Universidad Politécnica
de Madrid

Escuela Técnica Superior de Ingenieros Informáticos

Máster Universitario en Inteligencia Artificial

Trabajo Fin de Máster

Autocodificador Evolutivo de Red Bayesiana para Detección de Anomalías Aplicado a Ciberseguridad

Autor: Jorge Casajús-Setién
Tutores: Concha Bielza y Pedro Larrañaga

Madrid, Septiembre de 2022

Este Trabajo Fin de Máster se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de Máster
Máster Universitario en Inteligencia Artificial

Título: Autocodificador Evolutivo de Red Bayesiana para Detección de Anomalías Aplicado a Ciberseguridad

Septiembre de 2022

Autor: Jorge Casajús-Setién
Tutores: Concha Bielza
Pedro Larrañaga
Departamento de Inteligencia Artificial
ETSI Informáticos
Universidad Politécnica de Madrid

Agradecimientos

El estudio que se expone en las páginas que suceden a esta breve introducción, y que es mi trabajo final para el Máster Universitario en Inteligencia Artifical de la UPM, se ha realizado a lo largo de más de medio año, por lo que hay una larga lista de personas que, de un modo u otro, me han apoyado durante el camino.

Este trabajo forma parte del proyecto SLISE: *network SLIcing SEcurity for next generation communications* (2021-2024), que ha sido financiado por el Centro para el Desarrollo Tecnológico Industrial del Ministerio de Industria y en el que ha participado el *Computational Intelligence Group* de la UPM en colaboración con *Titanium Industrial Security* S.L. Por ello, quería agradecer en primer lugar a Borja, a Dani y a todo el equipo de *Titanium* Madrid el espacio tan productivo de trabajo que me han proporcionado estos últimos meses, lo calurosamente que me han acogido, los recursos y conocimiento que han puesto a mi servicio y, como no, los descansos para el café.

También he recibido un enorme apoyo del propio *Computational Intelligence Group*, en especial de mis tutores Concha y Pedro, junto a los que he aprendido casi todo lo que sé sobre Inteligencia Artificial y que, semana tras semana, se han asegurado de alinear mi esfuerzo en la dirección correcta para que este trabajo llegara a buen puerto. Igualmente, les debo un agradecimiento especial a Paula y a Carlos por haber formado equipo conmigo cada día y haber sido lo más parecido a unos "compañeros de clase" que he tenido este semestre.

Finalmente, tengo que darle las gracias a Marisa por haber estado a mi lado desde que la llamé tras salir de mi primera reunión con Concha y Pedro hasta ahora mismo, que siendo las cuatro de la tarde, está esperando a que acabe de escribir los agradecimientos y mande la primera versión completa del trabajo para comer juntos.

Resumen

El presente Trabajo de Fin de Máster persigue el objetivo de diseñar e implementar un sistema novedoso de detección de anomalías con redes Bayesianas que incorpore un proceso de aprendizaje generativo-adversarial basado en la técnica de entrenamiento para redes neuronales de igual nombre.

Los problemas de detección de anomalías constituyen una rama de estudio fundamental en *machine learning*. La forma más frecuente que toma este problema en escenarios reales es la de la detección semisupervisada de instancias irregulares a partir de datos etiquetados positivamente, es decir, sin anomalías. Esto requiere la construcción de un modelo capaz de capturar el comportamiento normal de los datos de un conjunto de entrenamiento compuesto exclusivamente por datos categorizados como normales. Así, nuevas instancias nunca antes vistas por el modelo pueden ser comparadas con la noción de normalidad inducida del conjunto de datos de entrenamiento para ser etiquetadas – o no – como anómalas.

En particular, el detector de anomalías desarrollado constituye una nueva forma de resolver un problema real de ciberseguridad: la detección de intrusiones en una red de computadores a través del análisis del tráfico que la atraviesa. Este enfoque al problema, si bien supone la asunción (no siempre correcta) de que una anomalía en el tráfico de red implica una amenaza de intrusión, permite la detección de ciberataques nuevos, pues no depende del reconocimiento de un patrón de ataque previamente observado.

En la tarea de modelización de un patrón de comportamiento, los modelos generativos y, concretamente, las redes generativo-adversariales (GANs), han demostrado tener un excelente rendimiento. Por este motivo, en la última década se han propuesto numerosos algoritmos de detección de anomalías basados en GANs, muchos de ellos empleando además una estructura de autocodificador para la detección de irregularidades en los datos. Lo más frecuente es que este tipo de algoritmos empleen redes neuronales profundas como agente inteligente, lo que conlleva ciertas desventajas. En este trabajo se presenta una metodología diferente para la detección de anomalías semisupervisada basada en el uso de redes Bayesianas cuyo proceso de aprendizaje se lleva a cabo a través de un entrenamiento generativo-adversarial siguiendo una estrategia evolutiva, con el objetivo principal de paliar la falta de interpretabilidad intrínseca de las redes neuronales profundas.

El uso de redes Bayesianas obedece al objetivo de mejorar la capacidad del modelo detector de anomalías de ser interpretado por un operador o usuario. Este punto es clave en ámbitos en los que se requiere comprender por qué un sistema inteligente ha llegado a una conclusión determinada, especialmente cuando ésta es crítica para

la toma de una decisión que puede tener un efecto perjudicial sobre una persona. Este dilema ético ha provocado que en áreas como la medicina o la ciberseguridad no puedan ser utilizados sistemas "de caja negra", como las redes neuronales profundas, que por su estructura en capas se hallan en el extremo menos interpretable de los modelos de inteligencia artifical. No obstante, son las arquitecturas más complejas y, por tanto, menos interpretables, las que frecuentemente proveen los mejores resultados, lo que abre una línea de investigación para tratar de encontrar el equilibrio ideal entre interpretabilidad y rendimiento en cada aplicación práctica.

El modelo presentado ha sido evaluado en un problema real de detección de anomalías en ciberseguridad, obteniendo resultados parejos a los de otros algoritmos de detección de anomalías basados en GANs y en redes Bayesianas puras, pero estableciendo una alternativa híbrida a ambos en términos de interpretabilidad y rendimiento.

Abstract

This master's thesis pursues the main goal of designing and implementing a novel anomaly detection system based on Bayesian networks, including a generative-adversarial learning algorithm, in the sense that it mimics the way generative-adversarial neural networks are trained.

Anomaly detection is nowadays a fundamental problem in machine learning. Semi-supervised detection of outliers from only positively labeled instances (without anomalies), which is among the most frequent forms of this problem in real scenarios, requires for a model to capture the normal behaviour of data from a training set exclusively comprised of normal-labelled instances, so new unseen samples can be afterwards compared to the induced notion of normality to be flagged -or not- as anomalous.

Particularly, the developed anomaly detection model represents a new way of solving a real cybersecurity problem: intrusion detection in a computer network by traffic data analysis. This perspective of the problem, though based on the (not always correct) assumption that an anomaly in network traffic implies an intrusion in the net, allows for the detection of previously unseen cyberattacks, since it is not dependent on recognising the signature of an attack.

In modelling a certain pattern of behaviour, generative models such as generative-adversarial networks (GANs) have proved to have great performance. Thus, numerous anomaly detection algorithms with GANs at their core have been proposed, most of them relying on an autoencoder for the AD task. Most often, GAN-based algorithms are powered by deep neural networks, with the downsides this entails. In the present work, a novel approach to semi-supervised AD with Bayesian networks using generative-adversarial training and an evolutive strategy is proposed, which aims to palliate the intrinsic lack of interpretability of deep neural networks.

The usage of Bayesian networks responds to the need of improving the model's inherent capability to be understood by an operator or a user. The interpretability capacity of a model remains a key aspect in fields where comprehension of the reasoning underlying a certain result of an intelligent system is required, specially in cases when decisions stemming from these results can have a negative impact in people's lives. This ethical dilemma has pushed areas such as medicine or cybersecurity to refuse black-box intelligent systems, as deep neural networks, which stand at the less-interpretable side of the spectrum of models due to their layered symbolic structure. However, it is the more complex architectures (and thus, the less-interpretable ones) that tend to offer the best performance, opening a way for a new research line aiming to find the ideal balance between interpretability and performance for each practical

scenario.

The proposed model will be tested on a real-world anomaly detection problem in cybersecurity, performing on par with other AD algorithms based on GANs and Bayesian networks, therefore providing a hybrid alternative to both in terms of interpretability and performance.

Tabla de contenidos

Capítulo 1

Introducción

La detección de anomalías previamente desconocidas, definiéndolas como patrones dentro de un conjunto de datos que no se comportan de acuerdo a lo que cabe esperar observando el resto de instancias del conjunto, constituye un problema central en numerosas áreas de investigación (Chandola *et al.*, 2009) y es, por tanto, una de las tareas fundamentales dentro de la disciplina del aprendizaje automático o *machine learning*.

1.1. Detección de anomalías: un problema de clasificación

El problema de detección de anomalías dado un conjunto de instancias puede formularse como una tarea de clasificación cuyo objetivo es asignar correctamente una categoría binaria a una instancia dada: 0 si es considerada normal y 1 si es una anomalía. En la mayor parte de las aplicaciones reales, la detección de anomalías, por infrecuentes, requiere que se trabaje con datos no etiquetados que se asumen de categoría normal (no anómalos), lo que reduce el problema a la categoría de aprendizaje no supervisado.

La situación más frecuente es aquella en la que las anomalías se deben señalar mediante comparación de las muestras con una noción de normalidad previamente inducida de un conjunto de instancias de entrenamiento etiquetados como normales, sin necesidad de observar anomalías durante la fase de aprendizaje (Madhuri y Rani, 2018). Este tipo de problema, que supone el cuadro en el que se desarrolla este trabajo, se enmarca dentro de las tareas de aprendizaje semisupervisado, concretamente en las de aprendizaje únicamente a partir de datos etiquetados como positivos, que en el caso en cuestión se corresponden con las instancias normales.

Formalmente, sea $\mathbf{X}$ el espacio muestral y dado un conjunto $\mathcal{S}$ conteniendo una serie de instancias $\mathbf{x} \in \mathbf{X}$ que se consideran representativas del comportamiento normal y, por tanto, están etiquetadas como normales, el problema de detección de anomalías mediante aprendizaje semisupervisado se puede definir como la optimización de los parámetros $\boldsymbol{\theta}$ de un modelo que aprenda la distribución probabilística de los datos normales de $\mathcal{S}$, denotada por $p_{\mathbf{X}}$. El modelo debe ser capaz de extraer, para una nueva instancia dada, una puntuación de anomalía $A(\mathbf{x})$, de forma que datos anormales obtengan puntuaciones más altas y tal que, estableciendo un umbral conveniente, ϕ, se pueda clasificar dicha nueva instancia como normal si su puntuación de anomalía

no sobrepasa el umbral ($A(\mathbf{x}) \leq \phi$) o, de lo contrario, como anómala ($A(\mathbf{x}) > \phi$).

1.2. Aplicaciones de la detección de anomalías

En escenarios reales, los motivos que causan la aparición de una anomalía en un conjunto de datos son de diversa naturaleza, pero a menudo indican que un evento, que puede comprender una instancia de datos o una serie de éstas, es merecedor de ser analizado en profundidad. Los sistemas de detección de anomalías han sido aplicados con éxito en un amplio abanico de situaciones, siendo algunos ejemplos representativos los siguientes (Chandola *et al.*, 2009):

- Ciertas organizaciones comerciales cuya actividad se ve perjudicada por el uso de sus recursos de forma abusiva o no autorizada (compañías bancarias, telefónicas y aseguradoras, entre otras) emplean detección de anomalías para descubrir fraudes, o usos maliciosos de sus servicios, y prevenir así pérdidas económicas.
- En medicina, a menudo se puede establecer una correlación entre la presencia de anomalías en el historial de un paciente y la existencia de una condición anormal en su salud. También puede aplicarse a la detección de brotes de enfermedades epidémicas estudiando, a un tiempo dado, los historiales médicos de una población.
- En el entorno industrial es una práctica frecuente monitorizar mediante sensores los parámetros de funcionamiento de todos los componentes de una máquina. Esto permite efectuar un análisis de anomalías sobre los datos recopilados, lo que puede poner de manifiesto defectos producidos por el uso y desgaste de la maquinaria, habilitando su reparación inmediata para evitar que se propaguen o se detenga la producción.

De forma general, la detección de anomalías puede ser utilizada en cualquier ámbito que se requiera o se beneficie de la distinción de casos especiales que difieren de una noción bien definida de normalidad, bien porque representen un fallo o porque sean de interés especial.

1.2.1. Detección de anomalías en ciberseguridad

En el campo de la ciberseguridad se puede comprobar que la presencia de ciberataques está relacionada con la existencia de anomalías en dos tipos de datos:

- En las trazas de llamadas al sistema operativo, donde pueden aparecer anomalías en forma de secuencias anormales de trazas que codifican programas o comportamientos maliciosos. Se dice que este tipo de detección de intrusiones se realiza a nivel de *host*, y requiere que se tenga en cuenta que la naturaleza de las anomalías es secuencial, pues es la coincidencia temporal de una serie de instrucciones tomadas de un alfabeto finito lo que denota la anomalía, pero nunca la instrucción puntual.
- En el tráfico de red, estudiando los datos que caracterizan el envío y recepción de paquetes IP en una red de ordenadores, aunque también se puede profundizar en el análisis desgranando los paquetes en sus estructuras internas. Los

sistemas que utilizan este mecanismo se conocen como detectores de intrusión a nivel de red.

En particular, el detector de anomalías introducido en este trabajo ha sido desarrollado con el objetivo de construir un sistema de detección de intrusiones a nivel de red en colaboración con Titanium Industrial Security S.L.

1.3. Algoritmos generativos y GANs

La última pieza de introducción necesaria para cimentar el presente trabajo corresponde a los algoritmos generativos, de los que toma inspiración el modelo detector de anomalías propuesto e implementado.

La meta de un algoritmo generativo es la de modelizar una distribución compleja de datos para generar nuevas instancias sintéticas conforme a la distribución aprendida. Esto sugiere que puedan ser utilizados en tareas de detección semisupervisada de anomalías (Mattia *et al.*, 2019), primero aprendiendo la distribución del conjunto de datos etiquetados como normales, $\mathcal{S}$, y posteriormente comprobando si cada instancia de prueba cumple o no con el comportamiento esperado.

La propuesta de las redes generativo-adversariales, del inglés *Generative Adversarial Networks* o GANs (Goodfellow *et al.*, 2014a) intrudujo un novedoso procedimiento de entrenamiento para redes neuronales basado en la competencia entre dos agentes inteligentes: un generador y un discriminador, originalmente materializados como dos redes de neuronas artificiales.

En su trabajo, Goodfellow et al. tratan de explicar la estructura del sistema adversarial de dos partes en términos del modelo generativo funcionando como un estafador cuyo objetivo es falsificar billetes y el modelo discriminador como un policía que procura distinguir los billetes falsos de otros legítimos. El entrenamiento de las redes neuronales generadora y discriminadora se produce a través de este juego competitivo de suma cero, es decir, en el que la ganancia de una de las dos partes equivale a una pérdida para la otra. Así, se trata de optimizar a la vez el comportamiento de los modelos policía y falsificador, de modo que cuando uno de éstos experimenta una mejora en su "oficio", fuerza al otro a mejorar de acuerdo a la métrica que se haya establecido para cuantificar el éxito en sus relativas tareas.

Matemáticamente, el fin de la red generadora es el de representar, a través de sus parámetros $\boldsymbol{\theta}_G$, una aplicación $G(\mathbf{z};\boldsymbol{\theta}_G)$ que va de un espacio m-dimensional de variables de entrada $\mathbf{Z}$ a un espacio de salida n-dimensional $\mathbf{X}$, con la particularidad de que el espacio de entrada siempre es de dimensionalidad menor que el de salida (es decir, $m < n$). El espacio $\mathbf{Z}$ se denomina *espacio latente*, y contiene los vectores de ruido $\mathbf{z} = (z_1, \ldots, z_m)$ que constituyen las semillas aleatorias a partir de las cuales se producirá la generación de muestras artificiales. Por su parte, el espacio $\mathbf{X}$ o *espacio de datos* contiene las instancias normales, de la forma $\mathbf{x} = (x_1, \ldots, x_n)$ y caracterizadas por n atributos. En la Figura 1.1 viene representado un esquema de la aplicación que codifica G entre los espacios latente y de datos.

De otro lado, la red discriminadora trata de codificar en sus parámetros $\boldsymbol{\theta}_D$ la función $D(\mathbf{x};\boldsymbol{\theta}_D)$, que toma una instancia $\mathbf{x}$ del espacio de datos como entrada y devuelve un valor escalar cuya lectura significa la probabilidad de que tal $\mathbf{x}$ sea una instancia real frente a la posibilidad de que haya sido generada por la aplicación de $G(\mathbf{z};\boldsymbol{\theta}_G)$ sobre

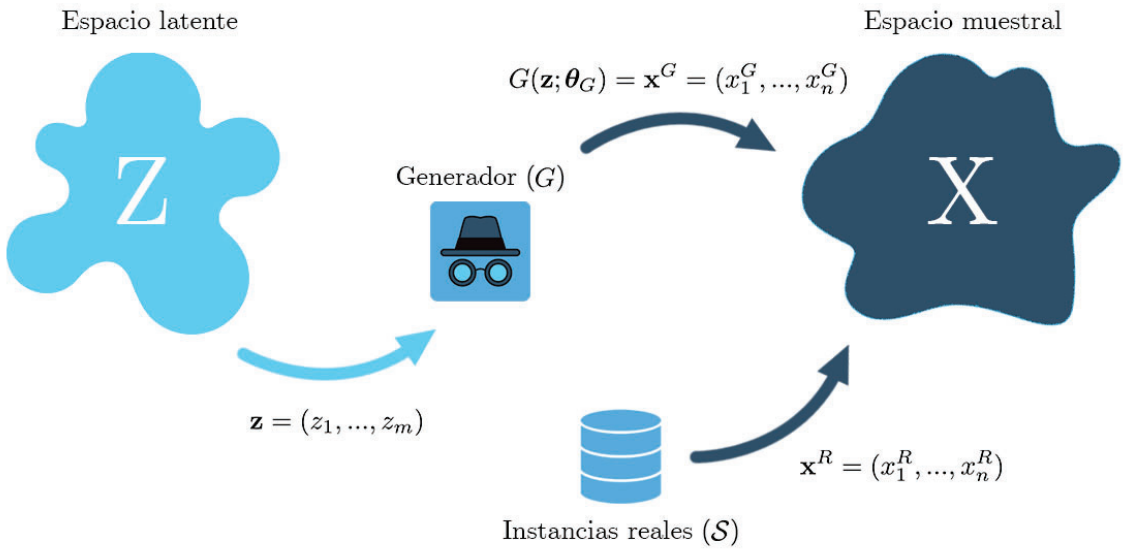


Figura 1.1: Representación esquemática de la aplicación codificada en los parámetros $\boldsymbol{\theta}_G$ del generador, G. Nótese que al espacio de datos no sólo pertenecen las instancias obtenidas de la aplicación de G sobre un vector del espacio latente $\mathbf{z} \in \mathbf{Z}$, sino también las instancias reales, que constituyen el conjunto de entrenamiento $\mathcal{S}$.

un vector de ruido aleatorio $\mathbf{z}$ actuando como semilla. El proceso de discriminación de instancias se representa en la Figura 1.2.

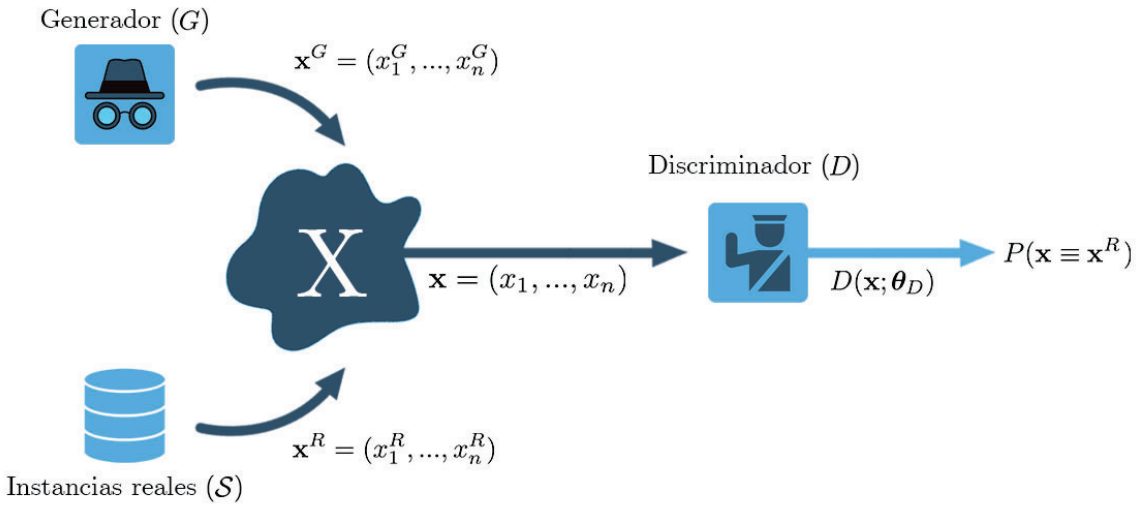


Figura 1.2: Esquema de la aplicación de discriminación que codifica D en sus parámetros $\boldsymbol{\theta}_D$. Las instancias del espacio de datos que se evalúan con el discriminador pueden provenir tanto de la aplicación de G sobre un vector del espacio latente $\mathbf{z} \in \mathbf{Z}$ como del conjunto de entrenamiento $\mathcal{S}$.

En el proceso de minimización-maximización por suma cero de acuerdo al cual se

optimizan los parámetros de las redes neuronales G y D, la red D aspira a maximizar la probabilidad de asignar correctamente a una instancia dada la etiqueta "real" o "falsa", mientras que G trata de minimizar la probabilidad de que D reconozca una instancia generada como tal. Matemáticamente, se expresa mediante:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\mathbf{x}}[\log D(\mathbf{x};\boldsymbol{\theta}_D)] + \mathbb{E}_{\mathbf{z}}[\log(1 - D(G(\mathbf{z};\boldsymbol{\theta}_G))].$$

El algoritmo de entrenamiento para redes generativo-adversariales presentado en Goodfellow *et al.* (2014a) viene detallado en el Algoritmo 1. En el artículo original, Goodfellow et al. emplean un parámetro adicional k para el ratio de entrenamiento, o número de veces que se actualizan los parámetros del generador antes de actualizar los del discriminador, pero en este caso se ha fijado $k = 1$ por simplicidad.

Algoritmo 1: Entrenamiento por descenso estocástico del gradiente en *minibatch* para redes generativo-adversariales (Goodfellow *et al.*, 2014a)

Datos: El conjunto de datos de entrenamiento $\mathbf{X}$ y un generador de ruido gaussiano $\mathbf{Z}$

Resultado: Parámetros optimizados de las redes G y D

para *número de iteraciones de entrenamiento* **hacer**

Tomar un *minibatch* de N instancias de ruido $\{\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(N)}\}$ de acuerdo a $\mathbf{Z}$

Tomar un *minibatch* de N instancias reales $\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}\} \subset \mathbf{X}$

Actualizar el discriminador por ascenso de su gradiente estocástico:

$$\nabla_{\boldsymbol{\theta}_D} \frac{1}{N} \sum_{i=1}^{N} \left[\log D\left(\mathbf{x}^{(i)};\boldsymbol{\theta}_D\right) + \log\left(1 - D\left(G\left(\mathbf{z}^{(i)};\boldsymbol{\theta}_G\right);\boldsymbol{\theta}_D\right)\right)\right]$$

Actualizar el generador por descenso de su gradiente estocástico:

$$\nabla_{\boldsymbol{\theta}_G} \frac{1}{N} \sum_{i=1}^{N} \log\left(1 - D\left(G\left(\mathbf{z}^{(i)};\boldsymbol{\theta}_G\right);\boldsymbol{\theta}_D\right)\right)$$

fin

1.4. Estructura del trabajo

El trabajo que sigue se desarrolla siguiendo una estructura dividida en cinco capítulos, siendo el primero (Capítulo 1) la introducción. El Capítulo 2 comprende una discusión integral sobre la tarea de detección de anomalías con especial énfasis en los modelos generativos que se han diseñado para este fin en los últimos años. También se realiza en el mismo un comentario sobre el paradigma de interpretabilidad, enfrentando redes Bayesianas y redes neuronales profundas en extremos opuestos del espectro.

El Capítulo 3 supone el grueso teórico del trabajo, pues recoge con detalle todo el proceso de diseño, construcción e implementación algorítmica del aprendizaje del BEAA, el modelo propuesto como una solución novedosa para el problema de detección de anomalías semisupervisada. El modelo propuesto tendrá los objetivos de:

- Poder desempeñar la función de detección de intrusiones en una red de ordenadores de forma competente en un escenario práctico de ciberseguridad.
- Suponer un modelo generativo de detección semisupervisada de anomalías capaz de proveer una interpretabilidad y un rendimiento a medio camino entre un modelo de red Bayesiana y una arquitectura GAN compleja con redes neuronales profundas.

Estos objetivos del modelo se tratan de demostrar en el Capítulo 4, donde se detallan los métodos experimentales y los resultados obtenidos al entrenar y probar el BEAA sobre dos muestras de datos: un *dataset* público para la evaluación de modelos de detección de intrusiones en red por anomalías y un conjunto de datos proporcionado por *Titanium Industrial Security* S.L. sintetizado en un laboratorio virtual con ataques ejecutados en tiempo real por un experto en ciberseguridad. En el Capítulo 5, que cierra el trabajo, se exponen las conclusiones finales sobre el rendimiento del BEAA, se comenta su posición en el mapa de modelos de detección de anomalías y se introducen las futuras líneas de investigación previstas para su mejora.

Capítulo 2

Estado de la cuestión: detección de anomalías e interpretabilidad

2.1. Detección de anomalías con modelos generativos

De entre los algoritmos generativos más ampliamente extendidos, las GANs destacan por su gran rendimiento y versatilidad, por lo que también han sido aplicadas con éxito para tareas de detección de anomalías (Deecke *et al.*, 2019). Tras entrenar adversarialmente un par discriminador-generador sobre un conjunto de datos $\mathcal{S}$ de instancias etiquetadas como normales, el propio discriminador puede ser considerado como un detector de anomalías muy rudimentario, sencillamente utilizando su valor de salida como una puntuación de anomalía para la clasificación de nuevos datos. Este método, no obstante, sufre de un grave inconveniente; depende fuertemente de cómo concluyó el entrenamiento:

- Si el discriminador D era capaz de etiquetar correctamente la mayor parte de los datos reales y falsos en el momento en que termina el entrenamiento, entonces la salida $D(\mathbf{x};\boldsymbol{\theta}_D)$ obtenida para nuevas instancias será próxima a cero para anomalías, ya que el discriminador podrá determinar con confianza que no provienen de la distribución de la muestra de datos con comportamiento normal, por lo que les asignará una baja probabilidad de ser reales. Igualmente, la probabilidad $D(\mathbf{x};\boldsymbol{\theta}_D)$ será próxima a uno para instancias normales, indicando que el discriminador considera que es altamente probable que sigan la misma distribución que la muestra de instancias reales sobre las que ha entrenado el modelo adversarial.
- Si, por el contrario, era el generador G el que estaba consiguiendo engañar al discriminador una fracción significativa de los intentos cuando terminó el entrenamiento, entonces el generador habría capturado de forma precisa la distribución subyacente a los datos normales. En tal caso, la salida del discriminador $D(\mathbf{x};\boldsymbol{\theta}_D)$ para nuevos datos puede llegar a devolver un valor próximo a uno para anomalías, ya que el discriminador habrá entrenado con instancias generadas muy fieles a los datos de $\mathcal{S}$, etiquetados como normales. De este modo considerará que una anomalía no puede haber sido producida por G y le será asignada una alta probabilidad de ser real.

El discriminador, por tanto, aunque puede ser utilizado para distinguir instancias

reales de sus contrapartes generadas, no está equipado para trabajar con ejemplos que difieran significativamente de la muestra sobre las que ha entrenado. Para construir modelos detectores de anomalías más robustos con GANs, otros autores han explorado el uso de la red generadora para esta tarea. En esta línea de investigación existen dos estudios que se han utilizado como referencia para comenzar la construcción teórica del modelo presentado en este trabajo, que se exponen en las siguientes subsecciones.

2.1.1. La arquitectura AnoGAN

Presentada por Schlegl *et al.* (2017), esta estructura emerge como un modelo detector de anomalías basado en GANs estándar entrenadas exclusivamente en instancias normales (no anómalas), entrenando al generador para sintetizar instancias normales realistas mediante la aplicación $G(\mathbf{z}; \boldsymbol{\theta}_G) : \mathbf{z} \to \mathbf{x}$. Tal función puede ser también utilizada para encontrar la aplicación inversa, que devuelve un dato dado a su representación en el espacio latente, $E(\mathbf{x}) : \mathbf{x} \to \mathbf{z}$. Para encontrar el vector de ruido $\mathbf{z}$ que lleva a la reconstrucción $G(\mathbf{z})$ más próxima a la $\mathbf{x}$ original, las coordenadas de $\mathbf{z}$ en el espacio latente se hallan a través de un proceso iterativo. En cada paso, numerado según $\gamma = 1, \ldots, \Gamma$, se actualiza el vector de ruido en el espacio latente que devuelve la aplicación $E(\mathbf{x})$, denotado por $\mathbf{z}_\gamma$. La regla de actualización de $\mathbf{z}_\gamma$ persigue la minimización de dos funciones de pérdida que se computan a partir de los modelos generador y discriminador aprendidos;

- Una pérdida de reconstrucción o residual, $\mathcal{L}_R(\mathbf{z}_\gamma)$, que cuantifica el error cometido al reconstruir la instancia $\mathbf{x}$ a través de la aplicación compuesta $G(E(\mathbf{x}); \boldsymbol{\theta}_G)$. Esta función asegura que el entrenamiento del modelo tenderá a buscar los parámetros que aseguren la similaridad entre los ejemplos generados y la muestra de entrenamiento:
$$\mathcal{L}_R(\mathbf{z}; \boldsymbol{\theta}_G) = ||\mathbf{x} - G(\mathbf{z}; \boldsymbol{\theta}_G)||. \tag{2.1}$$

- Una pérdida de discriminación o contextual, $\mathcal{L}_D(\mathbf{z}_\gamma)$, que restringe a las instancias generadas a pertenecer a la variedad dada por la representación aprendida del espacio de datos a partir de los ejemplos de entrenamiento, etiquetados como normales. La función de pérdida contextual se vale de la entropía cruzada binaria, $H_\alpha(\mathbf{x})$, para forzar a las instancias sintéticas provenientes de G a obtener alta probabilidad de ser consideradas reales por a D. Así, la clase objetivo α deberá ser 1, representativa de la probabilidad máxima de que una instancia dada constituya una real:
$$\mathcal{L}_D(\mathbf{z}; \boldsymbol{\theta}_G, \boldsymbol{\theta}_D) = H_1\big(D(G(\mathbf{z}; \boldsymbol{\theta}_G); \boldsymbol{\theta}_D)\big) = -\log\big(D(G(\mathbf{z}; \boldsymbol{\theta}_G); \boldsymbol{\theta}_D)\big), \tag{2.2}$$
con la entropía cruzada binaria formulada como:
$$H_\alpha(\mathbf{x}) = -[\alpha \log(\mathbf{x}) + (1 - \alpha) \log(1 - \mathbf{x})].$$

Así, para cada paso, la función de pérdida que compone ambas partes por medio de su suma ponderada resulta ser:

$$\mathcal{L}(\mathbf{z}_\gamma; \boldsymbol{\theta}_G, \boldsymbol{\theta}_D) = (1 - \lambda)\mathcal{L}_R(\mathbf{z}_\gamma; \boldsymbol{\theta}_G) + \lambda \mathcal{L}_D(\mathbf{z}_\gamma; \boldsymbol{\theta}_G, \boldsymbol{\theta}_D),$$

donde se tiene que $\lambda \in [0, 1]$.

Finalmente, la detección de anomalías en la arquitectura AnoGAN se ejecuta utilizando como puntuación de anomalía la propia función de pérdida para el último paso del proceso iterativo de cálculo de la representación de $\mathbf{x}$ en el espacio latente. En esta forma, denotando por $r(\mathbf{x})$ y $d(\mathbf{x})$, respectivamente, las pérdidas residuales y de discriminación calculadas en el último paso iterativo, es decir, $\mathcal{L}_R(\mathbf{z}_\Gamma)$ y $\mathcal{L}_D(\mathbf{z}_\Gamma)$, la puntuación de anomalía, S_A, se escribe como:

$$S_A(\mathbf{x}) = (1 - \lambda)r(\mathbf{x}) + \lambda d(\mathbf{x}).$$

El funcionamiento de este sistema detector de anomalías recae sobre la idea según la cual, como el generador ha sido entrenado sobre una muestra de datos normales, una instancia anómala recibirá una reconstrucción más pobre que una cuyo comportamiento cumpla con la norma, ya que será difícil para el generador hallar un punto en el espacio latente que, a la vez, proporcione una representación precisa del ejemplo original y pertenezca a la representación aprendida del espacio de datos.

2.1.2. La arquitectura GANomaly

Otro marco de detección de anomalías a partir de un entrenamiento exclusivamente sobre datos etiquetados como normales y empleando redes generativo-adversariales es el sistema GANomaly (Akcay *et al.*, 2018). En el núcleo de su funcionamiento se encuentra un submodelo autocodificador o *autoencoder*, A, ejerciendo de agente generador. Los autocodificadores son sistemas integrales de codificación-decodificación con la particularidad de que los espacios de entrada y salida corresponden al mismo dominio, en este caso, el espacio de datos.

En la arquitectura GANomaly el autocodificador A viene implementado mediante dos redes neuronales:

- Un codificador A_E, que recibe una instancia de entrada $\mathbf{x}$ y devuelve una representación comprimida (por ser de menor dimensionalidad) de ésta, $\mathbf{z}$, perteneciente al espacio latente. Las dimensiones del espacio latente, idealmente, deben ser aquellas que proporcionen la representación más fiel de $\mathbf{x}$ con menor dimensionalidad.
- Un decodificador A_D, que acepta como entrada un vector de ruido $\mathbf{z}$ y lo manda al espacio de datos, aumentando su dimensión.

Componiendo estas dos redes neuronales en forma de pajarita, es decir, conectando la salida de A_E a la entrada de A_D, se crea un tipo especial de generador: $A = A_D \circ A_E$. El autocodificador A toma como entrada una instancia perteneciente al espacio de datos, $\mathbf{x}$ y devuelve una reconstrucción de ésta, denotada por $\mathbf{x}'$, que se forma primero proyectando la instancia original sobre el espacio latente y seguidamente transformando el vector de ruido obtenido de nuevo al espacio de datos, lo que equivale a hacer pasar el dato por un cuello de botella al disminuir la dimensionalidad de su representación y posteriormente volverla a ampliar.

La estructura GANomaly, representada en la Figura 2.1, incluye dos redes neuronales adicionales:

- Un codificador, E, que realiza la tarea de volver a proyectar la instancia reconstruida $\mathbf{x}'$ al espacio latente, devolviendo $\mathbf{z}' = E(\mathbf{x}')$.

- Un discriminador, D, cuyo objetivo es la clasificación del par compuesto por una instancia $\mathbf{x}$ y su reconstrucción autocodificada $\mathbf{x}'$ como adecuada o pobremente reconstruida (o, equivalentemente, real y falso).

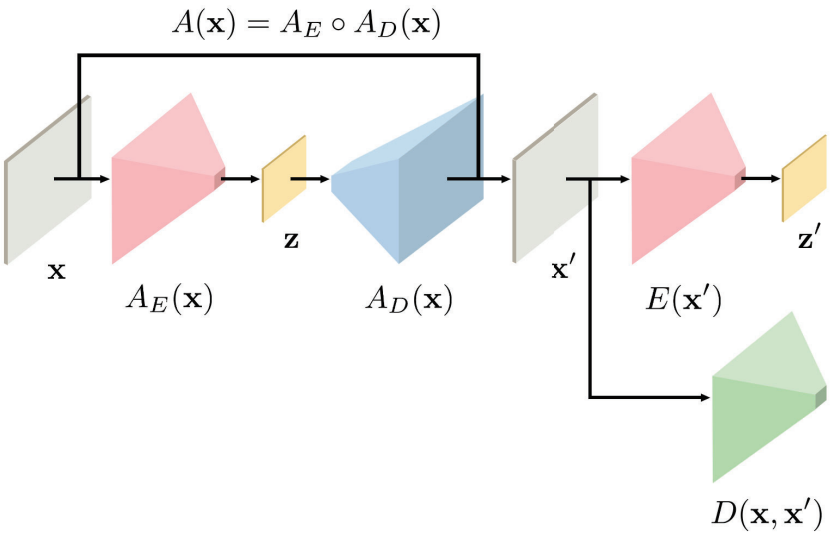


Figura 2.1: Estructura GANomaly (Akcay *et al.*, 2018).

El entrenamiento de este modelo se lleva a cabo por optimización de tres funciones de pérdida: las pérdidas de reconstrucción, $\mathcal{L}_R$, y discriminación, $\mathcal{L}_D$, ya introducidas en la Ecuación 2.1 y en la Ecuación 2.2, respectivamente, y una pérdida de codificación, $\mathcal{L}_E$, cuyo objetivo es minimizar la distancia Euclídea entre las características codificadas del dato de entrada, $\mathbf{z} = A_E(\mathbf{x})$, y las del dato reconstruido, que viene dado por:

$$\mathbf{z}' = E(A_D(A_E(\mathbf{x}))) = E(A_D \circ A_E(\mathbf{x})) = E(A(\mathbf{x})),$$

quedando así escrita la pérdida de codificación como:

$$\mathcal{L}_E(\mathbf{x}) = \|\mathbf{z} - \mathbf{z}'\| = \|A_E(\mathbf{x}) - E(A(\mathbf{x}))\|.$$

La tarea de detección de anomalías en la arquitectura GANomaly se lleva a cabo empleando como puntuación de anomalía a una muestra $\mathbf{x}$ dada la pérdida de codificación que se calcula reconstruyéndola, $\mathcal{L}_E(\mathbf{x})$.

La hipótesis tras el funcionamiento de este modelo se fundamenta en que cuando el sistema se alimenta con una instancia anormal, aunque A_E es capaz de mapearla al espacio latente y obtener de ella un vector de ruido, la parte generadora del autocodificador, A_G, no será capaz de reproducir las anomalías presentes en la instancia de entrada a partir de su representación en el espacio latente calculada por el codificador, pues al haber entrenado exclusivamente sobre ejemplos etiquetados como normales, la parametrización del sistema no permite la generación de datos anómalos. Así, cuando la reconstrucción $\mathbf{x}'$ pasa por alto las anomalías que había presentes

a la entrada, su representación en el espacio latente $\mathbf{z}'$ tampoco las contendrá, provocando la aparición de una cierta disimilitud entre $\mathbf{z}$ y $\mathbf{z}'$.

Nótese que la principal diferencia de esta arquitectura con respecto a la AnoGAN reside en que la aplicación de codificación $E(\mathbf{x})$ viene dada por una red que se entrena en conjunto con el resto del modelo, en lugar de ser calculada iterativamente.

2.2. El papel de la interpretabilidad en modelos generativos de detección de anomalías

Citando textualmente el trabajo de Ian Goodfellow y sus colaboradores sobre GANs: "El *framework* de modelización adversarial se aplica en su forma más directa cuando ambos modelos implicados son perceptrones multicapa" (Goodfellow *et al.*, 2014a). Las dos arquitecturas de detección de anomalías con modelos generativos previamente discutidas emplean perceptrones multicapa (i.e., redes de neuronas artificiales) como el paradigma matemático para codificar las funciones que gobiernan el funcionamiento de ambos sistemas.

No obstante, aunque las redes neuronales son sencillas de entrenar de forma adversarial, están sujetas por diseño a una serie de limitaciones que derivan de su falta de interpretabilidad. Dicho efecto se ve tanto más acentuado cuanto más profunda es la red, es decir, cuantas más neuronas y capas la componen, por lo que es especialmente problemático cuando la aplicación en cuestión requiere hallar patrones y relaciones de alta dimensionalidad con un gran número de variables, siendo ésta la coyuntura habitual en los escenarios de ciberseguridad.

2.2.1. Introducción a la interpretabilidad

Se dice que un sistema de inteligencia artificial es interpretable si, por diseño, se pueden comprender las decisiones provistas por los algoritmos que lo componen, siempre procurando mantener la transparencia en su funcionamiento interno que, además, debe estar formulado en términos matemáticos de forma clara y coherente (Tjoa y Guan, 2019).

A la hora de emplear un modelo de *machine learning* para tomar una decisión que sea susceptible de afectar a la economía de una empresa o incluso a la vida de una persona y, en general, en cualquier situación en que una parte pueda salir perjudicada por utilizar un algoritmo, existe la necesidad de entender las razones que han llevado al sistema inteligente hacia una determinada conclusión, de manera que sea posible garantizar que su uso ha sido responsable (Kaur *et al.*, 2022).

Especialmente en contextos críticos, como pueden representar los campos de la medicina o de la ciberseguridad, la demanda de interpretabilidad y transparencia en los modelos de inteligencia artificial ha crecido ante el peligro de generar predicciones y decisiones que no sean justificables por no poderse obtener una explicación detallada del funcionamiento del modelo (Tjoa y Guan, 2019). La comunidad se está volviendo reticente a confiar en técnicas que no sean fácilmente interpretables, lo que se ve reflejado en la legislación: en el Artículo 22 del Reglamento General de Protección de Datos (GDPR por sus siglas en inglés, *General Data Protection Regulation*), efectivo desde 2018 en toda la Unión Europea, se prohíbe la toma de cualquier decisión capaz

de producir efectos legales o significativos basándose exclusivamente en un procesamiento automatizado (Goodman y Flaxman, 2017). De la lectura comprensiva del resto de artículos, se puede extraer la conclusión de que del GDPR emana un nuevo "Derecho a la explicación" para el ciudadano, de acuerdo al cual las compañías de procesamiento de datos tienen el deber de informar al usuario no solo de cómo han sido sus datos recopilados, sino también de cómo han sido analizados, detallando de forma concisa y accesible el procedimiento de generación de resultados.

De acuerdo con Barredo-Arrieta *et al.* (2020), pueden encontrarse tres razones por las que la interpretabilidad debe considerarse como un objetivo de diseño durante el desarrollo e implementación de un modelo de *machine learning*:

- La posibilidad de entender el por qué de las conclusiones de un modelo incrementa la capacidad del operador de detectar sesgos en el conjunto de datos de entrenamiento, lo que resulta en una mejora de la imparcialidad del modelo.
- Los ejemplos adversariales (Szegedy *et al.*, 2013) constituyen instancias especializadas de entrada para un modelo de inteligencia artificial creadas a propósito para confundirlo, provocando que se comporte incorrectamente. Los ejemplos adversariales fuerzan su clasificación errónea, si bien difieren solo ligeramente de otras instancias que se clasifican acertadamente (Goodfellow *et al.*, 2014b). Estoe supone un problema especialmente serio en ámbitos como el de ciberseguridad, en el que los ejemplos adversariales suponen una vía por la que camuflar un ataque a una red o un comportamiento malicioso de modo que sea clasificado erróneamente como benigno por un sistema de detección de intrusiones basado en detección de anomalías.

  La generación de este tipo de ejemplos se suele realizar por perturbación de una instancia inicial, añadiéndole ruido que trate de aumentar el error en su clasificación. La técnica de producción de instancias adversariales más extendida es el método rápido de signo del gradiente (FGSM, por sus siglas en inglés, *Fast Gradient Sign Method*), que emplea la derivada de la función de error del modelo con respecto de la entrada para hallar el ruido a sumar a la instancia original. Un modelo que sea interpretable facilitará el diagnóstico de sus potenciales debilidades adversariales.
- La capacidad de interpretabilidad de un modelo potencia la habilidad de operadores y desarrolladores para entender cómo y qué variables afectan al resultado que devuelve, por lo que puede ser utilizada para garantizar que en el razonamiento del modelo solo tienen impacto variables con significado y relevantes para el problema, volviendo al sistema más robusto. Además, contribuye a asegurar que el razonamiento siga, de existir, la causalidad subyacente a los datos de entrenamiento.

2.2.2. Modelos de caja negra y modelos transparentes

Surgen, en consecuencia, numerosas ventajas al utilizar técnicas de *machine learning* inteligibles para el operador o usuario. Al respecto suele hablarse con dos términos que a menudo se usan de forma intercambiable, pero que para esta discusion merecen ser desambigüados: interpretabilidad y explicabilidad. Doshi-Velez y Kim (2017) denominan interpretabilidad a la característica de un modelo que permite que pueda ser explicado en términos comprensibles por un humano. Separadamente, Rudin

(2019) establece la necesidad de diferenciar de distinguir aquellos modelos que inherentemente son interpretables de aquellos que proveen explicaciones a posteriori por ser incomprensibles para un ser humano. Así, se puede identificar la frontera entre interpretabilidad y explicabilidad con la que separa los modelos que, por diseño, funcionan de una forma que un humano puede entender y aquellos que ofrecen información adicional sobre sus procesos de razonamiento tras haber llegado a una conclusión de una forma ininteligible.

Los sistemas de inteligencia artificial de más alto rendimiento y precisión son, frecuentemente, modelos que ocultan la lógica de sus procesos internos bajo numerosas capas de transformaciones matemáticas, por lo que para cumplir con los estándares de interpretabilidad se apoyan en el uso de técnicas de explicabilidad a posteriori (también llamadas *post-hoc*) (Bodria *et al.*, 2021). Así, estos modelos se denominan de caja negra, siendo justamente el caso de las redes neuronales profundas (DNN, *deep neural networks*).

En contraposición, existen arquitecturas algorítmicas denominadas transparentes que, aunque suelen proporcionar rendimientos más bajos en problemas de alta dimensionalidad, son interpretables por diseño. Entre los modelos transparentes más interpretables se encuentran las redes Bayesianas, o BNs, por sus siglas en inglés: *Bayesian Networks* (Mihaljević *et al.*, 2021).

Es precisamente por este motivo que en el presente trabajo se introduce un nuevo enfoque generativo-adversarial para el aprendizaje de redes Bayesianas que trata de proponer una solución alternativa al problema de la detección de anomalías, representando un compromiso entre los modelos de detección de anomalías basados en GANs, que emplean DNNs, y aquellos que explotan las propiedades clasificadoras de una BN. El objetivo es, por tanto, construir un sistema detector de anomalías que sea capaz de proveer un mejor rendimiento que una BN aunque reteniendo la mayor parte posible de su interpretabilidad inherente, justificando así su uso frente a un modelo basado en DNNs en aplicaciones que requieran cierta comprensión del razonamiento del algoritmo, como se da dentro del contexto de ciberseguridad.

2.3. Redes Bayesianas generativo-adversariales para detección de anomalías

2.3.1. Detección de anomalías con una red Bayesiana

La tarea de detección de anomalías se puede llevar a cabo de forma elemental utilizando una única BN. Para ello, es preciso comenzar efectuando el aprendizaje de su estructura, que por definición viene dada por un grafo acíclico dirigido (DAG por sus siglas en inglés, *Directed acyclic graph*) $\mathcal{G}$, y de sus parámetros $\theta_{\mathcal{G}}$, que son las distribuciones de probabilidad condicionadas (CPDs por sus siglas en inglés, *Conditional probability densities*) de cada variable según los valores de sus padres de acuerdo al grafo (Pearl, 1988; Koller y Friedman, 2009).

Una vez acaba el proceso de aprendizaje se dice que la BN, dada por el par $(\mathcal{G}, \theta_{\mathcal{G}})$, ha sido ajustada al problema. Tras ajustar una BN a un conjunto de datos de entrenamiento etiquetados como normales, la detección de anomalías se realiza calculando la log-verosimilitud para una instancia, $\mathbf{x}$, que se desee clasificar, $l(\mathbf{x})$, y empleándola como una medida de cuán normal es. Ésta constituye una asunción muy natural,

dado que la log-verosimilitud de una instancia es una métrica que cuantifica en escala logarítmica cómo de probable es que ocurra a partir de las relaciones aprendidas por la red de los datos de entrenamiento:

$$l(\mathbf{x}; \mathcal{G}, \boldsymbol{\theta}_{\mathcal{G}}) = \log(P(\mathbf{x}; \mathcal{G}, \boldsymbol{\theta}_{\mathcal{G}})).$$

Para detección de anomalías tal y como se presentó en la introducción (ver Capítulo 1), el conjunto de entrenamiento estará compuesto por una muestras representativa del comportamiento normal. De este modo, los datos anómalos estarán asociados a una menor probabilidad de aparición y, por tanto, con una log-verosimilitud más baja. Dada una estructura de BN en la que cada variable X_i con $i \in \{1, \ldots, n\}$ representa un nodo en el DAG y está gobernada por una distribución de probabilidad condicionada en sus padres según el grafo $\mathcal{G}$, $P(X_i|Pa^{\mathcal{G}}(X_i))$, para una instancia $\mathbf{x} = (x_1, \ldots, x_n)$, la log-verosimilitud se escribe:

$$l(\mathbf{x}; \mathcal{G}, \boldsymbol{\theta}_{\mathcal{G}}) = \log\left(\prod_{i=1}^{n} P(X_i = x_i|Pa^{\mathcal{G}}(X_i) = pa_i^{\mathcal{G}})\right),$$

donde $pa_i^{\mathcal{G}}$ denota la instanciación de los padres de la variable X_i de acuerdo a la instanciación provista por $\mathbf{x}$ y según $\mathcal{G}$. Este método, a pesar de su sencillez, presenta el inconveniente de que requiere fijar un umbral con el propósito de separar datos normales de anomalías según el valor de su log-verosimilitud (Atienza *et al.*, 2021a).

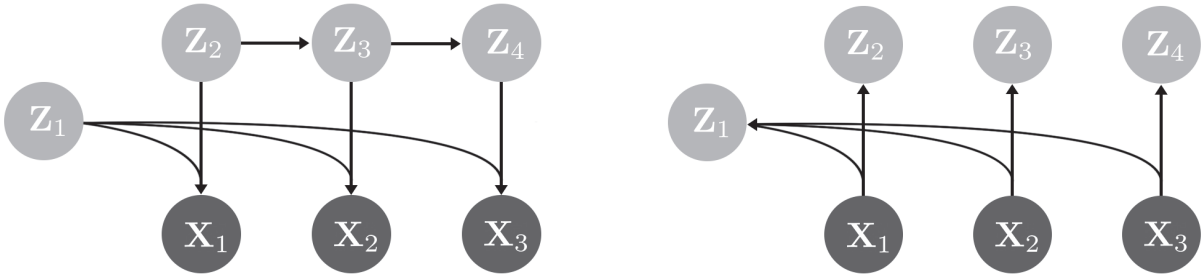
2.3.2. Técnicas generativo-adversariales para redes Bayesianas

El primer paso para construir un modelo generativo-adversarial basado en redes Bayesianas consiste en establecer una forma de entrenar adversarialmente una pareja de BNs actuando como el discriminador y el generador de una GAN.

El trabajo de Li *et al.* (2018a) sobre GANs gráficas establece un marco teórico para la modelización de relaciones de datos de alta dimensión utilizando un generador que tiene asociado un DAG, uniendo así el poder de representación de dependencias entre variables de las BNs con la capacidad de expresión de dependencias funcionales de las GANs.

La estructura de DAG del generador en este modelo permite que la producción de nuevos datos falsos sea directa mediante muestreo ancestral, instanciando como evidencia una serie de variables latentes que actúan de padres en el grafo $\mathcal{G}$ de las variables observables, que se extraen simulando desde los nodos raíz hacia los nodos hoja. No obstante, si bien la estructura de dependencias se representa por medio de un DAG, las funciones que caracterizan las relaciones entre variables están parametrizadas como DNNs, por lo que las funciones de verosimilitud aparecen en forma implícita y pueden llegar a ser intratables computacionalmente.

Para el aprendizaje del modelo se emplea un submodelo de reconocimiento auxiliar al de generación y ejerciendo como discriminador. Como se puede apreciar en la Figura 2.2, atendiendo exclusivamente a las aristas (arcos no dirigidos) de los grafos que conectan varibles latentes con variables observables, el esquema de ambos submodelos es el mismo, con la peculiaridad de que al tener en cuenta también la dirección de los arcos, en el submodelo generativo las relaciones de dependencia siempre tienen



(a) Submodelo generativo

(b) Submodelo de reconocimiento

Figura 2.2: Ejemplos de submodelos generativo y de reconocimiento de la arquitectura de GAN gráfica de Li *et al.* (2018a). Las variables latentes aparecen representadas en color blanco y las variables observables en gris. Nótese que en el submodelo generativo pueden existir relaciones entre variables latentes, y los arcos siempre parten de nodos de variables latentes a nodos de variables observables o latentes. Por el contrario, en el submodelo de reconocimiento los arcos siempre parten de un nodo correspondiente a una variable observable y llegan a un nodo que representa una variable latente.

como padre a un nodo de variable latente y en el submodelo de reconocimiento, a uno que represente una variable observable. En el submodelo generativo, además, están permitidos los arcos entre dos nodos correspondientes a variables latentes.

Es precisamente en el tratamiento implícito de las probabilidades, definiendo un proceso estocástico cuyo objetivo es la producción de instancias de acuerdo a una distribución no especificada, así como en el uso de un modelo de reconocimiento y uno de generación, donde estas GANs gráficas se asemejan al algoritmo propuesto en este trabajo. En contraste, en la arquitectura propuesta por Li y sus colaboradores se parametrizan las funciones de dependencia entre variables a través de DNNs, por lo que la novedad principal que implementa el modelo introducido en el presente trabajo consiste en el tratamiento en forma de BN de las funciones de dependencia entre variables, ya que los submodelos equivalentes a la parte generativa y de reconocimiento serán BNs.

El segundo trabajo que ha servido de referencia durante la fase de desarrollo conceptual del modelo es el de Ding *et al.* (2021) sobre GANs que emplean grafos para representar las relaciones de independencia condicional. En el artículo en que presentan su modelo, Ding y sus colaboradores subrayan que las GANs son, según su formulación original, diseñadas de forma independiente de modelo (*model-free*), refiriéndose a que no son capaces ni de explotar ni de proporcionar ninguna información sobre la estructura de dependencias de la distribución subyacente de las variables que procura aprender el modelo. No obstante, tal información sí que está presente en modelos que se construyen a partir de un grafo de dependencias, como una BN, si el grafo es dirigido, o un campo aleatorio de Markov (MRF por sus siglas en inglés, *Markov random field*), si el grafo es adireccional.

El problema que tratan de resolver responde, por tanto, a la pregunta: ¿Se puede elaborar un sistema de tipo GAN cuyo diseño tenga en cuenta el modelo (*model-based*) que sea capaz de utilizar información *a priori* sobre la estructura de dependencias entre las variables capturada por una BN o un MRF? Al respecto, el modelo que

introducen sustituye al discriminador de la GAN estándar por un conjunto de discriminadores más simples definidos localmente sobre un vecindario, que en una BN vendría dado por la unión de un nodo y sus padres. De este modo, la estructura de discriminadores locales constituye el elemento mediante el que se introduce en la GAN el esquema de dependencias condicionales.

Al entrenar un generador por competición adversarial con el conjunto de discriminadores locales mencionado, éste llegará a codificar una distribución Q que minimice la suma de divergencias (puede tomarse como divergencia cualquier métrica que permita representar la distancia estadística entre dos distribuciones de probabilidad) entre las marginales de P, la distribución objetivo, y Q sobre cada uno de los vecindarios locales en los que se ha definido un discriminador.

El verdadero objetivo, no obstante, debe ser minimizar la divergencia calculada entre las distribuciones conjuntas P y Q. Para satisfacer este objetivo, Ding y sus colaboradores teorizan que si la métrica escogida como divergencia cumple con una condición que denominan *subaditividad generalizada*, entonces la suma de divergencias tomadas sobre las marginales de P y Q en cada vecindario local establece una cota superior a la divergencia conjunta, con lo que minimizar la suma de divergencias locales equivale a efectos prácticos a minimizar la divergencia conjunta mediante la disminución de su cota superior.

Para dos distribuciones de probabilidad P y Q representadas por dos BNs definidas sobre el mismo espacio muestral y con el mismo DAG, $\mathcal{G}$, de nodos $X_1, \ldots, X_n$, una divergencia δ satisface la condición de subaditividad α-lineal si:

$$\delta(P, Q) \leq \alpha \sum_{i=1}^{n} \delta(P_{X_i \cup X_{\mathrm{Pa}_i}}, Q_{X_i \cup X_{\mathrm{Pa}_i}}),$$

donde el sufijo $X_i \cup X_{\mathrm{Pa}_i}$ indica que la distribución es una marginal local en el vecindario compuesto por la unión del nodo i con el conjunto de sus padres. Utilizando la cota superior de subaditividad (la parte derecha en la ecuación superior) como la función objetivo a minimizar en el entrenamiento adversarial se puede formular una GAN sobre una estructura condicional dada por el grafo subyacente a una BN:

$$\min_{Q} \sum_{i=1}^{n} \delta(P_{X_i \cup X_{\mathrm{Pa}_i}}, Q_{X_i \cup X_{\mathrm{Pa}_i}}).$$

La distribución generada, Q, se identifica con $G(\mathbf{Z}; \boldsymbol{\theta}_G)$ en notación de GAN, donde G representa la función codificada por el generador y $\mathbf{Z}$ es una distribución gaussiana. De este modo, y sea $\mathbf{X}$ la distribución objetivo, la meta del entrenamiento se puede reescribir como:

$$\min_{\boldsymbol{\theta}_G} \sum_{i=1}^{n} \delta_{X_i \cup X_{\mathrm{Pa}_i}}(\mathbf{X}, G(\mathbf{Z}; \boldsymbol{\theta}_G)).$$

Nótese que la acción de los discriminadores locales en el entrenamiento se realiza de forma implícita a través de cada función de divergencia local.

Adicionalmente, dado que los vecindarios locales son usualmente mucho más pequeños que el grafo completo, el uso de un conjunto de discriminadores más simples goza de propiedades computacionales mejoradas frente al uso de un gran discriminador que modeliza el espacio de variables completo.

Capítulo 3

Desarrollo del modelo

En el modelo que se introduce en este trabajo, no solo los submodelos generativo y discriminador de la GAN se construyen sobre una estructura gráfica probabilística, sino que son propiamente BNs. Tanto los parámetros de las redes, es decir, las CPDs, como sus estructuras de DAG se aprenden de forma adversarial en un proceso iterativo de optimización que sigue una estrategia evolutiva. Por añadidura, la tarea de detección de anomalías se ejecuta mediante un sistema autocodificador, de aquí el nombre del modelo propuesto: *Bayesian Evolutive Adversarial Autoencoder*, o BEAA por sus siglas.

El BEAA aparece de forma sintetizada en un artículo ya aceptado para la decimoprimera conferencia internacional sobre modelos gráfico-probabilísticos, PGM por sus siglas en inglés: *Probabilistic Graphical Models* (Casajús-Setién *et al.*, 2022).

La descripción del BEAA requiere anticipar que emplea para su funcionamiento dos tipos de variables, análogas a las que emplean otros modelos generativos:

- Variables observables o atributos, que pertenecen al espacio de datos de dimensión n. Cada variable atributo corresponde a una de las características (*features*) que describen una instancia del espacio de datos.
- Variables latentes o de ruido, que pertenecen al espacio latente de dimensión $m < n$. Suponen las semillas aleatorias a partir de las cuales se realizará la generación de instancias.

Ambos tipos de variables vienen representados en la estructura gráfica por nodos de ruido o nodos atributo, respectivamente. Nótese que cada instancia generada debe provenir de una representación en el espacio latente en forma de valores concretos de las variables de ruido (proceso de generación) y, a su vez, cada instancia real tomada del espacio de datos debe poder ser transformada al espacio latente (proceso de codificación o generación inversa).

3.1. Construcción del BEAA

Para comprender los motivos tras algunas de las decisiones de diseño del BEAA, es necesario saber que el modelo fue concebido para poder ser implementado en Python utilizando la librería PyBNesian (Atienza *et al.*, 2022a,b). Las características de modelización, así como las ventajas y limitaciones de esta librería han constituído un

factor clave a la hora de idear el BEAA, pues su estructura, el flujo de aprendizaje y el algoritmo detector de anomalías debían ser formulables en términos de las herramientas implementadas en PyBNesian.

Uno de los aspectos críticos de PyBNesian consiste en que los nodos que pueden ser instanciados para utilizarse como evidencia, que en la documentación de la librería denominan *nodos interfaz*, deben ser nodos raíz, esto es, no pueden tener padres en la estructura de DAG.

3.1.1. Las componentes del modelo

En su forma compatible con la librería PyBNesian, el BEAA integra tres BNs:

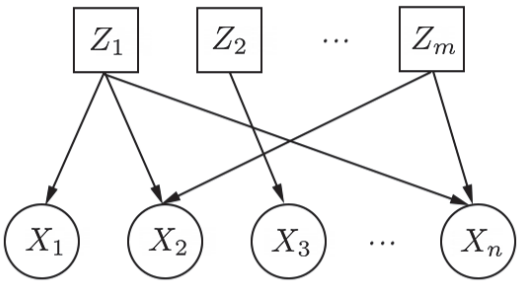
1. Una red generadora, $\mathcal{G}$, cuyos parámetros se aprenden mediante un procedimiento evolutivo por competición adversarial con el discriminador. A diferencia del concepto original de GAN, en el cual el entrenamiento del generador y del discriminador funciona de manera iterativa, alternando la optimización de uno y de otro, en el BEAA el discriminador y el generador no se entrenan conjuntamente, sino que el generador optimiza su estructura y parámetros compitiendo con un discriminador global y estático previamente entrenado. La estructura de DAG de $\mathcal{G}$ en el BEAA fue diseñada para cumplir con dos restricciones:

   - Los nodos de variables latentes (nodos de ruido) deben ser nodos raíz, pues constituirán los nodos interfaz de la BN al implementarse en PyBNesian.
   - Los nodos correspondientes a variables del espacio de datos (nodos atributo) deben ser nodos hoja, sin descendientes.

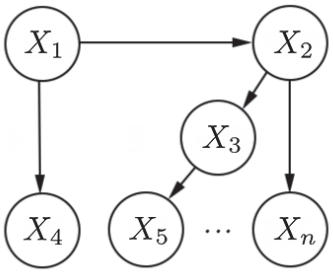
   Las dos restricciones combinadas tienen el efecto de restringir todos los arcos entre variables exceptuando aquellos que conectan un nodo de ruido con un nodo atributo. Esto permite emplear el método de muestreo ancestral para producir nuevos datos generados que sigan la distribución codificada por $\mathcal{G}$ a partir de una instanciación de las variables de ruido que ejerce de semilla. Mediante esta técnica, los valores de los nodos de ruido se establecen como evidencia y son propagados hacia los nodos hoja, obteniendo finalmente una estimación probabilística de los nodos atributo dependiente del ruido empleado, que constituirá la nueva instancia.

2. Una red discriminadora, $\mathcal{D}$, que se emplea como red auxiliar durante la fase de entrenamiento para que el generador aprenda sus parámetros y estructura en competición con ésta. Se trata de una BN estándar que se ajusta directamente a los datos de entrenamiento, compuestos por instancias etiquetadas como normales (no anómalas). Los nodos del discriminador se corresponden con las variables atributo, que son las que caracterizan una instancia en el espacio de datos.

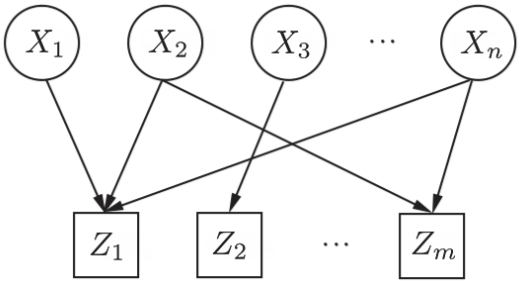
3. Un generador inverso, $\mathcal{R}$, dado por una BN aprendida a partir de la red generadora $\mathcal{G}$ y que representa la función inversa de ésta. La red $\mathcal{R}$, dada una instancia, devuelve una instanciación de las variables latentes correspondiente al vector de ruido que se esperaría que fuera la semilla de la instancia de haber sido producida por $\mathcal{G}$. Sobre el generador inverso se aplican restricciones análogas a las ya nombradas para $\mathcal{G}$, pero revertidas:



(a) Ejemplo de estructura DAG de $\mathcal{G}$



(b) Ejemplo de estructura DAG de $\mathcal{D}$



(c) Ejemplo de estructura DAG de $\mathcal{R}$

Figura 3.1: Las tres componentes del BEAA para un caso genérico con n nodos atributo (representados por círculos) y m nodos de ruido (representados por cuadrados).

- Los nodos de ruido deben ser nodos hoja, pues ahora son las variables latentes las que se pretende muestrear.
- Los nodos atributo deben ser nodos raíz, pues, al ser éstos los nodos que recibirán la evidencia, constituirán los nodos interfaz para $\mathcal{R}$ al implementarse en PyBNesian.

La estructura en DAG de cada componente del BEAA viene ilustrada en la Figura 3.1. La decisión de utilizar dos redes independientes para la generación de instancias a partir de ruido (decodificación) y la predicción de la semilla a partir de un dato dado (codificación) se tomó teniendo en cuenta que la librería PyBNesian solo soporta muestreo ancestral basado en evidencia si los nodos que se instancian son nodos raíz.

Se puede apreciar en esta división por componentes un cierto paralelismo con la estructura de modelo generativo y modelo de reconocimiento de las GANs con grafos de independencia condicional de Li *et al.* (2018a), cuyo trabajo quedó esquematizado en la Sección 2.3.2. Al igual que en la arquitectura que presentan en tal trabajo, el BEAA también cuenta con un modelo generativo cuyos arcos disponibles siempre parten de un nodo que representa una variable latente o de ruido y un modelo de generación inversa, equivalente al de reconocimiento en el modelo de Li y sus colaboradores, cuyos arcos permitidos solo pueden partir de los nodos atributo (véase la Figura 2.2).

3.1.2. Parámetros del BEAA

En la implementación práctica llevada a cabo en este trabajo, los tres modelos que componen el BEAA se materializan como BNs basadas en Gaussianas condicionadas

linealmente, en las que cada CPD tiene la forma funcional:

$$f(x_i|\mathbf{e}) = \mathcal{N}(x_i; \beta_0 + \sum_{j=1}^{k} \beta_{ij} e_j, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{ - \frac{\left(x_i - (\beta_0 + \sum_{j=1}^{k} \beta_{ij} e_j)\right)^2}{2\sigma^2} \right\},$$

donde x_i denota la variable para la que está definida la CPD, $\mathbf{e}$ representa la evidencia utilizada, que toma la forma de un vector k-dimensional, $\mathbf{e} = (e_1, \ldots, e_k)$, cuyos elementos forman una instanciación de los padres de la variable en cuestión, $\boldsymbol{\beta} = (\beta_{i0}, \beta_{i1}, \ldots, \beta_{ik})$ es el vector de parámetros que modeliza las dependencias entre variables y σ^2 representa, como es usual, la varianza de la Gaussiana condicional (Koller y Friedman, 2009).

La elección de BNs basadas en Gaussianas condicionadas linealmente responde a un motivo de practicidad, pues suponen un modelo del que es especialmente fácil disponer en la librería PyBNesian, siendo además ligeros computacionalmente, habilitando así la posibilidad de efectuar numerosas pruebas de concepto en un tiempo razonable. No obstante, las componentes del BEAA pueden tomar la forma de cualquier tipo de BN, incluyendo las basadas en estimación condicional de densidad de kernel (KDE) y las BN semiparamétricas (Atienza *et al.*, 2021b), ambas implementadas de serie en el paquete PyBNesian.

3.2. El proceso de aprendizaje del BEAA

3.2.1. Algoritmo genérico de aprendizaje del BEAA

A lo largo del desarrollo teórico-experimental del modelo se han probado numerosos enfoques para el aprendizaje de los parámetros y la estructura de las BNs que componen el BEAA. La versión del algoritmo de aprendizaje de propósito más general y en su forma más simplificada que sea compatible con su implementación en la librería PyBNesian sigue los pasos que se exponen a continuación:

1. En primera instancia se realiza un ajuste del discriminador $\mathcal{D}$ conforme al conjunto de datos de entrenamiento, $\mathcal{S}$, compuesto exclusivamente por instancias etiquetadas como normales. De este modo, el discriminador aprende las relaciones entre las variables características del comportamiento normal.

   Los datos empleados para el entrenamiento deben ser previamente normalizados. En los experimentos presentados más adelante, la normalización se lleva a cabo por estandarización de los datos conforme a una distribución Gaussiana de media cero y desviación típica uno.

2. Seguidamente se inicializa la red generadora $\mathcal{G}$. El objetivo de este proceso de inicialización es proporcionar una estructura y unos parámetros para el generador que sirvan de punto de partida para su posterior optimización.

   Para poder comenzar el entrenamiento del generador, la primera muestra generada por $\mathcal{G}$ debe contener individuos lo suficientemente distintos entre sí como para poder seleccionar un subconjunto de éstos que sea representativo del conjunto de entrenamiento formado por las instancias reales. Con tal fin, la estructura y parámetros iniciales de $\mathcal{G}$ se establecen a ser:

- Un DAG en el que existe conexión completa de nodos de ruido hacia nodos atributo, es decir, en el que de cada nodo de ruido parte un arco hacia cada nodo atributo.
- Un conjunto de CPDs con parámetros aleatorizados conforme a una distribución Gaussiana en torno a cero. Al estar los nodos de ruido completamente conectados a cada nodo atributo, si las CPDs fueran uniformes, la dependencia de los nodos atributo de los nodos de ruido sería trivial, pues los nodos de ruido serían indistinguibles entre sí.

  Las CPDs que emplean todas las redes del modelo son combinaciones lineales de Gaussianas, por lo que los coeficientes a aleatorizar son los β_{ij}, que especifican la forma en que la variable i-ésima depende de la j-ésima. Los coeficientes β_{i0}, que representan el término independiente de cada relación y caracterizan el desplazamiento base de la media de la Gaussiana, se establecen a 0, pues el valor esperado (la media) de las variables atributo es nulo por haberse estandarizado previamente.

3. Posteriormente deben hallarse los parámetros y la estructura óptima del generador $\mathcal{G}$, lo que se realiza recurriendo a un proceso evolutivo de optimización cuyo punto de partida es la inicialización de $\mathcal{G}$ descrita en la cláusula precedente.

   El procedimiento en cuestión se desarrolla de forma evolutiva, y está inspirado en los algoritmos de estimación de la distribución, o EDAs (*Estimation of Distribution Algorithms)*, un tipo de algoritmo evolutivo en el que cada generación de individuos se estima mediante una simulación de una distribución de probabilidad inferida a partir de una muestra seleccionada de la generación anterior por contener las instancias más representativas de la distribución objetivo (Larrañaga y Lozano, 2002). Esta manera de operar se reitera hasta que la distribución que se infiere de los individuos de cada generación converge o, en algunos casos, hasta que se alcanza un número máximo de generaciones.

   Las iteraciones para el aprendizaje del generador comprenden las siguientes fases:

   a) Se genera un lote (*batch*) de instancias conformes con la distribución que codifica $\mathcal{G}$ usando ruido Gaussiano como evidencia para instanciar los nodos de ruido. Muestreando ancestralmente se logra obtener un valor de los nodos atributo por cada instanciación, de modo que las muestras de instancias de los nodos atributo obtenidas de acuerdo a este proceso serán los individuos de cada generación.

   b) De cada individuo se calcula su log-verosimilitud de acuerdo al discriminador $\mathcal{D}$, atendiendo, por tanto, exclusivamente al valor de sus variables atributo. Para que la siguiente generación constituya una representación más fiel de la distribución de datos con comportamiento normal de la que ha aprendido el discriminador, se selecciona una fracción de los individuos con la log-verosimilitud más alta.

   c) Se reaprenden los parámetros y la estructura del generador $\mathcal{G}$ sobre la muestra seleccionada (cuyas instancias comprenden los valores de las variables atributo y del vector de ruido del que provienen), de forma que en la siguiente iteración el generador será más propenso a proveer datos gene-

rados con mayor log-verosimilitud de acuerdo a $\mathcal{D}$. El valor de β_0 de cada CPD se mantiene cercano a cero siempre que el lote de entrenamiento sea lo suficientemente grande, ya que los datos se someten al proceso de estandarizado, por lo que se puede prescindir de su actualización.

Este proceso se repite hasta que se considera que ha convergido la verosimilitud media de cada generación o, de no darse esta situación, hasta que se llegue a un número máximo de iteraciones.

4. Cuando el generador ha aprendido sus parámetros definitivos, debe aprenderse el generador inverso. A nivel práctico, la transferencia de conocimiento del generador $\mathcal{G}$ al generador inverso $\mathcal{R}$ se ha llevado a cabo de forma simple generando un lote de instancias según $\mathcal{G}$ y aprendiendo los parámetros de $\mathcal{R}$ de éste. La única diferencia de implementación entre $\mathcal{G}$ y $\mathcal{R}$ reside en que el primero tiene prohibido aprender arcos cuya dirección sea de una variable atributo a una de ruido, mientras que para el segundo son los únicos permitidos, de modo que los nodos interfaz puedan ser, respectivamente, los nodos de ruido en $\mathcal{G}$ y los nodos atributo en $\mathcal{R}$.

En el Algoritmo 2 está desarrollada una versión en pseudocódigo del procedimiento de aprendizaje del BEAA.

Algoritmo 2: Algoritmo de aprendizaje evolutivo-adversarial para un autocodificador Bayesiano

Datos: El conjunto de datos de entrenamiento $\mathcal{S}$, compuesto exclusivamente por instancias etiquetadas como normales
Resultado: Parámetros y estructuras optimizadas de las BNs que componen el BEAA: $\mathcal{D}$, $\mathcal{G}$ y $\mathcal{R}$

1 Aprendizaje de los parámetros y la estructura de $\mathcal{D}$ a partir de $\mathcal{S}$
2 Inicialización de $\mathcal{G}$: **para** *cada nodo de ruido* i *en* $\mathcal{G}$ **hacer**
3 Generar n coeficientes aleatorios β_{ij} para el nodo de ruido i, donde $j = 1, \ldots, n$
4 Conectar el nodo de ruido i a cada nodo atributo j con coeficiente β_{ij}
5 El coeficiente β_{i0} se establece a cero
6 **fin**
7 Entrenamiento de $\mathcal{G}$: **repetir**
8 Muestreo de un lote de $N_{\text{generación}}$ instancias de acuerdo a $\mathcal{G}$
9 Cálculo de la log-verosimilitud de cada individuo de la generación según $\mathcal{D}$
10 Selección de los $N_{\text{selección}}$ mejores individuos por su log-verosimilitud
11 Actualización de $\mathcal{G}$ aprendiendo su estructura y parámetros de la muestra seleccionada en la instruccion anterior
12 **hasta que** *convergencia o número máximo de generaciones*;
13 Muestreo de $N_{\mathcal{R}}$ instancias de $\mathcal{G}$
14 Aprendizaje de los parámetros y la estructura de $\mathcal{R}$ a partir de las $N_{\mathcal{R}}$ instancias generadas

3.2.2. Generación de semillas para el muestreo de instancias

En las líneas 8 y 13 del Algoritmo 2 se requiere la generación, a partir de la red $\mathcal{G}$, de un lote de instancias que sean conformes con la distribución codificada en sus parámetros y estructura a cada paso del entrenamiento. Este proceso requiere primero

disponer de un conjunto de semillas aleatorias sobre las que aplicar la función de $\mathcal{G}$ que sea de igual tamaño al que se desea generar.

La producción de semillas aleatorias es un factor crítico para el correcto desempeño del entrenamiento, pues de su distribución dependerá la de las instancias producidas por $\mathcal{G}$ y, por tanto, la evolución del sistema en cada iteración. El uso de semillas cuyo módulo Euclídeo sea demasiado grande impedirá la mejora evolutiva del modelo, pues darán lugar a la producción de datos anómalos (véase la Sección 3.4.1), que alejarán de la distribución objetivo a la versión de $\mathcal{G}$ de la siguiente iteración. El uso de semillas cuyas componentes sean demasiado próximas a cero tendrá, no obstante, el efecto de estancar el aprendizaje de $\mathcal{G}$, pues las muestras carecerán de la exploración necesaria para asegurar que se consideran ejemplos lo suficientemente distintos como para ser representativos de todas las dependencias entre variables inherentes a la verdadera distribución de datos normales.

Para resolver este problema se ha implantado un algoritmo de generación de semillas en forma de corteza de una hiperesfera m-dimensional (Mustard, 1964) en torno al origen de coordenadas del espacio latente, que es de dimensión m. Los valores aleatorios para cada variable de ruido que conforma la semilla se obtienen de una distribución normal bimodal formada por la combinación de dos Gaussianas cuyas medias se encuentran desplazadas simétricamente respecto del origen una cantidad μ_{seed} y con la desviación típica establecida al mismo valor, σ_{seed}.

3.3. Implementación del aprendizaje del BEAA

Los algoritmos utilizados para el aprendizaje de los parámetros y la estructura de las diferentes BN en la implementación práctica del BEAA en Python (líneas 1, 11 y 14 del Algoritmo 2) están limitados a aquellos ya programados en la librería PyBNesian. Aquellos que incluye para la búsqueda de estructuras y que han sido probados en el BEAA son:

- El algoritmo de escalada voraz (*greedy hill-climbing*), que trabaja partiendo de una solución arbitraria y aplicando pequeños cambios locales iterativamente y quedándose con aquellos que mejoran una métrica de la estructura hasta que encuentra una solución satisfaciente. Para mejorar la exploración durante el proceso de búsqueda, cuando la puntuación de la estructura no mejora se emplea un conjunto tabú de operaciones sobre la estructura, de forma que el algoritmo no vuelva a explorar la misma solución a lo largo de una serie de iteraciones.
- El algoritmo PC (Spirtes *et al.*, 1993), que comienza con un grafo no dirigido completamente conexo, procede recurrentemente empleando tests de independencia condicional para eliminar las aristas que conectan entre los que se puede determinar independencia condicional de acuerdo a los datos de entrenamiento. Este proceso inicial aparece recopilado en pseudocódigo en el Algoritmo 3. Posteriormente efectúa un procedimiento similar para establecer la dirección de las aristas que han permanecido, obteniendo una clase de equivalencia de DAG (Le *et al.*, 2015).

  Para comprobar la independencia condicional entre dos nodos dado un subconjunto de otros nodos del grafo, el algoritmo PC requiere testar una hipótesis de

independencia condicional (hipótesis nula) dado el conjunto de datos para el aprendizaje, para lo que se pueden utilizar distintos tests estadísticos. PyBNesian recoge la implementación de una variedad de éstos, pero solo se ha utilizado el test de independencia condicional por información mutua, que permite trabajar con datos categóricos y continuos al mismo tiempo, aunque bajo la asunción de que los datos continuos están distribuidos de forma Gaussiana.

Algoritmo 3: Algoritmo PC para la inducción de la estructura de una BN

Datos: El conjunto de datos $\mathcal{D}$, compuesto por instancias de todos los nodos de la red $\mathcal{N}$

Resultado: El grafo parcialmente dirigido de $\mathcal{N}$

```
Inicialización de Γ, el grafo completamente conexo y no dirigido de la red N
t = 0
repetir
    repetir
        Seleccionar un par de nodos de Γ ordenados y adyacentes: A y B
        Seleccionar un vecindario C de A de tamaño t
        Borrar la arista A − B si A y B son condicionalmente independientes dado C
    hasta que todos los vecindarios son de tamaño menor que t;
    t = t + 1
hasta que todos los vecindarios son de tamaño menor que t;
```

Para proporcionar un marco experimental que permitiera comparar justamente los cambios de implementación y conceptuales en el modelo, así como su rendimiento en diferentes situaciones era requisito escoger uno de los algoritmos disponibles y emplearlo en todos los experimentos, por lo que en la obtención de los resultados que se presentarán en las siguientes secciones se ha empleado el algoritmo PC para la búsqueda de estructuras.

En cuanto a los parámetros de las redes, en PyBNesian solo existe una implementación para hallar los óptimos por medio del método de estimación por máxima verosimilitud (MLE, *Maximum Likelihood Estimation*). La función de verosimilitud, evaluada sobre una serie de instanciaciones de las variables de la red, proporciona una medida de cuánto de probable es obtener tales observaciones bajo los parámetros y estructura calculados (Redner y Walker, 1984). Así, se puede definir de forma sumamente natural el estimador de parámetros de máxima verosimilitud, denotado por $\boldsymbol{\theta}^*$, como aquel que maximiza la función de verosimilitud calculada sobre el conjunto de datos de aprendizaje, $\mathcal{D}$:

$$\boldsymbol{\theta}^* = (\theta_1^*, \ldots, \theta_r^*) = \text{argmax}_{(\theta_1^*,\ldots,\theta_r^*)} P(\mathcal{D}|\boldsymbol{\theta}, \mathfrak{S}),$$

donde $\boldsymbol{\theta}$ representa el conjunto de parámetros de la red, de dimensión r, y $\mathfrak{S}$ el DAG sobre el que se construye.

3.4. Detección de anomalías con el BEAA

Los submodelos discriminador y generador, aunque son de por sí útiles para la generación mediante BNs de muestras coherentes con la distribución que han aprendido,

solo constituyen un medio para el fin de este trabajo, que persigue el objetivo de construir un sistema detector de anomalías.

El elemento que desempeña la función detectora de anomalías es el generador inverso, $\mathcal{R}$. Al introducir un dato $\mathbf{x} = (x_1, \ldots, x_n)$ como evidencia en $\mathcal{R}$ y operar por muestreo ancestral se obtiene un valor para los nodos de ruido de acuerdo a los parámetros aprendidos que se considera la representación de $\mathbf{x}$ en el espacio latente. A partir de esta representación, denominada vector semilla y denotada por $\mathbf{z} = (z_1, \ldots, z_m) = \mathcal{R}(\mathbf{x})$, se evalúa si la instancia es anómala, para lo cual se proponen dos métodos, ambos basados en el uso de una puntuación de anomalía.

3.4.1. Método del módulo del vector semilla

En este primer sistema de detección de anomalías, las irregularidades en los datos se reconocen observando el módulo Euclídeo del vector de ruido $\mathbf{z}$ que se extrae al aplicar $\mathcal{R}(\mathbf{x})$ al introducir como evidencia una instancia dada, $\mathbf{x}$. La intuición que soporta esta idea recae en que, tras entrenar el modelo completo, cada nodo de ruido de $\mathcal{R}$ contendrá codificada en su CPD una relación estadística entre un subconjunto de variables atributo que será característica del comportamiento normal de los datos. Nótese que el número de nodos de ruido determina el número de relaciones que se capturarán y el número de arcos que conecta cada uno a los nodos atributo determinará el grado de cada relación.

Asumiendo que las muestras de datos empleadas tanto para el entrenamiento como para las pruebas se estandarizan previamente, cuando los datos se comportan de acuerdo a la noción de normalidad aprendida por el modelo, el valor inducido por $\mathcal{R}$ para los nodos de ruido deberá ser próximo a cero, garantizando que:

- El valor de cada variable atributo se encuentra próximo al esperado. Tras el proceso de estandarización de los datos, el valor esperado de cada variable atributo calculado en el conjunto de datos de entrenamiento (su media muestral) se transforma en cero. Al introducir en $\mathcal{G}$ una semilla formada por un vector de ceros y operar por muestreo ancestral se generará una instancia de las variables atributo en la que el valor de cada una solo dependerá del desplazamiento β_0 de la media de la Gaussiana que caracteriza su CPD (que es nulo según lo expuesto en la Sección 3.2) y de la desviación típica que se haya aprendido para ésta.

  Equivalentemente, tras aprender $\mathcal{R}$ a partir de $\mathcal{G}$, si una instancia de las variables atributo está compuesta por valores próximos a los esperados (es decir, a cero), la semilla que se obtendrá al emplearla como evidencia será próxima al vector nulo.

- Se satisfacen las relaciones entre variables atributo codificadas en las CPDs de los nodos de ruido. Las expresiones de las CPDs de los nodos de ruido representan, al igualarse a cero, una relación entre una serie de variables atributo que es característica del comportamiento normal capturado por el BEAA.

  Cuando alguna de estas relaciones no se satisface, lo que implica la presencia de anomalías, la evaluación de las CPDs de los nodos de ruido bajo la instanciación dada de las variables atributo proporciona valores diferentes de cero (tanto positivos como negativos). Por el contrario, un elemento del espacio de datos que cumpla con las relaciones de normalidad formuladas en las CPDs de los

nodos de ruido deberá retrotraerse al espacio latente hacia un vector semilla de módulo cercano a cero.

Atendiendo a las razones mencionadas, una vez se calcula por muestreo ancestral mediante $\mathcal{R}$ el vector semilla que contiene la previsión de las instanciaciones de los nodos de ruido para un dato proporcionado, se puede utilizar el módulo de tal vector como una puntuación de anomalía a partir de la cual se puede implementar un clasificador de anomalías a la manera que se introdujo en la Sección 1.1.

Por medio de una métrica de distancia genérica, d_p, el módulo de un vector $\mathbf{v} = (v_1, \ldots, v_t)$ queda definido como:

$$d_p(\mathbf{v}) = \sqrt[p]{\sum_{i=1}^{t} v_i^p}.$$

Se puede emplear cualquier valor positivo de p para calcular el módulo del vector semilla, donde $p = 2$ correspondería al caso Euclídeo, $d(\mathbf{v}) = \sqrt{v_1^2 + \cdots + v_t^2}$. Dependiendo del valor de p se puede ajustar la importancia relativa de los efectos que puede causar una anomalía en un dato: la desviación de la media y el incumplimiento de las relaciones capturadas por las CPDs de los nodos de ruido.

El caso $p = 1$ (distancia Manhattan) supone el escenario en que ambos efectos están balanceados, pues la distancia corresponde simplemente a la suma de todas las componentes del vector: $d(\mathbf{v}) = v_1 + \cdots + v_t$. En el extremo en que $p \to \infty$ (distancia de Chebyshev) domina completamente la coordenada de valor absoluto más alto (Deza y Deza, 2009), pues se tiene:

$$\lim_{p\to\infty} d_p(\mathbf{v}) = \lim_{p\to\infty} \sqrt[p]{\sum_{i=1}^{t} v_i^p} = \text{máx}(|v_i|),$$

lo que tiene el efecto de proporcionar la máxima importancia al valor más alejado de cero de entre todas las componentes del vector semilla, priorizando por tanto la relación que codifica en su CPD el nodo al que corresponde frente a todas las demás y frente a la desviación de la media de las variables atributo. Por el contrario, el uso de $p \in (0, 1)$ suavizará el efecto que producen sobre la puntuación de anomalía las coordenadas más alejadas del origen, igualando la contribución de todas las componentes del vector semilla y aumentando así la importancia relativa de la desviación con respecto a la media de cada variable atributo frente a la de las relaciones codificadas en las CPDs de los nodos de ruido.

Para una instancia x dada, la puntuación de anomalía por módulo del vector de ruido caracterizada por el parámetro de suavizado p se formula matemáticamente como:

$$A^p_{\text{mod}} = d_p\left[\mathcal{R}(\mathbf{x})\right] = d_p(\mathbf{z}) = \sqrt[p]{\sum_{i=1}^{m} z_i^p}.$$

Para facilitar la comparación de resultados, en todos los experimentos presentados se ha tomado el módulo Euclídeo para la definición de la puntuación de anomalía por

módulo del vector semilla, por lo que de ahora en adelante se omitirá el superíndice p cuando se hable de A_{mod}, pues queda fijado a $p = 2$.

3.4.2. Método del error de reconstrucción

Este método explota el error en la reconstrucción de una instancia para obtener una puntuación de anomalía de una manera similar a la arquitectura AnoGAN (Schlegl *et al.*, 2017), presentada en la Sección 2.1.1.

El procedimiento para asignar a una instancia, $\mathbf{x}$, una puntuación de anomalía comienza igual que en el método anterior: calculando, mediante $\mathcal{R}$, el vector semilla de $\mathbf{x}$. Tras este paso se procura reconstruir el dato introducido a partir de su representación en el espacio latente, para lo cual se muestrea ancestralmente y de acuerdo a $\mathcal{G}$ la instanciación de los nodos atributo empleando como evidencia el vector semilla previamente calculado. Ejecutar esta secuencia equivale a rehacer la instancia original, obteniendo una reconstrucción, denotada por $\hat{\mathbf{x}} = (\hat{x}_1, \ldots, \hat{x}_n) = \mathcal{G}(\mathcal{R}(\mathbf{x}))$.

Calcular la distancia Euclídea entre la instancia original y la reconstruída proporciona una medida de cuán precisa es la reconstrucción. Como el BEAA se entrena exclusivamente sobre datos considerados normales, será dificil para las redes que lo componen procesar anomalías correctamente. Así, en el proceso de autocodificación que resulta de componer las funciones codificadas por $\mathcal{R}$ y $\mathcal{G}$, de existir anomalías en $\mathbf{x}$, el dato de entrada, no se reconstruirán correctamente, resultando en una distancia mayor entre $\mathbf{x}$ y $\hat{\mathbf{x}}$. La expresión de la puntuación de anomalía por error de reconstrucción para una instancia $\mathbf{x}$ dada queda:

$$A_{\text{rec}} = d_2\left[\mathbf{x} - \mathcal{G}\left(\mathcal{R}(\hat{\mathbf{x}})\right)\right] = d_2(\mathbf{x} - \hat{\mathbf{x}}) = \sqrt{\sum_{i=1}^{n}(x_i - \hat{x}_i)^2}.$$

Siguiendo este razonamiento, cuanto más grande sea la distancia entre una instancia y su reconstrucción, más probable es que la original contuviese anomalías. Por el contrario, si la distancia entre ambas es baja, puede concluirse que la reconstrucción habrá sido fiel, lo que indica que el dato de entrada es coherente con la muestra de distribución normal sobre la que ha entrenado el BEAA.

3.4.3. Convergencia probabilística de las puntuaciones de anomalía

A la hora de disponer de las dos puntuaciones de anomalía expuestas en los apartados anteriores para utilizarlas en un problema de clasificación debe tenerse en cuenta que el BEAA no es un modelo determinista, sino estocástico. Esto es, el resultado que se devuelve a la salida del modelo no queda completamente definido por los valores que se introducen a su entrada, sino que a lo largo del proceso de propagación de evidencia por las BNs que componen el modelo se aplica una cierta incertidumbre estadística.

Para mitigar las desviaciones puntuales que puedan producirse en el cálculo de las puntuaciones de anomalía por módulo del vector de ruido o por reconstrucción se puede repetir el muestreo del vector semilla, $\mathbf{z}$, o del dato reconstruido, $\hat{\mathbf{x}}$, respectivamente, hasta que converjan a un cierto valor.

El fundamento matemático tras esta solución se halla en la ley débil de los grandes números, cuyo origen se remonta al teorema de Jacob Bernoulli (Seneta, 2013), formulado a principios del siglo XVIII. De este teorema se puede deducir que, dada una sucesión de observaciones $\phi_1, \ldots, \phi_N$ de una variable aleatoria Φ cuyo valor esperado y varianza son μ y σ^2, respectivamente, entonces el promedio de las observaciones converge en probabilidad (es decir, cuando el número de observaciones tiende a infinito) al valor esperado. Fomalmente se escribe en términos de la probabildad de convergencia como:

$$\forall \varepsilon > 0 \ : \ \lim_{N \to \infty} P\left(\left| \frac{1}{N} \sum_{i=1}^{N} \phi_i - \mu \right| < \varepsilon \right) = 1.$$

En vista de este resultado, cabe reformular las puntuaciones de anomalía descritas en la sección anterior empleando el promedio de un conjunto de observaciones. Para el método del módulo del vector semilla, dada una instancia de entrada, $\mathbf{x}$, se realiza el proceso de muestreo ancestral del vector de ruido usando $\mathbf{x}$ como evidencia un número N de veces. En cada experimento se obtiene un valor de $\mathbf{z}$, lo que permite construir el conjunto de observaciones $\{\mathcal{R}^{(j)}(\mathbf{x})\}_{j=1}^{N} = \{\mathbf{z}^{(j)}\}_{j=1}^{N}$. El promedio de los vectores de tal conjunto se calcula componente a componente según $\bar{\mathbf{z}} = (\bar{z}_1, \ldots, \bar{z}_m)$, donde $\bar{z}_i = \frac{1}{N} \sum_{j=1}^{N} z_i^{(j)}$. A partir de este promedio se escribe:

$$\bar{A}_{\text{mod}} = d_2(\bar{\mathbf{z}}) = \sqrt{\sum_{i=1}^{m} \bar{z}_i^2} = \sqrt{\sum_{i=1}^{m} \left(\frac{1}{N} \sum_{j=1}^{N} z_i^{(j)} \right)^2},$$

donde la barra sobre el símbolo de la puntuación denota que en su cálculo se ha empleado un promedio estadístico de las observaciones tomadas de las BNs del modelo.

Una reformulación análoga para la puntuación de anomalía obtenida por el método de reconstrucción requiere realizar múltiples reconstrucciones de una instancia para obtener el promedio de la reconstrucción, $\bar{\hat{\mathbf{x}}} = (\bar{\hat{x}}_1, \ldots, \bar{\hat{x}}_n)$. Realizando N experimentos de reconstrucción puede expresarse cada componente como $\bar{\hat{x}}_i = \frac{1}{N} \sum_{j=1}^{N} \hat{x}_i^{(j)}$, quedando escrita la puntuación por error en la reconstrucción promedio de la manera siguiente:

$$\bar{A}_{\text{rec}} = d_2(\mathbf{x} - \bar{\hat{\mathbf{x}}}) = \sqrt{\sum_{i=1}^{n} (x_i - \bar{\hat{x}}_i)^2} = \sqrt{\sum_{i=1}^{n} \left[x_i - \left(\frac{1}{N} \sum_{j=1}^{N} \hat{x}_i^{(j)} \right) \right]^2}.$$

3.4.4. Implementación práctica del clasificador de anomalías

En la implementación del BEAA como detector de anomalías se ha utilizado también como puntuación para la clasificación de éstas una combinación de las medidas del módulo del vector semilla y del error en la reconstrucción, A_{comb}, en la que quedan sintetizadas ambas métricas por multiplicación: $A_{\text{comb}} = A_{\text{mod}} \cdot A_{\text{rec}}$. Igualmente se puede calcular la puntuación combinada a partir de las puntuaciones del módulo del vector semilla y del error de reconstrucción promediadas, obteniéndose $\bar{A}_{\text{comb}} = \bar{A}_{\text{mod}} \cdot \bar{A}_{\text{rec}}$

La idea tras la formulación de esta nueva puntuación de anomalía consiste en probar la conjetura que establece que ambas puntuaciones están alineadas, es decir, coinciden en gran medida al etiquetar la existencia de anomalías en el conjunto de datos de prueba. De darse tal caso, la combinación de las puntuaciones ayudará a subsanar los errores de ambas, inclinando la balanza a favor de dar la alarma de anomalía detectada cuando una puntuación no lo indique con certeza siempre y cuando la otra sí lo haga.

El procedimiento de asignación de las tres puntuaciones de anomalía a una instancia dada viene condensado en forma de pseudocódigo en el Algoritmo 4.

Algoritmo 4: Asignación de las puntuaciones de anomalía por módulo del vector semilla, por error en la reconstrucción y por combinación de ambas

Datos: La instancia de entrada, $\mathbf{x}$ y las BN generadora, $\mathcal{G}$, y generadora inversa, $\mathcal{R}$, del BEAA, así como N, el número de observaciones sobre las que se obtienen las puntuaciones de anomalía promediadas: $\bar{A}_{\text{mod}}$ y $\bar{A}_{\text{rec}}$.

Resultado: Las puntuaciones $\bar{A}_{\text{mod}}$, $\bar{A}_{\text{rec}}$ y $\bar{A}_{\text{comb}}$ calculadas para $\mathbf{x}$

Muestreo de N instanciaciones $\mathbf{z}^{(j)}$ de los nodos de ruido de $\mathcal{R}$ empleando $\mathbf{x}$ como evidencia

Con las N semillas $\{\mathbf{z}^{(j)}\}_{j=1}^{N}$, calcular $\bar{\mathbf{z}}$: **para** *cada nodo de ruido* i **hacer**

  Obtener el promedio de todas las instanciaciones del nodo: $\bar{z}_i = \frac{1}{M}\sum_{j=1}^{M} z_i^{(j)}$

fin

Componer el vector $\bar{\mathbf{z}} = (\bar{z}_1, \ldots, \bar{z}_m)$

Calcular $\bar{A}_{\text{mod}} = d_2(\bar{\mathbf{z}})$

Con las N semillas $\{\mathbf{z}^{(j)}\}_{j=1}^{N}$: **para** *cada instanciación* j *de los nodos de ruido* **hacer**

  Introducir $\mathbf{z}^{(j)}$ como evidencia en $\mathcal{G}$ y propagarla para obtener $\hat{\mathbf{x}}^{(j)}$

fin

Calcular $\bar{\hat{\mathbf{x}}}$: **para** *cada nodo atributo* i **hacer**

  Obtener el promedio de todas las instanciaciones del nodo: $\bar{\hat{x}}_i = \frac{1}{N}\sum_{j=1}^{N} \hat{x}_i^{(j)}$

fin

Componer el vector $\bar{\hat{\mathbf{x}}} = (\bar{\hat{x}}_1, \ldots, \bar{\hat{x}}_n)$

Calcular $\bar{A}_{\text{rec}} = d_2(\mathbf{x} - \bar{\hat{\mathbf{x}}})$

Obtener $\bar{A}_{\text{comb}} = \bar{A}_{\text{mod}} \cdot \bar{A}_{\text{rec}}$

3.5. Comparación del BEAA con otros modelos de detección de anomalías

Merece la pena hacer un alto en el desarrollo del trabajo para remarcar los puntos en los que el BEAA difiere de los algoritmos de detección de anomalías basados en GANs de redes neuronales presentados en la Sección 2.1: las arquitecturas AnoGAN (Schlegl *et al.*, 2017) y GANomaly (Akcay *et al.*, 2018).

El Cuadro 3.1 recoge las diferencias más importantes entre el BEAA y los sistemas AnoGAN y GANomaly. La interpretabilidad de cada uno de los modelos a la hora de detectar anomalías se refiere exclusivamente a su capacidad de ser inherentemente comprensible por un humano, no contempla la aplicación de técnicas de explicación *post-hoc*.

Característica	AnoGAN	GANomaly	BEAA
Forma que toman el generador y el discriminador	Redes neuronales	Redes neuronales	BNs
Tipo de competencia entre generador y discriminador	Dinámica (el discriminador también entrena)	Dinámica (el discriminador también entrena)	Estática (el discriminador es fijo)
Modo de codificación $\mathbf{x} \rightarrow \mathbf{z}$	Proceso iterativo de reconstrucción	Red neuronal	BN
Puntuación para la detección de anomalías	Suma de pérdida residual y de discriminación	Pérdida de codificación (reconstrucción)	Módulo del vector semilla y error de reconstrucción
Interpretabilidad	Caja negra	Caja negra	Transparente

Cuadro 3.1: Comparación de las características de las dos arquitecturas de detección de anomalías con GAN introducidas en la Sección 2.1 frente al BEAA, propuesto en este trabajo.

3.6. Algoritmos alternativos para el aprendizaje del BEAA

El algoritmo introducido en la Sección 3.2.1 es la versión más refinada y simple desarrollada para entrenar una BN generadora de forma adversarial. No obstante, durante el proceso de diseño del algoritmo genérico, se han ingeniado paralelamente variaciones de éste que han acabado por dar lugar a flujos de aprendizaje alternativos de diferentes características.

El esquema de redes del BEAA se mantiene para todas sus variantes, lo que justifica que sigan considerándose extensiones del mismo modelo. Es en el algoritmo de aprendizaje donde se incluyen nuevos módulos para variar la forma en que el generador, $\mathcal{G}$, aprende sus parámetros y estructura para el siguiente paso del entrenamiento a partir de los ejemplos seleccionados en cada iteración.

3.6.1. BEAA de modificación del espacio latente (BEAAm)

Esta versión del BEAA incluye un módulo adicional en el algoritmo de aprendizaje que modifica las instancias que se seleccionan en cada iteración, cambiando así la perspectiva del espacio latente que captura $\mathcal{G}$ y favoreciendo la generación de nuevos datos que sean fieles a la distribución de los datos etiquetados como normales, previamente aprendida por $\mathcal{D}$.

El BEAAm está pensado para ser utilizado con el detector de anomalías por módulo del vector semilla. Funciona aplicando sobre los valores de los nodos de ruido de las instancias seleccionadas en cada iteración un escalado atractor o repulsor respecto al origen de coordenadas del espacio latente. El signo de la transformación que se aplicará sobre una instancia dada dependerá de la log-verosimilitud que le asigne $\mathcal{D}$: si la instancia se considera verosímil, ya sea en sentido absoluto o comparada con el resto de individuos de su generación, se le aplicará un escalado reductor, acercando su vector semilla al origen de coordenadas mientras que se conserva su dirección. Por el contrario, si la instancia es defectuosa como falsificación de los datos

etiquetados como normales, se le aplicará una transformación repulsora, enviando sus componentes del vector de ruido hacia valores más altos y, por tanto, marcándolo como una anomalía.

Al atraer hacia el centro del espacio latente las instancias consideradas más verosímiles, se favorece su generación al introducir en $\mathcal{G}$ valores para los nodos de ruido correspondientes a un dato normal, es decir, de bajo valor absoluto, de acuerdo a lo explicado en la Sección 3.4.1. Contrariamente, una vez se aprenda el generador inverso $\mathcal{R}$ a partir de $\mathcal{G}$, éste proporcionará valores más próximos a 0 para las componentes del vector semilla en las instancias semejantes a las que se han atraído hacia el origen del espacio latente a lo largo del entrenamiento del generador.

La ventaja principal de esta versión del algoritmo reside en que tolera entrenar con un cociente más alto entre la cardinalidad de la muestra seleccionada en cada generación y el número de individuos que componen la generación, lo que se denomina el ratio de selección, $\frac{N_{\text{selección}}}{N_{\text{generación}}}$. Incluso se puede llegar a prescindir del proceso de selección, dependiendo exclusivamente del reescalado de la muestra seleccionada para guiar la evolución del generador $\mathcal{G}$. Como el muestreo de instancias a partir de $\mathcal{G}$ es uno de los procesos que implica mayores tiempos del entrenamiento, disminuye notablemente el tiempo de ejecución, ya que el BEAAm permite trabajar generando menos ejemplos en cada iteración sin comprometer el número de instancias sobre las que aprende $\mathcal{G}$ en cada paso, que debe ser lo suficientemente grande como proporcionar una buena representación de todas las relaciones de dependencia inherentes a los datos de entrenamiento.

El proceso de modificación de las instancias se realiza de forma proporcional a la log-verosimilitud que les asigna $\mathcal{D}$, por medio de una fórmula matemática que depende de un parámetro ρ, al que se denomina factor de reducción/repulsión. El valor de ρ determina cuánto se alejarán o acercarán como máximo las instancias seleccionadas al origen de coordenadas del espacio latente, por lo que actúa como un regulador de la velocidad de convergencia del algoritmo. Si toma un valor demasiado elevado el aprendizaje convergerá rápidamente, pero el riesgo de sobreentrenamiento será muy alto.

La fórmula que se ha utlizado para la modificación de cada componente $z_i^{(j)}$ del vector de ruido que ha generado una instancia $\mathbf{x}^{(j)}$, cuya log-verosimilitud asignada es $l^{(j)}$, comienza aplicando a la instancia más verosímil de entre las seleccionadas en cada iteración la atracción más fuerte, dividiendo componente a componente por ρ, y a la menos verosímil la repulsión más fuerte, multiplicando individualmente los valores de sus componentes por ρ. Al resto de instancias de la muestra seleccionada se les aplica una repulsión o atracción linealmente proporcional a su log-verosimilitud, dividiendo cada una de sus componentes por f_ρ, calculado como:

$$f_\rho = \frac{1}{\rho} + \frac{l^{(j)} - \text{mín}_j(l^{(j)})}{\text{máx}_j(l^{(j)}) - \text{mín}_j(l^{(j)})} \cdot (\rho - \frac{1}{\rho}).$$

Aunque su rango adecuado de valores se ha determinado experimentalmente a ser $1.001 \leq \rho \geq 1.2$, en general ρ debe estar en equilibrio con el número de iteraciones que se establecen para el entrenamiento, lo que se denomina el número de épocas. Esto se debe a que el total de veces que se aplica el proceso de repulsión y atracción de la muestra seleccionada actúa como una suerte de potencia a la que se eleva

el factor de reducción/repulsión para dar lugar al factor de modificación efectivo, $\rho_{\text{eff}} = \rho^{N_{\text{épocas}}}$, una métrica definida para cuantificar el grado de reescalado del espacio latente. El pseudocódigo que recoge el funcionamiento del BEAAm, marcando en negrita la diferencia con el BEAA genérico (líneas 11 a 13), se puede hallar en el Algoritmo 5.

Algoritmo 5: Algoritmo de aprendizaje del BEAA con modificación de la geometría del espacio latente (BEAAm).

Datos: El conjunto de datos de entrenamiento $\mathcal{S}$, compuesto exclusivamente por instancias etiquetadas como normales

Resultado: Parámetros y estructuras optimizadas de las BNs que componen el BEAA: $\mathcal{D}$, $\mathcal{G}$ y $\mathcal{R}$

1 Aprendizaje de los parámetros y la estructura de $\mathcal{D}$ a partir de $\mathcal{S}$
2 Inicialización de $\mathcal{G}$: **para** *cada nodo de ruido* i *en* $\mathcal{G}$ **hacer**
3 Generar n coeficientes aleatorios β_{ij} para el nodo de ruido i, donde $j = 1, \ldots, n$
4 Conectar el nodo de ruido i a cada nodo atributo j con coeficiente β_{ij}
5 El coeficiente β_{i0} se establece a cero
6 **fin**
7 Entrenamiento de $\mathcal{G}$: **repetir**
8 Muestreo de un lote de $N_{\text{generación}}$ instancias de acuerdo a $\mathcal{G}$
9 Cálculo de la log-verosimilitud de cada individuo de la generación según $\mathcal{D}$
10 Selección de los $N_{\text{selección}}$ mejores individuos por su log-verosimilitud
11 **Cálculo del factor de repulsión f_ρ para cada instancia seleccionada**
12 **Obtención de la muestra de individuos modificados dividiendo cada instancia seleccionada por su f_ρ**
13 **Actualización de $\mathcal{G}$ aprendiendo su estructura y parámetros de la muestra de individuos modificados**
14 **hasta que** *convergencia o número máximo de generaciones*;
15 Muestreo de $N_{\mathcal{R}}$ instancias de $\mathcal{G}$
16 Aprendizaje de los parámetros y la estructura de $\mathcal{R}$ a partir de las $N_{\mathcal{R}}$ instancias generadas

3.6.2. BEAA con generación genética (BEAAg)

Otra variación que se puede incluir en el proceso de aprendizaje del BEAA consiste en implementar un apoyo genético a la generación de instancias por $\mathcal{G}$ durante la fase de entrenamiento. El objetivo de esta propuesta es refinar la calidad de la muestra seleccionada en cada iteración en términos de su log-verosimilitud calculada según $\mathcal{D}$, favoreciendo así la convergencia del proceso de entrenamiento.

Los algoritmos genéticos (Goldberg, 1989), englobados dentro de los algoritmos evolutivos, constituyen un tipo de algoritmos de búsqueda que tratan de imitar la manera en que funciona la selección natural, transformando un conjunto de objetos matemáticos individuales con respecto al tiempo usando operaciones genéticas y actuando de acuerdo al principio Darwiniano de supervivencia del más apto. Los individuos en los algoritmos genéticos son los objetos matemáticos con los que trabaja el problema, y sus genes vienen dados por cada una de sus características, por lo que a menudo los individuos toman la forma de vectores, con cada componente codificando un gen. En líneas generales todos los algoritmos genéticos siguen cinco pasos:

1. Generación aleatoria de una población inicial
2. Cálculo del nivel de aptitud de cada individuo
3. Selección probabilística de los mejores individuos según su aptitud
4. Aplicación de operadores genéticos sobre los individuos seleccionados para componer la siguiente población
5. Iteración del proceso desde el segundo paso hasta que se solucione el problema o se satisfaga una condición de convergencia

Los operadores genéticos más extendidos son el de cruce y el de mutación. El cruce, análogo a la reproducción natural, toma características de dos individuos exitosos (seleccionados dentro de los más aptos) para producir un conjunto de descendientes. Puede darse con o sin reemplazo de los progenitores, y en caso de ser con reemplazo es crucial no cruzar todos los individuos seleccionados para asegurarse de que la siguiente generación no pierde todos los individuos aptos. El operador de mutación también se inspira en la que se produce de forma natural en el genoma de los seres vivos: de forma aleatoria puede cambiar el valor que codifica un gen. En la implementación práctica de los algoritmos evolutivos pueden emplearse uno o ambos operadores, pero siempre se debe establecer una probabilidad de que se apliquen sobre cada individuo o pareja de estos, según sea el caso.

El BEAA que incluye un módulo genético de generación de instancias se denomina BEAA con generación genética, BEAAg, y solo implementa el operador de cruce entre individuos. Tras una generación aleatoria de semillas inicial del modo detallado en la Sección 3.2.2, se evalúa cada individuo. La forma más coherente con el resto del trabajo de obtener una evaluación de cuánto de bueno es cada vector semilla es, naturalmente, generar para cada semilla una instancia de los nodos atributo de acuerdo a $\mathcal{G}$ y calcular su log-verosimilitud. Teniendo en cuenta que el proceso de muestreo de los nodos atributo es probabilístico, el resultado de este paso podría perfeccionarse realizando el proceso de generación un número significativo de veces y finalmente tomar el valor medio de la log-verosimilitud. No obstante, no se ha puesto en práctica este promedio para evitar que el tiempo de ejecución del algoritmo creciese por encima de lo aceptable.

Seguidamente se toman aquellas semillas que dan lugar a instancias más verosímiles para formar la población de individuos sobre la que se aplicará el operador de cruce. La implementación del operador de cruce se realiza intercambiando un subconjunto aleatorio de los atributos de un progenitor, que se reemplazará, con los de otro individuo seleccionado aleatoriamente. El Algoritmo 6 ofrece un detalle del BEAAg, con generación genética de muestras.

Para construir el BEAAg completo, el Algoritmo 6, que corresponde al proceso de generación genética de instancias del BEAAg, se integra en el flujo de aprendizaje genérico del BEAA (detallado en el Algoritmo 2) sustituyendo las instrucciones correspondientes a la generación, evaluado y selección de instancias en el entrenamiento de $\mathcal{G}$, dadas por las líneas 8, 9 y 10 en el Algoritmo 2. El uso del algoritmo genético para la producción de mejores muestras sobre las que entrenar $\mathcal{G}$ en cada iteración del algoritmo supone una enorme mejora de la calidad del conjunto seleccionado sobre el que se entrena el generador, lo que repercute en un avance más rápido de la convergencia de la log-verosimilitud de las instancias generadas en cada época de

Algoritmo 6: Algoritmo de generación genética de instancias del BEAAg.

Datos: Instancias aprendidas de $\mathcal{G}$ y $\mathcal{D}$, número de generaciones, probabilidad de cruce p_{cruce} y los tamaños de generación y selección, $N_{\text{generación}}$ y $N_{\text{selección}}$

Resultado: Un conjunto de tamaño $N_{\text{selección}}$ formado por instanciaciones completas de los nodos de $\mathcal{G}$ (de ruido y de atributo)

Generación de un conjunto de vectores semilla aleatorios según el proceso descrito en la Sección 3.2.2, que constituye la primera generación de individuos
repetir
  Generar para cada individuo de la población una instancia de los nodos atributo que le corresponderían de acuerdo a $\mathcal{G}$
  Calcular la log-verosimilitud según $\mathcal{D}$ del valor de los nodos atributo generado para cada individuo
  Selección de las $N_{\text{selección}}$ mejores instancias de la generación por su log-verosimilitud
  para *Cada individuo* $\mathbf{z}_i$ *de la selección* **hacer**
    Tomar un número aleatorio r de una distribución uniforme entre 0 y 1
    si $r < p_{cruce}$ **entonces**
      Seleccionar un índice aleatorio i_{cruce} del vector semilla al que corresponde el individuo $\mathbf{z}_i$
      Tomar aleatoriamente otro individuo $\mathbf{z}_j$ del conjunto seleccionado
      Formar al descendiente de $\mathbf{z}_i$ copiándolo y tomando el valor de las componentes $(i_{\text{cruce}}, i_{\text{cruce}}+1, \ldots, m)$ del vector $\mathbf{z}_j$
      Sustituir en la población a $\mathbf{z}_i$ por su descendiente
  fin
  Generar un nuevo conjunto de semillas de tamaño $N_{\text{generación}} - N_{\text{selección}}$ para unirlo a la muestra cruzada y formar la población de la siguiente generación
hasta que *Número máximo de generaciones*;
Generar para cada individuo de la población una instancia de los nodos atributo que le corresponderían de acuerdo a $\mathcal{G}$
Calcular la log-verosimilitud según $\mathcal{D}$ del valor de los nodos atributo generado para cada individuo
Selección de las $N_{\text{selección}}$ mejores instancias de la generación por su log-verosimilitud

entrenamiento.

El uso del generador genético de instancias, sin embargo, supone un gran aumento de la potencia computacional necesaria para ejecutar el algoritmo completo, pues implica la realización de numerosos muestreos de instanciaciones de los nodos atributo de $\mathcal{G}$ a partir de los de ruido, lo que comporta un impacto negativo sobre el tiempo de computación total.

3.7. Evaluación del BEAA

El último aspecto práctico que debe tenerse en cuenta a la hora de emplear el detector de anomalías del BEAA se refiere al propio acto de clasificación. Tal y como se menciona en la Sección 1.1, asignar a un dato una puntuación de cuán anómalo es no implica clasificarlo como una anomalía, sino que se debe establecer una frontera

de decisión. En este caso, tal frontera se materializa como un umbral a partir del cual se considera que un dato es anómalo.

En una aplicación real se requiere establecer un umbral de decisión, pero en los experimentos realizados en la siguiente sección no ha sido necesario, pues para evaluar el rendimiento de los detectores de anomalías implementados se ha empleado la métrica del área bajo la curva ROC (Spackman, 1989).

La curva ROC, cuyas siglas corresponden a característica operativa del receptor (*Receiver operating characteristic*), es una representación gráfica de cómo varía el rendimiento de un clasificador binario al variar el umbral de discriminación que separa las dos clases de acuerdo a una puntuación que se le asigna a cada instancia.

La curva ROC se representa en un espacio en el que el eje Y representa el ratio de verdaderos positivos (TPR, *true positive rate*) y el eje X, el ratio de falsos positivos (FPR, *false positive rate*), correspondiéndose cada punto de la curva con un valor para el umbral de detección. La curva se construye a partir de un conjunto de datos de prueba que, en primer lugar, son puntuados. Para obtener el máximo número posible de puntos no redundantes, los valores para el umbral que se seleccionan son aquellos que resultan en el cambio de clase de una instancia. Comenzando desde la puntuación más baja asignada, que corresponde al caso en que toda la muestra se clasifica como anómala y, por tanto, $\text{TPR} = \text{FPR} = 1$, se incrementa el umbral de detección hasta el punto en que toda la muestra se clasifica como normal, situación que equivale a $\text{TPR} = \text{FPR} = 0$.

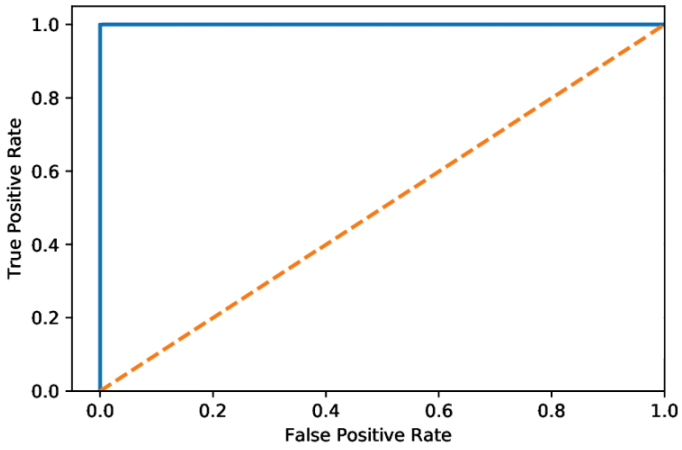


Figura 3.2: Curvas ROC de un clasificador perfecto (azul) y de un clasificador aleatorio (naranja). El primero clasifica todas las instancias correctamente desde el primer punto de la gráfica, que establece el umbral como el valor más bajo de la puntuación obtenido de entre las instancias evaluadas. El segundo tiene, para cualquier valor del umbral, una probabilidad igual de asignar a una instancia una clase u otra.

Un clasificador perfecto tendrá una curva ROC con área 1, es decir, formada por un segmento vertical conectando el punto $(0, 0)$ con el $(0, 1)$ y uno horizontal conectando éste con el $(1, 1)$. Por el contrario, el peor clasificador posible obtendrá una curva ROC diagonal, conectando con una recta los extremos $(0, 0)$ y $(1, 1)$, lo que implica un área bajo la curva de 0.5. En la Figura 3.2 se puede apreciar una representación de ambos casos.

Nótese que el peor clasificador posible no es aquel que tiene área nula bajo la curva ROC (el opuesto al perfecto), dado que clasificar incorrectamente todas las instancias de la muestra equivale a una separación perfecta de los datos, luego en términos de teoría de la información es también un clasificador perfecto, aunque con las clases invertidas. No obstante, el clasificador cuya curva ROC viene dada por una recta diagonal que divide en dos el espacio FPR – TPR corresponde a un clasificador aleatorio con la misma probabilidad de asignar una muestra a una clase o a otra para cualquier valor del umbral, lo que supone la situación de mayor entropía (Shannon, 1948).

Capítulo 4

Resultados experimentales

El prototipo del BEAA que se ha utilizado para todos los experimentos que se presentan en esta sección es el desarrollado en Python usando para la construcción y aprendizaje de BNs la librería PyBNesian, y está disponible en el repositorio jogecodes/BEAA en GitHub. El Cuadro 4.1 recoge un resumen de las especificaciones del modelo que se mantienen para todos los experimentos.

Tipo de red	***Basada en gaussianas condicionadas linealmente***
Algoritmo de aprendizaje de parámetros	*Estimación por máxima verosimilitud*
Algoritmo de aprendizaje de estructura	*PC con test de independencia condicional por información mutua*
Iteraciones para el promedio de las puntuaciones de anomalía	*50*

Cuadro 4.1: Especificaciones fijas de la implementación del BEAA para los experimentos expuestos a lo largo de este capítulo.

Cada experimento particular estará determinado, además de por los valores previamente señalados, por una serie de hiperparámetros que se indicarán en cada caso en forma de tabla, comprendiendo toda la información determinante para el aprendizaje de las redes del modelo y para la tarea de detección de anomalías.

4.1. Conjuntos de datos para la experimentación

Los datos que maneja el BEAA son conexiones: conjuntos de paquetes IP que se envían en una ventana de tiempo continua desde un mismo puerto emisor de una dirección IP de origen al mismo puerto receptor de una dirección IP de destino.

El BEAA fue concebido para aprender a partir del tráfico de una red de computadores su comportamiento normal. Para ello requiere de una muestra significativa de datos procedentes de conexiones no maliciosas entre los equipos de la red y de éstos con el exterior. Igualmente, para efectuar un proceso de prueba y comprobar que los modelos desarrollados detectan con éxito ciberataques a partir de las anomalías en el

tráfico de red, es necesario disponer de conjuntos de datos de prueba que contengan conexiones correspondientes a intrusiones malintencionadas en la red mezcladas con las de naturaleza benigna.

Para entrenar y probar los modelos, se han realizado experimentos con dos conjuntos de datos, uno de ellos dado por un repositorio público de datos de conexiones para el entrenamiento de modelos de detección de intrusiones para ciberseguridad y el segundo, sintetizado en un laboratorio simulado, desarrollado específicamente para el proyecto del que este trabajo forma parte, del que se dispone gracias a la colaboración de *Titanium Industrial Security S.L.*

4.1.1. El *dataset* UNSW-NB15

La evaluación de los sistemas inteligentes de detección de intrusiones en red y, en particular, de los que funcionan por detección de anomalías, sufre a menudo de resultados poco satisfactorios dado que, hasta mediados de la década anterior, los conjuntos de datos disponibles públicamente carecían de entornos de red modernos así como de ataques actualizados (Moustafa y Slay, 2016).

Para resolver estos problemas se desarrolló en la Universidad de Nueva Gales del Sur el conjunto de datos UNSW-NB15 (Moustafa, 2017; Moustafa *et al.*, 2019, 2017; Sarhan *et al.*, 2021), que incluye más de dos millones de conexiones a lo largo de 31 horas de simulación en tres redes con 45 direcciones de equipo diferentes y recopila cerca de 320.000 ataques representativos de las técnicas de invasión más populares, entre los que se encuentran:

- *Fuzzers*: Intentos de suspender un programa o una red introduciendo datos generados aleatoriamente.
- Ataques de análisis y de reconocimiento: Operaciones de exploración de la red, como el escaneo de puertos, cuyo objetivo es ganar información sobre ésta.
- *Backdoors*: Intentos de sobrepasar la seguiridad de un equipo de forma sigilosa para ganar acceso a éste o a los datos que contiene.
- DOS (*Deny Of Service*): Procuran interrumpir temporalmente los servicios de un *host* conectado a Internet, haciendo inaccesibles los servicios de un servidor o una red para sus usuarios.
- *Exploits*: Aprovechamiento de vulnerabilidades conocidas de un sistema operativo o de un componente de software para penetrar la seguridad del sistema.
- Ataques genéricos, que comprenden los que funcionan contra todo tipo de clave cifrada por bloques.
- Código de *shell*: Fragmentos de código transportados por un paquete que pueden vulnerar la seguridad de un equipo en una red.
- *Worms*: Replicación del software atacante para esparcirse por otros ordenadores, a menudo utilizando fallos de seguridad en un ordenador y diseminándose por la red a la que pertenece.

El conjunto de datos original incluye 47 variables descriptivas de cada dato. No obstante, no todas pueden ser utilizadas para el objetivo práctico que persigue este trabajo, ya que muchas de ellas no corresponden a información sobre la conexión o el

paquete, sino que son obtenidas mediante un postprocesado de los datos disponibles que no sería posible efectuar en un escenario real y, por tanto, no permiten demostrar las capacidades de detección de ataques del BEAA exclusivamente a nivel de red.

Las variables atributo que se han retenido para cada instancia del conjunto de datos se recogen en el Cuadro 4.2. Existen variables que se almacenan en dos versiones: una correspondiente a los datos enviados en la dirección origen → destino (*Source to Destination*) y otra correspondiente a las respuestas, que llevan la dirección opuesta, destino → origen (*Destination to Source*). Las primeras vienen precedidas por la letra s y las segundas por la letra d.

Nº	**Nombre**	**Descripción**
1	*dur*	Duración de la conexión
$2_s/2_d$	*sbytes/dbytes*	Bytes enviados
$3_s/3_d$	*sttl/dttl*	Tiempo de vida de los paquetes
$4_s/4_d$	*sloss/dloss*	Paquetes retransmitidos o perdidos
$5_s/5_d$	*sload/dload*	Bits por segundo de la conexión
$6_s/6_d$	*spkts/dpkts*	Número de paquetes enviados en la conexión
$7_s/7_d$	*swin/dwin*	Tamaño de la ventana de recepción
$8_s/8_d$	*stcpb/dtcpb*	Número inicial de secuencia TCP
$9_s/9_d$	*smeansz/dmeansz*	Tamaño medio de los paquetes enviados
$10_s/10_d$	*sjit/djit*	Variabilidad temporal del reloj
$11_s/11_d$	*sintpkt/dintpkt*	Espaciado temporal entre paquetes
12	*tcprtt*	Tiempo de establecimiento de la conexión
13	*synack*	Tiempo de confirmación de la sincronización
14	*ackdat*	Tiempo de confirmación de la conexión
15	*is_sm_ips_ports*	Si los puertos e IPs de origen y destino son iguales vale 1, de lo contrario vale 0
16	*ct_dst_ltm*	Número de conexiones con la misma dirección destino de entre las últimas 100
17	*ct_src_ltm*	Número de conexiones con la misma dirección origen de entre las últimas 100
18	*ct_dst_sport_ltm*	Número de conexiones con la misma dirección destino y el mismo puerto origen de entre las últimas 100
19	*ct_src_dport_ltm*	Número de conexiones con la misma dirección origen y el mismo puerto destino de entre las últimas 100
20	*ct_dst_src*	Número de conexiones con las mismas direcciones origen y destino de entre las últimas 100

Cuadro 4.2: Variables seleccionadas del *dataset* UNSW-NB15.

4.1.2. Los datos sintéticos de *Titanium*

Desde *Titanium Industrial Security S.L.* se ha contribuido a este trabajo con la fabricación de un conjunto de datos para detección de intrusiones en red mediante detección de anomalías. Este *dataset* está compuesto por conexiones entre ordenadores, al igual que el UNSW-NB15, que en este caso fueron simulados con máquinas virtuales.

El conjunto de datos de *Titanium* comprende dos partes; una simulación de la red de ordenadores, que constituye los datos normales sobre los que entrenar el modelo, y una sección que recoge dos tipos de ataque ejecutados *in situ* por un experto, a saber: un escaneo de puertos y un movimiento lateral.

El escaneo de puertos (Bhuyan *et al.*, 2011) constituye un tipo de ataque en el que el incursor explora las diferentes direcciones de protocolos de Internet de un equipo para encontrar sus puntos vulnerables. En sí no son maliciosos, pero suelen preceder a un ataque más serio, por lo que su detección es un tópico importante en ciberseguridad. Los ataques de movimiento lateral (Chen *et al.*, 2018) comprenden un tipo especial de intrusión en la que un invasor de la red, tras haber obtenido acceso a una máquina de ésta, lleva a cabo una misión de reconocimiento de la red para posteriormente extraer credenciales e infiltrarse en otros equipos. De este modo se obtiene un control más profundo de la red, pudiendo hacer uso de privilegios superiores, así como acceder a recursos y datos más valiosos.

Los datos del tráfico de red fueron extraídos a partir de la información de los paquetes enviados y recibidos con un capturador de tráfico. Estos datos "en bruto" no podían ser utilizados para entrenar y probar las maquetas de los modelos detectores de anomalías preparadas, pues no se graban directamente en un formato legible por las librerías de Python utilizadas. Gracias a la colaboración de *Titanium*, los datos fueron agrupados en conexiones siguiendo el estándar marcado por el *dataset* UNSW-NB15, transformados al formato adecuado y etiquetados siguiendo la clasificación binaria en conexiones normales (clase 0) y maliciosas (clase 1).

En total, los datos de laboratorio de *Titanium* incluyen las conexiones establecidas por seis máquinas de diferentes sistemas operativos funcionando cuatro de ellas como usuarios de oficina y dos como servidores de correo, así como de un atacante y de un equipo de registro. Las variables utilizadas para la descripción de los datos fueron seleccionadas para ser compatibles con las empleadas en el UNSW-NB15, y se recogen en el Cuadro 4.3. Como se puede apreciar en la tabla, corresponden a un subconjunto de las variables contenidas en el UNSW-NB15, pero solo está disponible la información sobre los datos que se envían en la conexión.

4.2. Objetivos de la experimentación

Los experimentos que siguen en las posteriores secciones pretenden alcanzar los siguientes objetivos:

1. Demostrar que la aplicación iterativa del esquema de entrenamiento introducido en la Sección 3.2.1 resulta en una mejora continua y consistente de las instancias generadas por $\mathcal{G}$, lo cual debe verse reflejado en una disminución de la log-verosimilitud calculada por el discriminador $\mathcal{D}$, que en primera instancia aprende el comportamiento de los datos considerados normales.

Nº	Nombre	Descripción
1	*dur*	Duración de la conexión
2	*sbytes*	Bytes enviados del origen al destino
3	*sttl*	Tiempo de vida de los paquetes
4	*sload*	Bits por segundo de la conexión
5	*spkts*	Número de paquetes enviados
6	*swin*	Tamaño de la ventana de recepción
7	*stcpb*	Número inicial de secuencia TCP
8	*smeansz*	Tamaño medio de los paquetes enviados
9	*sjit*	Variabilidad temporal del reloj
10	*tcprtt*	Tiempo de establecimiento de la conexión
11	*synack*	Tiempo de confirmación de la sincronización
12	*ackdat*	Tiempo de confirmación de la conexión
13	*is_sm_ips_ports*	Si los puertos e IPs de origen y destino son iguales vale 1, de lo contrario vale 0

Cuadro 4.3: Variables atributo del conjunto de datos sintético de *Titanium*.

2. Probar que una mejora en la log-verosimilitud esperada de los ejemplos generados por $\mathcal{G}$ está directamente correlacionada con una mejor separación entre anomalías y datos normales al aplicar cualquiera de los métodos de clasificación de instancias anómalas presentados en la Sección 3.4. Hay que tener en cuenta que un sobreentrenamiento del modelo puede llevar también al colapso de las CPDs Gaussianas multivariantes en distribuciones Gaussianas univariantes, lo que hace que disminuya dramáticamente la capacidad de detección de anomalías del modelo.

El colapso de las CPDs multivariantes es un tema que aparecerá recurrentemente a lo largo de este capítulo, siendo especialmente relevante la situación en que, tras ejecutar el entrenamiento de $\mathcal{G}$, las CPDs de los nodos atributo carecen de dependencias condicionales porque todos los arcos han desaparecido. En este caso, al aprender $\mathcal{R}$ de una muestra generada por $\mathcal{G}$, no se extraerán relaciones estadísticas entre las variables atributo en las expresiones de las CPDs de los nodos de ruido, y de hacerlo, no serán significativas, pues no corresponderán a un comportamiento característico de los datos normales capturado por $\mathcal{G}$ durante su aprendizaje.

La meta final del trabajo es comprobar que el modelo propuesto provee una solución al problema de detección de intrusiones en ciberseguiridad. Para poder justificar que el funcionamiento del modelo es satisfactorio, su rendimiento debe ser superior al que tendría un detector de anomalías usando una BN simple (véase la Sección 2.3.1), pues proporciona una interpretabilidad más baja que éste.

4.3. Experimentos sobre el conjunto de datos UNSW-NB15

4.3.1. Preparación de los datos

Todos los ensayos presentados en esta sección vienen precedidos de la preparación de los datos del UNSW-NB15. Para poder entrenar el modelo se requiere extraer una fracción de las conexiones recogidas en el conjunto de datos constituido exclusivamente por instancias normales, y para probar de forma justa el modelo desarrollado, se necesita de un subconjunto de los datos originales que no hayan sido nunca antes vistos por el modelo entrenado y que contengan una representación equilibrada de conexiones benignas y maliciosas.

La división del conjunto de datos que se ha empleado para el entrenamiento y prueba de los modelos se ha mantenido a lo largo de todos los experimentos para facilitar su equiparación. Tal configuración consiste en el uso de un 90 % de los datos etiquetados como normales para el entrenamiento del modelo y del 10 % de datos normales restantes junto con todos los datos clasificados como una de las categorías de ataque, englobados bajo la clase de amenaza, para la prueba del modelo.

Para el correcto funcionamiento del modelo, es pertinente también normalizar los datos de las conexiones antes de alimentar el sistema con ellos. La técnica de normalización escogida para los experimentos presentados es la de estandarización, por medio de la cual se hace cero la media de las variables atributo de los datos y se escalan las instancias para que la desviación típica de cada variable sea uno (véase la Sección 3.2.1). Esto se consigue calculando para cada variable, X_i, su media muestral, u_i, y su desviación típica muestral, s_i, quedando así cada atributo reescalado, $\tilde{X}_i$, como:

$$\tilde{X}_i = \frac{X_i - u_i}{s_i}$$

La elección de la estandarización como proceso normalizador constituye una elección coherente teniendo en cuenta que el tipo de BNs que integra el modelo emplea Gaussianas condicionadas linealmente como CPDs de sus variables. Si bien la media y la desviación típica que emplea la estandarización se calculan sobre el conjunto de entrenamiento, el normalizado de prueba debe seguir las mismas reglas geométricas, pues de lo contrario el modelo trabajará sobre la asunción errónea de que los datos están sujetos a la misma escala, invalidando así sus cálculos.

El preprocesamiento de los datos, englobando tanto su separación en subconjuntos de entrenamiento y prueba como su normalización, se ha realizado empleando la librería *SciKit-Learn* (Pedregosa *et al.*, 2011).

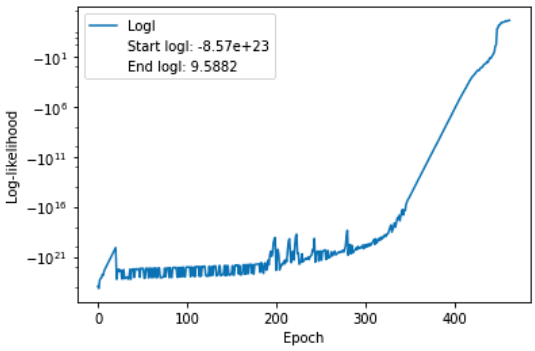
4.3.2. Resultados experimentales

A la hora de evaluar el rendimiento del BEAA sobre los datos del *dataset* público UNSW-NB15 se han probado numerosos hiperparámetros para el modelo, variando, entre otros, el número de épocas para el entrenamiento (referido en las figuras como *Epochs*), el tamaño de cada muestra generada por $\mathcal{G}$ durante el entrenamiento, $N_{\text{generación}}$ o el número de instancias seleccionadas en cada generación como las más representativas de la distribución a la que se desea llegar, $N_{\text{selección}}$. Cada cambio en un parámetro tiene un efecto distinto sobre el resultado del entrenamiento y en la

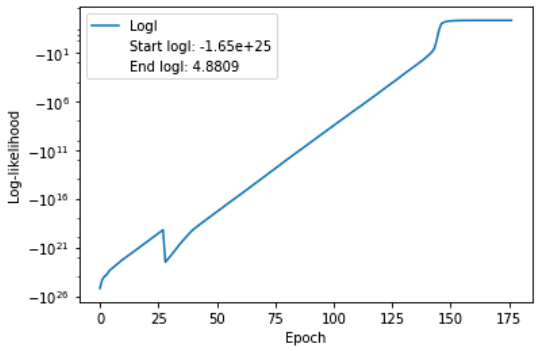
posterior tarea de clasificación de datos en normales o anómalos. En el Cuadro 4.5 se incluye un resumen de los parámetros alterables en el modelo, así como de su impacto en el funcionamiento del BEAA y sus valores idóneos, ambos determinados experimentalmente.

Además de para la caracterización de los parámetros del modelo, los experimentos sobre el set de datos UNSW-NB15 han servido para probar los objetivos enumerados en la Sección 4.2. En primer lugar, todos los modelos entrenados con éxito exhiben un comportamiento similar en cuanto a la convergencia de la log-verosimilitud media de acuerdo a $\mathcal{D}$ de las muestras generadas por $\mathcal{G}$ hacia valores que denotan una mayor fidelidad en la producción.

Dos ejemplos de convergencia de la log-verosimilitud para dos modelos de diferente configuración se exponen en la Figura 4.1. En cada figura viene representada en escala logarítmica la evolución del valor medio de la log-verosimilitud calculado sobre todas las instancias de la muestra generada por $\mathcal{G}$ en cada iteración del proceso de aprendizaje adversarial del BEAA. Los valores de la log-verosimilitud inicial son en ambos casos muy elevados, dado que corresponden a la generación aleatoria que se da en la primera iteración del algoritmo de aprendizaje, con la red $\mathcal{G}$ inicializada de forma que cada nodo atributo esté completamente conectado a todos los nodos de ruido con coeficientes aleatorios para las Gaussianas de la CPD.



(a) Caso con ratio $\frac{N_{\text{selección}}}{N_{\text{generación}}} = 0.1$.



(b) Caso con ratio $\frac{N_{\text{selección}}}{N_{\text{generación}}} = 0.001$.

Figura 4.1: Evolución de la log-verosimilitud media calculada mediante $\mathcal{D}$ de las muestras generadas por $\mathcal{G}$ para dos ejemplos de BEAA con diferentes parámetros. El eje y se presenta en escala logarítmica para facilitar la apreciación de la tendencia de la curva.

En la Subfigura 4.1a se puede observar cómo la log-verosimilitud media según $\mathcal{D}$ de las muestras producidas por $\mathcal{G}$ en cada generación, incluso partiendo de un valor inicial muy negativo, logra converger en la iteración número 463 a 9.59. Este valor da a entender que la red generadora desarrolla un gran poder falsificador de instancias coherentes con la distribución de datos normales, dado que el promedio de la log-verosimilitud de acuerdo a $\mathcal{D}$ tomado sobre todas las instancias etiquetadas como normales del conjunto de datos de prueba vale 5.95. En contraste, el mismo promedio tomado sobre las instancias correspondientes a las conexiones maliciosas del conjunto de datos de prueba toma el valor -413.51.

El entrenamiento del modelo cuyo desarrollo se muestra en la Subfigura 4.1a, en lugar de abarcar un número fijo de épocas, implementa una comprobación de la va-

Parámetro	Efecto de su variación y valores idóneos
Número de nodos de ruido, m	A mayor número de nodos de ruido, más relaciones se extraen, por lo que disminuye la cardinalidad de éstas (el número de variables que implican) y aumenta la susceptibilidad al colapso de los nodos atributo de $\mathcal{G}$. Se ha determinado $0.25n \leq m \leq 0.75n$ como la mejor elección, donde n es la dimensión del espacio de datos.
Parámetros para la generación aleatoria de semillas, μ_{seed} y σ_{seed}	Los valores de la media y la desviación típica para la generación de semillas en el proceso de entrenamiento (véase la Sección 3.2.2) alteran la convergencia del algoritmo. Un valor cercano a cero de ambas hará que converja más rápidamente, pero simplificará las relaciones de dependencia en los nodos de ruido de los nodos atributo de $\mathcal{G}$, lo que puede propiciar su colapso durante el entrenamiento. Un valor más alto de los parámetros de generación de semillas fomentará el aprendizaje de relaciones estadísticas más ricas para la detección de anomalías. Se ha determinado que $\mu_{\text{seed}} \sim 1$ junto con $\sigma_{\text{seed}} = \frac{\mu_{\text{seed}}}{2}$ proporciona buenos resultados experimentales.
Tamaño de cada generación en el entrenamiento de $\mathcal{G}$, $N_{\text{generación}}$	Cuanto más grande sea la muestra que se produce en cada generación, mayor será la exploración del modelo y, por tanto, más ricas las relaciones extraídas. Además, favorece la convergencia, pues aumenta la probabilidad de generar instancias coherentes con la distribución objetivo. La generación de instancias masiva tiene un impacto muy negativo sobre el tiempo de ejecución del algoritmo, no obstante.
Tamaño de la muestra seleccionada en cada generación, $N_{\text{selección}}$	Determina el número de instancias sobre las que entrenará el generador en cada iteración. Cuanto más grande sea este número más variadas serán las relaciones extraídas, a costa de la calidad media (en términos de log-verosimilitud de acuerdo a $\mathcal{D}$) de las muestras sobre las que aprende $\mathcal{G}$, lo que repercute negativamente en la velocidad de convergencia. El tamaño adecuado de $N_{\text{selección}}$ depende del $N_{\text{generación}}$ escogido, siendo el ratio de selección adecuado $\frac{N_{\text{selección}}}{N_{\text{generación}}} \lesssim 0.2$.
Número de iteraciones de entrenamiento de $\mathcal{G}$ (épocas)	A mayor número de épocas, más se refina la producción de instancias verosímiles por $\mathcal{G}$, indicando la maduración de las relaciones de dependencia entre las variables atributo capturadas. El número de épocas también está directamente relacionado con la simplificación (y posible colapso) de las relaciones extraídas, así como con un mayor tiempo de entrenamiento. El valor idóneo del número de épocas depende fuertemente del ratio de selección, necesitándose más épocas (hasta 1000) cuando este ratio es cercano a 1 y pudiendo funcionar con algunas decenas de épocas cuando el cociente es bajo.

Cuadro 4.5: Parámetros del BEAA y su efecto, determinado experimentalmente sobre el *dataset* UNSW-NB15.

riación del valor medio de la log-verosimilitud en cada iteración. Cuando la diferencia entre la log-verosimilitud media de una iteración y la anterior es menor que una cierta tolerancia (tol), que en este ejemplo vale tol $= 0.001$, se detiene el entrenamiento. Tal modelo genera por cada iteración 50000 instancias, de las cuales selecciona 5000, dando lugar a un ratio de selección de $\frac{N_{\text{selección}}}{N_{\text{generación}}} = 0.1$. El elevado cociente entre el tamaño de la muestra seleccionada y la generada justifica que la convergencia de la log-verosimilitud tarde del orden de 500 iteraciones, si bien estas son razonablemente rápidas, con el entrenamiento suponiendo menos de un minuto de computación.

La gráfica de la Subfigura 4.1b proviene de un modelo con un número de instancias generadas por iteración 100 veces más alto, dando lugar a un ratio de selección $\frac{N_{\text{selección}}}{N_{\text{generación}}} = 0.001$. Así se explica que la log-verosimilitud media haya necesitado muchas menos épocas para converger, pues en cada iteración el generador entrena sobre una muestra más refinada. No obstante, es sensiblemente más lento que el modelo anteriormente expuesto, llegando el entrenamiento a durar hasta 25 minutos.

La tendencia global de la evolución de la log-verosimilitud es creciente, probando así el primer punto expuesto en la Sección 4.2. A pesar de ello, como es propio de los comportamientos adversariales en competiciones de suma cero (véase la Sección 1.3), pueden observarse oscilaciones en el crecimiento. Dichas oscilaciones son especialmente prominentes en el modelo de mayor ratio de selección, pues el uso de muestras de menor calidad para el aprendizaje de $\mathcal{G}$ en cada iteración lleva a un entrenamiento más inestable. En cambio, en el modelo que emplea un ratio menor la tendencia a crecer es sensiblemente más monótona, reflejando que la muestra seleccionada tiene una probabilidad más alta de contener ejemplos de alta verosimilitud, tornando más consistente la mejora del BEAA en cada iteración.

El poder de clasificación que demuestra el BEAA debe ponerse en comparación con el que tendría un sistema detector de anomalías más interpretable y ligero, como el que podría representar un clasificador formado por una única BN que entrena sobre una muestra de instancias normales y emplea la log-verosimilitud de cada nuevo ejemplo para clasificarlo como anómalo o no. En tal sentido, al ejecutar una serie de pruebas con la división de datos del UNSW-NB15 anteriormente introducida y sobre la que trabajan todos los modelos del BEAA probados, se determinó que un clasificador de BN simple actuando por log-verosimilitud puntúa con un área bajo la curva ROC de $\text{AUC}_{\text{BN}} = 0.9803$. Para ser considerado una opción viable, el BEAA debería puntuar mejor que este umbral en un test estadístico, pues ofrece menor interpretabilidad que una única BN al tiempo que emplea más recursos computacionales.

En los experimentos estadísticos conducidos mediante numerosos aprendizajes y pruebas del BEAA con parámetros en el rango óptimo calculado empíricamente se ha podido comprobar que el entrenamiento del BEAA, al depender de una inicialización aleatoria e incluir pasos estocásticos a lo largo de todo el proceso, no siempre resulta en un clasificador de rendimiento adecuado. El problema más evidente se ha concretado a ser el colapso de las CPDs multivariantes en univariantes (previamente mencionado en la Sección 4.2), que causa el fracaso de numerosos ejemplares en entrenamiento del BEAA. Aún así, de entre los modelos que aprenden con éxito a detectar anomalías se han obtenido instancias del BEAA con rendimiento significativamente superior a la cota $\text{AUC}_{\text{BN}} = 0.9803$ que establece el uso de una única BN usando como puntuación de anomalía la log-verosimilitud de las instancias de entrada.

En tal sentido destaca particularmente un modelo que se entrenó con las características que detalla el Cuadro 4.6, y cuya curva ROC, bajo la cual el área es de $\text{AUC}_{\text{BN}} = 0.9866$, se presenta en la Figura 4.2. Esta instancia del BEAA, que constituye el modelo más exitoso de entre los aprendidos, emplea la estrategia de modificación del espacio latente, por lo que es propiamente un BEAAm. Esto le permite prescindir del proceso de selección de instancias en cada generación, necesitando de menos tamaño poblacional para proveer al generador de una base muestral lo suficientemente grande para su aprendizaje.

Parámetro	**Valor que toma**
Tipo de BEAA	BEAA con modificación del espacio latente, BEAAm
Número de nodos de ruido	$m = 0.5n$, la mitad de nodos de ruido que de nodos atributo
Parámetros para la generación aleatoria de semillas	$\mu_{\text{seed}} = 0.75$ con $\sigma_{\text{seed}} = \frac{\mu_{\text{seed}}}{2}$
Tamaño de generación y selección	$N_{\text{generación}} = N_{\text{selección}} = 10000$, indicando que se prescinde de la selección de instancias
Número de iteraciones de entrenamiento	100 épocas
Factor de repulsión/atracción	$\rho = 1.01$

Cuadro 4.6: Especificaciones de la implementación del BEAA que ha proporcionado los mejores resultados.

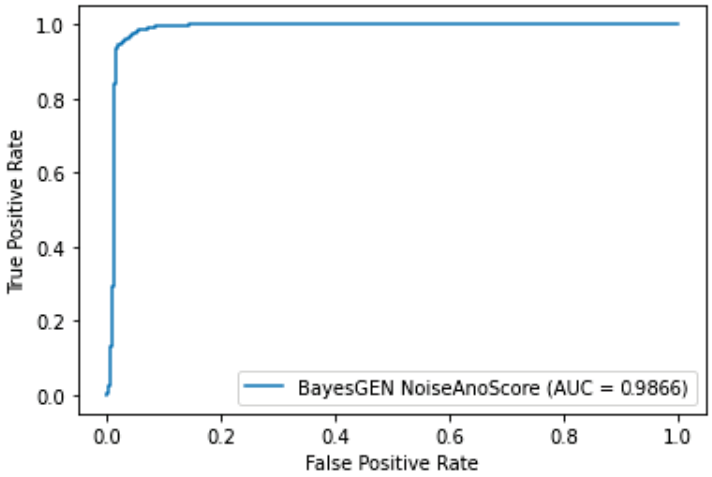


Figura 4.2: Curva ROC del mejor modelo de BEAA aprendido sobre el conjunto de datos UNSW-NB15.

El valor exacto del área bajo la curva ROC del modelo presentado usando el método de detección de anomalías por módulo del vector semilla es de $\text{AUC}_{\text{BEAAm}} = 0.9866$, indicando un poder de separación entre datos normales y anomalías muy próximo al que tendría un clasificador ideal. En particular, se puede comprobar que la curva de la Figura 4.2 logra alcanzar un $\text{TPR} = 1$ con $\text{FPR} < 0.2$, lo que implica que se podría

tomar un umbral tal que el sistema logre distinguir todas las anomalías del conjunto de datos de prueba causando menos de un $20\,\%$ de falsos positivos. Este resultado, aunque produce más falsos positivos de lo tolerable en el área de ciberseguridad, implica que el modelo sí que podría constituir una parte útil de un sistema de detección de intrusiones más complejo, por ejemplo, basado en un *ensemble* de clasificadores que emplee varios modelos apilados para proveer una separación óptima entre clases reduciendo al mínimo los falsos positivos, como en los trabajos de Ucar *et al.* (2021) y de Rajagopal *et al.* (2020).

La presencia de inestabilidades en el algoritmo de aprendizaje relacionadas con el colapso de las Gaussianas multivariantes, que ha quedado a resolver en futuras versiones del modelo, ha hecho imposible la tarea de probar estadísticamente que, en un gran número de experimentos, el modelo propuesto obtiene un rendimiento consistentemente mejor que el de la BN discriminadora, $\mathcal{D}$, actuando como detector de anomalías. A pesar de esto, como en iteraciones aisladas el BEAA ha logrado puntuaciones de área bajo la curva ROC (AUC) bien por encima del umbral que marca el uso de una única BN para la detección de anomalías, se puede albergar la esperanza de que la continuación del refinamiento del BEAA pueda dar lugar a un modelo altamente competitivo.

4.3.3. Interpretabilidad de los resultados sobre el *dataset* UNSW-NB15

La posibilidad de interpretar los resultados obtenidos supone la ventaja que permite defender que el algoritmo presentado en este trabajo es superior a otros modelos más potentes pero de caja negra, como las GANs basadas en redes neuronales. En comparación con una BN, el BEAA es tan solo parcialmente interpretable, pues pierde parte de la transparencia inherente de éstas. Aún así, el BEAA ha sido diseñado para proporcionar información relativa a los resultados que provee para complementar su funcionamiento. Estas explicaciones, si bien requieren de conocimiento experto para ser adecuadamente descifradas, son una garantía del correcto funcionamiento del modelo, lo hacen robusto además de confiable y habilitan su uso en ámbitos en los que la toma de decisiones usando un sistema de inteligencia artificial puede desencadenar consecuencias perjudiciales para una persona, un grupo de éstas o una compañía.

El elemento interpretable del BEAA se halla en las expresiones de las CPDs de los nodos de ruido del generador inverso, $\mathcal{R}$. En esta red, los arcos siempre toman la dirección que va de los nodos atributo, representativos de las características que describen las instancias del espacio de datos, a los nodos de ruido. Al haber escogido que las CPDs que caracterizan los nodos de las BNs que integran BEAA sean Gaussianas condicionadas linealmente, cada variable de ruido Z_i estará gobernada por una expresión del siguiente tipo:

$$P(Z_i = z_i|\mathrm{Pa}_i^{\mathcal{R}}) = \mathcal{N}(z_i; \beta_0 + \sum_{x_j \in \mathrm{Pa}_i^{\mathcal{R}}} \beta_{ij} x_j, \sigma^2),$$

en la que $\mathrm{Pa}_i^{\mathcal{R}}$ denota el conjunto de nodos atributo que son los padres de Z_i según el grafo de $\mathcal{R}$. En cada coeficiente β_{ij} se codifica cuánto de relevante es el atributo X_j para la CPD de Z_i, de modo que ante la presencia de un dato anómalo se puede deducir cuáles han sido las variables atributo involucradas y cómo se han combinado

para dar lugar a la irregularidad. Tanto si la puntuación de detección de anomalía en uso es la que se basa en el módulo del vector semilla como si es la que emplea el error de reconstrucción se puede obtener información adicional sobre la causa de la anomalía observando qué nodos de ruido resaltan y atendiendo a las CPDs que los gobiernan.

Para ilustrar cómo se interpretan los resultados del BEAA se extraerán una serie de ejemplos de clasificación utilizado un modelo aprendido sobre un subconjunto simplificado de los datos del UNSW-NB15 para facilitar la explicación del proceso de interpretación. El *dataset* simplificado cuenta tan solo con las variables indicadas en el Cuadro 4.7.

Nº	**Nombre**	**Descripción**
1	*dur*	Duración de la conexión
2	*sbytes*	Bytes enviados del origen al destino
3	*sloss*	Paquetes retransmitidos o perdidos
4	*sload*	Bits por segundo de la conexión
5	*spkts*	Número de paquetes enviados
6	*smeansz*	Tamaño medio de los paquetes enviados
7	*sintpkt*	Espaciado temporal entre paquetes

Cuadro 4.7: Subconjunto reducido de variables atributo del conjunto de datos UNSW-NB15 para experimentos de interpretabilidad.

Asimismo, el modelo de BEAA empleado para la detección de anomalías en este apartado se materializa con los parámetros que recoge el Cuadro 4.8.

Parámetro	**Valor que toma**
Tipo de BEAA	BEAA genérico
Número de nodos de ruido	$m \sim 0.5n$, con 4 nodos de ruido para 7 nodos atributo
Parámetros para la generación aleatoria de semillas	$\mu_{\text{seed}} = 1$ con $\sigma_{\text{seed}} = \frac{\mu_{\text{seed}}}{2}$
Tamaño de generación y selección	$N_{\text{generación}} = 100000$ y $N_{\text{selección}} = 10000$, quedando un ratio de selección $\frac{N_{\text{selección}}}{N_{\text{generación}}} = 0.1$
Tolerancia en la convergencia de la log-verosimilitud	$\text{tol} = 0.005$

Cuadro 4.8: Parámetros de la implementación del BEAA empleada para los experimentos de interpretabilidad.

En la Figura 4.3 se incluyen dos diagramas correspondientes, cada uno, a un ejemplo de anomalía detectada por medio de la puntuación del módulo del vector semilla, dado que es el método que ha proporcionado la mejor separación de muestras para el *dataset* reducido sobre el que se ha realizado el ejercicio de interpretabilidad. Las

gráficas muestran qué valor absoluto recibe cada nodo de ruido de $\mathcal{R}$ tras muestrear ancestralmente dado un dato anómalo (un ataque) como evidencia para instanciar sus nodos atributo. Esto permite determinar cuál es el nodo de ruido más relevante para el cálculo de la puntuación de anomalía en cada caso y, por tanto, cuál es la relación estadística de entre las codificadas en las CPDs de los nodos de ruido que no se satisface para el dato evaluado. La contribución de cada nodo de ruido al módulo del vector semilla se representa a través de su valor absoluto y en escala logarítmica para facilitar la visualización.

(a) Vector semilla calculado para un ataque *Fuzzer*.

(b) Vector semilla calculado para un ataque DoS.

Figura 4.3: Comparación de los valores absolutos de las variables de ruido generadas por $\mathcal{R}$ dadas dos instancias correspondientes a ciberataques. Los valores se presentan escalados logarítmicamente.

En la Subfigura 4.3a se puede apreciar que el nodo de ruido que comporta la mayor contribución para el módulo del vector semilla es el denotado como *noise2*. Visualmente cabe pensar que el nodo denominado *noise4* también contribuye significativamente, pero no se debe olvidar que el gráfico presenta los valores en escala logarítmica, por lo que la importancia relativa de *noise2* es mayor de lo que aparenta. De hecho, numéricamente, *noise2* vale cerca del doble de *noise4* para esta instancia determinada, lo que implica una contribución cuatro veces mayor que la de *noise4* al módulo Euclídeo del vector semilla. Una vez justificada la toma de *noise2* como la componente dominante se debe estudiar la CPD que lo gobierna, que toma la forma de la siguiente Gaussiana:

$$P(\textit{noise2} = z_2 | \textit{dur}, \textit{sload}) = \mathcal{N}(\mu = -0.081 + 4.025 \cdot \textit{dur} + 3.474 \cdot \textit{sload}, \sigma = 0.052),$$

donde z_2 representa el valor asignado al nodo *noise2* en la instancia obtenida de los nodos de ruido para el dato anómalo introducido en $\mathcal{R}$ y los coeficientes que acompañan a *dur* y a *sload* son, respectivamente y de acuerdo a la notación introducida en la Sección 3.1.2, $\beta_{2,\textit{dur}}$ y $\beta_{2,\textit{sload}}$.

Es directo comprobar que, dada la anterior fórmula, un valor anormalmente alto de este nodo de ruido (en el ejemplo dado recibe $\textit{noise2}_2 = 133.68$) se da cuando

se produce una discordancia entre el tiempo de duración de la conexión, *dur*, y el número de bis por segundo que se transfieren entre los equipos, *sload*. Teniendo en cuenta que cuando en un dato original una variable toma un valor alrededor de su media valdrá cerca de 0 tras aplicar la normalización, para que la relación estadística codificada en la CPD de *noise2* no diverja pueden seguirse dos tendencias:

- Al disminuir *dur* por debajo de su media y, por tanto, tomar un valor negativo tras su normalización, debe aumentar *sload* por encima de la suya en una proporción cercana a la que se obtiene de calcular el cociente $\frac{\beta_{2,dur}}{\beta_{2,sload}}$.
- Alternativamente, si *dur* subiera, para una conexión de prueba, por encima de lo que entiende el modelo como normal, *sload* debería responder subiendo en la proporción que marca la división entre sus cocientes.

Juntos, estos hechos que se derivan del desglose de la CPD de *noise2*, implican que *dur* y *sload* deben estar inversamente relacionados, manteniendo estable el equilibrio en la escala que especifican sus coeficientes $\beta_{2,dur}$ y $\beta_{2,sload}$. En general, cualquier conexión saludable entre dos máquinas debería satisfacer esta premisa, pues aunque la duración de una conexión dependerá también de otros factores, es trivial comprobar que, en efecto, existe una proporción inversa entre el número de bits enviados por segundo y la duración de la conexión, que se hace evidente cuando se mantienen constantes el resto de variables.

En particular, el dato cuyos valores del vector semilla se representan en la Subfigura 4.3a y sobre el que se está realizando este ejercicio de interpretabilidad corresponde a un ataque de *fuzzing* (Sutton *et al.*, 2007). Este tipo de intrusiones tiene el objetivo de encontrar vulnerabilidades en una aplicación o sistema del que se tiene poca o ninguna información a priori, para lo cual se le alimenta con entradas construidas a propósito introduciendo ruido en entradas normales del programa para intentar que falle su ejecución (Li *et al.*, 2018b). Cabe concluir que en un ataque de *fuzzing* podría verse aumentada en gran medida la tasa de envío de bits por el equipo de origen (el atacante), pues debe enviar numerosos datos de entrada, sin que la duración disminuya en concordancia, ya que si se encuentra una vulnerabilidad se detendrá la ejecución del programa en el equipo destino. Este es precisamente el escenario que se da en el caso en estudio, repitiéndose el mismo patrón en todos los ejemplos de *fuzzers* del conjunto de datos de prueba.

Nótese que el BEAA sólo captura relaciones lineales, pues las BN que lo constituyen emplean Gaussianas condicionadas linealmente para representar las CPDs de sus nodos. Esto implica que únicamente se puede garantizar que las relaciones entre variables que codifican los nodos de ruido de $\mathcal{R}$ son una aproximación a primer orden de las que subyacen al conjunto de datos de entrenamiento, compuesto por instancias normales. Tal aproximación, además, sólo funciona en un entorno del origen del espacio de datos, que equivale al punto en que todos los datos toman como valor su promedio en el conjunto de entrenamiento, ya que es esta la región geométrica sobre la que se aprenden los parámetros y estructura de las redes del BEAA.

El segundo estudio de interpretabilidad, recogido en la Subfigura 4.3b, corresponde a un ataque de tipo DoS, siglas de *Denial of Service*: denegación de servicio. Los ataques DoS (Loukas y Öke, 2010) engloban cualquier tipo de intento de evitar que los recursos o servicios de un sistema o red puedan ser utilizados por sus usuarios, lo que a menudo se consigue agotando la capacidad de respuesta del sistema sobre-

cargando la banda ancha de la que dispone para el tráfico de datos. En este ejemplo se puede apreciar que, indiscutiblemente, es la componente *noise3* la responsable de la mayor parte del valor del módulo del vector semilla. La CPD que gobierna el comportamiento de este nodo de ruido se escribe como sigue:

$$P(\textit{noise3} = z_3 | \textit{sintpkt}, \textit{spkts}, \textit{smeansz}) = \\ \mathcal{N}(\mu = -1.127 - 1.663 \cdot \textit{sintpkt} - 2.581 \cdot \textit{spkts} + 4 \cdot \textit{smeansz}, \sigma = 0.147),$$

En este caso la expresión de la CPD conecta las variables correspondientes al espaciado temporal que se deja entre paquetes, *sintpkt*, número de paquetes enviados en la conexión, *spkts*, y tamaño medio de estos, *smeansz*. Las relaciones con más de dos variables son más complicadas de caracterizar. Concretamente, para tres variables, la región geométrica que las equilibra corresponde a una superficie, por lo que es más difícil de visualizar. No obstante, se puede extraer de la relación una idea de la proporción en que se deben encontrar las variables que involucra.

Concretamente, el incumplimiento de la relación anterior se produce porque el número de paquetes enviados es mucho mayor de lo que el modelo espera dado su tamaño medio. Esto se puede interpretar como un indicio de que, con intención de sobrecargar el sistema, se le están enviando numerosas solicitudes, así como otros paquetes ligeros que contienen poca información, siendo precisamente este el modo en que actúa un ataque DoS.

Las estructuras de las BNs que integran la instancia del BEAA usada para este ejercicio de interpretabilidad se recogen en la Figura A.1 del Apéndice A.

4.4. Experimentos sobre los datos sintéticos de *Titanium*

4.4.1. Preparación de los datos

El BEAA está pensado para operar sobre datos estandarizados, por lo que al igual que para el conjunto de datos UNSW-NB15 se debe aplicar el proceso de normalizado explicado en la Sección 4.3.1 para los datos sintéticos proporcionados por *Titanium*.

Este conjunto de datos no requiere someterse a una división para aprendizaje y prueba, pues viene de serie separado a propósito en dos partes: una para el entrenamiento, que contiene exclusivamente muestras normales, y una destinada para experimentación, que recoge tráfico de red normal junto con los ataques de escaneo de puertos y movimiento lateral ejecutados en vivo por un experto sobre la red simulada.

La sección de entrenamiento del *dataset* de Titanium contiene 8924 conexiones en total, por lo que supone una base mucho más escasa sobre la que aprender el BEAA comparada con la que proveía el conjunto UNSW-NB15. Pese a esto, su valor como ejemplo de aplicación práctica del BEAA en un escenario realista lo convierte en una de las piezas más importantes de este trabajo. En cuanto a la sección del conjunto de datos preparada para probar los modelos entrenados, esta contiene 11846 datos de conexiones, de las cuales un 46 % corresponden a entradas maliciosas, que forman parte de un ataque ejecutado por el experto en ciberseguridad. Aunque es extraño

disponer de un conjunto de datos de entrenamiento más pequeño que el de prueba, debe tenerse en cuenta que el de entrenamiento es más extenso a nivel temporal, ya que contiene exclusivamente instancias normales, mientras que algunas secciones de los ataques ejecutados en el conjunto de datos de entrenamiento implican el establecimiento de numerosas conexiones maliciosas.

Del mismo modo en que se realizó para los experimentos sobre el UNSW-NB15, el proceso de normalización comienza ejecutándose sobre el conjunto de datos para el entrenamiento del modelo y, seguidamente, sobre los datos de prueba empleando las reglas geométricas de reescalado de los datos calculadas para el primero. La implementación de la normalización se materializa, de nuevo, mediante la libreria *SciKit-Learn* (Pedregosa *et al.*, 2011).

4.4.2. Resultados experimentales

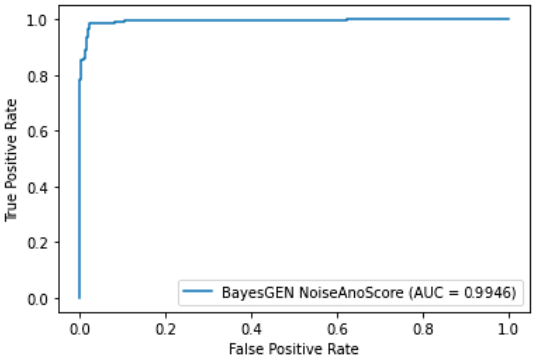
El rendimiento del BEAA en el escenario realista suministrado por *Titanium* se ha evaluado aprendiendo y poniendo a prueba instancias del modelo con diferentes estructuras de parámetros hasta encontrar la configuración ópima para el tratamiento del problema.

En este apartado, como el impacto que tiene cada parámetro del BEAA en el funcionamiento del modelo ya se caracterizó en la Sección 4.3.2, que incluye también una descripción detallada de la capacidad de interpretabilidad del sistema, no se han repetido estos estudios, sino que directamente se han extraído conclusiones de la aplicación del BEAA al conjunto de datos sintético y de su rendimiento obtenido sobre este.

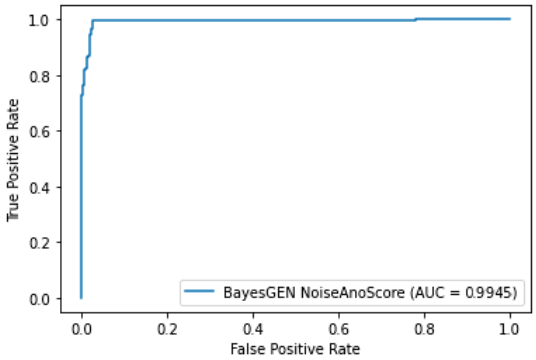
Los resultados obtenidos sobre los datos de *Titanium* reflejan que el BEAA, en comparación con un detector de anomalías compuesto por una simple BN aprendida sobre un conjunto de datos normales y clasificando las instancias de entrada según su log-verosimilitud, es un modelo pesado y se beneficia de un espacio de variables complejo y de alta dimensionalidad, pues en ningún caso experimental sobre este *dataset* se ha conseguido superar el umbral de puntuación AUC que marca el detector de anomalías de BN por log-verosimilitud: $\text{AUC}_{\text{BN}} = 0.9954$. Aún así, se ha conseguido obtener un rendimiento similar de forma muy consistente con numerosas instancias del BEAA, demostrando que el modelo también funciona a nivel práctico en escenarios realistas. De hecho, proporciona un ratio de falsos positivos aún más bajo que en el conjunto de datos UNSW-NB15 para el punto en que se clasifican correctamente todos los ataques como anomalías, llegando a obtenerse un $\text{TPR} = 1$ con $\text{FPR} < 0.1$.

En la Figura 4.4 se muestran las curvas ROC de las dos mejores instancias del BEAA obtenidas para este problema, con sus parámetros especificados en los Cuadros 4.9 y 4.10, respectivamente.

Ambos modelos ofrecen una puntuación AUC casi perfecta, llegando a señalar la totalidad de las conexiones anómalas con menos de un $5\,\%$ de falsos positivos en ambos casos. En un escenario de ciberseguridad real, si bien es crucial detectar todas las conexiones maliciosas, un ligero porcentaje de falsos positivos puede suponer un enorme número de falsas alarmas, pues la tasa de conexiones por unidad de tiempo en una red de una compañía grande puede llegar a ser muy alta. Este problema puede resolverse estableciendo una banda de desconfianza alrededor del umbral de separación de clases, marcando ciertas conexiones para ser enviadas a un clasifica-



(a) Instancia de BEAA con AUC $= 0.9946$.



(b) Instancia de BEAA con AUC $= 0.9945$.

Figura 4.4: Mejores instancias del BEAA aprendidas sobre los datos sintéticos de *Titanium*. Para ambas, la mejor detección de anomalías se obtiene mediante la puntuación combinada por multiplicación de las puntuaciones del módulo del vector semilla y del error en la reconstrucción.

Parámetro	**Valor que toma**
Tipo de BEAA	BEAA genérico
Número de nodos de ruido	$m = 6$ para 13 nodos atributo
Parámetros para la generación aleatoria de semillas	$\mu_{\text{seed}} = 0.5$ con $\sigma_{\text{seed}} = \frac{\mu_{\text{seed}}}{2}$
Tamaño de generación y selección	$N_{\text{generación}} = 100000$ y $N_{\text{selección}} = 10000$, quedando un ratio de selección $\frac{N_{\text{selección}}}{N_{\text{generación}}} = 0.1$
Número de épocas	25 iteraciones de entrenamiento

Cuadro 4.9: Parámetros de la mejor instancia obtenida del BEAA sobre los datos sintéticos de *Titanium*, correspondiente a la Subfigura 4.4a. El entrenamiento se detiene por número máximo de épocas para esta instancia del modelo.

dor más pesado – dando lugar a un sistema de clasificación en cascada (Gama y Brazdil, 2000), como en el trabajo de Rahman *et al.* (2021) – o incluso a un revisor experto, dando lugar a un sistema híbrido de IA cuyo objetivo sería potenciar a un operador humano.

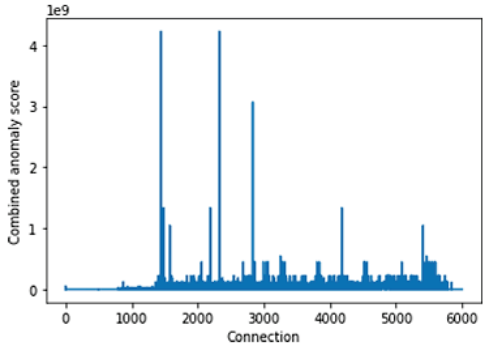
En particular, la distribución de la puntuación combinada de anomalía asignada a cada dato del conjunto de prueba por el modelo de la Subfigura 4.4a, que supone la configuración más exitosa obtenida sobre este problema, viene recogida en la Figura 4.5. La figura se ha distribuido en dos paneles separados para cada ataque porque la escala de las puntuaciones asignadas es muy diferente para el escaneo de puertos que para el movimiento lateral. El primero recibe puntuaciones muy altas al ser un tipo de ataque muy fácil de señalar como anómalo para el modelo, debido principalmente a la cantidad tan por encima de la media de paquetes de *ping* que se envían. El movimiento lateral, en cambio, es un ataque mucho más sofisticado y de perfil discreto, más difícil de detectar por anomalías.

Para comprobar que el BEAA funciona como detector de ataques en la red de labo-

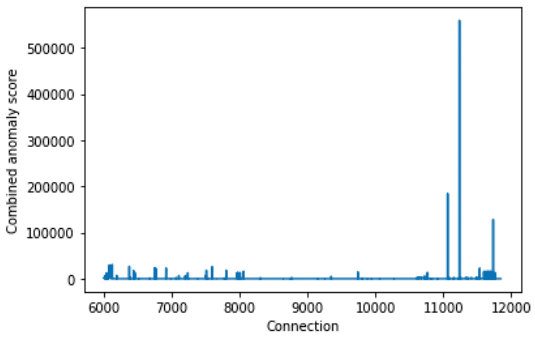
Parámetro	**Valor que toma**
Tipo de BEAA	BEAA genérico
Número de nodos de ruido	$m = 8$ para 13 nodos atributo
Parámetros para la generación aleatoria de semillas	$\mu_{\text{seed}} = 1$ con $\sigma_{\text{seed}} = \frac{\mu_{\text{seed}}}{2}$
Tamaño de generación y selección	$N_{\text{generación}} = 10000$ y $N_{\text{selección}} = 5000$, quedando un ratio de selección $\frac{N_{\text{selección}}}{N_{\text{generación}}} = 0.5$
Tolerancia en la convergencia de la log-verosimilitud	tol $= 0.1$

Cuadro 4.10: Parámetros de la segunda mejor instancia obtenida del BEAA sobre los datos sintéticos de *Titanium*, correspondiente a la Subfigura 4.4b. Nótese que se llega al final del entrenamiento por convergencia de la log-verosimilitud y no por número máximo de épocas.

ratorio se debe conocer cuándo comenzó cada ataque el experto en ciberseguridad según el reloj de la máquina de registro. Así se podrá comprobar si, en efecto, el sistema los habría detectado exitosamente. El escaneo de puertos se inicia a la hora 13:38 marcada por la máquina de registro correspondiendo a las conexiones numeradas en la franja entre la 1000 y la 6000. El ataque de movimiento lateral se ejecuta sobre las 14:50 de la máquina de registro, por lo que corresponde a las conexiones numeradas a partir de la 10500.



(a) Puntuación de anomalía para las conexiones 0-6000, correspondientes al ataque de escaneo de puertos.



(b) Puntuación de anomalía para las conexiones 6000-11846, correspondientes al ataque de movimiento lateral.

Figura 4.5: Puntuación de anomalía combinada asignada a cada conexión del conjunto de prueba del *dataset* de laboratorio de *Titanium* por el mejor modelo de BEAA desarrollado para este experimento, cuya curva ROC se encuentra en la Subfigura 4.4a.

Nótese que, si bien el clasificador no tiene una puntuación AUC perfecta, sería sencillo establecer un umbral que permitiera detectar, al menos, una conexión maliciosa de cada ataque sin causar falsas alarmas. En ambos paneles se puede comprobar visualmente que en las franjas de realización de los ataques se asignan puntuaciones de anomalía mucho más elevadas a las conexiones que se establecen, indicando

una correlación directa entre la detección de una anomalía en el tráfico de red y la ejecución de un ciberataque.

4.4.3. Interpretabilidad sobre los datos de *Titanium*

De la misma forma que se realizó un estudio de la interpretabilidad de los resultados del BEAA al aplicarlo sobre los datos del UNSW-NB15 en la Subsección 4.3.3, se ha procedido análogamente con los datos de laboratorio suministrados por *Titanium*.

Siguiendo el mismo esquema, y con el fin de que este estudio sea más ilustrativo para lectores que no sean expertos en ciberseguridad, se ha efectuado para este propósito un recorte en las variables tenidas en cuenta, dejando un total de 10 atributos para describir cada dato. Igualmente, se ha utilizado deliberadamente un sistema BEAA con un número reducido de nodos de ruido de modo que se extraigan menos relaciones estadísticas entre las variables atributo y sea más sencillo interpretar los resultados del modelo.

En particular, la instancia del BEAA que se ha entrenado expresamente para interpretar sus resultados sobre los datos de *Titanium* cuenta con las especificaciones que se recogen en el Cuadro 4.11. A pesar de la reducción de variables, logra una puntuación de área bajo la curva ROC muy competente, llegando a obtener AUC $= 0.9918$ para la detección de anomalías por error en la reconstrucción.

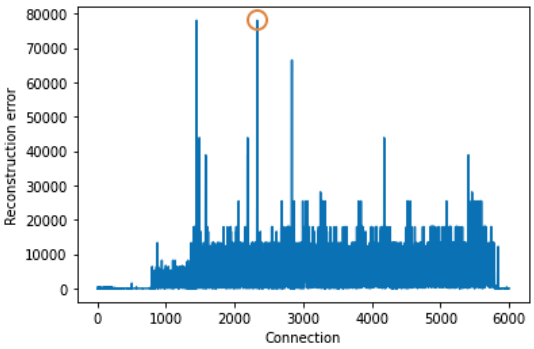
Parámetro	**Valor que toma**
Tipo de BEAA	BEAA genérico
Número de nodos de ruido	$m = 4$ para 10 nodos atributo
Parámetros para la generación aleatoria de semillas	$\mu_{\text{seed}} = 1$ con $\sigma_{\text{seed}} = \frac{\mu_{\text{seed}}}{2}$
Tamaño de generación y selección	$N_{\text{generación}} = 500000$ y $N_{\text{selección}} = 10000$, quedando un ratio de selección $\frac{N_{\text{selección}}}{N_{\text{generación}}} = 0.02$
Número de épocas	50 iteraciones de entrenamiento

Cuadro 4.11: Parámetros de la instancia de BEAA utilizada para el ejercicio de interpretabilidad sobre los datos sintéticos de *Titanium*.

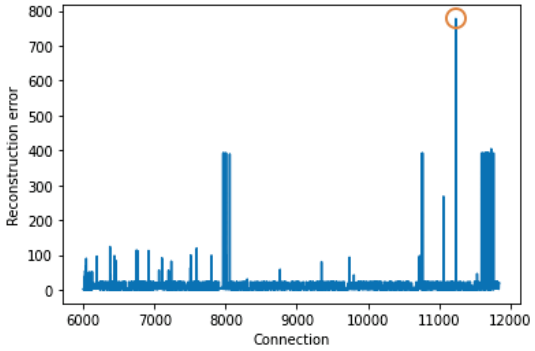
Como en este caso el método de detección de anomalías por módulo del vector semilla proporciona una separación algo peor de las irregularidades y, en adición, puntúa en una escala muy diferente los ataques de escaneo de puertos y de movimiento lateral, se ha realizado el ejercicio de interpretabilidad empleando la puntuación de anomalía obtenida por error en la reconstrucción. Esta elección, además, compensa que en la Sección 4.3.3 solo se estudiase la interpretabilidad mediante el método del módulo del vector semilla.

En la Figura 4.6 se recoge, en dos paneles separados para cada ataque, la puntuación de anomalía por error en la reconstrucción proporcionada por el BEAA preparado de acuerdo con los parámetros expuestos en el Cuadro 4.11 para cada conexión del conjunto de datos de prueba provisto por *Titanium*. Con el fin de estudiar la interpretabilidad para ambos tipos de ataque de un modo útil, tal y como se realizaría

en un escenario real, se han seleccionado para el ejercicio las conexiones de mayor puntuación en cada ataque. Estas conexiones, indicadas en la Figura 4.6, equivalen a la numerada como 2327 para el escaneo de puertos y a la 11062 para el ataque de movimiento lateral.



(a) Puntuación de anomalía para las conexiones 0-6000, correspondientes al ataque de escaneo de puertos.



(b) Puntuación de anomalía para las conexiones 6000-11846, correspondientes al ataque de movimiento lateral.

Figura 4.6: Puntuación de anomalía por error en la reconstrucción asignada a cada conexión del conjunto de prueba del *dataset* de laboratorio de *Titanium* por el modelo de BEAA desarrollado para el ejercicio de interpretabildad. Las instancias marcadas con un círculo naranja corresponden a las que han obtenido la puntuación más alta en el tramo de cada ataque.

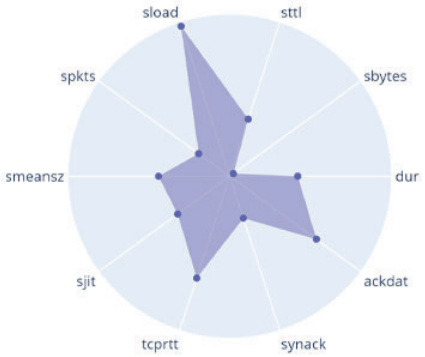
En la Figura 4.7 se representan en escala logarítmica los valores absolutos de la diferencia entre cada nodo atributo de la instancia original y de la reconstrucción, obtenida tras propagar el dato de entrada primero por $\mathcal{G}$ y seguidamente por $\mathcal{R}$.

Para el caso expuesto en la Subfigura 4.7a, que pertenece al tramo del escaneo de puertos y se corresponde con la instancia que obtiene la mayor puntuación de todo el conjunto de prueba, se puede apreciar que *sload* constituye la componente en la que el BEAA detecta una mayor anomalía por medio de la reconstrucción, seguida por *ackdat* y *tcprtt*, si bien hay que tener en cuenta que al haberse aplicado un escalado logarítmico sobre los datos de las diferencias la contribución a la puntuación de anomalía de *sload* en relación a *ackdat* y *tcprtt* es mayor de lo que parece en el gráfico.

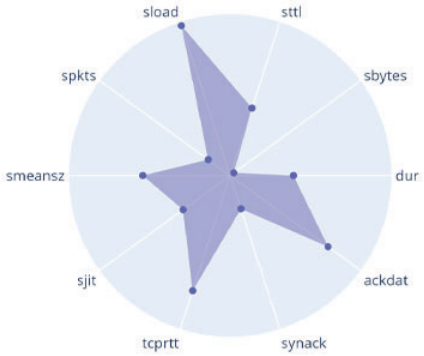
Todas las conexiones procedentes del ataque de escaneo de puertos presentan un patrón de error en la reconstrucción similar, en el que las diferencias en la reconstrucción de *sload*, *ackdat* y *tcprtt* son especialmente prominentes. Este esquema recuerda en cierto modo al que presenta el ejemplo de interpretabilidad correspondiente al ataque de *Fuzzing* en la Subsección 4.3.3, en el que la anomalía se detecta porque la tasa de transmisión de bits entre los equipos no concuerda con la duración de la conexión.

El signo que toma la diferencia de valores entre el dato original y el reconstruido es positivo para *sload*, indicando que en la reconstrucción se le ha asignado un valor más bajo que el que presentaba la conexión orignal anómala, y negativo para *ackdat* y *tcprtt*, lo que significa que en la reconstrucción se han sobreestimado sus valores.

Los ataques de escaneo de puertos tratan de sondear qué puertos de un equipo obje-



(a) Diferencia en la reconstrucción de cada nodo atributo calculada para la conexión más anómala del ataque de escaneo de puertos.



(b) Diferencia en la reconstrucción de cada nodo atributo calculada para la conexión más anómala del ataque de movimiento lateral.

Figura 4.7: Comparación de los valores absolutos de las diferencias en la reconstrucción de cada variable atributo, obtenida tras pasar el dato original por un proceso de codificación según $\mathcal{R}$ y de generación por $\mathcal{G}$, para dos conexiones correspondientes a los ataques de escaneo de puertos y movimiento lateral, respectivamente. Los valores se presentan escalados logarítmicamente.

tivo están abiertos a conexiones entrantes, y sirven para obtener un mapa de la red o el equipo que se desea invadir, por lo que suelen realizarse como fase preliminar de un ataque mayor. Se ejecutan enviando numerosos paquetes de solicitud de conexión a un equipo víctima para después dejar que esta expire esperando la confirmación del equipo atacante (Messer, 2007). De este modo se evita sobrecargar el sistema objetivo, esquivando las alarmas que pudieran saltar por un ataque de tipo DoS (véase la Subsección 4.3.3). Esta descripción concuerda con el patrón que se observa en la Subfigura 4.7a, dado que:

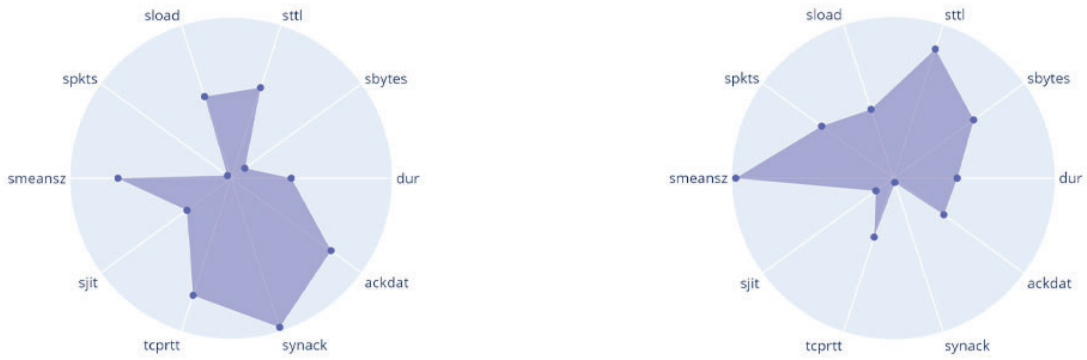
- La transferencia de numerosos paquetes de solicitud de conexión explica el aumento en la tasa de envío de bits, *sload*, pues se envían especialmente rápido dado que son ligeros por carecer de contenido y, además, están sujetos al protocolo de transporte UDP, que gobierna la transferencia de mensajes de sincronización entre equipos y es más rápido que el TCP, que se usa para transferencias ordinarias.
- La corta vida de la conexión, *tcprtt*, encuentra su razón de ser en que durante un escaneo de puertos las conexiones se cierran inmediatamente después de obtener la primera respuesta del equipo víctima para intentar que el escaneo pase desapercibido. Esto explica el corto tiempo de confirmación de la conexión, *ackdat*.

En la Subfigura 4.7b se puede observar la diferencia en la reconstrucción de la conexión más anómala que pertenece al ataque de movimiento lateral. Este tipo de ataque es mucho más sofisticado que un escaneo de puertos, que a nivel matemático destaca como una bruta desviación de la media de casi todas las variables que caracterizan

una conexión saludable. Así se explica que la conexión más anómala del ataque de movimiento lateral reciba una puntuación de anomalía hasta 100 veces menor que la más anómala del escaneo de puertos, pues los ataques de movimiento lateral se dan una vez el atacante ya posee el control de uno de los equipos de la propia red; al producirse la intrusión desde dentro de la red es más probable que se confunda con el tráfico normal.

En la conexión que se recoge en la Subfigura 4.7b, no obstante, puede apreciarse un patrón muy similar al del panel contiguo, correspondiente a un escaneo de puertos. La detección de los puertos disponibles en los demás equipos de la red es un paso clave en un proceso de movimiento lateral, ya que a menudo, el intruso trata de expandirse a otros equipos para ganar privilegios mayores o acceder a recursos valiosos. La discusión de la interpretabilidad de esta instancia seguiría el mismo discurso que para el caso dado por la conexión más anómala del escaneo de puertos, pues las variables involucradas son las mismas y en idéntica proporción, con la única diferencia siendo que *sload* toma un valor 10 veces menor.

Para enriquecer este apartado con ejemplos de interpretabilidad variados, en la Figura 4.8 se presentan las diferencias en la reconstrucción correspondientes a la segunda y tercera conexiones más anómalas pertenecientes al ataque de movimiento lateral.



(a) Diferencia en la reconstrucción de cada nodo atributo calculada para la segunda conexión más anómala del ataque de movimiento lateral.

(b) Diferencia en la reconstrucción de cada nodo atributo calculada para la tercera conexión más anómala del ataque de movimiento lateral.

Figura 4.8: Comparación de los valores absolutos de las diferencias en la reconstrucción de cada variable atributo, obtenida tras pasar por un proceso de codificación según $\mathcal{R}$ y de generación por $\mathcal{G}$, para dos conexiones correspondientes al ataque de movimiento lateral. Los valores se presentan escalados logarítmicamente.

De un vistazo se puede apreciar que las instancias cuyo error en la reconstrucción recoge la Figura 4.8 distan de parecerse a las discutidas en la Figura 4.7, por lo que el ejercicio de interpretabilidad deberá recorrer un camino diferente para deducir los principios tras el razonamiento del modelo.

Comenzando con la Subfigura 4.8a, es fácil comprobar que en este ejemplo es la

variable *synack*, que corresponde al tiempo de confirmación de la sincronización para la conexión, la que contribuye en mayor medida a la puntuación de anomalía que se le asigna. Igualmente, también el tiempo que tarda en establecerse la conexión (*tcprtt*) y el tiempo de confirmación de la conexión (*ackdat*) destacan en el gráfico. En cuanto al signo de la diferencia, puede consultarse que este es negativo para las tres variables mencionadas: en la reconstrucción se les ha asignado un valor más alto que en el dato original, denotando que el modelo las encuentra anómalamente bajas.

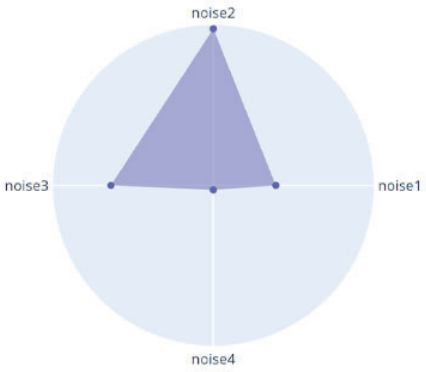
Entre las tres, *synack*, *tcprtt* y *ackdat* caracterizan el tiempo que tarda en configurarse una conexión entre dos equipos, que precede al intercambio de paquetes. Un tiempo de configuración inferior a lo esperado en una conexión puede ser indicativo de un comportamiento sospechoso. Por ejemplo, ciertos protocolos de conexión realizan durante su fase de configuración confirmaciones de seguridad para garantizar que el intercambio de paquetes sucede de forma fiable y sin errores, especialmente cuando se van a transferir archivos grandes o de forma continua. La omisión deliberada de estos pasos en la configuración de la conexión puede ser utilizada por un atacante para explotar vulnerabilidades, para enviar paquetes con contenido malicioso y que sean especialmente pesados o para enviar un gran número de éstos a través de un protocolo no seguro.

Por lo que respecta a la Subfigura 4.8b, la variable más relevante para el cálculo de la puntuación de anomalía es *smeansz*, que simboliza el tamaño medio de los paquetes enviados, seguida de *sttl*, el tiempo de vida medio de los paquetes, que especifica cuántas veces pueden ser redirigidos por otro dispositivo antes de llegar a su destino. La diferencia es positiva para *smeansz*, indicando que los paquetes enviados son considerablemente más grandes en el dato original de lo que el modelo ha aprendido como normal, por lo que al reconstruir el dato le asigna un valor mucho más bajo. Contrariamente, para *sttl* sucede al revés: resulta ser menor de lo esperado en el dato original. Es complicado establecer una relación entre estas dos variables, si bien, independientemente, pueden de por sí ser representativas de un ataque:

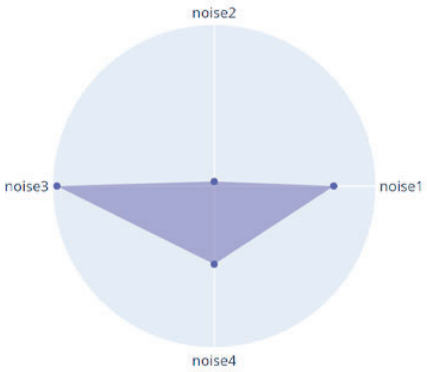
- El enorme tamaño medio de los paquetes transferidos podría provenir de la ausencia de mensajes de control y sincronización durante la transferencia, indicando que el proceso ha sido intervenido por un experto con privilegios elevados.
- Igualmente, el tiempo de vida medio de los paquetes enviados en una conexión se asigna de forma automática para cada protocolo. Su alteración por un atacante experto explicaría lo anómalamente bajo que se ha registrado a ser *sttl* para esta conexión.

Para complementar la información que nos proporciona el estudio de las diferencias en la recontrucción se puede realizar un análisis por componentes del módulo del vector semilla de la misma forma que se hizo en la Subsección 4.3.3 para los datos del conjunto UNSW-NB15. En la Figura 4.9 está representado para las dos instancias cuyas diferencias en la reconstrucción se exponen en la Figura 4.8 el peso de la contribución de cada variable de ruido al módulo del vector semilla.

La segunda conexión más anómala del ataque de movimiento lateral, representada en las Subfiguras 4.8a y 4.9a, resulta sufrir de una anomalía en el valor de la relación estadística codificada en la CPD del nodo de ruido *noise2*, cuya expresión se escribe como sigue:



(a) Contribución al módulo del vector semilla de cada nodo atributo calculada para la segunda conexión más anómala del ataque de movimiento lateral.



(b) Contribución al módulo del vector semilla de cada nodo atributo calculada para la tercera conexión más anómala del ataque de movimiento lateral.

Figura 4.9: Comparación de los valores absolutos de las contribuciones al módulo del vector semilla de cada variable de ruido para dos conexiones correspondientes al ataque de movimiento lateral. Los valores se presentan escalados logarítmicamente.

$$P(\textit{noise2} = z_2 | \textit{sjit}, \textit{synack}, \textit{sload}) = \\ \mathcal{N}(\mu = -1.211 + 3.072 \cdot \textit{sjit} + 2.329 \cdot \textit{synack} - 3.303 \cdot \textit{sload}, \sigma = 0.249).$$

En este caso, la enorme desviación de *synack* respecto a lo considerado normal por el modelo es lo que produce que diverga la relación codificada por *noise2*. Una disminución simultánea de la tasa de envío de bits, *sload*, podría haber compensado el bajo tiempo de confirmación de la sincronización, reforzando el razonamiento que se había llevado a cabo sobre esta anomalía: lo que hace que sea sospechoso el corto tiempo de configuración de la conexión no es en sí su valor, sino que no se haya reescalado consecuentemente *sload*, tal y como correspondería a un protocolo de transferencia no fiable, capaz de realizar una configuración más rápida o incluso prescindir de esta, pero no habilitado para el envío de paquetes a altas tasas de transferencia de bits.

Por otro lado, para la tercera conexión más anómala del movimiento lateral, cuyo patrón de anomalía se expone en las Subfiguras 4.8b y 4.9b, es el nodo *noise3* el que codifica la CPD que causa la divergencia de la puntuación de anomalía. Esta CPD tiene la expresión siguiente:

$$P(\textit{noise3} = z_3 | \textit{sttl}, \textit{smeansz}) = \mathcal{N}(\mu = -4.179 + 9.112 \cdot \textit{sttl} - 3.448 \cdot \textit{smeansz}, \sigma = 0.000),$$

que relaciona de forma directa las dos variables que ya se habían observado a ser anómalas en la Subfigura 4.8b. Si bien ya se había comentado que ambas variables podían ser utilizadas independientemente como señal de una conexión sospechosa,

el modelo ha sido capaz de hallar una correspondencia entre ambas, capturando que un aumento del tamaño medio de los paquetes enviados en una conexión suele conllevar un aumento del tiempo de vida con el que se envían y viceversa.

En este caso particular se da la situación opuesta: ante un aumento incoherente del tamaño medio de los paquetes enviados en una conexión, el tiempo de vida medio de estos se ve extrañamente reducido, lo que puede interpretarse como un síntoma de que la conexión ha sido manipulada por un experto con el objetivo de tomar control de otras máquinas y ascender en la escala de privilegios, ya que un número corto de vida para los paquetes ayuda a asegurar que no son enrutados más allá del entorno que desea controlar el atacante, indicando una incursión deliberadamente dirigida a un equipo objetivo cuya situación en la red es conocida por el atacante, siendo además el gran tamaño medio de los paquetes enviados un factor susceptible de implicar que la transferencia contiene ejecutables o que intenta sobrecargar a la víctima.

Los grafos de dependencias condicionales de las BNs que componen el BEAA aprendido para el ejercicio de interpretabilidad sobre los datos provistos por *Titanium* vienen recogidos en la Figura B.1 del Apéndice B.

Capítulo 5

Conclusiones y futuras líneas de investigación

En el trabajo que precede a este capítulo se ha introducido un novedoso modelo de detección de anomalías probabilística empleando BNs que aprenden sus parámetros y estructura en un proceso generativo-adversarial, detallando las etapas de diseño e implementación y demostrando su funcionamiento en dos casos prácticos de ciberseguridad: un *dataset* público que se ha utilizado como referencia en numerosos trabajos del ámbito y un conjunto de datos generado en un laboratorio virtual en colaboración con *Titanium*.

5.1. Conclusiones

El competidor más directo contra el que medir el BEAA viene dado por una BN clasificando anomalías por su log-verosimilitud, pues ofrece mejor interpretabilidad siendo más ligero, ya que es un componente del propio BEAA. El uso del BEAA, por tanto, sólo estaría justificado si su rendimiento es significativamente más alto que el de un detector de anomalías de BN. Aunque no se ha podido probar en ninguno de los dos conjuntos de datos disponibles que el BEAA ofrezca estadísticamente un rendimiento mejor que el de la BN simple, sí que se han obtenido instancias del BEAA entrenadas sobre el *dataset* UNSW-NB15 que superan la puntuación AUC que obtiene para el mismo problema el detector de anomalías por log-verosimilitud con una única BN.

El BEAA es un sistema que todavía está en fase de maduración, pero a pesar de esto ha demostrado ser capaz de proveer un rendimiento adecuado para su uso en escenarios industriales prácticos de detección de ciberataques. Por consiguiente cabe guardar esperanza respecto a la perspectiva de llegar a convertirlo en un modelo que supere robustamente el rendimiento del detector de anomalías de BN estándar. Esto situaría al BEAA como un modelo de buena interpretabilidad y rendimiento a medio camino entre una BN y un complejo de redes neuronales, que si bien no han sido probadas para este trabajo por motivos de tiempo y extensión, deberían ser más efectivas para la detección de anomalías, aunque dado el alto nivel de rendimiento exhibido en ambos problemas tanto por el BEAA como por el detector de anomalías de BN por log-verosimilitud, el margen de mejora en que pueden superarlos es escaso.

Como tal, se ha puesto especial énfasis en la facultad de interpretabilidad del BEAA,

pues es la característica que respalda el uso del BEAA frente a otros modelos que se desenvuelvan mejor en el mismo problema pero no sean capaces de proporcionar el mismo nivel de interpretabilidad, lo que puede ser clave en ciertos contextos prácticos. El BEAA no sólo es capaz de determinar qué variables han influido más en un resultado, sino que además puede deducir un conjunto de relaciones estadísticas entre las variables que caracterizan los datos, para después señalar cuáles son las que fallan en un dato etiquetado como anómalo.

5.2. Futuras líneas de investigación

El diseño e implementación del BEAA ha abierto una línea de investigación que no se ha podido abarcar en su totalidad en el presente trabajo. Al ser un desarrollo en proceso, el continuo esfuerzo por mejorar el algoritmo ha impedido caracterizar algunos de sus aspectos en la profundidad deseada. Al respecto cabe señalar tres estudios que serán próximamente efectuados para complementar el análisis del funcionamiento del BEAA realizado en este trabajo:

- Comparar el rendimiento del BEAA como detector de anomalías frente a los dos modelos generativos en los que se inspira: el modelo GANomaly y el AnoGAN (véase la Sección 2.1).
- Estudiar las competencias del BEAA como modelo generativo con capacidad de producir de manera altamente personalizable muestras sintéticas conformes con una distribución previamente aprendida. Para ello aprovecha sus propiedades de interpretabilidad en el sentido inverso, introduciendo desviaciones sobre las muestras producidas en la dirección deseada.
- En relación a su uso para detección de intrusiones en red en el ámbito de la ciberseguridad, realizar un análisis del tipo de ataques que son más difícilmente detectados por el BEAA para determinar sus puntos débiles como detector de anomalías.

En el apartado anterior se ha mencionado que el interés del BEAA reside principalmente en que es, en potencia, un modelo capaz de consolidarse entre los detectores de anomalías ya existentes como una solución de alto rendimiento con capacidad mejorada de interpretabilidad respecto a otros modelos generativos. No obstante, aún necesita de ciertas mejoras para convertirlo en un modelo más robusto. Las medidas que se han considerado como las más importantes para el futuro desarrollo del BEAA son:

- Desarrollar el BEAA sin apoyarse en la librería PyBNesian podría ayudar a simplificar el algoritmo de aprendizaje y el modelo en general, pues no estaría sujeto a las restricciones de diseño que impone PyBNesian y no necesitaría del generador inverso $\mathcal{R}$, ya que se podría diseñar una red que pudiera muestrear en ambas direcciones.

  Alternativamente, valdría considerar el diseño de un proceso de transferencia formal del conocimiento de $\mathcal{G}$ a $\mathcal{R}$ que permitiese aprender los arcos y las CPDs del generador inverso directamente a partir de las del generador directo, sustituyendo así el paso cuarto en el algoritmo de aprendizaje detallado en la Subsección 3.2.1. Así podría obtenerse un resultado similar sin necesidad de desarrollar desde cero todos los algoritmos empleados de PyBNesian, tras los que

hay un enorme trabajo.

- Implementar un nuevo módulo de entrenamiento que incluya un proceso de determinación del número óptimo de nodos de ruido para cada caso. Esta idea ya se comenzó a probar, pero fue desestimada para incluirse en el trabajo por motivos de extensión y tiempo. No obstante, los resultados obtenidos fueron prometedores.
- Estudiar la clasificación de nuevos datos en normales y anómalos empleando una frontera de decisión bidimensional que considere las dos puntuaciones de anomalía presentadas en la Sección 3.4 conjuntamente y en función del valor de ambas, en lugar de combinarlas simplísticamente por multiplicación.
- Estudiar la resistencia del sistema detector de anomalías ante *Data Poisoning* (Bovenzi *et al.*, 2022), que se refiere a la contaminación de los datos de entrenamiento de un sistema de detección de anomalías semisupervisada. La muestra de datos sobre la que entrena el BEAA debe estar compuesta exclusivamente por instancias normales, pero en un escenario práctico no se puede garantizar que un registro muy extenso del tráfico de la red no contenga ninguna conexión maliciosa. Se ha demostrado que los modelos generativos proveen una considerable robustez frente a este problema (Rahman *et al.*, 2021), pero aunque esta amenaza es de gran relevancia en el campo de la ciberseguridad, escasean los trabajos que desarrollan sistemas de detección de intrusiones por detección de anomalías usando modelos generativos para aumentar su robustez ante la contaminación de datos.
- La implementación de un seguro que proteja a los modelos del colapso de las CPDs de los nodos atributo durante el entrenamiento de $\mathcal{G}$ podría abrir las puertas hacia la obtención de un modelo con un perfil de funcionamiento más robusto en cuanto a rendimiento, pudiendo así llegar a probar que ofrece un rendimiento estadísticamente superior al de otros modelos más ligeros e interpretables, como el detector de anomalías que emplea una única BN y que clasifica los datos según su log-verosimilitud.
- El BEAA presentado en este trabajo emplea siempre BNs basadas en Gaussianas condicionadas linealmente. Este tipo de BNs tienen la desventaja de que presuponen que los datos sobre los que entrenan y funcionan están distribuidos conforme a una Gaussiana. Utilizar el BEAA con otros tipos de BNs, como las semiparamétricas (Atienza *et al.*, 2021b), ya implementadas en PyBNesian, podría hacer del BEAA un modelo más versátil y sofisticado, capaz de ajustarse mejor al problema de detección de anomalías en ciberseguridad.

Bibliografía

Akcay, S., Atapour-Abarghouei, A., y Breckon, T. P. (2018). GANomaly: Semi-supervised anomaly detection via adversarial training. En *Proceedings of the* 14^{th} *Asian Conference on Computer Vision*, pp. 622–637. Springer.

Atienza, D., Bielza, C., Diaz-Rozo, J., y Larrañaga, P. (2021a). Efficient anomaly detection in a laser-surface heat-treatment process via laser-spot tracking. *IEEE/ASME Transactions on Mechatronics*, 26(1):405–415.

Atienza, D., Bielza, C., y Larrañaga, P. (2021b). Semiparametric Bayesian networks. *Information Sciences*, 584:564–582.

Atienza, D., Bielza, C., y Larrañaga, P. (2022a). PyBNesian: An extensible python package for Bayesian networks. *Neurocomputing*, 504:204–209.

Atienza, D., Larrañaga, P., y Bielza, C. (2022b). Hybrid semiparametric Bayesian networks. *TEST: An Official Journal of the Spanish Society of Statistics and Operations Research*, 31(2):299–327.

Barredo-Arrieta, A., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., Garcia, S., Gil-Lopez, S., Molina, D., Benjamins, R., Chatila, R., y Herrera, F. (2020). Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, 58:82–115.

Bhuyan, M. H., Bhattacharyya, D., y Kalita, J. (2011). Surveying port scans and their detection methodologies. *The Computer Journal*, 54(10):1565–1581.

Bodria, F., Giannotti, F., Guidotti, R., Naretto, F., Pedreschi, D., y Rinzivillo, S. (2021). Benchmarking and survey of explanation methods for black box models. *CoRR*, abs/2102.13076.

Bovenzi, G., Foggia, A., Santella, S., Testa, A., Persico, V., y Pescapé, A. (2022). Data poisoning attacks against autoencoder-based anomaly detection models: A robustness analysis. En *Proceedings of the ICC 2022 - IEEE International Conference on Communications*, pp. 5427–5432. IEEE Computer Society.

Casajús-Setién, J., Bielza, C., y Larrañaga, P. (2022). Evolutive adversarially-trained Bayesian network autoencoder for interpretable anomaly detection. En *Proceedings of the* 11^{th} *International Conference on Probabilistic Graphical Models*. Aceptado.

Chandola, V., Banerjee, A., y Kumar, V. (2009). Anomaly detection: A survey. *ACM Computing Surveys*, 41.

Chen, M., Yao, Y., Liu, J., Jiang, B., Su, L., y Lu, Z. (2018). A novel approach for identifying lateral movement attacks based on network embedding. En *Proceedings*

of the 2018 IEEE International Conference on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom), pp. 708–715. IEEE Computer Society.

Deecke, L., Vandermeulen, R. A., Ruff, L., Mandt, S., y Kloft, M. (2019). Image anomaly detection with generative adversarial networks. En *Machine Learning and Knowledge Discovery in Databases (ECML/PKDD)*, pp. 3–17. Springer International Publishing.

Deza, M. M. y Deza, E. (2009). *Encyclopedia of Distances*. Springer.

Ding, M., Daskalakis, C., y Feizi, S. (2021). GANs with conditional independence graphs: On subadditivity of probability divergences. En *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics*, volumen 130 de *Proceedings of Machine Learning Research*, pp. 3709–3717. PMLR.

Doshi-Velez, F. y Kim, B. (2017). Towards a rigorous science of interpretable machine learning. *arXiv e-prints*.

Gama, J. y Brazdil, P. (2000). Cascade generalization. *Machine Learning*, 41:315–343.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., y Bengio, Y. (2014a). Generative adversarial networks. En *Proceeding of the* 27^{th} *Conference on Advances in Neural Information Processing Systems (NIPS)*, volumen 27, pp. 2672–2680. Curran Associates, Inc.

Goodfellow, I. J., Shlens, J., y Szegedy, C. (2014b). Explaining and harnessing adversarial examples. *arXiv 1412.6572*.

Goodman, B. y Flaxman, S. (2017). European union regulations on algorithmic decision-making and a "right to explanation". *AI Magazine*, 38(3):50–57.

Kaur, H., Adar, E., Gilbert, E., y Lampe, C. (2022). Sensible AI: Re-imagining interpretability and explainability using sensemaking theory. En *2022 ACM Conference on Fairness, Accountability, and Transparency*, p. 702–714. ACM.

Koller, D. y Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press.

Larrañaga, P. y Lozano, J. (2002). *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Springer.

Le, T., Hoang, T., Li, J., Liu, L., y Liu, H. (2015). A fast PC algorithm for high dimensional causal discovery with multi-core PCs. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 16:1483–1495.

Li, C., Welling, M., Zhu, J., y Zhang, B. (2018a). Graphical generative adversarial networks. En *Advances in Neural Information Processing Systems*, volumen 31. Curran Associates, Inc.

Li, J., Zhao, B., y Zhang, C. (2018b). Fuzzing: A survey. *Cybersecurity*, 1:2523–3246.

Loukas, G. y Öke, G. (2010). Protection against denial of service attacks: A survey. *The Computer Journal*, 53:1020–1037.

Madhuri, S. y Rani, M. U. (2018). Anomaly detection techniques causes and issues. *International Journal of Engineering Technology*, 7(3.24):449–453.

Mattia, F. D., Galeone, P., Simoni, M. D., y Ghelfi, E. (2019). A survey on GANs for anomaly detection. *CoRR*, abs/1906.11632.

Messer, J. (2007). *Secrets of Network Cartography: A Comprehensive Guide to Nmap*. Professor Messer.

Mihaljević, B., Bielza, C., y Larrañaga, P. (2021). Bayesian networks for interpretable machine learning and optimization. *Neurocomputing*, 456(C):648–665.

Moustafa, N. (2017). *Designing an Online and Reliable Statistical Anomaly Detection Framework for Dealing with Large High-Speed Network Traffic.* University of New South Wales, Canberra, Australia.

Moustafa, N., Creech, G., y Slay, J. (2017). *Data Analytics and Decision Support for Cybersecurity: Trends, Methodologies and Applications*, capítulo Big Data Analytics for Intrusion Detection System: Statistical Decision-Making Using Finite Dirichlet Mixture Models, pp. 127–156. Springer.

Moustafa, N. y Slay, J. (2016). The evaluation of network anomaly detection systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set. *Information Security Journal: A Global Perspective*, 25(1-3):18–31.

Moustafa, N., Slay, J., y Creech, G. (2019). Novel geometric area analysis technique for anomaly detection using trapezoidal area estimation on large-scale networks. *IEEE Transactions on Big Data*, 5(4):481–494.

Mustard, D. (1964). Numerical integration over the n-dimensional spherical shell. *Mathematics of Computation*, 18:578–589.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., y Duchesnay, E. (2011). Scikit-Learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Rahman, S., Uddin, A., Halder, S., y Acharjee, U. (2021). An efficient hybrid system for anomaly detection in social networks. *Cybersecurity*, 4(1):1–11.

Rajagopal, S., Kundapur, P., y S., H. (2020). A stacking ensemble for network intrusion detection using heterogeneous datasets. *Security and Communication Networks*, 2020:1–9.

Redner, R. A. y Walker, H. F. (1984). Mixture densities, maximum likelihood, and the em algorithm. *Sociery for Industrial and Applied Mathematics Review*, 26:195–239.

Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215.

Sarhan, M., Layeghy, S., Moustafa, N., y Portmann, M. (2021). NetFlow datasets for machine learning-based network intrusion detection systems. En *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pp. 117–135. Springer.

Schlegl, T., Seeböck, P., Waldstein, S. M., Schmidt-Erfurth, U., y Langs, G. (2017). Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. *CoRR*, abs/1703.05921.

Seneta, E. (2013). A tricentenary history of the law of large numbers. *Bernoulli*, 19(4).

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423.

Spackman, K. A. (1989). Signal detection theory: Valuable tools for evaluating inductive learning. En *Proceedings of the Sixth International Workshop on Machine Learning*, p. 160–163. Morgan Kaufmann.

Spirtes, P., Glymour, C., y Scheines, R. (1993). *Causation, Prediction, and Search*. Springer.

Sutton, M., Greene, A., y Amini, P. (2007). *Fuzzing: Brute Force Vulnerability Discovery*. Pearson Education.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., y Fergus, R. (2013). Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

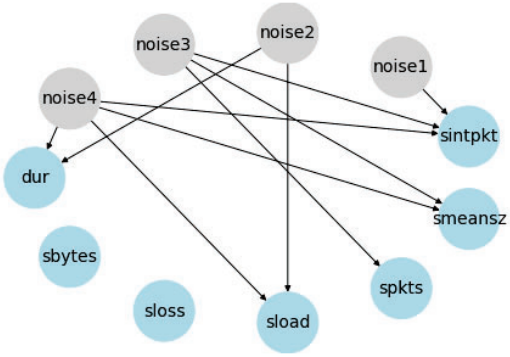
Tjoa, E. y Guan, C. (2019). A survey on explainable artificial intelligence (XAI): Towards medical XAI. *CoRR*, abs/1907.07374.

Ucar, M., Uçar, E., e Incetaş, M. (2021). A stacking ensemble learning approach for intrusion detection system. *Düzce Üniversitesi Bilim ve Teknoloji Dergisi*, 9:1329–1341.

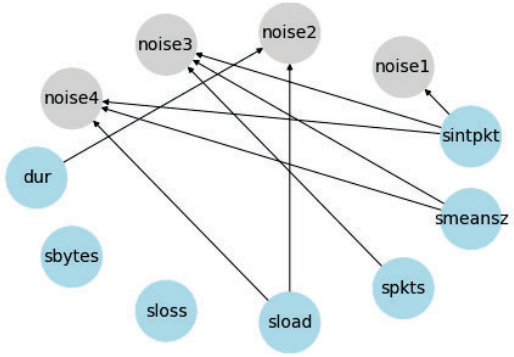
Apéndice A

Grafos de las BNs que integran el BEAA usado en la Sección 4.3.3

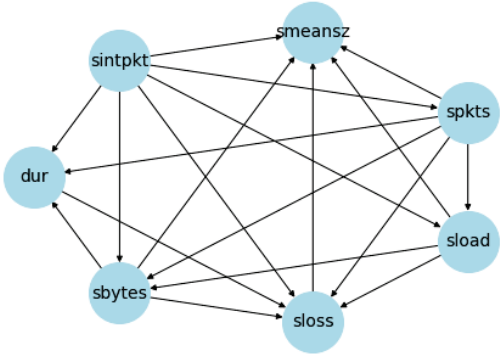
Las estructuras de las BNs generadora ($\mathcal{G}$), discriminadora ($\mathcal{D}$) y generadora inversa ($\mathcal{R}$) que componen el BEAA utilizado en la Sección 4.3.3 para la realización del ejercicio de interpretabilidad sobre el *dataset* UNSW-NB15 se recogen en la Figura A.1.



(a) DAG del generador $\mathcal{G}$



(b) DAG del generador inverso $\mathcal{R}$



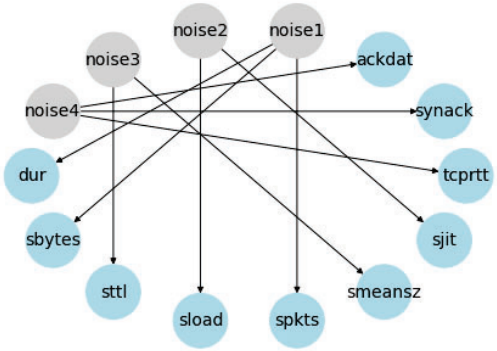
(c) DAG del discriminador $\mathcal{D}$

Figura A.1: Estructura gráfica de las componentes del BEAA empleado en la Sección 4.3.3. Los nodos de ruido aparecen en color gris y los nodos atributo en color azul.

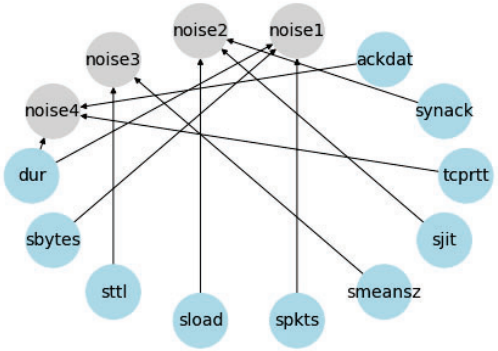
Apéndice B

Grafos de las BNs que integran el BEAA usado en la Sección 4.4.3

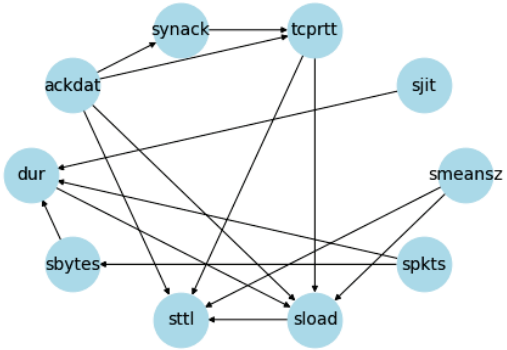
Las estructuras de las BNs generadora ($\mathcal{G}$), discriminadora ($\mathcal{D}$) y generadora inversa ($\mathcal{R}$) que componen el BEAA utilizado en la Sección 4.4.3 para la realización del ejercicio de interpretabilidad sobre los datos de laboratorio proporcionados por *Titanium* se recogen en la Figura B.1.



(a) DAG del generador $\mathcal{G}$



(b) DAG del generador inverso $\mathcal{R}$



(c) DAG del discriminador $\mathcal{D}$

Figura B.1: Estructura gráfica de las componentes del BEAA empleado en la Sección 4.4.3. Los nodos de ruido aparecen en color gris y los nodos atributo en color azul.