red neuronal con una capa oculta como:

$$\begin{array}{c} \{\alpha_{0m}, \alpha_m : m = 1, 2, \cdots, M\} \quad con \quad M(p+1)pesos, \\ \{\beta_{0k}, \beta_k : k = 1, 2, \cdots, K\} \quad con \quad K(M+1)pesos \end{array} \tag{3.9}$$

Para la regresión, usamos como medida de ajuste de parámetros (o función de error), la suma de errores cuadráticos:

$$R(\theta) = \sum_{k=1}^{K} \sum_{i=1}^{N} (y_{ik} - f_k(x_i))^2 \tag{3.10}$$

Para la clasificación, usamos la función anterior o bien, el error de entropía cruzada (*cross-entropy*):

$$R(\theta) = -\sum_{k=1}^{K} \sum_{i=1}^{N} y_{ik} log(f_k(x_i)) \tag{3.11}$$

y su correspondiente clasificador es $G(x) = argmax_k\, f_k(x)$. Con la función *softmax* como función de activación y la función de error de *cross-entropy*, la red neuronal es exactamente un modelo de regresión logística lineal en las unidades ocultas, y todos los parámetros son estimados por el método de máxima verosimilitud.

Normalmente no queremos alcanzar el mínimo global de $R(\theta)$, ya que conduciría a una función que se aproxima demasiado a los datos, razón por la que seguramente dejaría de ser "suave", y perdería la capacidad predictiva (es decir, de medir la tendencia que presenta la variable Y en función de las entradas X_1, X_2,..., Xp). Es el defecto conocido como "sobreajuste". En su lugar, se necesita algún otro tipo de regularización que detallaremos en la siguiente sección.

Para la minimización de $R(\theta)$ se usa el método de descenso por gradiente, también conocido en este entorno como método de retropropagación (optimización). Por la composición del modelo, el gradiente puede ser fácil de de derivar usando la regla de la cadena. Es más, esto puede ser calculado mediante un barrido hacia delante y hacia atrás sobre la red teniendo en cuenta los valores locales de cada unidad.

A continuación detallamos la **retropropagación para la función de error correspondiente a la suma cuadrática de errores**, algoritmo iterativo que mide la calidad de aproximación del método resultante.

Sea $z_{mi} = \sigma(\alpha_{0m} + \alpha_m^T x_i)$ por (3.5) y sea $z_i = (z_{1i}, z_{2i}, \cdots, z_{Mi})$. Entonces, tenemos que

$$\begin{aligned} R(\theta) &\equiv \sum_{i=1}^{N} R_i \\ &= \sum_{i=1}^{N} \sum_{k=1}^{K} (y_{ik} - f_k(x_i))^2 \end{aligned} \tag{3.12}$$

cuyas derivadas son:

$$\frac{\partial R_i}{\partial \beta_{km}} = -2(y_{ik} - f_k(x_i)) g_k'(\beta_k^T z_i) z_{mi}$$

$$\frac{\partial R_i}{\partial \alpha_{ml}} = -\sum_{k=1}^{K} 2(y_{ik} - f_k(x_i)) g_k'(\beta_k^T z_i) \beta_{km} \sigma'(\alpha_m^T x_i) x_{il} \tag{3.13}$$

Dadas estas derivadas, la actualización del descenso del gradiente en la iteración $(r+1)$ es:

$$\beta_{km}^{(r+1)} = \beta_{km}^{(r)} - \gamma_r \sum_{i=1}^{N} \frac{\partial R_i}{\partial \beta_{km}^{(r)}}$$

$$\alpha_{ml}^{(r+1)} = \alpha_{ml}^{(r)} - \gamma_r \sum_{i=1}^{N} \frac{\partial R_i}{\partial \alpha_{ml}^{(r)}} \tag{3.14}$$

dónde γ_r es la **ratio** o velocidad **de aprendizaje**.

Expresamos (3.13) como:

$$\frac{\partial R_i}{\partial \beta_{km}} = \delta_{ki} z_{mi}$$

$$\frac{\partial R_i}{\partial \alpha_{ml}} = s_{mi} x_{il} \tag{3.15}$$

Los valores de δ_{ki} y s_{mi} son ***errores*** del modelo actual, en la salida y las unidades de la capa oculta, respectivamente. Por sus definiciones, estos errores satisfacen que:

$$s_{mi} x_{il} = \sigma'(\alpha_m^T x_i) \sum_{k=1}^{K} \beta_{km} \delta_{ki} \tag{3.16}$$

conocidas como las **ecuaciones de retropropagación**. Usando esto, las actualizaciones de 3.14 pueden implementarse con un algoritmo de dos pasadas.

- En el barrido hacia adelante, los pesos actuales están fijados y se calcula $\hat{f}_k(x_i)$ con (3.5).

- En el barrido hacia atrás, δ_{ki} son utilizados para obtener los errores s_{mi} (3.16). Ambos errores son utilizados después para calcular los gradientes de las actualizaciones en (3.14), vía (3.15).

La actualización en (3.14) es un tipo de *batch learning* en el que los parámetros son actualizados por una suma de todos los casos de entrenamiento. Para el ratio de aprendizaje γ_r suele tomarse una constante.

Otro tipo de actualización es mediante *online learning*, en el que los parámetros se actualizan tras el procesamiento de cada entrada. De este modo, se puede trabajar con un conjunto de datos entrenamiento muy grande. En este caso, el ratio de aprendizaje γ_r debe tender a cero a medida que se realizan las iteraciones ($r \to \infty$).

En cuanto a la computación, la retropropragación puede ser muy lenta y por ello no se suele utilizar. Se usan otras técnicas conocidas como "gradientes conjugadosz "métodos de métrica variable", que no comentaremos en este trabajo.

3.5. Entrenando una red neuronal

Entrenar una red neuronal no es trivial puesto que, por lo general, es un modelo sobreparametrizado y el problema de optimización es no-convexo e inestable.

En esta sección se detallarán una serie de cuestiones importantes a tener en cuenta durante el entrenamiento de una red neuronal.

3.5.1. Definiendo los valores de inicio: los pesos

- Si tomamos el cero como valor inicial de los pesos, las derivadas se anulan, y el algoritmo no mueve la iteración inicial: no avanza.

- Si los pesos son cercanos a cero, la función sigmoide σ es prácticamente lineal y la red neuronal se convertiría en una aproximación de un modelo lineal.

- Se suele empezar con valores aleatorios próximos a cero, el modelo comienza siendo lineal y se convierte en no-lineal a medida que el valor de los pesos incrementa.

- Si tomamos como pesos iniciales valores grandes, es fácil que el algoritmo converja a una solución muy lejana de ser óptima.

3.5.2. Combatiendo el sobreajuste

A menudo, las redes neuronales tienen demasiados pesos y se pueden sobreajustar los datos en el mínimo global de R. Existen diferentes enfoques para la regularización:

- Interrupción temprana.
  - Utilizado en los primeros desarrollos de las redes neuronales.
  - Se previene el sobreajuste deteniendo el modelo justo antes de alcanzar el mínimo global de $R(\theta)$.
  - Solo se puede entrenar durante un tiempo determinado.
  - Como los pesos iniciales son próximos a 0, inicialmente se tiene una solución lineal altamente regularizada y además, la interrupción temprana reduce el modelo hacia un modelo lineal.
  - Se puede utilizar un conjunto de datos de validación para determinar cuándo parar.
- Decaimiento o rebaja de los pesos.
  - Añade una penalización a la función de error R: $R(\theta) + \lambda J(\theta)$, con $J(\theta) = \sum_{km} \beta_{km}^2 + \sum_{ml} \alpha_{ml}^2$ y $\lambda \geq 0$ parámetro de ajuste.
  - Cuanto mayores son los valores de λ, valores de pesos grandes aumentan la función de error, y la búsqueda del error óptimo (mínimo) tiende a reducir dichos pesos.
  - Se suele usar validación cruzada para elegir el valor de λ.
  - En la figura 3.4, podemos ver un ejemplo de un modelo con y sin decaimiento de pesos.
  - En la figura 3.5, vemos que el decaimiento de pesos ha amortiguado los pesos en ambas capas: los pesos resultantes se distribuyen de manera bastante uniforme en las últimas diez unidades ocultas.

3.5.3. Escalado de entradas

El escalado de entradas determina la escala de pesos de la capa inferior.

Al principio, lo mejor es estandarizar todas las entradas para que tengan una media cero y desviación estándar uno. Debemos estar seguros de que todas las entradas son tratadas por igual en el proceso de regularización.

Con la estandarización de entradas, es corriente tomar pesos uniformes y aleatorios dentro del rango [-0.7, +0.7].

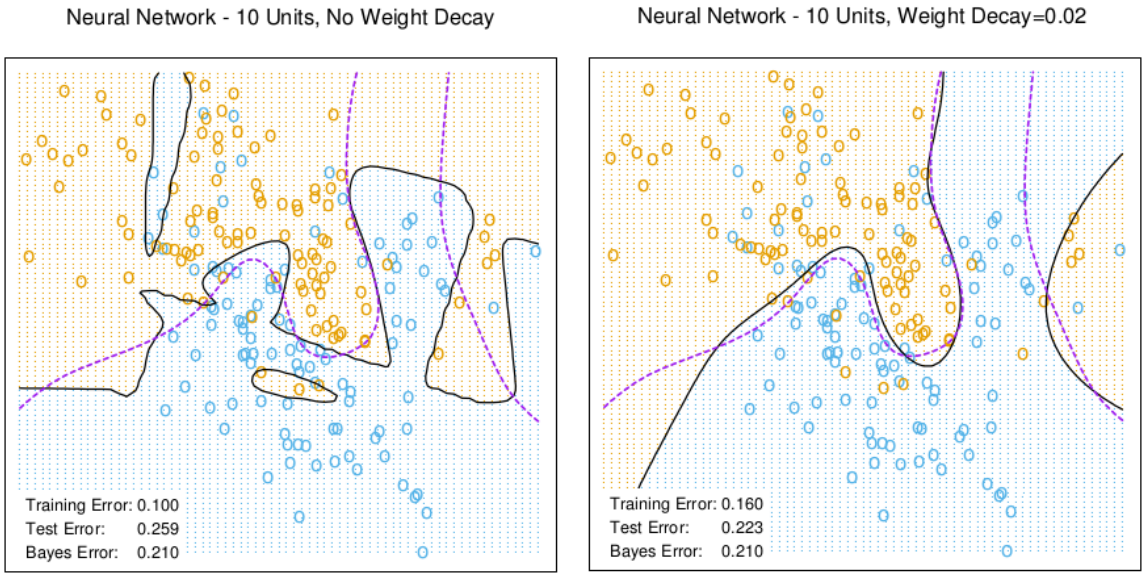

Figura 3.4: Respresentación de los resultados de una red neuronal que usa una función de activación *softmax* y *error de entropía cruzada*. A la izquierda: Sin usar decadencia de peso. A la derecha: Usando decadencia de peso y logra aproximarse a el ratio de error de Bayes (morado).

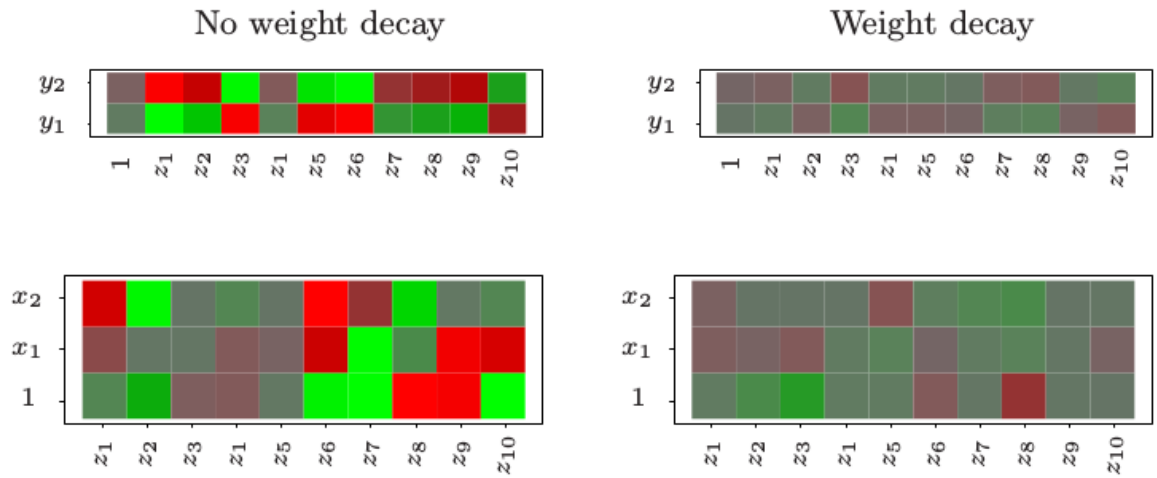

Figura 3.5: Mapas de calor de los pesos estimados a partir de entrenamiento de las redes neuronales de la figura 3.4. La gama de colores varía de verde brillante (negativo) a rojo brillante (positivo).

3.5.4. Número de unidades ocultas y capas

En general, es preferible tener demasiadas, que pocas unidades ocultas. Una baja cantidad de unidades ocultas implica un modelo poco flexible, mientras que con demasiadas capas, los pesos que sobren se pueden ir perdiendo fuerza usando una regularización adecuada.

La utilización de múltiples capas ocultas permite la construcción de funciones jerárquicas en distintas resoluciones.

3.5.5. Múltiples mínimos

La función de error $R(\theta)$ no es convexa, y tiene muchos mínimos locales. Como consecuencia, la solución final obtenida depende bastante de la elección de los pesos iniciales.

Tenemos varias opciones para evitar este problema de caída en mínimos locales:

1. Entrenar diversas redes neuronales cuyos pesos iniciales se han seleccionado al azar. Finalmente, escoger aquella red neuronal cuya penalización de error sea mínima.
2. Entrenar diversas redes neuronales cuyos pesos iniciales han sido seleccionados al azar para realizar un ejemplo de prueba promedio de la predicción de cada red.
3. Bagging. Entrenar diversas redes neuronales seleccionando los subconjuntos de datos de entrenamiento de forma aleatoria para realizar un ejemplo de prueba promedio de la predicción de cada red.

3.6. Consideraciones computacionales

Siendo N la cantidad de observaciones, P la cantidad de atributos de X, M las unidades ocultas y L las épocas de entrenamiento, el ajuste de una red neuronal, generalmente, requiere un coste de $O(NPML)$ operaciones.

Capítulo 4

Análisis sobre la información necesaria para la clasificación

Índice

4.1. DNS: Una puerta entreabierta para las ciberamenazas 35
4.1.1. Sistema de Nombres de Dominio 35
4.1.2. Nombre de dominio DNS 36
4.1.3. Seguridad en el DNS 36

4.1. DNS: Una puerta entreabierta para las ciberamenazas

En esta sección, se definirán los conceptos básicos relacionados con el DNS, y después, se tratará de explicar cual es su papel en las ciberamenazas.

4.1.1. Sistema de Nombres de Dominio

El Sistema de Nombres de Dominio (DNS) es un sistema de nomenclatura jerárquica para cualquier recurso conectado a Internet o a una red privada. Asocia información variada a nombres de dominio y además es capaz de traducirlos a direcciones IP físicas. Existen tres partes que componen este sistema:

- Clientes de fase 1. Programa cliente DNS que se ejecuta en un ordenador y genera peticiones

DNS a un servidor DNS para la resolución de nombres de dominio.

- Servidores DNS. Tratan de resolver las peticiones de los clientes. Si se trata de un servidor recursivo y no disponen de la dirección solicitada, reenvían la petición a otro servidor.
- Zonas de autoridad. Parte del espacio de nombres sobre la que un servidor DNS es responsable.

4.1.2. Nombre de dominio DNS

Un nombre de dominio, es una cadena de texto formada por un conjunto de etiquetas separadas por puntos. Por ejemplo, *e-ujier.uji.es.*. Estas etiquetas siguen un orden jerárquico, de derecha a izquierda.

El dominio de nivel superior (TLD) es el situado lo más a la derecha posible. En el ejemplo anterior sería *es.*. El resto de etiquetas a su izquierda son sumbdominios de nivel inferior. La etiqueta de nivel más bajo suele especificar el nombre de la máquina.

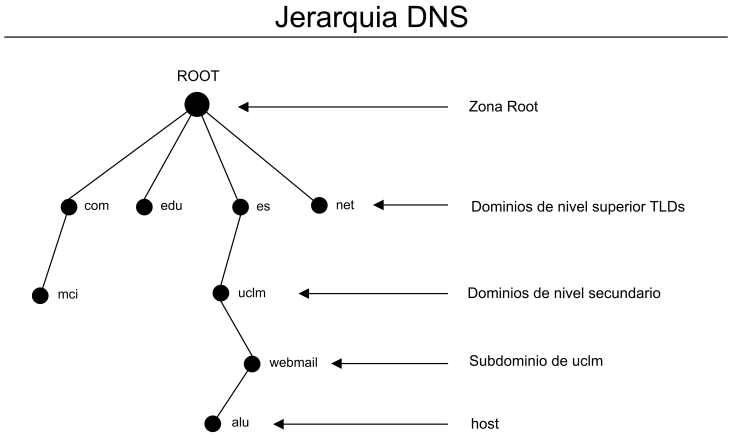

Figura 4.1: Sistema de nomenclatura jerárquico del DNS. [4]

4.1.3. Seguridad en el DNS

Primeras amenazas

A partir de 1990, con la expansión de internet al sector comercial, empezaron a cambiar los requisitos de las medidas de seguridad del diseño software DNS para proteger la integridad de los datos y autenticación de usuarios.

Se descubrieron diversas vulnerabilidades en el sistema que fueron explotadas por usuarios maliciosos. Uno de estos problemas es el envenenamiento de la memoria caché del DNS, en el que

se introducen datos en la memoria caché de resolución del DNS, provocando que el servidor de nombres devuelva información falsa, y poder desviar el tráfico hacia la máquina de un atacante.

Problemas de autenticidad

Las respuestas DNS no suelen estar firmadas [1] lo que es un punto débil del sistema fácil de atacar. Domain Name System Security Extensions (DNSSEC) modificaron el DNS para añadir soporte para el firmado criptográfico de respuestas. Tras esto, ha habido otras alternativas como DNSCurve o TSIG para dar también, apoyo criptográfico.

Ataques typosquatting

Algunos nombres de dominio se pueden usar para lograr efectos de suplantación de identidad. Por ejemplo, *paypal.com* y *paypa1.com* son diferentes nombres, sin embargo, los usuarios pueden ser incapaces de distinguirlos en una interfaz gráfica, este tipo de ataque se conoce como typosquatting. Esta vulnerabilidad se explota de vez en cuando en el phishing. [1]

Técnicas tales como la resolución DNS inversa (rDNS) puede ayudarnos a validar los resultados de DNS.

Bootnets

Las bootnets tienen como objetivo extenderse lo máximo posible. Si hay algo que caracterice a las bootnets, esto es la capacidad de sus miembros o bots de comunicarse entre ellos y con los sistema C&C. De ese modo, los bots son capaces de obtener nuevas instrucciones y actualizaciones.

Para establecer este tipo de conexiones, hay dos posibilidades, usar direcciones IP fijas dentro del código del malware o, usar nombres de dominio fijos o generados mediante un algoritmo DGA.

Puesto que un dominio puede estas asociado a múltiples direcciones IP, esto dificulta aún mas la detección de una actividad sospechosa.

Esto se complica todavía más si los desarrolladores utilizan técnicas como Fast-Flux, en la que un nombre de dominio resuelve una dirección IP distinta en función del momento en el que se realice la petición. De este modo, se consigue descentralizar los servidores C&C y que sea mucho más compleja su identificación.

[1] Como firma se entiende el hecho de cifrar algo (normalmente el resultado de una función hash del archivo a firmar) con la clave privada del autor, para que todo aquel que tenga acceso a su certificado pueda comprobar su autenticidad descifrando con su clave pública.

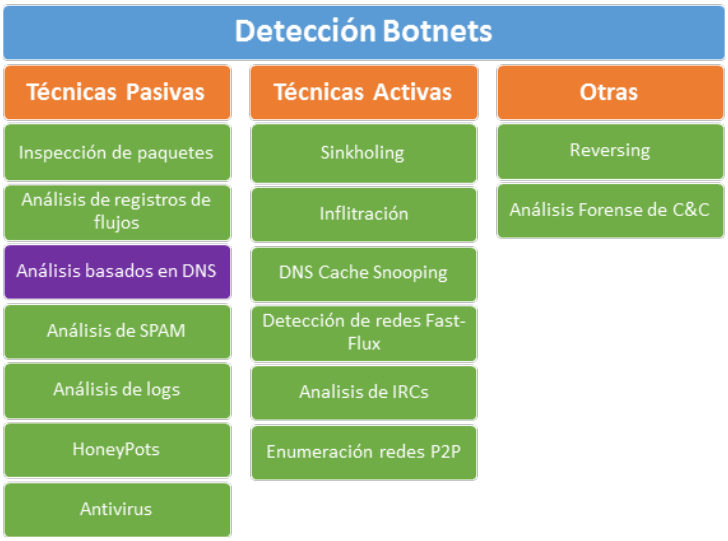

Figura 4.2: Técnicas de detección de bootnets.

Si por el contrario se usan direcciones IP fijas para la conexión, existen técnicas basadas en DNS que dificultan su detección. Aún así, por medio de técnicas como el análisis de flujo, permiten detectarlas puesto que las direcciones IP no varían. [5]

Amenazas Persistentes Amanzadas (APT)

Las APT son por excelencia la niña bonita del ciberespionaje. Se trata de ataques dirigidos en los que se pretende la infección de la máquina de forma sigilosa para poder permanecer en ella el mayor periodo de tiempo posible. Su objetivo es lograr sustraer información de un sistema informático y enviarla al sistema de Comando & Control.

Se trata de malware capaz de permanecer dormido (inactivo) durante un largo periodo de tiempo si descubre que está en riesgo de poder ser detectado.

Los métodos de comunicación con el exterior del sistema de información infectado es muy similar al utilizado por las bootnets. Pero debido a su sofisticación y su diseño específico, está preparado para evadir las técnicas de detección de dicho sistema.

Capítulo 5

Preparación de los datos para ser procesados por la red neuronal

Índice

5.1. Datos iniciales 39
5.2. Métodos de obtención de información sobre los nombres de dominio DNS y normalización de los datos 40
5.2.1. Cotejo del FQDN del dominio en diferentes BlackLists 40
5.2.2. Obtención de la resolución del FQDN y cotejo de la misma en diferentes BlackLists de direcciones IP infectadas 40
5.2.3. Obtención de los servidores de nombres asociados al FQDN y cotejo de la misma en diferentes BlackLists de servidores de nombres infectados . 41
5.2.4. Obtención de la información de registro del FQDN y cotejo de la misma en la base de datos de información disponible sobre los FQDN infectados 42
5.2.5. Detección de ataques typosquatting 42
5.2.6. Detección de FQDN creados mediante un DGA 43
5.2.7. Comprobación de que no se trata de un NXDOMAIN 43
5.2.8. Comprobación de que no pertenece a una red Fast-Flux 44
5.2.9. Detección de contenido malicioso 44

5.1. Datos iniciales

La única información que disponemos a priori de los nombres de dominio DNS es su FQDN. Por ello, necesitamos obtener información sobre ellos que nos permita detectar amenazas como las descritas en la sección anterior.

5.2. Métodos de obtención de información sobre los nombres de dominio DNS y normalización de los datos

Para la obtención de información sobre los nombres de dominio DNS, desarrollé una serie de programas modulares en Python.

5.2.1. Cotejo del FQDN del dominio en diferentes BlackLists

Se comprueba si el FQDN se encuentra en una base de datos con diferentes BlackLists que contienen nombres de dominio DNS maliciosos.

Resultado de la función

La función devuelve un número real entre 0 y 1 que representa el grado de coincidencia:

$$\frac{n}{N} \quad si \quad \frac{n}{N} > \frac{1}{N} \tag{5.1}$$

$$0 \quad si \quad \frac{n}{N} \neq \frac{1}{N} \tag{5.2}$$

siendo N el número de subdominios del FQDN y n el número de subdominios que coinciden de forma contigua, de mayor a menor nivel.

Detalles de implementación

La base de datos, es un diccionario que se almacena en archivos JSON. Para ello se utilizaron las librerías de JSON para Python.

5.2.2. Obtención de la resolución del FQDN y cotejo de la misma en diferentes BlackLists de direcciones IP infectadas

Se resuelve el FQDN para conseguir las direcciones IP asociadas al dominio. Se comprueba si alguna de las direcciones IP se encuentra en la base de datos.

Resultado de la función

La función devuelve un número real entre 0 y 1 que representa el grado de coincidencia:

$$\frac{n}{N} \tag{5.3}$$

siendo N el número de direcciones IP cotejadas y n el número de direcciones IP coincidentes.

Detalles de implementación

La resolución del dominio se hace mediante el uso de las librerías: pythonwhois , dns.resolver y mediante una consulta a BFK. EDV-Consulting ya que esta última utiliza *Passive DNS* para resolverlo.

La base de datos, es un diccionario que se almacena en archivos JSON. Para ello se utilizaron las librerías de JSON para Python.

5.2.3. Obtención de los servidores de nombres asociados al FQDN y cotejo de la misma en diferentes BlackLists de servidores de nombres infectados

Se realiza una consulta 'NS' sobre el FQDN para la obtención de las direcciones de los servidores de nombres. Se comprueba si alguna de los servidores de nombres se encuentra en la base de datos.

Resultado de la función

La función devuelve un número real entre 0 y 1 que representa el grado de coincidencia:

$$\frac{n}{N} \tag{5.4}$$

siendo N el número de direcciones IP cotejadas y n el número de direcciones IP coincidentes.

Detalles de implementación

La resolución del dominio se hace mediante el uso de las librerías: pythonwhois , dns.resolver y mediante una consulta a BFK. EDV-Consulting ya que esta última utiliza *Passive DNS* para resolverlo.

La base de datos, es un diccionario que se almacena en archivos JSON. Para ello se utilizaron las librerías de JSON para Python.

5.2.4. Obtención de la información de registro del FQDN y cotejo de la misma en la base de datos de información disponible sobre los FQDN infectados

Los FQDN de reciente creación tienen más posibilidades de haber sido creados explícitamente para la realización de una oleada de ataques. Al igual, que los que han sufrido modificaciones en los datos de registro recientemente. [11]

En este método se obtiene la fecha de creación del dominio, la fecha de su última actualización, e información sobre el registrante del FQDN (nombre, correo electrónico, fax, número de teléfono y organización)

Resultado de la función

Se devuelven tres valores:

- Indicador de reciente creación. Si el FQDN ha sido creado hace más de dos meses devuelve 0, si ha sido durante los últimos dos meses se devuelve 0.5, y en el último mes 1.
- Indicador de actualizaciones recientes. Si el FQDN ha sido actualizado hace más de dos meses devuelve 0, si ha sido durante los últimos dos meses se devuelve 0.5, y en el último mes 1.
- Indicador de coincidencia de información de registro.

  $$\frac{n}{N} \tag{5.5}$$

  siendo N el número de campos obtenidos sobre la información de registro y n el número de campos coincidentes.

Detalles de implementación

La resolución del dominio se hace mediante el uso de la librería: pythonwhois.

La base de datos, es un diccionario que se almacena en archivos JSON. Para ello se utilizaron las librerías de JSON para Python.

5.2.5. Detección de ataques typosquatting

Se coteja si el FQDN se asemeja a algún dominio de los catalogados por Alexa como más visitados en España.

Resultado de la función

Para cada uno del TOP Sites de España, se calcula: $\frac{L-distance(TopSiteAlexa,FQDN)}{L}$ siendo $distance(TopSiteAlexa, FQDN)$ la distancia de Levenshtein entre el FQDN de Alexa y el FQDN a analizar y L la longitud de cadena mayor de ambos FQDN.

Detalles de implementación

La base de datos, es un diccionario que se almacena en archivos JSON. Para ello se utilizaron las librerías de JSON para Python.

5.2.6. Detección de FQDN creados mediante un DGA

Se pretende calcular el nivel de entropía o aleatoriedad del FQDN.

Resultado de la función

Relación entre el número de vocales que contiene el FQDN con respecto al número de consonantes o símbolos.

5.2.7. Comprobación de que no se trata de un NXDOMAIN

Según [5], obtener una petición fallida al resolver un FQDN es un indicio de que este dicho nombre de dominio DNS esté infectado o pertenezca al sistema de bootnet.

Resultado de la función

Si la petición se resuelve sin problemas se devuelve 0, sino 1.

Detalles de implementación

La resolución del dominio se hace mediante el uso de la librería: dns.resolver.

5.2.8. Comprobación de que no pertenece a una red Fast-Flux

Se utiliza una herramienta en Python implementada por Alejandro Nolla disponible en GitHub en el siguiente enlace: `https://github.com/z0mbiehunt3r/pffdetect`. Dicha herramienta está basada en el artículo de investigación [8].

Resultado de la función

Se devuelve la puntuación obtenida tras pasar el nombre de dominio DNS por el programa de Alejandro Nolla.

5.2.9. Detección de contenido malicioso

Utilización de la API de VirusTotal para la detección de contenido malicioso.

Resultado de la función

La función devuelve un número real entre 0 y 1 que representa el porcentaje de test de VirusTotal en los que se ha dado positivo.

$$\frac{n}{N} \tag{5.6}$$

siendo N el número total de pruebas y n el número de pruebas pasados como positivo.

Capítulo 6

Conclusiones

A lo largo de mi estancia en prácticas, pude mejorar mis aptitudes como programadora en Python. Además, aprendí a planificar un proyecto desde cero e intentar llevarlo a cabo, dándome cuenta de los problemas e inconvenientes que acaban surgiendo durante el desarrollo del mismo.

La verdad es que la experiencia en S2 Grupo fue muy amena puesto que había un buen ambiente de trabajo en la oficina y a pesar de trabajar sola, mis compañeros siempre estaban predispuestos a ayudarme en lo que necesitara.

Además, en S2 Grupo me dieron la oportunidad de colaborar en su blog Security Art Work, realizando una entrada sobre Detección de dominios DNS maliciosos utilizados por APT.

Asimismo, gracias a este TFG he podido estudiar a fondo las redes neuronales, un método de clasificación que me parece muy interesante por capacidad de aprendizaje.

Para concluir, me gustaría agradecer a mi tutor, Pablo Gregori Huerta, por su esfuerzo a pesar de realizar el TFG a distancia; a mis supervisores, Antonio Villalón y Antonio Sanz, por darme la oportunidad de realizar este TFG junto a ellos en su empresa; y a mi familia y amigos por todo el apoyo durante este último año.

Glosario

A

aprendizaje supervisado

Técnica para deducir una función a partir de datos de entrenamiento. Los datos de entrenamiento son una tupla en la que una componente son los datos de entrada y otra, los resultados deseados. En los problemas de regresión, la salida del método es un valor numérico y en los de clasificación, la etiqueta de una clase. Su objetivo es crear una función capaz de predecir la salida de un objeto, después de a analizar lo que ocurre con los datos de entrenamiento, siendo capaz de generalizar para poder predecir del mismo modo en situaciones no vistas previamente. 24

APT

Amenaza Persistente Avanzada. Conjunto de procesos de piratería informática sigilosos y continuos, comúnmente controlados por humanos dirigidos a una entidad específica. Utilizan sofisticadas técnicas para la extracción de información. Además, estos procesos requieren de una monitorización para llevar a cabo la extracción de datos. 2, 10, 11, 12, 13, 38

B

bootnet

Grupo de ordenadores conectados a internet que, de forma involuntaria, una vez infectados por un malware, pueden ser controlados de forma remota para realizar acciones sin la autorización y el conocimiento del propietario. 37, 38

C

ciberespionaje

Espionaje por medio de internet, redes y/o utilizando cualquier tipo de software malicioso. 10, 38

criptología

Disciplina científica que se dedica al estudio de mensajes crifrados. 11

D

DGA

Algoritmo de Generación de Dominios. Algoritmos implementados para la generación de dominios, que suelen ser aleatorios, comúnmente utilizados por malwares, para dificultar la detección de la comunicación entre la botnet y el sistema C2. 37

DNS

Sistema de nombres de dominio. Sistema de nomenclatura jerárquica para cualquier recurso conectado a Internet o a una red privada. Asocia información variada a nombres de dominio y además es capaz de traducirlos a direcciones IP físicas. 2, 11, 12, 13, 19, 20, 21, 49

E

econometría

Parte de la ciencia económica que aplica las técnicas matemáticas y estadísticas a las teorías económicas para su verificación y para la solución de los problemas económicos mediante modelos. 25

épocas de entrenamiento

Barrido a través de todo el conjunto de datos de entrenamiento. 34

F

Fast-Flux

Técnica de los sistemas DNS usada por botnets para esconder el robo de información de los usuarios (phishing) y los lugares de entrega de información del malware, tras una red en constante cambio en la que equipos comprometidos actúan como proxies. Para ello, se asocian numerosas direcciones IP a un único nombre de dominio completo, donde las direcciones IP cambian con una frecuencia extremadamente alta mediante cambios de registro DNS. 5, 37, 39, 43

FQDN

Full Qualified Domain Name. Nombre de dominio que especifica la localización exacta dentro del DNS (Domain Name System). Especifica todos los niveles de dominio, de este modo no da lugar a ambigüedades, solo puede ser interpretado de un modo. 39

función spline

Función numérica definidas a trozos por funciones polinómicas, y que posee un alto grado de suavidad en los puntos dónde las ramas polinomiales conectan. 51

función hash

Función determinista (un mensaje siempre tiene el mismo valor hash) y de bajo coste, capaz de calcular la imagen de un conjunto de datos, y que su resultado sea una cadena finita desde la cual sea imposible obtener su anti-imagen. 37

H

HTTP

Hypertext Transfer Protocol. Protocolo de aplicación usado en cada transacción de la World Wide Web. 12

I

IPC

Comunicación entre procesos. Función básica de un sistema operativo en el que existe una comunicación entre procesos por medio de memoria compartida. 12

M

malware

Software malicioso. 11, 13, 19, 37, 38

método de retropropagación (optimización)

El método calcula el gradiente de una función de pérdida con respecto a todos los pesos de la red. El gradiente se alimenta del método de optimización que a su vez lo utiliza para actualizar los pesos, en un intento de minimizar la función de pérdida. Requiere que la función de activación utilizad sea diferenciable.. 29

método de quasi-Newton

Métodos utilizado para encontrar ceros, o bien, máximos y mínimos locales de funciones. Es una alternativa al método de Newton, ya que se puede utilizar si el jacobiano o el hessiano no está disponible o es demasiado caro calcularlo en cada iteración. 25

modelo de regresión lineal

Método estadístico que modela la relación entre una variable dependiente Y, las variables independientes X y el termino aleatorio ε.

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \varepsilon$$

donde β_0 es el termino constante, β_i con $(i > 0)$ son los parametros de cada variable independiente X_i, y p es el número de parámetros independientes que se tienen en cuenta en la regresión. 25

modelo de índice único

Single Index Model (SIM). Sencillo modelo de precios de activos para medir tanto el riesgo como el retorno de una acción, comúnmente usado en la industria financiera. 25

MySQL

Sistema de gestión de bases de datos relacionales. 19

N

nombres de dominio DNS maliciosos

Nombres de dominio DNS utilizados de forma ilegítima por algún tipo de malware. 11

P

Python

Lenguaje de programación interpretado. `https://www.python.org/`. 2, 12, 19, 20, 21, 22, 39

R

ransomware

Malware que restringe el acceso a un sistema informático o parte de él y pide a cambio de un rescate para eliminar la restricción. 10

red neuronal feed-forward

Red neuronal feedforward en la que las conexiones entre las unidades no forman un ciclo dirigido. 7, 26

regresión por mínimos cuadrados

Técnica de regresión paramétrica que basada en el método de mínimos cuadrados. 25

retropropagación

Algoritmo de aprendizaje supervisado para entrenar redes neuronales. Emplea un ciclo de dos fases:

- Una vez que se ha aplicado un patrón a la entrada de la red como estímulo, este se propaga desde la primera capa a través de las capas superiores de la red, hasta generar una salida. La señal de salida se compara con la salida deseada y se calcula una señal de error para cada una de las salidas.
- Las salidas de error se propagan hacia atrás, partiendo de la capa de salida, hacia todas las neuronas de la capa oculta que contribuyen directamente a la salida. Sin embargo las neuronas de la capa oculta solo reciben una fracción de la señal total del error, basándose aproximadamente en la contribución relativa que haya aportado cada neurona a la salida original.

. 26

S

scatterplot smoother

Método para ajustar una función a través de los puntos de un diagrama de dispersión, que representan la relación entre las variables. 25

servidor de exfiltración

Servidor utilizado para la exfiltración (extracción no autorizada) de datos de un sistema informático. 12

sinapsis

Relación funcional de contacto entre las terminaciones de las células nerviosas. 28

sistema C&C

Sistema Mando & Control. Herramienta de control de la botnet. Se encarga de gestionar la información enviada por las máquinas infectadas, y de enviar instrucciones a éstas. 12, 37

smoothing spline

Método de ajuste de una curva suave a un conjunto de datos con ruido utilizando una función spline. 25

SMTP

Protocolo de Transferencia de Correo Simple. Estandar de internet para la transmisión de correo electrónico. 12

sobreajuste

Overfitting. Efecto de sobreentrenar un algoritmo de aprendizaje con ciertos datos para los que se conoce el resultado deseado. Cuando esto ocurre, el algoritmo de puede quedar ajustado a unas características muy específicas de los datos de entrenamiento que no tienen relación causal con la función objetivo. 25

suavizado

Smoothing. En estadística, suavizar un conjunto de datos es crear una función de aproximación que intente capturar patrones significantes en los datos, dejando de lado el ruido. 24, 25

Bibliografía

[1] APWG. Global phishing survey: Domain name use and trends in 1h2010. http://docs.apwg.org/reports/APWG_GlobalPhishingSurvey_1H2010.pdf. [Consulta: 1 de Noviembre de 2015].

[2] CCN-Cert del Centro Criptológico Nacional (CCN). Carmen: Herramienta para la detección de ataques avanzados/apt. https://www.ccn-cert.cni.es/herramientas-de-ciberseguridad/carmen.html. [Consulta: 16 de Noviembre de 2015].

[3] CCN-Cert del Centro Criptológico Nacional (CCN). Resumen ejecutivo del informe de ciberamenazas 2014 y tendencias 2015 (ccn-cert ia-9/15). https://www.ccn-cert.cni.es/informes/informes-ccn-cert-publicos/795-ccn-cert-resumen-ia-09-15-ciberamenazas-2014-tendencias-2015/file.html. [Consulta: 16 de Noviembre de 2015].

[4] DNSSEC. El sistema de nombres de dominio (dns). http://www.dns-sec.es/index.php/sistema-de-nombres-de-dominio-dns/el-sistema-de-nombres-de-dominio-dns/. [Consulta: 22 de Octubre de 2015].

[5] David Cantón (INCIBE). https://www.incibe.es/blogs/post/Seguridad/BlogSeguridad/Articulo_y_comentarios/Botnets_DNS. Publicado el 20 de Enero de 2015, [Consulta: 1 de Noviembre de 2015].

[6] Pete Chapman (NCR), Julian Clinton (SPSS), Randy Kerber (NCR), Thomas Khabaza (SPSS), Thomas Reinartz (DaimlerChrysler), Colin Shearer (SPSS) y Rüdiger Wirth (DaimlerChrysler). Crisp-dm 1.0. step-by-step data mining guide. https://the-modeling-agency.com/crisp-dm.pdf. [Consulta: 16 de Noviembre de 2015].

[7] S2 Grupo. Dossier coorporativo de S2 Grupo. http://s2grupo.es/pdf/dossier_es.pdf. [Consulta: 16 de Noviembre de 2015].

[8] Thorsten Holz, Christian Gorecki, Konrad Rieck, and Felix C. Freiling. Measuring and detecting fast-flux service networks. http://www.internetsociety.org/doc/measuring-and-detecting-fast-flux-service-networks-paper. Proc. of 15th Network and Distributed System Security Symposium (NDSS), 1998, [Consulta: 22 de Octubre de 2015].

[9] Trevor Hastie,Robert Tibshirani y Jerome Friedman. The elements of statistical learning: Data mining, inference, and prediction. http://statweb.stanford.edu/~tibs/ElemStatLearn/. Segunda edición, Febrero 2009, [Consulta: 22 de Octubre de 2015].

[10] Universitat Jaume I. Documentación de la asignatura de seguridad informática en el grado en ingeniería informática y en el de matemática computacional. http://sim.nisu.org/ ó http://siv.nisu.org/. [Consulta: 16 de Noviembre de 2015].

[11] Wei Xu, Kyle Sanders, Yanxin Zhang. Virus bulletin conference september 2014.we know it before you do: Predicting malicious domains. https://www.virusbtn.com/pdf/conference/vb2014/VB2014-XuZhangSanders.pdf. Septiembre de 2014, [Consulta: 1 de Noviembre de 2015].

[12] Wikipedia. Cross industry standard process for data mining. https://en.wikipedia.org/wiki/Cross_Industry_Standard_Process_for_Data_Mining. [Consulta: 16 de Noviembre de 2015].