Konputagailuen Arkitektura eta Teknologia Saila
Departamento de Arquitectura y Tecnología de Computadores

Universidad del País Vasco
Euskal Herriko Unibertsitatea

INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Aportaciones a la clasificación no supervisada y a su validación. Aplicación a la seguridad informática.

Memoria que para optar al grado de Doctor en Informática presenta
Ibai Gurrutxaga Goikoetxea

Dirigida por
Olatz Arbelaitz Gallego
Javier Muguerza Rivero

Donostia, 2010


BY NC SA

Konputagailuen Arkitektura eta Teknologia Saila
Departamento de Arquitectura y Tecnología de Computadores

Universidad del País Vasco
Euskal Herriko Unibertsitatea

INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Aportaciones a la clasificación no supervisada y a su validación. Aplicación a la seguridad informática.

Memoria que para optar al grado de Doctor en Informática presenta
Ibai Gurrutxaga Goikoetxea

Dirigida por
Olatz Arbelaitz Gallego
Javier Muguerza Rivero

Donostia, 2010

Este trabajo ha contado con las subvenciones de la Diputación Foral de Gipuzkoa (y la Unión Europea) y la UPV/EHU (proyectos 1/UPV 00139.226-T-15920/2004 y EHU 08/39).

"Daría todo lo que sé,
por la mitad de lo que ignoro"

René Descartes

Aimarri

Eskerrak

Lehenik eta behin, eskerrik asko nire bi zuzendariei. Eskerrik asko, Olatz eta Javi, zuzendari izateaz gain lagun izateagatik. Eskerrik asko edozein lanetan laguntzeko prest azaltzeagatik. Eskerrik asko behar izan denean estutzeagatik. Eskerrik asko tesi hau sortzen hasi baino askoz lehenagotik emandako laguntzagatik. Eskerrik asko, azken finean, guztiagatik.

Eskerrik asko baita ere, noski, taldekide guztiei. Txus, Natxo, Iñigo, Iñaki... zer esan? Eskerrik asko hasiera hasieratik emandako laguntza guztiagatik. Eskerrik asko bihotzez nik tesia behingoz buka dezadan egin duzuen esfortzuagatik. Eskerrik asko taldean hain gustora egon izanaren arrazoi izateagatik. Lan hau, badakizue, talde osoarena da.

Eta ezin utzi alde batera lanean hain gustora sentitzen lagundu didazuen guztioi. Bazkalkideei, tertuliakideei... eskerrik asko denoi. Baina batez ere, eskerrik asko nire bulegokide ohiei. Eskerrik asko pantailatik begirada kenduarazi eta beste gauza batzuetan pentsatzen laguntzeagatik, eskerrik asko lanean hain momentu onak pasatu izanaren arrazoi izateagatik eta eskerrik asko lankide baino gehiago lagun izateagatik.

Esker berezi bat baita ere Yosuri, lanaren arlo matematiko zehatz batzuetan laguntzeagatik, baina batez ere beti irribarre batekin laguntzeko prest azaldu izanagatik. Eskerrik asko.

Eskerrik asko, aita eta ama, bizitzako beste gauza guztiak bezala tesi hau ere zuei esker lortu baitut. Eskerrik asko beti hor egon zaretelako laguntzeko prest, beti babesa emanaz. Zenbat errazten den bizitza horrela! Eskerrik asko ama, aita eta Oier, bizitzaz disfrutatzen erakusteagatik eta zoriontasuna gauza txikietan bilatzen erakusteagatik.

Eta bukatzeko, nola eskertu zuri, Iratxe... Eskerrik asko, tesi honek emandako burukominak sufritu behar izan dituzun arren beti animatu nauzulako. Baina eskerrik asko, batez ere, zure ondoan zoriontsu naizelako. Eskerrik asko zurekin hobea izaten ikasi dudalako. Maitatzen eta maitatua izaten erakutsi didazulako.

Eta eskerrik asko Aimar. Zenbat eman didazun hain denbora gutxian!

Índice general

Índice de figuras

Índice de tablas

Parte I

Introducción

Capítulo 1

Introducción

1.1. Motivación y objetivos

La minería de datos es una disciplina cada vez más importante en el área de la inteligencia artificial debido a la incalculable cantidad de datos que se generan cada día en diversos ámbitos de la sociedad. Cámaras digitales, transacciones telemáticas de todo tipo, tarjetas inteligentes, registro digital de todo tipo de compra-ventas... generan una cantidad de datos a partir de los cuales se puede obtener un conocimiento muy valioso para diversos fines. Sin embargo, la gran cantidad de datos obliga a que las técnicas de extracción de conocimiento sean automáticas.

Por otra parte, el uso de las redes de computadores con fines económicos como las transacciones bancarias y comerciales, el acceso a servicios administrativos... está cada vez más extendido. Este hecho, unido a un bajo nivel de concienciación respecto a la seguridad informática por parte de los usuarios, hace que las redes informáticas sean objetivo de ladrones y mafias organizadas.

Desafortunadamente, la seguridad informática total es imposible. Dado que las distintas aplicaciones informáticas no pueden dejar de ofrecer sus servicios, la posibilidad de un ataque es inevitable. Por ello, existen multitud de métodos para minimizar el riesgo de sufrir ataques de este tipo y detectarlos en caso de sufrirlos. Entre ellos se encuentran los antivirus, los cortafuegos, los sistemas de detección de intrusos... e incluso las, a menudo olvidadas, campañas de concienciación y educación del usuario.

El código malicioso o *malware* (virus, troyanos, gusanos...) es uno de los peligros a los que se expone todo usuario de las redes informáticas. Por desgracia, el peligro de este tipo de código no se limita a un funcionamien-

to ineficiente del ordenador o a la pérdida de datos. Actualmente existe *malware* diseñado para conseguir información relevante como códigos de acceso, números de cuenta, tarjetas de crédito, etc. Dado el elevado número de *malware* generado cada día y las dificultades para generar reglas de detección de intrusos manualmente, cada vez es más habitual la unión de la seguridad informática y las técnicas de la minería de datos (Barbará y Jajodia, 2002; Lee, 1999).

El trabajo descrito en esta memoria nace en este contexto y surge gracias a dos proyectos que tratan de buscar soluciones a dos problemas de la seguridad informática mediante técnicas de minería de datos. El primer proyecto surge por iniciativa de una de las empresas más importantes de seguridad informática a nivel mundial: S21sec Information Security Labs (S21sec)[1]. Una de las múltiples responsabilidades de dicha empresa consiste en analizar y catalogar todo tipo de *malware* detectado en los millones de ordenadores propiedad de las empresas e instituciones que protege.

Sin embargo, la cantidad de código malicioso descubierto cada día va en aumento y ya se cuenta por miles. Este hecho imposibilita el análisis manual del *malware* y requiere un análisis automático. Afortunadamente, la gran mayoría del código malicioso es una variante de *malware* conocido por lo que se considera factible que los sistemas automáticos los clasifiquen, dejando a los expertos en seguridad únicamente código malicioso de nueva generación.

Por otra parte, el análisis estático tradicional en el que se analiza el código del programa malicioso está resultando ser ineficiente debido, entre otras razones, a técnicas de polimorfismo y metamorfismo utilizadas en la actualidad. Estas técnicas consiguen ocultar el código de un programa malicioso, e incluso modificarlo periódicamente. La única solución a este problema es el análisis dinámico, es decir, ejecutar el código malicioso en un entorno controlado y monitorizar todas las acciones realizadas por él y por los procesos que pueda ir generando.

Por consiguiente, el objetivo del proyecto planteado por la empresa S21sec consiste en la realización de un sistema de agrupamiento de *malware* en familias mediante análisis dinámico. Una vez se construyan e identifiquen las familias la mayor parte del nuevo *malware* detectado podrá ser asignado a alguna de las familias, quedando automáticamente catalogado.

El segundo proyecto del que surge el trabajo descrito en esta memoria está indirectamente relacionado con el código malicioso. En muchas ocasio-

[1]Agradecemos la colaboración prestada por los expertos de la empresa S21sec así como los datos que nos han proporcionado.

nes el código malicioso se utiliza con el objetivo de conseguir acceso ilegitimo a sistemas protegidos, sea robando credenciales de acceso, abriendo puertas traseras, realizando escaneos, etc. Una de las técnicas para detectar y evitar estas intrusiones es la detección de intrusos. Los sistemas de detección de intrusos (IDS) analizan información, tanto de la red como de un ordenador, para tratar de detectar comportamientos que denoten una posible intrusión al sistema.

Debido a la rápida evolución de estos ataques y al elevado tráfico de datos en las redes actuales, existen sistemas de detección de intrusos basados en el aprendizaje automático desde hace más de dos décadas (Denning, 1987). Sin embargo, las dos principales aproximaciones de este tipo sufren un inconveniente fundamental: la necesidad de datos etiquetados. En concreto, una aproximación, denominada detección de uso indebido, requiere datos etiquetados como conexiones normales o ataques. Por otra parte, la otra aproximación, denominada detección de anomalías, requiere únicamente datos normales.

No obstante, dado el elevado tráfico y la variedad de la información transmitida en una red normal, la obtención de tráfico representativo puramente normal (sin ningún ataque) es un trabajo altamente dificultoso. El etiquetado de toda la información es, obviamente, aún más complicado. Debido a ello, existe una tercera aproximación más reciente, denominada detección de anomalías no supervisada, que trabaja sobre datos en los que se entremezcla el tráfico normal con ataques y no se conoce la etiqueta de cada uno. La obtención de datos para esta aproximación es extremadamente sencilla por lo que consideramos que será la aproximación principal en el futuro.

Por otra parte, la mayoría de los sistemas de detección de intrusos existentes que se basan en el análisis del tráfico de red, denominados sistemas de detección de intrusos de red, centran sus decisiones en la información obtenida a partir de las cabeceras de los paquetes interceptados. Es decir, ignoran los datos transmitidos por el paquete y analizan sólo las cabeceras de control. La razón es que los protocolos que producen este tipo de cabeceras son pocos y conocidos, por lo que su análisis se puede hacer de forma sencilla. Normalmente este análisis se reduce a las cabeceras de los protocolos principales de la arquitectura TCP/IP (TCP, UDP e IP), aunque pueden incluir algún protocolo auxiliar como ICMP o ARP.

No obstante, existen algunos sistemas de detección de intrusos que sí analizan los datos transmitidos en la red (Krügel et al., 2002; Wang y Stolfo, 2004). Sin embargo, estos datos siguen distintos protocolos y formatos dependiendo de la aplicación que los haya generado. Por ello, el análisis se suele centrar en algunos protocolos bien conocidos como DNS, HTTP o SMTP, en

base a reglas obtenidas para cada protocolo en particular. También existe alguna aproximación que analiza los datos de cualquier aplicación, pero genera distintos modelos de tráfico dependiendo del puerto utilizado (en base a la premisa de que el puerto identifica la aplicación). Uno de sus inconvenientes es que algunos de los modelos se definen en base a muy pocos ejemplos y que alguna aplicación ausente en los datos de entrenamiento puede quedar ignorada. Además, no es posible combinar los análisis de diferentes variaciones de una aplicación que utilicen puertos distintos para comunicarse.

Por todo ello, planteamos un proyecto con el objetivo de definir un sistema de detección de intrusos que, utilizando técnicas de aprendizaje automático, fuera capaz de analizar la información correspondiente a los datos de aplicación dentro del tráfico de una red. Asimismo, el sistema debería de ser capaz de analizar esta información de forma genérica, independientemente de la aplicación que la genere, y por lo tanto, sin conocer su estructura o formato. Además, el sistema de detección crearía un único modelo para todas las aplicaciones sin ningún tipo de dependencia sobre los puertos utilizados. Por las razones mencionadas anteriormente, dicho sistema de detección de intrusos debería seguir la aproximación no supervisada, es decir, con todo tipo de tráfico sin etiquetar.

El desarrollo de estos dos proyectos se ha fundamentado en la técnica del aprendizaje automático no supervisado conocido como agrupamiento, o más habitualmente, clustering. Aunque en el próximo capítulo se profundizará en el clustering, anticipamos aquí una posible definición. El clustering es una técnica que sirve para agrupar datos en grupos, o clusters, en base a sus características. El objetivo es dividir o particionar los datos de forma que los casos que estén en el mismo grupo sean similares y los que estén en distintos grupos sean diferentes.

El clustering es una técnica que ha avanzado mucho desde sus inicios y se afronta desde distintos ámbitos como la estadística o el aprendizaje automático. El número de algoritmos de clustering que sirven para agrupar los datos de una base de datos es incalculable (Jain, 2009). También existen multitud de aproximaciones para analizar si unos datos tienen cierta tendencia a agruparse y, si es así, para saber el número de grupos que forman. Por otra parte, existen multitud de aproximaciones para medir la bondad de una agrupación o partición concreta o comparar diversas particiones.

Sin embargo, a diferencia de otras áreas relacionadas como la estadística, el reconocimiento de patrones o la clasificación supervisada, el clustering es un área que ha resultado difícil de fundamentar matemáticamente (Kettenring, 2006). Aunque la idea de grupo o cluster resulte intuitivamente simple, la realidad es que su definición no es en absoluto sencilla. Tal vez ésa sea

la razón principal por la que el clustering no se fundamente aún sobre unas bases teóricas robustas.

Esta carencia del clustering, aunque no ha sido impedimento para obtener grandes avances en el área, ha generado cierta arbitrariedad en algunos procedimientos y metodologías. De esta manera, algunos procedimientos se han ido estableciendo como estándares *de facto* sin un análisis teórico robusto que las respalde. El resultado es que distintas aportaciones en el área se basan o han sido evaluadas por procedimientos sesgados o incorrectos llevando, por tanto, a conclusiones erróneas o imprecisas.

El trabajo realizado para aplicar el clustering a los dos problemas del ámbito de la seguridad informática recién mencionados nos ha llevado a analizar de un modo crítico varios procedimientos o metodologías. El resultado ha derivado en una reflexión sobre dos metodologías habitualmente utilizadas en el clustering. Cada una de las reflexiones va acompañada de una descripción de los inconvenientes observados y una propuesta alternativa que se analiza empíricamente.

Una de las metodologías criticadas corresponde a la validación de clustering. Una de las técnicas de validación de clustering, denominada validación interna, consiste en medir la bondad o calidad de una partición sin conocer la partición ideal de los datos. Existen multitud de índices que tratan de medir esa bondad por lo que es habitual evaluar y comparar dichos índices. Aunque no existe un estándar para realizar dicha comparación existe una metodología que engloba la mayoría de experimentos realizados en este contexto. Dicha metodología se basa en evaluar cada índice frente a uno de los problemas principales del clustering: determinar el número de clusters en un conjunto de datos determinado.

Hemos observado que la metodología de evaluación mencionada funciona bajo una suposición que no siempre se cumple. Por lo tanto, uno de los objetivos del trabajo realizado es ilustrar de la forma más clara posible dicho inconveniente y definirlo formalmente para, posteriormente, proponer una solución alternativa que evite dicho problema.

Una de las dificultades principales de este trabajo es la de demostrar que la solución propuesta es mejor que la metodología existente. Dado que la bondad de los distintos índices es desconocida, validar una metodología que las evalúa no es sencillo. Sin embargo, una reflexión profunda junto con una experimentación adecuada pueden servir para determinar la aptitud de cada una de las metodologías.

Otra de las aportaciones realizadas para mejorar una metodología o procedimiento existente se centra en el clustering jerárquico. Algunos algoritmos de clustering construyen un conjunto de particiones anidadas denominada

jerarquía de clusters. Estas jerarquías se han utilizado ampliamente en biología (Sneath y Sokal, 1973) ya que se adaptan muy bien a las relaciones filogenéticas de distintos organismos. De manera similar, estas jerarquías se adaptan a las relaciones de distintos programas maliciosos, ya que, como se ha apuntado anteriormente, muchos son variaciones de otros.

Por todo ello, el clustering jerárquico ha sido una de las modalidades con las que hemos trabajado en el entorno de la seguridad informática. Este trabajo nos ha llevado a realizar dos aportaciones en este terreno. La primera de ellas trata de mejorar un procedimiento de post-proceso de jerarquías mientras que la segunda nos ha llevado a explorar un nuevo punto de vista en la actualización de jerarquías.

A pesar de que una jerarquía de clusters aporta mucha más información que una partición, es habitual simplificar una jerarquía para obtener una única partición. El trabajo realizado con jerarquías de clusters nos ha llevado a detectar ciertos contextos en los que el procedimiento de post-proceso de jerarquías para reducirlas a una única partición no realiza un trabajo satisfactorio. Por lo tanto, el objetivo relativo a este punto será describir el método de post-proceso de la forma más clara posible e ilustrar con ejemplos las situaciones donde falla. El objetivo final es proponer un nuevo método que supere dichos inconvenientes y analizar empíricamente sus ventajas. Este segundo trabajo tiene relación directa con el anterior ya que los métodos de post-proceso de jerarquías, tanto el habitual como el propuesto, se basan en los mencionados índices de validación.

Finalmente, la necesidad de adaptar dinámicamente las familias de *malware* generadas en el proyecto propuesto por la empresa S21sec, nos ha llevado a ver la necesidad de un algoritmo de clustering jerárquico incremental con ciertas características particulares. Aunque en la literatura científica existen aproximaciones de este tipo, la actualización de las jerarquías provoca cambios importantes en las estructuras previamente creadas. Esta inestabilidad es un serio inconveniente ya que dificulta la aplicación práctica del conocimiento extraído. Por un lado las frecuentes modificaciones de las estructuras principales dificultan su asimilación por parte del usuario del sistema. Por otra parte, aunque los resultados obtenidos sean buenos, la inestabilidad provoca desconfianza en el usuario, que puede terminar por desechar el sistema de aprendizaje automático (Turney, 1995).

Por consiguiente, proponemos un último objetivo relacionado con la estabilidad de los algoritmos jerárquicos incrementales. La idea fundamental es diseñar, implementar y analizar un algoritmo jerárquico incremental estable. Es decir, la actualización incremental de las jerarquías deberá realizarse sin modificar de forma significativa las estructuras existentes. Obviamente,

habrá que comprobar que la estabilidad no afecte severamente a la calidad de las jerarquías construidas por el algoritmo.

1.2. Organización de la memoria

Esta memoria está organizada en cuatro partes principales: Introducción, Aportaciones, Aplicaciones y Conclusiones. La primera de ellas contiene dos capítulos. El primero de ellos es el que nos ocupa y sirve para introducir el trabajo realizado que se describe en esta memoria. El Capítulo 2 describe los fundamentos de la clasificación no supervisada y su validación, pilares fundamentales de este trabajo. Asimismo, se definen formalmente algunos conceptos importantes y se introduce la notación utilizada a lo largo del documento.

En la parte Aportaciones se describen, divididas en tres capítulos, las aportaciones realizadas al área de la clasificación no supervisada. Como ya se ha indicado en la sección anterior estas aportaciones giran principalmente en torno al clustering jerárquico y la validación de clustering.

El primer capítulo de esta parte (Capítulo 3) describe el trabajo relacionado con el algoritmo SIHC, un algoritmo jerárquico de clustering incremental y estable.

Por su parte, en el Capítulo 4 se describe un nuevo método de post-proceso de jerarquías, SEP, que permite obtener como resultado mejores particiones que el método habitual. Asimismo, se describe y analiza un nuevo índice de validación, COP, adecuado para este método.

La última aportación se describe en el Capítulo 5. Se trata de una nueva metodología para evaluar índices de validación de clusters y se argumenta por qué es más adecuada que la metodología tradicional.

La tercera parte, Aplicaciones, se compone de dos capítulos. Estos capítulos describen dos aplicaciones de la clasificación no supervisada a la seguridad informática. Por un lado, en el Capítulo 6 se describe el prototipo MALBEC, prototipo realizado para la agrupación de *malware*. Asimismo se describen las distintas comparativas que se han realizado para determinar las diferentes técnicas de aprendizaje automático a utilizar.

Por otro lado, en el Capítulo 7 se describe un nuevo sistema de detección de intrusos de red que analiza los datos del tráfico de red de forma genérica sin tener en cuenta los distintos formatos y protocolos posibles a nivel de aplicación.

Estos cinco capítulos forman el núcleo de la tesis y describen las contribuciones originales realizadas. Todos los capítulos tienen una estructura similar

donde primero se introduce el problema y se propone una solución. Posteriormente se realiza una experimentación para evaluar la solución propuesta y se finaliza con una sección donde se detallan las conclusiones obtenidas.

Debido a los compromisos de confidencialidad contraídos con la empresa S21sec, el Capítulo 6 es el que más se aleja de este esquema general, y que menos profundiza en el trabajo realizado. Sin embargo, la parte más técnica se describe con detalle omitiendo tan sólo aspectos particulares de la implementación del prototipo.

La última parte de la memoria consta de un único capítulo, el Capítulo 8, donde se resumen las conclusiones obtenidas en los distintos trabajos realizados y se describen posibles líneas de trabajo futuras. Asimismo, este capítulo consta de una lista de las publicaciones en las que se ha publicado parte de este trabajo.

Finalmente, se detallan en orden alfabético las referencias bibliográficas citadas a lo largo de todo el documento y se incluye un apéndice con información adicional sobre el Capítulo 5.

Capítulo 2

Clasificación no supervisada

2.1. Introducción

Los avances en las tecnologías de almacenamiento de datos junto con un espectacular aumento de las aplicaciones de Internet, imágenes digitales, vídeo... han generado una ingente cantidad de bases de datos de diversa índole. Se estima que el universo digital consumió 281 exabytes en el año 2007 y que el valor de 2006 se multiplicará por 10 en 5 años (Gantz, 2008). La mayor parte de esa información está almacenada en soportes electrónicos lo que supone una inmejorable oportunidad para un análisis de datos automático.

Pero el aumento de la cantidad de datos ha venido acompañado de un aumento en la variedad de los datos. Cámaras fotográficas y de vídeo digitales generan cantidades ingentes de fotografías y vídeos. El despliegue de tarjetas RFID ha resultado en millones de sensores transmitiendo datos de forma regular. Sitios web, mensajes de correo electrónico, blogs, transacciones comerciales... generan terabytes de información cada día. Este aumento de la cantidad de datos y su variedad requiere serios avances en las técnicas de análisis de datos.

La minería de datos o *Data Mining* consiste en la extracción de la información, o el conocimiento, que reside de forma implícita en los datos (Han y Kamber, 2006; Hernández et al., 2004). El proceso completo consiste en varios pasos desde la obtención de los datos y su transformación, hasta la validación de los resultados y su presentación. Dependiendo de la fuente podemos encontrar diversas variantes del proceso. Algunos agrupan varios pasos en uno o incluso ignoran algún paso por considerarlo obvio. A continuación se describen brevemente los pasos seguidos en un proceso general

de minería de datos.

1. Obtención de los datos: El primer paso de un proceso de minería de datos consiste en obtener los datos a analizar.

2. Limpieza e integración: Los datos se limpian, en la medida de lo posible, de datos inconsistentes y ruido. Asimismo, si los datos proceden de distintas fuentes se integran de forma adecuada.

3. Selección y transformación de los datos: En este paso se seleccionan los datos más adecuados para la extracción del conocimiento y se transforman de forma apropiada. Por ejemplo, para ciertos métodos de aprendizaje es necesario normalizar las características para que todas estén en un rango similar.

4. Extracción del conocimiento: Es en este paso donde se aplican los métodos estadísticos, de aprendizaje automático, de reconocimiento de patrones... capaces de extraer conocimiento de los datos. Estos métodos generalmente extraen modelos que representan los datos en forma de reglas, modelos estadísticos, redes neuronales...

5. Validación del modelo: En este paso se valida el modelo obtenido en el paso anterior. Para ello es posible que haya sido necesario reservar parte de los datos. De esta manera se puede validar el modelo obtenido utilizando datos ignorados en el proceso de aprendizaje de forma que se evitan posibles problemas derivados de un sesgo excesivo hacia la muestra de entrenamiento.

6. Presentación de los resultados: En este paso final se presentan los resultados de forma sencilla de interpretar.

Hay que tener claro que el proceso descrito es un proceso iterativo y que los resultados en cada uno de los pasos pueden llevar a repetir parte del proceso o incluso el proceso completo.

El paso de extracción del conocimiento es el que más atención ha despertado en la comunidad científica y existen multitud de algoritmos para ello. La mayoría de estos algoritmos se engloban en el aprendizaje automático.

El aprendizaje automático o *Machine Learning* es una rama de la Inteligencia Artificial que trata de generalizar patrones a partir de una información suministrada en forma de ejemplos. Es decir, los algoritmos de

aprendizaje automático reciben como entrada un conjunto de casos[1] y realizan una generalización para obtener un modelo que represente el contexto de donde han sido extraídos los datos. Este conjunto de datos de entrada se conoce como muestra de entrenamiento.

El objetivo de los algoritmos de aprendizaje suele ser tanto predictivo como descriptivo. En el primero de los casos el objetivo es utilizar la muestra de entrenamiento para predecir algo sobre los datos que no se encuentran en la muestra. En el segundo, se analiza la muestra y se trata de describir su contenido de forma fácil de entender, analizando para ello las relaciones entre los distintos casos que la componen.

El aprendizaje automático se ha dividido tradicionalmente en dos ramas: la clasificación supervisada y la clasificación no supervisada. La primera de ellas trabaja con datos etiquetados, es decir, cada uno de los casos de la muestra de entrenamiento tiene asociada una etiqueta o clase. Por lo tanto, cada caso está descrito por una serie de características (variables independientes) y etiquetada con una clase (variable dependiente). La segunda rama, en cambio, trabaja sobre datos sin etiquetar. Esta "pequeña" diferencia hace que los objetivos y algoritmos de una y otra rama sean completamente distintos.

En la clasificación supervisada (o aprendizaje supervisado) los algoritmos relacionan las variables independientes con la dependiente. El objetivo principal de estos algoritmos es asignar una clase a casos desconocidos no etiquetados en base al aprendizaje realizado sobre la muestra de entrenamiento. Dado que los casos a clasificar son desconocidos, es decir, no se encuentran en la muestra de entrenamiento, el algoritmo ha de realizar una generalización de los datos de entrenamiento y crear un modelo. Si la clase a predecir es un valor continuo el proceso se conoce como regresión, mientras que si es discreto se conoce como clasificación.

El modelo creado por los algoritmos supervisados puede tener diversas formas —un árbol, una serie de reglas, una red neuronal...— pero la mayoría implícitamente tesela el espacio de donde provienen los datos y asigna una clase a cada región del espacio. La región en la que se encuentran los nuevos casos a clasificar determinará su clase.

Existen diversos paradigmas de clasificación supervisada y multitud de algoritmos (Han y Kamber, 2006; Hernández et al., 2004). Sirva como ejemplo esta lista no exhaustiva:

- Árboles de decisión: C4.5, CHAID, CART, CTC...

[1] Aparte de caso, en la bibliografía es habitual encontrar otros términos para referirse a este concepto: ejemplo, objeto, instancia, patrón, dato...

- Reglas de inducción: RIPPER, CN2...
- Métodos bayesianos: *Naïve-Bayes*, redes bayesianas...
- Clasificadores basados en vecindad: K-NN...
- Redes neuronales artificiales: perceptrón multicapa, funciones de base radial...
- Máquinas de vector soporte (*Support Vector Machines*): SMO...
- Clasificadores múltiples: *Bagging*, *Boosting*, *Random Subspace Method*, *Random Forest*...

A diferencia del valor predictivo de la clasificación supervisada, los algoritmos de clasificación no supervisada tienen un valor principalmente descriptivo. Dado que la clase de los casos de la muestra de entrenamiento es desconocida, e incluso se ignora el número de clases existentes, los algoritmos tratan de describir de forma más o menos sencilla la posible estructura de los datos.

Al partir de un estado con menos información la clasificación no supervisada es generalmente más difícil que la supervisada. Por ello, ha recibido menos atención y existen menos paradigmas y algoritmos. La técnica más estudiada y desarrollada es el agrupamiento o clustering aunque existen otros como las reglas de asociación y el algoritmo APRIORI (Agrawal y Srikant, 1994). Dado que el clustering es un tema central en esta tesis, se realizará una descripción más exhaustiva en la próxima sección.

En los últimos años se ha empezado a considerar la clasificación supervisada y la no supervisada como los dos extremos de un mismo problema. En un extremo se conocen las etiquetas de todos los casos de la muestra de entrenamiento (clasificación supervisada) mientras que en el otro no se conoce ninguna (clasificación no supervisada). Entre ambos existe un escenario donde se conocen las etiquetas de sólo algunos casos. Este escenario se conoce como clasificación semi-supervisada (Chapelle et al., 2006).

La clasificación semi-supervisada es de gran utilidad en los contextos donde el etiquetado de los casos es posible, pero altamente costoso. En estos contextos el etiquetado total es inviable, pero se ha comprobado que el etiquetado de algunos de los casos, aunque sólo sean unos pocos, aporta una gran ventaja respecto a la clasificación no supervisada.

2.2. Clustering

El clustering es la técnica principal de la clasificación no supervisada. Aunque existen diversas definiciones de clustering podríamos decir que el objetivo del clustering es agrupar los casos de una muestra o base de datos en grupos naturales. Para aclarar el concepto de grupo natural se suele recurrir al concepto de cercanía. Cada grupo o cluster generado por un proceso de clustering debe ser homogéneo, y a su vez, los clusters deben ser heterogéneos entre sí.

Evidentemente, el concepto de cercanía dependerá de la distancia o medida de similitud utilizada para comparar los casos de la muestra, por lo que la definición de la distancia utilizada es uno de los puntos claves del clustering. Otra forma de definir un cluster es en base al concepto de densidad. Un cluster ideal podría definirse como un conjunto de objetos compacto y aislado.

En la práctica, un cluster es una entidad subjetiva que depende del observador. Desafortunadamente, mientras que los humanos somos excelentes detectores de clusters en 2 y posiblemente 3 dimensiones, necesitamos algoritmos automáticos para datos de más dimensiones. Este reto, junto con el desconocimiento del número de grupos naturales en los datos analizados, es el que ha generado, y sigue generando, multitud de algoritmos de clustering (Jain, 2009).

En la actualidad cada vez está más extendida una versión del clustering más flexible que asigna a cada caso una probabilidad de pertenencia a cada cluster. De este modo, se puede representar, por ejemplo, el hecho de que un caso esté en una zona de solapamiento de varios clusters. A esta versión de clustering se le conoce como clustering difuso o *fuzzy*, mientras que al clustering tradicional se le denomina *crisp*. En esta memoria nos centraremos exclusivamente en este último.

2.2.1. Definiciones y notación

Antes de entrar en una descripción más detallada del clustering en esta sección se describen varios conceptos y se determina la notación utilizada a lo largo de todo el documento.

Una base de datos, $X = \{x_1, x_2, \ldots, x_N\} \in \Re^F$, es un conjunto de N objetos o puntos representados como vectores de características en un espacio F-dimensional. Denotamos el valor del atributo correspondiente a la dimensión f de x_i como x_{if}.

Un cluster, $C \subseteq X$, es un subconjunto de objetos de la base de datos.

El centroide de un cluster, $\bar{C} = (1/|C|)\sum_{x \in C} x$, es el vector medio de todos los vectores del cluster. Asimismo, $\bar{X} = (1/N)\sum_{x \in X} x$ es el vector medio, o centroide, de toda la base de datos.

Llamamos partición parcial de una base de datos, $P^Y = \{C_1, C_2, \ldots, C_K\} : \bigcup_{C_k \in P^Y} C_k = Y, C_k \cap C_l = \emptyset \quad \forall k \neq l$ y $Y \subseteq X$, al conjunto de clusters disjuntos que contienen un subconjunto de datos de la base de datos. Una partición total, o simplemente una partición, es una partición que contiene todos los objetos de la base de datos, P^X. Por claridad, tanto las particiones parciales como totales serán mencionadas como particiones donde las diferencias sean irrelevantes o el contexto aclare el tipo de partición referido. Denominaremos partición correcta de una base de datos, P^*, aquella que describa los grupos naturales de la base de datos, es decir, cada cluster en P^* representará un grupo natural de objetos en la base de datos.

Una jerarquía, $H = \{P_1, P_2, \ldots, P_R\} \quad \forall P_r, P_s \quad r < s \Leftrightarrow \forall C_k \in P_r, \exists C_l \in P_s : C_k \subseteq C_l$, es un conjunto de particiones anidadas donde cualquier partición, excepto la primera, puede ser descrita a partir de un conjunto de uniones de clusters realizadas a la partición anterior.

Un índice de validación de clusters es una función, V(P), que mide la bondad o calidad de una partición. Salvo si expresamente se indica lo contrario, asumiremos que un valor menor de V denota una partición mejor.

2.2.2. Aplicaciones del clustering

El clustering o agrupamiento de datos se ha utilizado principalmente para tres propósitos fundamentales:

- Detectar la estructura subyacente en los datos: conocer mejor los datos, generar hipótesis, detectar anomalías e identificar las características principales.
- Clasificación natural: identificar el nivel de similitud de diferentes organismos (relaciones filogenéticas).
- Compresión: método para organizar los datos y resumirlos a través de prototipos o representantes de cluster.

Siguiendo cualquiera de estos objetivos el clustering se ha utilizado en numerosos campos científicos desde el análisis de imágenes hasta la biología y el estudio genético. Según Jain (2009) una búsqueda de los términos *data clustering* en *Google Scholar* da como resultado 1660 entradas sólo en el año 2007. Este dato da una pista acerca de la gran extensión del clustering.

A continuación enumeramos como ejemplo, y sin ánimo de ser exhaustivos, una serie de áreas en las que se ha utilizado el clustering.

- Biología: clustering de proteínas (Guralnik y Karypis, 2001; Paccanaro et al., 2006; Vijaya et al., 2006; Zhong et al., 2005), clustering de genes (Burke et al., 1999; Seal et al., 2005), filogenia (Kim et al., 2003; Planet, 2006) ...
- Procesamiento de imágenes (Chou et al., 2004; Ma y Sethi, 2007; Posse, 2001; Shi y Jitendra, 2000).
- Seguridad informática (Barbará y Jajodia, 2002; Julisch, 2003; Maloof, 2006; Perona et al., 2009).
- Psicología (Allik y McCrae, 2004; Holzinger y Harman, 1941; Stefurak et al., 2004).
- Medicina (John et al., 2003).
- Arqueología (Hall, 2004).
- Clustering de documentos (Aliguliyev, 2009; Maarek y Wecker, 1994; Slonim y Tishby, 2000).
- Marketing (Guralnik y Karypis, 2001).
- Web mining (Liu, 2007).
- Procesado de señal (Murtagh, 2009).
- Climatología (Gong y Richman, 1995; Posse, 2001)

Como se puede ver, el desarrollo del clustering se ha realizado gracias a investigadores de diversas disciplinas: taxónomos, sociólogos, psicólogos, biólogos, estadísticos, matemáticos, ingenieros, informáticos, médicos, etc. Dependiendo del área en la que se ha aplicado el clustering ha recibido diversos nombres como *Q-analysis*, *tipology*, *clumping* o *numerical taxonomy*.

2.2.3. Tipos de algoritmos

Debido al gran número de algoritmos de clustering y a las diversas estrategias seguidas a la hora de agrupar los datos, no es sencillo realizar una clasificación clara. La clasificación tradicional de los algoritmos los divide en dos categorías principales: jerárquicos y particionales.

Los algoritmos jerárquicos construyen una serie de particiones anidadas de forma aglomerativa o divisiva. En el primer caso los clusters se van uniendo formando particiones con cada vez menos clusters. En el segundo caso los clusters se van dividiendo de forma que las particiones tienen cada vez más clusters. La ventaja principal de este tipo de algoritmos es que es capaz de capturar la posible estructura jerárquica de los datos. La parte izquierda de la Figura 2.1 muestra unos datos con claro carácter jerárquico donde una solución de 3 clusters es tan buena como una de 6. Un algoritmo jerárquico puede crear una jerarquía de clusters donde se representen los distintos niveles de detalle en los que podemos estructurar los datos.

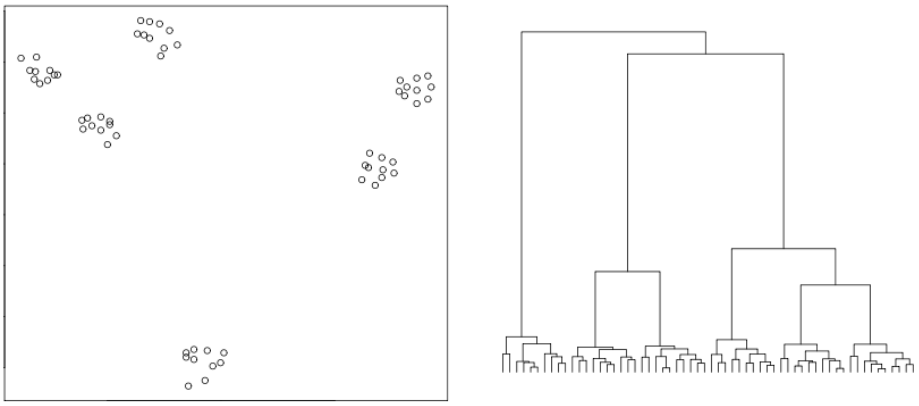

Figura 2.1: Una base de datos con estructura jerárquica y un posible dendrograma.

Típicamente una jerarquía de clusters se representa mediante un árbol denominado dendrograma. Los nodos hoja representan la primera partición de un proceso aglomerativo (o la última de uno divisivo), mientras que los nodos internos representan la unión de varios clusters, generalmente dos, en un proceso aglomerativo (o la división de un cluster en uno divisivo). La altura de cada nodo corresponde generalmente a la distancia entre sus nodos hijos. De este modo cada partición de la jerarquía se puede indicar mediante una recta horizontal que corte las distintas ramas del árbol. Para un análisis más detallado véase el Capítulo 4. La parte derecha de la Figura 2.1 muestra el dendrograma correspondiente a una posible jerarquía de los datos de la parte izquierda.

Por otra parte, un algoritmo particional genera una única partición de los datos con lo que en un caso como el de la Figura 2.1 deberá optar por un único nivel de detalle. De todos modos, este tipo de algoritmos son gene-

ralmente menos costosos que los jerárquicos y son ampliamente utilizados.

Una clasificación rigurosa y más detallada de los algoritmos de clustering no es tarea sencilla, pero existen criterios que permiten realizar cierta distinción entre ellos. Por ejemplo, los algoritmos de clustering utilizan distintas funciones objetivo, distintos modelos probabilísticos y distintos heurísticos. A continuación se revisan algunas de las aproximaciones principales.

- Algoritmos basados en densidad: Este tipo de algoritmos se basa en la idea de que un cluster es una región del espacio densa y aislada. Por lo tanto, detectan los clusters como puntos en el espacio conectados por zonas de una densidad mínima establecida como umbral. Este tipo de algoritmos puede detectar clusters de cualquier forma, pero no son adecuados para espacios con alta dimensionalidad. Algunos algoritmos de este tipo son DENCLUE (Hinneburg y Keim, 1998), DBSCAN (Ester et al., 1996) y OPTICS (Ankerst et al., 1999).

- Algoritmos de clustering de subespacios: Este tipo de algoritmos es útil para espacios de alta dimensionalidad ya que trata de detectar clusters en subespacios de baja dimensionalidad. Algunos algoritmos de este tipo son CLIQUE (Agrawal et al., 2005), SUBCLU (Kailing et al., 2004) y PROCLUS (Aggarwal et al., 1999).

- *Grid-clustering*: Este tipo de algoritmos dividen el espacio en celdas y analizan la densidad de cada una de las celdas. Normalmente este tipo de algoritmos consiguen reducir el coste computacional ya que el número de celdas no triviales, es decir, que contengan algún caso, es sensiblemente menor que el número de casos. STING (Wang et al., 1997) y WaveCluster (Sheikholeslami et al., 2000) son dos algoritmos de este tipo.

- Algoritmos basados en modelos probabilísticos: Algunos algoritmos como el EM (Dempster et al., 1977) o LDA (Blei et al., 2003) asumen que los datos han sido generados a partir de una mezcla de distribuciones y que cada cluster se describe a través de uno o varios de los componentes de la mezcla. Estos métodos tratan de inferir los parámetros de la mezcla.

- Algoritmos basados en la teoría de grafos y *spectral clustering*: Los algoritmos basados en la teoría de grafos representan una base de datos como un grafo donde los vértices corresponden a los casos y los pesos de las aristas representan la similitud (o la distancia) entre los

casos. El objetivo en estos casos es dividir los vértices en dos grupos de forma que la suma de los pesos de las aristas cortadas sea mínima. En muchas ocasiones el problema se resuelve mediante técnicas de *spectral clustering*, algoritmos que utilizan vectores propios de matrices calculadas de los datos. *Ratio cut* (Hagen y Kahng, 1992), *Normalized cut* (Shi y Jitendra, 2000; Yu y Shi, 2003) y MNCut (Meilă y Shi, 2001) son algoritmos de este tipo, así como el algoritmo optimizado de coloreado de grafos de Silva et al. (2006). El algoritmo propuesto por Ng et al. (2002) también se utiliza ampliamente.

- Algoritmos basados en la teoría de la información: Algunos algoritmos se formulan en base a la teoría de la información como por ejemplo el algoritmo *Minimum entropy method* de Roberts et al. (2001) o *Information bottleneck method* de Tishby et al. (1999).

- Algoritmos basados en la optimización: Algoritmos de optimización también han sido utilizados con el objetivo de particionar una base de datos (Bandyopadhyay y Saha, 2008; Laszlo y Mukherjee, 2007). Por ejemplo, tanto Handl y Knowles (2004), como más recientemente Aliguliyev (2009), utilizan una aproximación similar al ejecutar un algoritmo evolutivo combinado con varios índices de validación de clusters como función objetivo.

2.2.4. Algunos algoritmos de clustering

A continuación se describen en detalle 3 algoritmos de clustering que se utilizan en varios capítulos de esta tesis. El primero de ellos es un método general en el que se basan los algoritmos jerárquicos más utilizados. El segundo es el algoritmo de clustering por excelencia: K-means. Y finalmente, el tercero, es un algoritmo sencillo, pero no tan extendido, conocido con el nombre de *Fixed-Width* o *Leaders*.

El método jerárquico SAHN

Muchos de los algoritmos jerárquicos más utilizados como el *single-linkage*, *average-linkage*, *complete-linkage* o *Ward* se basan en un mismo método. Éste es el método jerárquico aglomerativo por excelencia, por lo que en muchos casos se conoce simplemente como el método aglomerativo. Sin embargo, siguiendo a Sneath y Sokal (1973), denominaremos este método como *Sequential Agglomerative Hierarchical Nonoverlapping* (SAHN).

El método es sencillo e intuitivo y se describe en el Algoritmo 2.1. En primer lugar se crea una partición donde todos los clusters están compuestos por un único caso. Posteriormente, y de forma iterativa, se van uniendo los dos clusters más cercanos hasta que la partición obtenida tenga un único cluster con todos los casos de la base de datos. Como consecuencia se crea una jerarquía con tantas particiones como casos donde en cada partición se unen exactamente 2 clusters de la partición anterior. Por lo tanto, el dendrograma resultante es un árbol binario.

Algoritmo 2.1 SAHN

1: Calcular la matriz de distancias correspondiente a los datos
2: Crear un cluster con cada caso
3: **for** $i = 2$ **to** N **do**
4: Unir los dos clusters más cercanos en un nuevo cluster
5: Actualizar la matriz de distancias
6: **end for**

Existen diversas medidas para calcular la distancia o proximidad entre clusters y dependiendo de la medida utilizada este método general permite obtener distintos algoritmos. Por ejemplo, el algoritmo *single-linkage* considera que la distancia entre clusters es la distancia entre los objetos más cercanos de cada cluster. Al contrario, *complete-linkage* define la distancia entre dos clusters como la distancia entre sus dos casos más lejanos.

Estos dos algoritmos se pueden analizar desde el punto de vista de la teoría de grafos si cada objeto se representa como un vértice del grafo. Supongamos un proceso iterativo donde se van añadiendo aristas en función de la distancia entre objetos (de menor distancia a mayor distancia). El estado del grafo en cualquiera de las iteraciones representa una partición. En el caso del *single-linkage* dos objetos estarán en el mismo cluster si existe un camino entre los vértices que representan los objetos. En cambio, en el caso de *complete-linkage* los vértices que representan los objetos de un cluster formarán un subgrafo completo.

Aparte de estas dos, existen otras muchas medidas de proximidad entre clusters, y por lo tanto, algoritmos basados en SAHN. Uno de ellos es el algoritmo *Ward* que, al igual que el algoritmo K-means, trata de minimizar el error cuadrático medio. Otros cuatro algoritmos comunes denominados UPGMA (o *average-linkage*), WPGMA, UPGMC y WPGMC (Jain y Dubes, 1988; Sneath y Sokal, 1973) surgen de la combinación de dos criterios. El prefijo "U" (*Unweighted*) significa que todos los casos pesan igual, mientras que "W" (*Weighted*) significa que todos los clusters pesan igual, de forma

que los casos en los clusters pequeños pesan más. El sufijo "A" (*Average*) significa que la distancia se calcula mediante la media entre todos los casos de cada cluster, mientras que "C" (*Centroid*) significa que la distancia entre clusters se define como la distancia entre centroides. "PGM" es común a los cuatro y significa *Pair Group Method*.

Aunque los métodos basados en centroides tienen la ventaja de tener una interpretación geométrica directa Jain y Dubes (1988) argumentan que sólo deberían ser usados cuando los casos se representan como vectores de características y la distancia utilizada es el cuadrado de la euclidiana. Además, pueden producir jerarquías un tanto "extrañas" ya que no son monótonas, es decir, no se garantiza que la altura de un nodo sea siempre igual o mayor que la de sus sucesores. Por consiguiente, algunos autores han mostrado sus reservas acerca de los algoritmos basados en centroides (Anderberg, 1973; Sneath y Sokal, 1973; Williams y Lance, 1977), pero los estudios comparativos no sugieren el abandono de tales algoritmos (Jain y Dubes, 1988).

El método SAHN necesita almacenar una matriz con las distancias entre todos los casos de la base de datos ($\frac{N(N-1)}{2}$ valores suponiendo una distancia simétrica). Dado que en cada iteración se unen dos clusters, es necesario actualizar la matriz de distancias. Por un lado, se eliminan las entradas correspondientes a los clusters unidos y, por otro lado, se crea una nueva entrada correspondiente al nuevo cluster. La mayoría de las medidas de proximidad entre clusters permiten llevar a cabo esta operación de forma combinatoria. Es decir, los valores correspondientes a la nueva entrada se calculan combinando valores de la matriz anterior. Es obvio que para el *single-linkage* cada valor de la nueva entrada corresponderá al mínimo de las dos entradas eliminadas y que para el *complete-linkage* será el máximo.

Para otras medidas de proximidad se pueden definir otro tipo de fórmulas matemáticas. Sin embargo, Lance y Williams (1967) propusieron una fórmula con 4 coeficientes que sirve para actualizar la matriz de distancias y se adecúa a la mayor parte de medidas de proximidad conocidas. La instanciación de los 4 coeficientes permite adaptar la fórmula a la medida deseada. A continuación se describe la fórmula y en la Tabla 2.1 se indican los valores de los coeficientes para las medidas de proximidad mencionadas anteriormente.

Supongamos que hemos unido los clusters C_i y C_j para crear el nuevo cluster C_{ij}. Entonces la distancia entre el nuevo cluster C_{ij} y un cluster C_k se puede calcular como

$$\mathrm{d}(C_{ij}, C_k) = \alpha_i \,\mathrm{d}(C_k, C_i) + \alpha_j \,\mathrm{d}(C_k, C_j) + \beta \,\mathrm{d}(C_i, C_j) + \gamma |\,\mathrm{d}(C_k, C_i) - \mathrm{d}(C_k, C_j)|$$

La fórmula de Lance-Williams es una forma de simplificar los métodos

Algoritmo	α_i	α_j	β	γ
Single-linkage	1/2	1/2	0	-1/2
Complete-linkage	1/2	1/2	0	1/2
UPGMA	$\frac{\lvert C_i\rvert}{\lvert C_{ij}\rvert}$	$\frac{\lvert C_j\rvert}{\lvert C_{ij}\rvert}$	0	0
WPGMA	1/2	1/2	0	0
UPGMC	$\frac{\lvert C_i\rvert}{\lvert C_{ij}\rvert}$	$\frac{\lvert C_j\rvert}{\lvert C_{ij}\rvert}$	$\frac{-\lvert C_i\rvert\lvert C_j\rvert}{\lvert C_{ij}\rvert^2}$	0
WPGMC	1/2	1/2	-1/4	0
Ward	$\frac{\lvert C_i\rvert+\lvert C_k\rvert}{\lvert C_{ij}\rvert+\lvert C_k\rvert}$	$\frac{\lvert C_j\rvert+\lvert C_k\rvert}{\lvert C_{ij}\rvert+\lvert C_k\rvert}$	$\frac{-\lvert C_k\rvert}{\lvert C_{ij}\rvert+\lvert C_k\rvert}$	0

Tabla 2.1: Valores de los coeficientes para la actualización de la matriz de distancias del método SAHN.

combinatorios en una única fórmula, pero también permite definir con sencillez nuevas medidas de proximidad conceptualmente complejas. Además, permite utilizar técnicas de optimización para instanciar los 4 coeficientes de forma que se pueda definir una medida de proximidad adaptada a cada problema.

El algoritmo K-means

Aunque el algoritmo K-means es conocido desde hace más de 5 décadas sigue siendo uno de los algoritmos de aprendizaje automático más utilizados y mejor valorados (Wu et al., 2008). Dado un conjunto de datos y un valor K, el algoritmo genera una partición de los datos con K clusters. El objetivo de K-means es minimizar el error cuadrático medio de todos los clusters. El error cuadrático de un cluster se define como la media de los cuadrados de las diferencias de los objetos de un cluster al centroide del cluster. El centroide de un cluster es el punto medio de todos los objetos del cluster.

El problema que trata de resolver el algoritmo K-means es NP-difícil, incluso para el caso de $K = 2$, por lo que K-means es un algoritmo voraz (*greedy*) que devuelve un óptimo local. No obstante, estudios recientes demuestran que si los clusters están bien separados el algoritmo K-means obtendrá el óptimo global con alta probabilidad (Meilă, 2006).

El Algoritmo 2.2 describe los pasos seguidos por K-means. En primer lugar se seleccionan K casos como centroides iniciales. Posteriormente, en un segundo paso, se asigna cada caso al cluster representado por el centroide más cercano. Finalmente, en el tercer paso se vuelven a calcular los centroides como la media de todos los casos asignados a cada cluster. Si los centroides calculados difieren de los anteriores se vuelve al paso 2, si no, el

algoritmo devuelve como resultado la última partición calculada.

Algoritmo 2.2 K-means

1: Seleccionar K casos como centroides iniciales
2: **repeat**
3: Asignar cada caso al cluster con centroide más cercano
4: Recalcular los centroides de cada cluster
5: **until** los centroides no se han modificado

Dado que al aumentar el valor de K el error cuadrático medio se reduce (siendo 0 con $K = N$) éste puede ser minimizado sólo para un valor prefijado de K. Dicho de otro modo, el error cuadrático medio no sirve para comparar la bondad de particiones con distinto número de clusters.

El algoritmo K-means requiere tres parámetros de usuario. El primero, y el principal, es K, el número de clusters de la partición. Existen varios heurísticos para calcular el valor de K ya que no existe método matemático válido (Tibshirani et al., 2001b). Una aproximación habitual es ejecutar el algoritmo en repetidas ocasiones con diferentes valores de K y utilizar algún criterio para seleccionar el más adecuado, ya sea mediante un experto en el área o bien mediante técnicas de validación. La validación de clustering se tratará en mayor profundidad en la Sección 2.3.

Otro parámetro a definir es la selección inicial de centroides. Dicha selección condiciona el óptimo local al que llega el algoritmo. Una posible forma de minimizar el impacto producido por este parámetro es ejecutar el algoritmo varias veces con distintas inicializaciones y seleccionar la partición que tenga un menor error cuadrático medio. Por otra parte existen diversas aproximaciones para seleccionar los centroides iniciales de forma inteligente (Khan y Ahmad, 2004; Su y Dy, 2004).

El tercer parámetro a definir es la función de distancia entre los casos de la base de datos. La distancia más habitual es la euclidiana con lo que se generan clusters de forma esférica. Para encontrar clusters hiper-elipsoidales Mao y Jain (1996) utilizaron la distancia de Mahalanobis. Otras distancias pueden servir para otros propósitos.

Existen multitud de variantes de K-means. Por ejemplo el Fuzzy c-means es una variante del K-means para clustering difuso (Bezdek, 1981; Dunn, 1973). Otra variante, llamada X-means, selecciona automáticamente el valor adecuado de K (Pelleg y Moore, 2000). También existen variantes para reducir el coste computacional de K-means o para reducir la cantidad de memoria necesaria (Bradley et al., 1998; Pelleg y Moore, 1999).

Otras variantes modifican la forma de calcular el centroide de cada clus-

ter. Por ejemplo, el algoritmo PAM (o K-medoids) utiliza como centroide el caso con menor distancia media al resto de los casos del cluster (Kaufman y Rouseew, 1990). A diferencia de lo que ocurre con K-means los centroides de los clusters serán siempre casos de la base de datos. Esta versión permite la ejecución del algoritmo sobre datos que no estén representados como vectores y de los que no se pueda calcular el caso medio como, por ejemplo, secuencias de longitud variable. También permite la ejecución del algoritmo cuando se desconozcan los datos y sólo se conozca la matriz de distancias correspondiente.

El algoritmo *Fixed-Width*

El algoritmo *Fixed-Width* está mucho menos extendido que los dos anteriores. Sin embargo, dadas sus características, es un algoritmo útil en ciertos entornos. Su principal bondad es que su coste computacional es reducido ya que su complejidad es de $O(K \cdot N)$ y generalmente $K \ll N$.

El algoritmo es sencillo y se describe en el Algoritmo 2.3. El algoritmo requiere un parámetro definido por el usuario, denominado radio. Se empieza definiendo un conjunto de clusters con un único cluster cuyo centro es el primer caso analizado. Posteriormente se analizan el resto de casos de la muestra de entrenamiento una única vez. Cada caso se va comparando con los centros de los clusters definidos hasta el momento. Si la distancia al más cercano es menor que el radio definido como parámetro se asigna el caso a ese cluster. Si no, se crea un nuevo cluster cuyo centro será el caso analizado.

Algoritmo 2.3 *Fixed-Width*

1: $O = \{x_1\}$
2: Crear un cluster asociado a x_1
3: **for** $i = 2$ **to** N **do**
4: $\quad \hat{o} = \arg\,\text{mín}_{o \in O}\, \mathrm{d}(x_i, o)$
5: $\quad$ **if** $\mathrm{d}(x_i, \hat{o}) <$ radio **then**
6: $\quad\quad$ Asignar x_i al cluster asociado a $\hat{o}$
7: $\quad$ **else**
8: $\quad\quad$ $O = O \cup x_i$
9: $\quad\quad$ Crear un cluster asociado a x_i
10: $\quad$ **end if**
11: **end for**

Los clusters generados serán esferas con centro en un caso de la muestra y radio definido por el usuario como parámetro. Sin embargo, este algoritmo

tiene severos inconvenientes. Por un lado, según el orden en que se analizan los casos el resultado puede diferir. Además, este orden también influye en el cluster que se asignará a cada caso, pudiendo darse la situación que un caso esté asignado a un cluster cuyo centro no sea el más cercano a éste. De todos modos, esto se soluciona de forma sencilla, sin aumentar la complejidad del algoritmo, realizando una nueva pasada de los datos y reasignando los casos a los clusters adecuados.

Otro inconveniente del algoritmo es que no es adecuado para clusters de diferente tamaño ya que el parámetro radio establece un tamaño fijo para todos. Este hecho generalmente provoca el sobreparticionamiento de los clusters más grandes. El hecho de que los clusters sean esféricos también imposibilita la detección de clusters no convexos.

Sin embargo, en algunos contextos particulares los inconvenientes del algoritmo no son importantes y el coste computacional reducido del algoritmo puede justificar su uso. Por ejemplo, tal y como se muestra en el Capítulo 7, el algoritmo puede ser adecuado para la detección de casos atípicos dado que en ese caso sólo interesan los clusters más pequeños, los cuales no se ven afectados por el problema del sobreparticionamiento.

Vijaya et al. (2004) proponen una modificación del algoritmo, a la que llaman *Leaders-Subleaders*, que consigue mejorar sus resultados de forma significativa. En particular, convierten el algoritmo en un algoritmo jerárquico. En primer lugar ejecutan el algoritmo con un valor alto para el parámetro radio. Posteriormente vuelven a ejecutar el algoritmo para cada uno de los clusters obtenidos en el primer paso utilizando un valor menor para el parámetro radio. Según los autores los resultados mejoran y el coste computacional no crece de forma significativa.

Por otra parte, al igual que con K-means, la función de distancia utilizada puede variar la forma de los clusters y la partición obtenida. Sin embargo, dado que no es un algoritmo tan extendido, no existe tal variedad de trabajos sobre el tema.

2.2.5. Retos fundamentales del clustering

El clustering ha demostrado ser un problema de difícil solución. Aunque otras áreas del aprendizaje automático, la estadística o la minería de datos están teóricamente bien analizadas y se basan en sólidos y claros análisis matemáticos, no ocurre lo mismo con el clustering (Kettenring, 2006). Jain (2009) atribuye la dificultad del clustering a la inherente imprecisión de la definición de cluster y a la dificultad de definir funciones de distancia y funciones objetivo adecuadas.

Como consecuencia, existen multitud de retos en el área del clustering como por ejemplo el autoescalado de las variables, el clustering semi-supervisado, el clustering múltiple, el clustering de variables, la reducción de dimensiones, el clustering de datos heterogéneos, la extracción de particiones a partir de jerarquías, el clustering de bases de datos de gran tamaño, la validación e interpretación de los resultados... (Jain, 2009; Kettenring, 2006).

Sin embargo, en los apartados siguientes nos vamos a centrar en los retos fundamentales del clustering. A pesar de los años de estudio del clustering y de la gran cantidad de algoritmos diseñados, algunas cuestiones básicas del clustering siguen sin ser resueltos de una forma satisfactoria. Es por ello que Kettenring (2006) afirma que el clustering es casi un arte: *"Effective clustering is very much an imprecise art"* .

Formalización del problema

Existen varias aproximaciones para formalizar los fundamentos del clustering de forma teórica (Ackerman y Ben-David, 2008, 2009; Fisher y Van Ness, 1971; Kleinberg, 2002; Meilă, 2005). Sin embargo, aunque la mayoría podemos formarnos una idea de lo que es el clustering, su formalización teórica y la búsqueda de bases matemáticas al respecto aún no ha dado unos resultados convincentes.

Hace ya cuatro décadas Fisher y Van Ness (1971) analizaron formalmente los algoritmos de clustering con el objetivo de compararlos. Ya que parece imposible encontrar un algoritmo que sea el mejor, trataron de definir una serie de criterios que un algoritmo ha de cumplir para que sea admisible. Definieron 9 criterios de admisibilidad que evalúan la calidad de los algoritmos de clustering. Algunos de los criterios tratan de medir la sensibilidad de los algoritmos respecto a cambios en los datos. Otros tratan de definir ciertas propiedades estructurales interesantes que las particiones de salida deberían cumplir. Veamos, como ejemplo, algunos de dichos criterios:

- Proporción de cluster: Un algoritmo cumple este criterio si los límites de los clusters no se alteran aunque los casos de algunos de los clusters se multipliquen un número arbitrario de veces.
- Omisión de cluster: Un algoritmo cumple este criterio si la eliminación de un cluster de los datos de entrenamiento no provoca ningún cambio en la forma de particionar el resto de los clusters.
- Monotonía: Un algoritmo cumple este criterio si una transformación

monótona en la matriz de distancias de los casos de entrenamiento no modifica la partición obtenida.

- Convexo: Un algoritmo cumple este criterio si las envolturas convexas (*convex hull*) de los clusters de las particiones que genera no se cruzan.

Los autores demostraron que no se pueden construir algoritmos que cumplan ciertos conjuntos de criterios de admisibilidad. Por ejemplo, demostraron que un algoritmo jerárquico no puede cumplir el criterio de monotonía. No obstante, no todos los criterios son deseables en cualquier situación por lo que es necesario analizar cada aplicación particular.

Más recientemente Kleinberg (2002) propuso un sistema similar y definió 3 criterios o axiomas. Los resultados coinciden, de algún modo, con los de Fisher y Van Ness, ya que demuestran que es imposible construir un algoritmo que cumpla los 3 axiomas. Veamos cuáles son esos 3 axiomas:

- Invarianza de escala: Un cambio de la escala de la función de distancia no debe cambiar los resultados del algoritmo.
- Riqueza: El algoritmo debe ser capaz de particionar los datos de entrada en todas las formas posibles. Es decir, cualquier partición válida de los datos debe ser una salida posible del algoritmo.
- Consistencia: Los resultados de un algoritmo de clustering no deben variar si reducimos las distancias intra-cluster (casos en un mismo cluster) y aumentamos las distancias inter-cluster (casos en distintos clusters).

Kleinberg demuestra que aunque los axiomas se pueden cumplir de dos en dos, es imposible que se cumplan los 3 a la vez. Por ello, llama a su trabajo "Una teoría de imposibilidad para el clustering". También presenta versiones relajadas de los axiomas y demuestra que algunos algoritmos ampliamente utilizados como el *single-linkage* o el K-means cumplen los 3 axiomas haciendo uso de la versión relajada de alguno de ellos.

Por otra parte, Meilă (2005) define una serie de axiomas que deberían cumplir las medidas de similitud de particiones (ver Sección 2.3.2). Una de las conclusiones principales del trabajo es que no puede existir una medida de similitud que cumpla 3 axiomas intuitivamente deseables.

Estos resultados parecen confirmar la dificultad del clustering y sugieren que la búsqueda de un algoritmo que cumpla una serie de axiomas simples y aparentemente lógicos es imposible. Sin embargo, el reciente trabajo de Ackerman y Ben-David (2008) da un giro a esta perspectiva.

Ackerman y Ben-David argumentan que el resultado pesimista de Kleinberg no refleja un problema inherente del clustering sino que surge debido a la particular definición de sus axiomas. En concreto, argumentan que el axioma de Consistencia es inapropiado. La solución que proponen cambia el punto de vista de trabajos anteriores ya que en lugar de definir los criterios básicos de los algoritmos de clustering, los definen para las medidas de calidad de las particiones, o índices de validación de clusters. Como su propio nombre indica, estas medidas miden la calidad de las particiones. En la Sección 2.3 se describen con mayor detalle este tipo de medidas relacionadas con la validación de clustering.

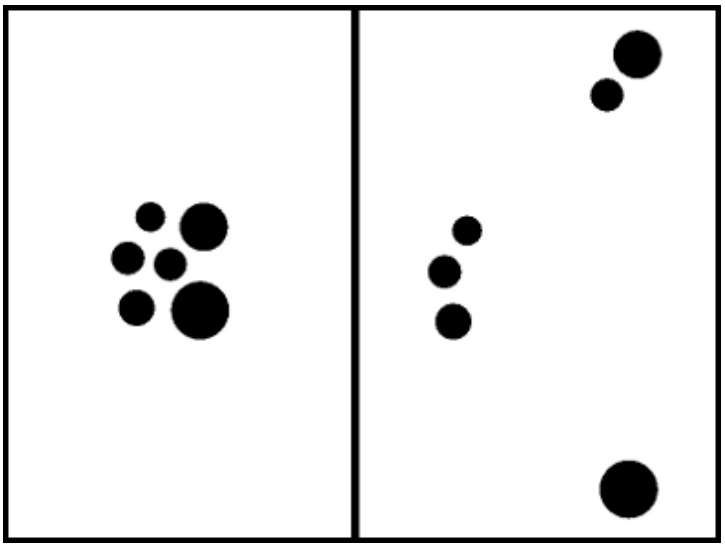

Figura 2.2: Una modificación consistente de una partición de 6 clusters.

Ackerman y Ben-David adaptan los axiomas de Kleinberg al nuevo esquema y demuestran que incluso una medida de calidad sencilla las cumple. El problema de la versión de Kleinberg radica, como ya se ha mencionado, en la definición del axioma de Consistencia. Ackerman y Ben-David ilustran el problema con un ejemplo como el de la Figura 2.2. La parte izquierda representa una partición de 6 clusters mientras que la parte derecha muestra una modificación consistente, según el criterio de Kleinberg. Ackerman y Ben-David argumentan que aunque la calidad de la partición de la derecha no debe ser menor que el de la izquierda, una partición de 3 clusters debería ser aún mejor. Es decir, a pesar de que se ha realizado un cambio consistente, no parece lógico exigir que un algoritmo de clustering no modifique su partición de salida. Esta grieta en los axiomas de Kleinberg abre una nueva

vía a la definición de unos axiomas para el clustering y su formalización teórica.

Otro concepto teórico importante es el de la agrupabilidad (*clusterability* o *cluster tendency*). Es decir, la tendencia que tiene una base de datos a estar estructurada en grupos y ser, por tanto, propensa a ser particionada. Ackerman y Ben-David (2009) realizan un estudio teórico interesante sobre los distintos conceptos de agrupabilidad existentes. Una de sus conclusiones más importantes es que si una base de datos tiene realmente una estructura de grupos, se puede obtener la partición correcta (o una similar) con un coste computacional bajo.

Definición de cluster

Ya hemos mencionado la dificultad de una definición precisa de cluster. Aunque a los humanos nos sea generalmente sencillo detectar clusters de forma intuitiva en entornos de 2 o 3 dimensiones, una definición formal de cluster no es tan sencilla. Los clusters pueden ser de diversa forma, tamaño y densidad e incluso pueden encontrarse parcial, o totalmente, solapados.

En muchos casos, el solapamiento entre clusters es lo que provoca un alto nivel de subjetividad en la determinación de los clusters. Para ilustrar el problema se describen a continuación una serie de ejemplos en un espacio bidimensional donde todos los clusters son gaussianos, con lo que se simplifica el problema provocado por la forma de los clusters.

Supongamos una base de datos generada de forma sintética donde cada cluster está formado por un número arbitrario de casos obtenidos a partir de una variable aleatoria con distribución normal. La matriz de covarianza de la distribución es igual para todos los clusters y la media se obtiene de forma aleatoria. ¿Qué ocurre si dos clusters tienen casualmente el mismo valor para la media? En ese caso los dos clusters serán absolutamente indistinguibles. De hecho, dado que coinciden en ambos parámetros (media y matriz de covarianza) parece lógico considerarlos como un único cluster.

Pero, ¿qué ocurre si las medias de dos clusters no coinciden, pero son muy similares? Puede ocurrir que a pesar de la similitud de los clusters pueda detectarse que son realmente dos clusters, sobre todo si el observador conoce que todos los clusters siguen una distribución normal. Pero, también puede ocurrir que sean absolutamente indistinguibles porque la diferencia entre las medias sea lo suficientemente pequeña comparada con la dispersión provocada por la matriz de covarianza. Por lo tanto ¿cuándo deberíamos considerarlos un único cluster y cuándo dos clusters solapados? La pregunta no tiene una única respuesta, sino que existe una respuesta por cada

observador.

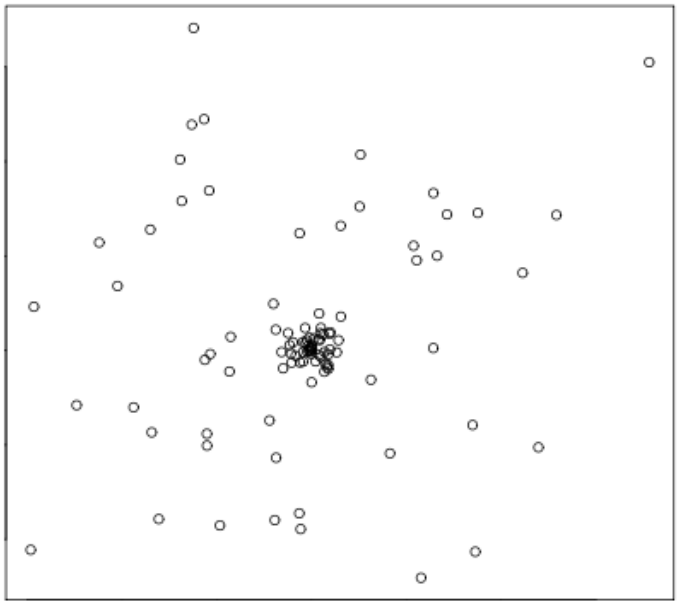

Figura 2.3: Dos clusters con el mismo centro, pero distinta varianza.

Ocurre algo similar cuando las medias coinciden, pero las matrices de covarianza difieren. La Figura 2.3 ilustra el problema. La figura muestra dos clusters con la misma forma esférica (matriz de covarianza diagonal) pero distinto tamaño y densidad. En este caso los dos clusters se pueden distinguir, pero si mantenemos los parámetros y reducimos el número de casos del cluster más pequeño llegará un momento en que las densidades se equiparen y, por lo tanto, los clusters sean indistinguibles.

Estos dos ejemplos ponen en relieve otra cuestión. Supongamos dos clusters solapados, pero suficientemente separados, de tal manera que no exista duda acerca de que son dos clusters y no uno. Sirva como ejemplo la Figura 2.4. Aunque no existan grandes dudas acerca del tamaño, forma y densidad de los clusters, hay una región del espacio en la que no podemos separar los clusters. En efecto, todos los casos que se encuentren en la región solapada serán imposibles de etiquetar con total fiabilidad. La aproximación habitual suele consistir en trazar una línea recta que separe ambos clusters en la zona solapada y asignar la etiqueta a los casos dependiendo del lado en el que estén. Sin embargo, la intuición nos dice que así no acertaremos a etiquetar

cada caso según la distribución de la que ha sido generado.

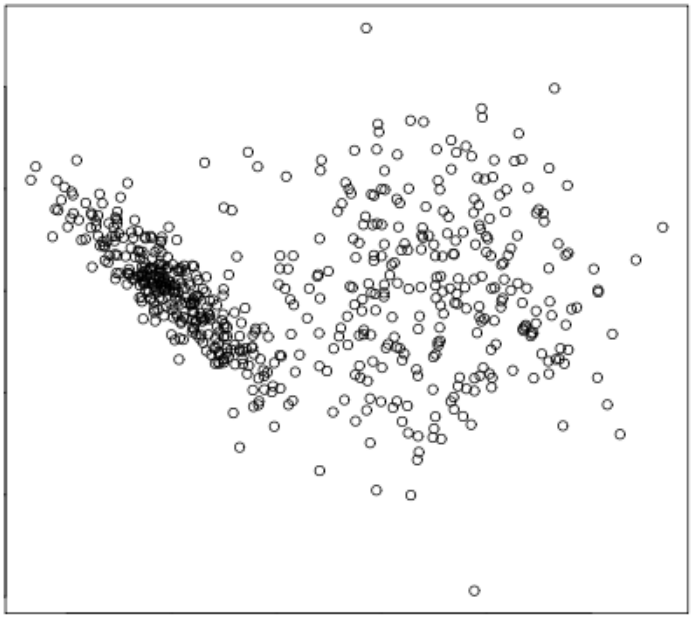

Figura 2.4: Dos clusters ligeramente solapados.

Otra posible solución, seguramente más acertada, es utilizar una partición difusa o *fuzzy*, donde cada caso tiene una probabilidad de pertenencia a cada cluster. Los casos en la región solapada tendrán una probabilidad menor que 1 para cada uno de los dos clusters y más cercana a 0,5 según se acerquen al punto central de la región solapada.

De todos modos, surgen problemas de identificación de clusters sin tener que irse a niveles de solapamiento tan extremos como algunos de los descritos en los párrafos anteriores. Veamos otro ejemplo descrito por Jain (2009). La Figura 2.5 muestra una base de datos generada a partir de 6 distribuciones gaussianas. Se podría argumentar que la base de datos está compuesta por 6 clusters. Sin embargo, dado que los clusters están parcialmente solapados, las particiones de 2 y 5 clusters mostradas en la figura también parecen correctas. Asimismo, aunque no se muestren en la figura, podríamos definir particiones de 3 o 4 clusters igual de justificables.

En definitiva, la definición precisa de cluster no es tarea sencilla por lo que la subjetividad sigue ejerciendo una gran influencia en los problemas de clustering. Este hecho es el origen de que la determinación automática del número de clusters en una base de datos sea una de las tareas más estudiadas

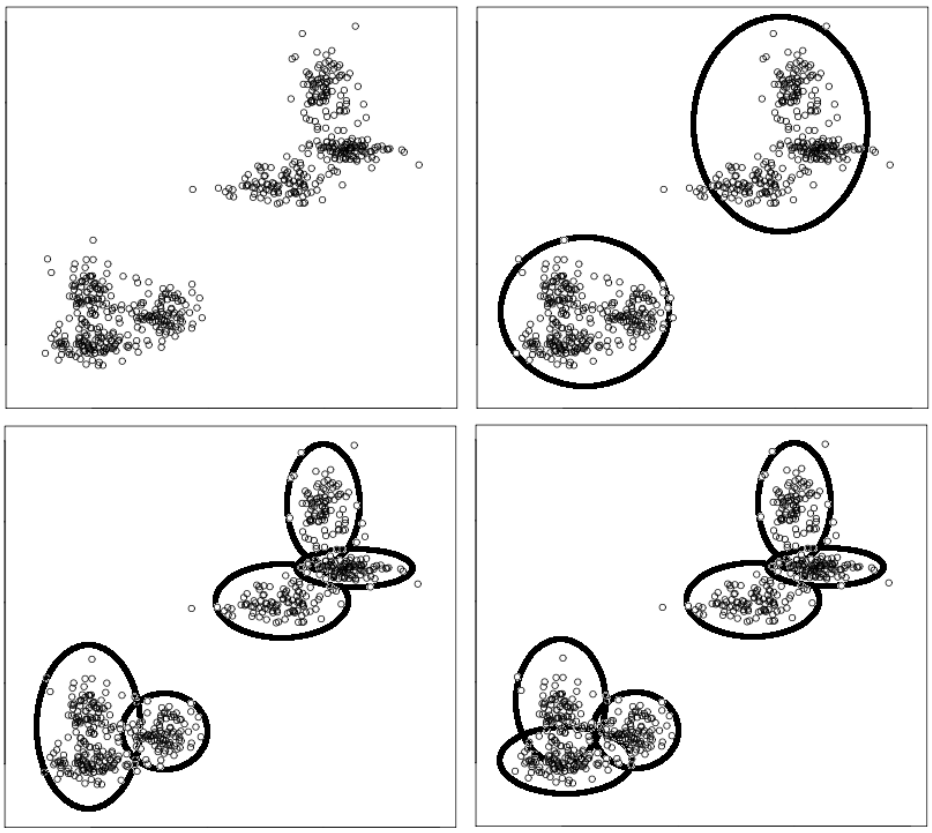

Figura 2.5: Base de datos generada a partir de 6 distribuciones gaussianas. Las particiones de 2, 5 y 6 clusters parecen todas válidas.

en el área del clustering.

Determinación del número de clusters

Existen diversas aproximaciones al cálculo del número de clusters de una base de datos. Una de las principales es la ejecución repetida de un algoritmo de clustering con distintos valores de K. Un análisis para medir la bondad de las particiones será el encargado de decidir el número de clusters al seleccionar la partición más adecuada. En el Capítulo 5 se argumenta que estas técnicas fallan si el algoritmo utilizado no es capaz de particionar los datos de forma adecuada.

La estabilidad de los algoritmos de clustering también se ha utilizado

para calcular el número de clusters de una muestra (Ben-Hur et al., 2002; Lange et al., 2004). Este método parte del supuesto de que un algoritmo será menos sensible a variaciones en los datos de entrada cuando se le indique el valor correcto de K. Aunque la idea sea intuitivamente correcta sufre de la misma dependencia del algoritmo de clustering que la técnica anterior. Si el algoritmo no es adecuado para particionar los datos, el método falla.

Otras técnicas se basan en métodos estadísticos que suponen ciertas características a los clusters, como por ejemplo la normalidad (Lago-Fernández y Corbacho, 2010). Existen también métodos visuales, como el VAT (Hathaway et al., 2006). Este método reorganiza la matriz de distancias y la representa gráficamente de forma que sea sencillo estimar el número de clusters en los datos y su tamaño.

Sin embargo, todas estas técnicas pierden su valor cuando los clusters no se pueden definir con claridad. Obviamente, el número de clusters no se puede calcular si el mismo concepto de cluster no está claro. Por lo tanto, este problema tan presente en el clustering no podrá ser resuelto si no se resuelve antes el problema de la definición de cluster de forma que desaparezca toda subjetividad posible. A su vez, esto no parece posible sin una base matemática adecuada y sólida.

Representación de los datos

La forma en que se representan los datos que se desean agrupar es un apartado fundamental del clustering. En realidad, la representación de los datos se considera un paso previo del proceso de la minería de datos. Sin embargo, los efectos de este proceso son de especial relevancia para el clustering por lo que realizamos una breve reflexión sobre el tema en este apartado. Aunque hoy en día los datos son tan heterogéneos que se pueden representar de forma muy variopinta, en este apartado nos centramos en el formato más común en el aprendizaje automático. Es decir, la representación en formato de tabla donde cada caso se representa como un vector de características de longitud fija.

La representación de los datos ejerce una influencia fundamental en cualquier proceso de extracción de conocimiento por lo que es habitual la eliminación de variables redundantes o altamente correladas, así como la creación de nuevas variables derivadas de otras. Un proceso muy habitual es la selección de un subconjunto de las variables en lo que se conoce como *Feature Subset Selection* (Liu y Motoda, 1998).

Para ilustrar la importancia de la representación de los datos en el clustering mostramos el ejemplo de la Figura 2.6. Un algoritmo como K-means

no es capaz de obtener la partición correcta de dos clusters correspondiente a los datos representados en la parte izquierda de la figura. La partición obtenida dividirá los dos anillos de forma que cada cluster se componga de aproximadamente la mitad de cada uno. Sin embargo, si los mismos datos se representan mediante los dos vectores propios principales de la matriz de similitud utilizando una función de base radial (RBF) obtenemos una distribución como la mostrada en la parte derecha de la figura. Estos datos son mucho más sencillos de analizar por cualquier algoritmo de clustering.

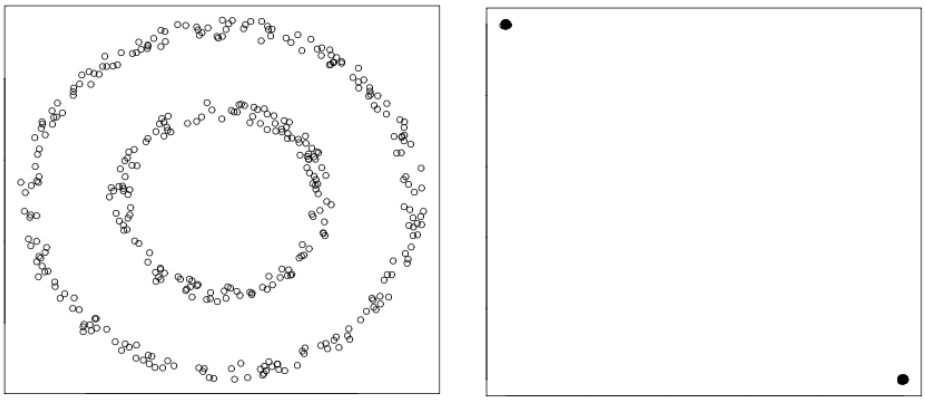

Figura 2.6: Dos representaciones distintas de los mismos datos.

Otra cuestión íntimamente relacionada con la representación de los datos es que unos mismos datos pueden ser agrupados de diversas formas dependiendo del propósito del usuario. Por ejemplo, supongamos una base de datos con características sobre ciertos animales. Las características representan información tanto sobre el aspecto como la actividad de los animales. Un mismo algoritmo puede generar dos particiones distintas, ambas de dos clusters y ambas válidas, dependiendo del peso que se le quiera dar a cada tipo de característica. Si damos mayor importancia a las características de aspecto el algoritmo podría, por ejemplo, agrupar mamíferos por un lado y aves por otro. Sin embargo, primando las variables de actividad podríamos obtener una partición que agrupe depredadores y presas. Las dos particiones son correctas y deducibles de los datos. La partición que debe generar el algoritmo depende del criterio del usuario y se puede conseguir mediante el adecuado preproceso de los datos.

A este respecto, Dasgupta y Ng (2009) proponen un algoritmo que es capaz de agrupar documentos siguiendo distintos criterios. Para ello, ofre-

cen al usuario de forma automática una lista de palabras que caracterizan las distintas posibilidades de agrupamiento. El usuario podrá seleccionar la posibilidad que mayor relación tenga con su objetivo.

2.3. Validación de clustering

El objetivo de la validación de clustering es medir de forma objetiva la bondad o calidad del resultado obtenido por un algoritmo de clustering. Como se ha descrito en este mismo capítulo, este resultado puede ser una partición o una jerarquía de particiones.

En realidad, la validación es un paso del proceso completo de minería de datos. Sin embargo, debido a la dificultad de formular una base matemática sólida y a conseguir una definición precisa de cluster, la validación es una parte fundamental en el clustering. La dificultad de determinar el número correcto de clusters en unos datos determinados ha sido una de las razones que han provocado el desarrollo de este área.

La validación de clustering se ha clasificado tradicionalmente en 3 categorías: externa, interna y relativa. Sin embargo, aunque las dos primeras categorías parecen estar claramente definidas, existe cierta confusión respecto a la tercera (Halkidi et al., 2001; Jain y Dubes, 1988; Yeung et al., 2001).

La validación externa asume el conocimiento de la partición correcta de los datos. Es decir, la verdadera partición es conocida (lo que se conoce como *ground truth* o *standard gold*). Evidentemente, en un contexto real de aplicación de clustering la partición correcta no es conocida, pero puede que en un contexto experimental de evaluación de algoritmos de clustering sí lo sea. El problema principal de esta aproximación es que, como hemos mencionado en la Sección 2.2.5, no siempre es posible determinar una única partición correcta. El solapamiento de los clusters o los distintos objetivos a la hora de particionar unos datos pueden imposibilitar la definición objetiva de una única partición correcta. Por otra parte, como ya hemos mencionado, algunos datos tienen una estructura jerárquica que hace imposible definir una única partición correcta (Dom, 2001).

La validación interna no necesita el conocimiento de la partición correcta ya que se basa únicamente en los datos agrupados. Es decir, evalúa la partición en base a los datos y las distancias entre ellos. La ventaja principal de esta aproximación es que no requiere el conocimiento de la partición correcta por lo que se puede utilizar en aplicaciones reales y no sólo en experimentos de laboratorio. Su desventaja principal es que evalúa la partición sobre los

mismos datos que se han utilizado para construirla.

Desafortunadamente, como ya observan Yeung et al. (2001), evaluar la adecuación de una partición sobre los datos de la misma partición evaluada es equivalente a evaluar los clusters bajo una diferente función objetivo. Por ejemplo, si utilizamos como medida de bondad de una partición el error cuadrático medio, es de suponer que un algoritmo como K-means obtenga mejores resultados que el *single-linkage*.

La validación, tanto externa como interna, se realiza a través de índices que dada una partición calculan una medida de la calidad o bondad de dicha partición. Si el valor devuelto por el índice es inusualmente alto (o bajo), podemos deducir que la partición representa la estructura interna de los datos. El problema principal es determinar cuándo un valor es inusualmente alto (o bajo) y cuando no. Esto nos lleva a tratar el problema como un problema estadístico de contraste de hipótesis (Halkidi et al., 2001; Jain y Dubes, 1988). Para ello es necesario tratar el índice como una variable aleatoria y conocer o estimar su distribución bajo la hipótesis de aleatoriedad.

Dado que no se puede calcular la distribución de probabilidad de los índices de validación de forma teórica, se utilizan técnicas de estimación como Monte Carlo o *Bootstrapping*. Sin embargo, estas aproximaciones suelen realizar suposiciones que no siempre se cumplen por lo que hay que ser extremadamente cautelosos, sobre todo en el caso de la validación interna (Jain y Dubes, 1988).

Un análisis de la bibiliografía sobre la validación de clustering demuestra que en la práctica no se utilizan los tests estadísticos para evaluar las particiones (Bezdek y Pal, 1998; Bouguessa et al., 2006; Chou et al., 2004; Günter y Bunke, 2003; Hardy, 1996; Kim y Ramakrishna, 2005; Mali y Mitra, 2003; Maulik y Bandyopadhyay, 2002; Milligan y Cooper, 1985; Pal y Biswas, 1997). La aproximación utilizada es la de comparar los valores de los índices de varias particiones. Se utiliza, por tanto, una aproximación comparativa donde no se analiza el valor del índice en términos absolutos sino en términos relativos.

El término validación relativa no parece estar claramente definido. Jain y Dubes (1988) lo utilizan para la validación mediante la comparación de varias particiones, pero sólo mediante índices de validación interna. En cambio, Halkidi et al. (2001) limitan la validación relativa a la comparación de particiones obtenidas mediante la variación de los parámetros de un mismo algoritmo. Otros, como Dom (2001) o Yeung et al. (2001), ni siquiera mencionan la validación relativa.

Dado que la práctica generalizada es la de analizar las particiones de manera comparativa, tanto para la validación externa como la interna, pa-

rece que la confusión puede surgir al situar la validación relativa al mismo nivel que las otras dos. Parece más adecuado limitar la validación a dos categorías: externa e interna. Para ambas categorías existe la posibilidad de realizar una comparativa en la que se calcule, mediante test de hipótesis, el umbral a partir del cuál una partición se puede considerar "buena" o comparar varias particiones para realizar una clasificación ordenada según su bondad. Podríamos denominar la primera como validación absoluta y la segunda como validación relativa.

Es importante recalcar que la validación absoluta y la relativa no son excluyentes. Al contrario, una validación combinada tendría la ventaja de indicar cuáles son las particiones aceptables y cuál es el orden de bondad entre ellas. Desde otro punto de vista, una validación absoluta marca el punto en la clasificación de una validación relativa a partir del cuál las particiones no parecen alcanzar un umbral mínimo de bondad.

A continuación se describen varios índices de validación, usados tanto para validación externa como interna. Dada su diferente naturaleza hemos dividido la descripción en dos apartados: validación de jerarquías y validación de particiones. La descripción se realiza desde el punto de vista de la validación relativa, en el sentido que acabamos de describir. Posteriormente, se realiza un breve repaso a otras técnicas de validación más novedosas basadas en la estabilidad (Sección 2.3.3) y finalmente, en la Sección 2.3.4, volvemos a mencionar someramente el concepto de la tendencia al clustering.

2.3.1. Validación de jerarquías

La validación de las jerarquías de clusters se ha realizado principalmente mediante la validación interna. La razón fundamental para ello es que si ya es complicado conocer la partición correcta de unos datos, es aún más difícil conocer una jerarquía correcta. Como posible solución intermedia Widyantoro et al. (2002) proponen la validación externa de una jerarquía en base a una partición correcta. La propuesta realizada suma, por cada cluster de la partición correcta, el número de casos que comparte con su nodo más parecido en la jerarquía. Posteriormente, se divide la suma por el número de casos total.

La medida utilizada por Widyantoro et al. para calcular la similitud entre dos clusters es la siguiente:

$$\mathrm{sim}(C_i, C_j) = \frac{|C_i \cap C_j|}{|C_i \cup C_j|}$$

Por lo que el índice de validación es el siguiente:

$$V(H) = \frac{\sum_{C_i \in P^*} |C_i \cap \arg\ \text{máx}_{C_j \in T_H} \text{sim}(C_i, C_j)|}{N}$$

donde $T_H = \bigcup_{P \in H} \bigcup_{C \in P} C$, es el conjunto de todos los clusters de la jerarquía H, que coincide con el conjunto de nodos del dendrograma correspondiente, N es el número total de casos y P^* es la partición correcta.

La validación interna de una jerarquía de clusters se realiza comparando la jerarquía con la matriz de distancias de los datos. Para ello, se calcula una matriz a partir de la jerarquía y se realiza una comparación entre las dos matrices utilizando el coeficiente de correlación momento-producto de Pearson (*product-moment correlation coefficient*) (Jain y Dubes, 1988).

PMCC =

$$\frac{(1/M)\sum_{i=1}^{N-1}\sum_{j=i+1}^{N}[A(i,j)B(i,j) - \mu_A\mu_B]}{\sqrt{[(1/M)\sum_{i=1}^{N-1}\sum_{j=i+1}^{N}A(i,j)^2 - \mu_A^2][(1/M)\sum_{i=1}^{N-1}\sum_{j=i+1}^{N}B(i,j)^2 - \mu_B^2]}}$$

A y B son las dos matrices a comparar, N es el número de casos en la base de datos y $M = N \times (N-1)/2$. Asimismo, μ_A y μ_B son las medias de las matrices A y B, respectivamente. Es decir,

$$\mu_A = (1/M)\sum_{i=1}^{N-1}\sum_{j=i+1}^{N}A(i,j) \qquad \mu_B = (1/M)\sum_{i=1}^{N-1}\sum_{j=i+1}^{N}B(i,j)$$

La matriz calculada a partir de la jerarquía es tradicionalmente la matriz cofenética, por lo que la medida utilizada se conoce también como coeficiente de correlación cofenética o *cophenetic correlation coefficient* (CPCC). La forma de obtener la matriz cofenética de una jerarquía se describe más fácilmente a través de un dendrograma. El valor (i, j) de la matriz cofenética es la altura del nodo en el que por primera vez se juntan los casos x_i y x_j.

Si en lugar de comparar una matriz cofenética con la matriz de distancias se comparan dos matrices cofenéticas correspondientes a dos dendrogramas se consigue una medida de similitud entre dendrogramas. Podani (2000) realiza un estudio al respecto y compara 5 modos distintos de extraer matrices a partir de dendrogramas. El autor observa que las matrices cofenéticas de dos dendrogramas pueden ser distintas aunque topológicamente sean idénticas y que las matrices PMD (*Partition Membership Divergence*) son más adecuadas en este caso. Una matriz PMD puede describirse de la siguiente manera: el valor (i, j) de la matriz PMD es el número de nodos internos con altura menor o igual que el nodo en el que por primera vez se juntan los

casos x_i y x_j. Dicho de otro modo, es el número de la iteración en la que se han unido los dos casos.

2.3.2. Validación de particiones

La validación de particiones es la validación más común en el clustering. Para ello existen multitud de índices que tratan de estimar la bondad de la partición evaluada.

Un método posible para evaluar una partición es convertirla en una matriz, A, donde $\mathrm{A}(i,j) = 0$ si los casos x_i y x_j están en el mismo cluster y $\mathrm{A}(i,j) = 1$ en caso contrario. De este modo se pueden utilizar medidas de similitud de matrices como índices de validación de particiones. Los mismos índices sirven para la validación interna (comparar con la matriz de similitud) como para la externa (comparar con la matriz correspondiente a la partición correcta) (Jain y Dubes, 1988).

Sin embargo, la mayoría de los índices de validación son específicos del tipo de validación. Típicamente la validación externa se reduce a la comparación de dos particiones: la evaluada y la correcta. Por ello, los índices de validación externa suelen ser medidas de similitud de particiones. En este trabajo, donde debido al contexto no haya confusión posible, nos referiremos a ellos simplemente como medidas de similitud. Los índices de validación interna, en cambio, suelen estimar la cohesión y la separación de los clusters y aunque la terminología no está unificada generalmente son denominados como índices de validación de clusters o *cluster validity index* (CVI) .

Validación externa

Como acabamos de mencionar, los índices de validación externa suelen ser medidas de similitud de particiones. Existen diversos tipos de medidas de similitud, dependiendo de la información en que se basan:

- **Recuento de pares**: Este tipo de medidas basan la estimación de la similitud en una matriz de contingencia entre ambas particiones. Cada fila de la matriz corresponde a un cluster de una de las particiones mientras que cada columna corresponde a un cluster de la otra partición. El valor de cada celda corresponde al número de casos que los clusters correspondientes a la fila y a la columna tienen en común.

  Supongamos $P = \{C_1, C_2, \ldots, C_K\}$ y $P' = \{C'_1, C'_2, \ldots, C'_{K'}\}$ dos particiones de K y K' clusters respectivamente. La matriz de contingencia correspondiente se representa en la Tabla 2.2. El valor n_{ij} corres-

	C_1	C_2	$\cdots$	C_K	
C'_1	n_{11}	n_{12}	$\cdots$	n_{1K}	$n_{1.}$
C'_2	n_{21}	n_{22}	$\cdots$	n_{2K}	$n_{2.}$
$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$
$C'_{K'}$	$n_{K'1}$	$n_{K'2}$	$\cdots$	$n_{K'K}$	$n_{K'.}$
	$n_{.1}$	$n_{.2}$	$\cdots$	$n_{.K}$	$n_{..} = N$

Tabla 2.2: Matriz de contingencia entre las particiones P y P'.

ponde al número de casos que los clusters C_i y C_j tienen en común, $n_{ij} = |C_i \cap C_j|$.

Sin embargo, la mayoría de medidas de similitud resumen la información de dicha matriz en una matriz de 2 filas y 2 columnas. En dicha matriz se recuentan las distintas situaciones en las que se encuentran todos los posibles pares de casos de la base de datos. Por cada par de casos existen 4 posibilidades: los casos están en el mismo cluster tanto en la partición P como en la partición P'; los casos están en el mismo cluster en P, pero no en P'; los casos están en el mismo cluster en P', pero no en P; los casos están en distinto cluster tanto en P como en P'. Tradicionalmente estos cuatro valores se han denominado a, b, c y d, respectivamente. La Tabla 2.3 muestra la matriz correspondiente.

		P'	
		Mismo cluster	Distinto cluster
P	Mismo cluster	a	b
	Distinto cluster	c	d

Tabla 2.3: Matriz de contingencia entre los pares de casos de las particiones P y P'.

Intuitivamente dos particiones serán similares si tienen valores de a y d altos y valores de b y c bajos. Estos valores se pueden obtener a partir de la Tabla 2.2 mediante las siguientes fórmulas:

$$a = (1/2)\sum_{i=1}^{K}\sum_{j=1}^{K'} n_{ij}^2 - (N/2)$$

$$b = (1/2)\sum_{j=1}^{K'} n_{.j}^2 - (1/2)\sum_{i=1}^{K}\sum_{j=1}^{K'} n_{ij}^2$$

$$c = (1/2)\sum_{i=1}^{K} n_{i.}^2 - (1/2)\sum_{i=1}^{K}\sum_{j=1}^{K'} n_{ij}^2$$

$$d = \frac{N(N+1)}{2} - (1/2)\left[\sum_{i=1}^{K} n_{i.}^2 + \sum_{j=1}^{K'} n_{.j}^2\right]$$

Se han definido multitud de medidas de similitud basados en el recuento de pares. A continuación se describen algunas de las más extendidas y que serán utilizadas en posteriores capítulos:

- Rand (Rand, 1971): $(a+d)/(a+b+c+d)$. Es probablemente la medida más utilizada. Representa la fracción de pares de casos en el mismo estado en ambas particiones.
- Jaccard (Jaccard, 1908): $a/(a+b+c)$. Es similar a la medida Rand, pero ignora los pares de casos que han sido asignados a distintos clusters en ambas particiones. Es habitual usarlo en entornos con muchos clusters, donde el valor de d puede ser muy alto.
- Fowlkes-Mallows (Fowlkes y Mallows, 1983): $a/\sqrt{(a+b)(a+c)}$. Al igual que la medida Jaccard esta medida también ignora el valor de d.
- Adjusted Rand (Hubert y Arabie, 1985):

$$\frac{\sum_{i=1}^{K}\sum_{j=1}^{K'}\binom{n_{ij}}{2} - \left[1/\binom{N}{2}\right]\sum_{i=1}^{K}\binom{n_{i.}}{2}\sum_{j=1}^{K'}\binom{n_{.j}}{2}}{(1/2)\left[\sum_{i=1}^{K}\binom{n_{i.}}{2} + \sum_{j=1}^{K'}\binom{n_{.j}}{2}\right] - \left[1/\binom{N}{2}\right]\sum_{i=1}^{K}\binom{n_{i.}}{2}\sum_{j=1}^{K'}\binom{n_{.j}}{2}}$$

  Es una versión ajustada del índice Rand para que el resultado sea cercano a 0 cuando las particiones comparadas han sido generadas aleatoriamente.

- **Teoría de la información**: Otro grupo importante de medidas de similitud se basan en conceptos de la teoría de la información. Antes de describir algunas de ellas describamos una serie de formulas relacionadas:
  - Entropía: $H(P) = -\sum_{C_i \in P} p(C_i)\log p(C_i)$ donde $p(C_i) = \frac{|C_i|}{N}$.
  - Entropía conjunta:
    $H(P,P') = -\sum_{C_i \in P}\sum_{C'_i \in P'} p(C_i, C'_i)\log p(C_i, C'_i)$ donde
    $p(C_i, C'_i) = \frac{|C_i \cap C'_i|}{N}$

- Información mutua: $I(P, P') = \sum_{C_i \in P} \sum_{C'_i \in P'} p(C_i, C'_i) \frac{\log p(C_i, C'_i)}{p(C_i)p(C'_i)}$

Las dos últimas fórmulas han sido definidas como medidas de similitud de particiones por Manning y Schütze (1999). Otros autores han propuesto diferentes combinaciones, como por ejemplo:

- Variation of Information (Meilă, 2003): $H(P)+H(P')-2I(P, P')$. Meilă analiza en detalle las propiedades de esta interesante medida.
- Normalized Mutual Information 1 (Malvestuto, 1986): $\frac{I(P,P')}{H(P,P')}$.
- Normalized Mutual Information 2 (Strehl y Ghosh, 2003): $\frac{I(P,P')}{\sqrt{H(P)H(P')}}$.

- **Número de ediciones**: El índice propuesto por Charon et al. (2006) es una distancia de edición. Calcula el número mínimo de casos que hay que mover de un cluster a otro para convertir una partición en otra.

A pesar del gran número de medidas de similitud propuestas no existe un criterio claro a la hora de seleccionar una u otra. Existen algunos trabajos comparativos que tratan de clasificar las medidas según su bondad o agruparlas en base al comportamiento.

Por ejemplo, Warrens (2008a) demuestra que el índice Adjusted Rand es equivalente al índice Kappa utilizado habitualmente en psicología. Por otra parte, el mismo autor analiza un conjunto de medidas y describe una serie de relaciones sobre sus límites. Demuestra que muchas medidas están relacionadas en el sentido de que el valor de unas es siempre superior a otras (Warrens, 2008b).

Baulieu (1989) encuentra una relación similar que llama *Global Order Equivalence* y significa que dos medidas ordenan de la misma manera el mismo conjunto de particiones. El autor define una serie de axiomas de los cuales deriva una serie de propiedades deseables. Analiza 21 medidas de similitud en relación a esas propiedades.

Batagelj y Bren (1995) también comparan una serie de medidas de similitud de particiones, agrupándolas por cercanía. Incluso utilizan el clustering para agruparlas y obtener un dendrograma de medidas de similitud. Albatineh et al. (2006) comparan 28 medidas de similitud y observan que, después de corregirlas para tener en cuenta la similitud entre particiones aleatorias, varias de estas medidas son equivalentes.

Más recientemente, Pfitzner et al. (2009) realizan una extensa evaluación de medidas de similitud. En concreto definen una serie de propiedades que las medidas de similitud deberían cumplir y analizan 56 medidas de similitud para tratar de averiguar cuáles de las propiedades cumple cada una. En base a ese análisis agrupan las medidas en 6 grupos.

En resumen, sigue sin haber un criterio claro para seleccionar una medida de similitud u otra aunque parece que la amplia gama de medidas no lo es tanto dado que muchas de ellas son similares o incluso equivalentes.

Validación interna

Existe un gran número de índices para la validación interna (Bezdek y Pal, 1998; Bouguessa et al., 2006; Chou et al., 2004; Günter y Bunke, 2003; Kim y Ramakrishna, 2005; Milligan y Cooper, 1985; Pal y Biswas, 1997; Wu et al., 2009). La mayoría se basan en los conceptos de cohesión de los clusters y la separación entre clusters. Tal y como ocurre en el clustering en general tampoco existe una base teórica robusta en este área, por lo que no hay un criterio claro para poder seleccionar el índice adecuado en cada caso. Kim y Ramakrishna (2005) realizan una reflexión teórica y definen dos categorías para los índices de validación de clusters: tipo ratio y tipo suma.

A pesar del gran número de índices propuestos no existen comparativas extensas y completas de ellas. Probablemente la comparativa más importante sea la realizada por Milligan y Cooper (1985) hace ya 25 años.

A continuación se describen varios CVI que se utilizan en capítulos posteriores y pueden servir como ejemplo:

- Calinski-Harabasz (CH) (Calinski y Harabasz, 1974):

$$\mathrm{CH}(P) = \frac{(N - |P|)\,\mathrm{inter}_{CH}(P)}{(|P| - 1)\,\mathrm{intra}_{CH}(P)}$$

  $\mathrm{inter}_{CH}(P) = \sum_{C \in P} |C|\,\mathrm{d}(\bar{C}, \bar{X})$ e $\mathrm{intra}_{CH}(P) = \sum_{C \in P} \sum_{x \in C} \mathrm{d}(x, \bar{C})$.

- C-Index (Hubert y Levin, 1976):

$$\mathrm{CI}(P) = \frac{\mathrm{S}(P) - \mathrm{S}_{\min}(\mathrm{P})}{\mathrm{S}_{\max}(P) - \mathrm{S}_{\min}(\mathrm{P})}$$

  donde $\mathrm{S}(P) = \sum_{C \in P} \sum_{x_i, x_j \in C} \mathrm{d}(x_i, x_j)$ es la suma de las distancias de todos los pares de casos en el mismo cluster. Sea n_w el número de dichos pares de objetos. Entonces, $\mathrm{S}_{\min}$ es la suma de las n_w menores distancias para toda la base de datos mientras que $\mathrm{S}_{\max}$ es la suma de las n_w mayores.

- Gamma (Baker y Hubert, 1975):

$$\text{Gamma}(P) = \frac{\sum_{C\in P}\sum_{x_i,x_j\in C} \text{dl}(x_i, x_j)}{n_w(N(N-1)/2 - n_w)}$$

donde $\text{dl}(x_i, x_j)$ denota el número de pares de objetos cuya distancia es menor que $\text{d}(x_i, x_j)$ y están en diferentes clusters. El denominador se utiliza para normalizar el resultado y asegurar que el índice está entre 0 y 1. Al igual que antes n_w denota el número de pares de casos en el mismo cluster.

- Davies-Bouldin (DB) (Davies y Bouldin, 1979):

$$\text{DB}(P) = \frac{1}{|P|}\sum_{C_k\in P} \max_{C_l\in P/C_k}\left\{\frac{\text{S}(C_k) + \text{S}(C_l)}{\text{d}(\bar{C_k}, \bar{C_l})}\right\}$$

donde $\text{S}(C) = 1/|C| \sum_{x\in C} \text{d}(x, \bar{C})$.

- Dunn (Dunn, 1973):

$$\text{Dunn}(P) = \frac{\text{inter}_{Dunn}(P)}{\text{intra}_{Dunn}(P)}$$

donde

$$\text{inter}_{Dunn}(P) = \min_{C_k\in P}\left\{\min_{C_l\in P/C_k}\{\delta(C_k, C_l)\}\right\}$$

$$\delta(C_k, C_l) = \min_{x_i\in C_k}\left\{\min_{x_j\in C_l} \text{d}(x_i, x_j)\right\}$$

$$\text{intra}_{Dunn}(P) = \max_{C\in P}\left\{\max_{x_i,x_j\in C} \text{d}(x_i, x_j)\right\}$$

- G(+) (Rohlf, 1974): Para calcular este índice se examinan todas las cuádruplas posibles, $(\alpha, \beta, \gamma, \delta)$. Decimos que una cuádrupla es inconsistente si una de las dos siguientes condiciones es verdadera:

  - $\text{d}(\alpha, \beta) < \text{d}(\gamma, \delta)$, α y β están en distintos clusters, y γ y δ están en el mismo cluster.
  - $\text{d}(\alpha, \beta) > \text{d}(\gamma, \delta)$, α y β están en el mismo cluster, y γ y δ están en distintos clusters.

Si denominamos $S(-)$ al número de cuádruplas inconsistentes y n_w al número de pares de casos en el mismo cluster, el índice G(+) se define como sigue:

$$\frac{2S(-)}{n_w(n_w-1)}$$

- McClain-Rao (McClain y Rao, 1975): Se calcula para cada cluster el ratio entre la distancia media entre los casos del cluster y la distancia media entre los casos dentro y fuera del cluster. El índice se define como la media de los ratios individuales para cada cluster.

Tanto el índice Calinski-Harabasz como Dunn difieren del resto en que un valor más alto es el que denota una mejor partición.

2.3.3. Validación en base a la estabilidad

Aunque la validación de clustering se ha caracterizado por evaluar particiones mediante índices de validación, tanto externas como internas, existe otra aproximación que evalúa los algoritmos de clustering en base a su estabilidad. La idea principal de estos métodos es que un algoritmo estable respecto a unos datos será más adecuado que uno inestable. La misma idea, tal y como se ha mencionado en la Sección 2.2.5, sirve para determinar el número correcto de clusters bajo la premisa de que un algoritmo de clustering será más estable al trabajar sobre unos datos con el número correcto de clusters.

Para evaluar un algoritmo de clustering se puede aplicar el algoritmo a varias muestras de una misma base de datos y comparar las particiones obtenidas (Ben-Hur et al., 2002; Falasconi et al., 2009). Otro método similar es aplicar el algoritmo a una serie de muestras, pero comparar las particiones obtenidas con la partición correspondiente a la base de datos completa (Levine y Domany, 2001). En general, según estos métodos, cuanto más se parezcan las particiones comparadas más estable y mejor será el algoritmo.

Otros autores utilizan una aproximación distinta (Lange et al., 2004; Tibshirani et al., 2001a). En este caso la base de datos se divide en dos muestras disjuntas del mismo tamaño. De esta forma, tal y como argumentan Lange et al. (2004), se evitan los resultados sesgados que puede provocar la dependencia existente entre las muestras solapadas. Se utiliza un algoritmo de clustering para generar una partición a partir de cada una de las dos muestras y se trata de predecir la partición de una muestra en base a la partición de la otra. Para ello se entrena un clasificador (clasificación supervisada) sobre una de las muestras, utilizando como etiqueta de cada caso el

número del cluster al que pertenece. Posteriormente se clasifican los casos de la otra muestra y se comparan los resultados obtenidos con el clasificador y los obtenidos con el algoritmo de clustering. En este tipo de validación la elección del clasificador es fundamental ya que una elección desacertada puede interferir en los resultados.

Otra aproximación diferente consiste en utilizar un método similar a la técnica de validación de la clasificación supervisada *leave-one-out*, pero ignorando un atributo en lugar de un caso (Yeung et al., 2001). Los autores de la propuesta sitúan dicha aproximación a caballo entre la validación interna y la externa. El método fue utilizado para agrupar datos sobre genes ignorando uno de los atributos (llamados "condición experimental" en el contexto particular de los datos de expresión génica). La bondad de la partición se mide en base al atributo ignorado. Desafortunadamente, dadas sus características, este método sólo parece apropiado para agrupar datos de expresión génica.

2.3.4. Tendencia al clustering

Uno de los problemas de los algoritmos de clustering es que generan una partición de los datos de entrada aunque estos no tengan ningún tipo de estructura interna. Dada la complejidad de la validación de clustering es recomendable evitar este tipo de problemas realizando un análisis previo con el objeto de establecer si los datos tienen algún tipo de estructura. Además, este análisis previo evita realizar un trabajo de clustering potencialmente costoso sobre unos datos imposibles de particionar de forma coherente. Este análisis previo para determinar si los datos tienen cierta predisposición a agruparse en grupos naturales se conoce como tendencia al clustering (*cluster tendency)* o agrupabilidad (*clusterability*).

A pesar de que el análisis de la tendencia al clustering se considera parte de la validación de clustering es un proceso que se realiza antes de la fase de aprendizaje. En este sentido, es un proceso que se sale de la posición habitual de la validación en el proceso completo de la minería de datos, que es posterior a la fase de aprendizaje. Por lo tanto, podríamos decir que el análisis de la tendencia al clustering es un proceso de análisis previo.

En este ámbito del clustering tampoco existe una base teórica sólida. No obstante, Ackerman y Ben-David (2009) realizan un análisis interesante sobre los distintos conceptos de agrupabilidad.

Las técnicas principales de análisis de la tendencia al clustering se basan en el concepto de aleatoriedad. Básicamente, tratan de determinar si los casos de la muestra de entrenamiento se han generado en base a un mismo

proceso aleatorio. Algunas técnicas se centran en analizar la aleatoriedad espacial, es decir, la aleatoriedad en las posiciones que ocupan los distintos casos. Otras técnicas, en cambio, tratan de medir la posible aleatoriedad de la matriz de distancias de los casos de la muestra (Jain y Dubes, 1988).

Existen otras aproximaciones que se basan en representaciones gráficas de la matriz de distancias. Bezdek y Hathaway (2002) proponen un sistema (VAT) para ordenar los casos de manera tal que la representación gráfica de la matriz de distancias sugiera el nivel de agrupabilidad de la muestra. Existen diversas variantes del mismo sistema para adaptarlo a bases de datos de gran tamaño como reVAT (Huband et al., 2004), bigVAT (Huband et al., 2005) o sVAT (Hathaway et al., 2006).

Otro método posible es el propuesto por Massey (2002) que utiliza técnicas de teoría de resonancia adaptativa, un modelo de red neuronal artificial.

Parte II

Aportaciones

Capítulo 3

SIHC: Un algoritmo de clustering estable, jerárquico e incremental

3.1. Introducción

Tal y como se menciona en el Capítulo 2 una posible clasificación de los algoritmos de clustering los divide en particionales y jerárquicos. Este capítulo se centra en los algoritmos jerárquicos y en particular en el conjunto de algoritmos basados en el método *SAHN* (*Sequential Agglomerative Hierarchical Nonoverlapping*). Para más detalle véase la Sección 2.2.4.

Los algoritmos basados en el método SAHN tienen varias ventajas:

- El método es sencillo e intuitivo: en cada paso se unen los dos clusters más cercanos. Esto hace que en procesos de depuración, control, análisis de resultados... sea sencillo e intuitivo seguir los pasos del algoritmo. Además es sencillo deducir cómo afecta cada dato o conjunto de datos en el proceso de construcción de la jerarquía.

- Es un método conocido desde hace varias décadas y muy utilizado. Esto hace que se conozcan los resultados que ha producido en distintos ámbitos. De este modo se consigue evitar la desconfianza que a veces surge respecto a algoritmos prometedores, pero poco estudiados. Además, esto permite que las aportaciones científicas relacionadas con el método SAHN puedan ser contrastadas y comparadas con otros trabajos.

- El método ha sido comparado teóricamente con otros algoritmos jerárquicos obteniendo resultados positivos para SAHN (Gascuel y McKenzie, 2004).

Sin embargo, a pesar de las ventajas mencionadas anteriormente, el método SAHN también tiene alguna desventaja. Seguramente la principal de ellas sea su coste computacional elevado, lo que hace que el método no sea adecuado para bases de datos de gran tamaño. De todas maneras, aunque el coste para el peor caso es de $O(N^3)$ existen versiones optimizadas que reducen el coste a $O(N^2)$ (Day y Edelsbrunner, 1984). Además, Olson (1995) ofrece un resumen de paralelizaciones que permiten ejecutar el algoritmo con un coste computacional reducido.

La evolución del clustering y su uso en distintos entornos ha hecho patente otra de las debilidades del método SAHN (y de muchos otros algoritmos de clustering). En entornos dinámicos, donde los datos llegan día a día, los algoritmos tradicionales no son válidos. Si un usuario quiere actualizar su partición o jerarquía con los nuevos datos que le llegan, no tiene otro remedio que volver a ejecutar el algoritmo de clustering con todos los datos (los nuevos junto con los que ya tenía de antemano).

Aunque este procedimiento puede ser factible con algún algoritmo de clustering particularmente eficiente en un entorno con una reducida cantidad de datos, no es el caso de los algoritmos basados en SAHN. Es cierto que podemos estabilizar el tamaño de la base de datos si eliminamos los datos más antiguos (considerándolos obsoletos), pero el procedimiento de volver a ejecutar el algoritmo sobre los datos antiguos y los nuevos no parece satisfactorio.

Esta desventaja no es exclusiva del método SAHN sino que la comparten la mayoría de algoritmos de clustering. A consecuencia de ello se han modificado algunos algoritmos y diseñado otros nuevos que permiten actualizar una partición o jerarquía según se van obteniendo nuevos datos. Este tipo de algoritmo se denomina algoritmo incremental (Fisher, 1987; Widyantoro et al., 2002).

Por lo tanto, el método SAHN no es un método incremental y si tenemos una jerarquía generada a partir de un conjunto de datos X y queremos calcular la jerarquía correspondiente a $X+Y$, tenemos que volver a ejecutar el mismo algoritmo partiendo del conjunto de datos completo, $X + Y$.

En la sección siguiente se describen dos aproximaciones existentes a un SAHN incremental. La primera aproximación explica cómo añadir nuevos casos en una jerarquía ya construida sin tener que volver a calcular toda la jerarquía. La segunda presenta una modificación al algoritmo *single-linkage*

de forma que lo convierte en incremental. Posteriormente, en la Sección 3.3 se propone una nueva versión incremental del algoritmo SAHN que afronta el problema desde una perspectiva diferente.

3.2. Aproximaciones a un SAHN incremental

En esta sección se describen dos aproximaciones existentes que tratan de dar cierto carácter incremental al método SAHN. Las dos aproximaciones siguen estrategias distintas. La primera de ellas define un procedimiento para calcular la región de una jerarquía afectada por la llegada de un nuevo elemento (Ribert et al., 1999). De esta manera sólo se recalcula la parte afectada de la jerarquía. La segunda aproximación define un nuevo algoritmo jerárquico incremental basado en el *single-linkage* (El-Sonbaty e Ismail, 1998).

La aproximación de Ribert et al. (1999) se basa en el concepto de región de influencia. La idea consiste en determinar la parte de la jerarquía que se mantendrá estable después de añadir un nuevo caso. De este modo el algoritmo permite reconstruir sólo una parte de la jerarquía.

Los resultados demuestran que los requerimientos de memoria se reducen drásticamente, pero el coste computacional se mantiene prácticamente constante. De hecho, los autores afirman que el coste computacional del algoritmo original era muy elevado por lo que los resultados mostrados corresponden a una versión optimizada.

Aunque Ribert et al. no aportan información respecto al tamaño medio de la parte de la jerarquía reconstruida, los resultados sobre coste computacional parecen indicar que en cada actualización se recalculan partes importantes de la jerarquía. De todos modos, aseguran que en ningún caso práctico es necesario reconstruir la jerarquía completa.

En resumen, el algoritmo consigue su objetivo de proporcionar la capacidad de añadir casos incrementalmente al método SAHN. Además, su requerimiento de memoria es significativamente más reducido. Por otra parte, es un algoritmo genérico que puede ser utilizado con distintas variantes del método SAHN.

La aproximación de El-Sonbaty e Ismail (1998) permite construir una jerarquía de forma incremental. La idea principal consiste en analizar los datos y almacenar los k vecinos más cercanos a cada caso. Este proceso se puede hacer de forma incremental y los autores demuestran empíricamente que la estructura de datos creada permite construir un dendrograma igual al correspondiente a *single-linkage*, siempre que el valor de k sea lo suficien-

temente alto. No obstante, las tres bases de datos reales utilizadas para la demostración empírica constan de 12, 9 y 8 casos por lo que la demostración se fundamenta principalmente en las bases de datos sintéticas.

Con el objeto de establecer un valor adecuado de k los autores realizan una experimentación con bases de datos sintéticas y establecen un valor recomendable de $k = 7$. Este algoritmo, junto con valores bajos de k (en relación al número de casos de la base de datos) permite conseguir ahorros en el espacio de memoria y el coste computacional requerido.

La mayor desventaja del algoritmo consiste en la necesidad de establecer un parámetro (k) y la limitación del algoritmo a la medida de proximidad *single-linkage*.

3.3. El algoritmo SIHC

Las aproximaciones mencionadas anteriormente siguen una misma premisa: la jerarquía generada a través del proceso incremental ha de ser igual a la jerarquía que generaría el método SAHN partiendo del conjunto completo de datos. Es una premisa necesaria si el objetivo es crear una versión incremental del método SAHN. Aunque este objetivo es interesante desde un punto de vista puramente científico, no lo es tanto desde un punto de vista práctico, dada la inestabilidad del método SAHN.

3.3.1. La inestabilidad del método SAHN

El método SAHN es inestable respecto a cambios en los datos. Es decir, jerarquías construidas sobre variaciones de unos mismos datos pueden diferir considerablemente. Esto hace que una variante incremental de SAHN, fiel al comportamiento del método original, tenga que variar considerablemente la estructura de la jerarquía según va siendo actualizada.

La Figura 3.1 muestra unos datos que servirán de ejemplo para ilustrar el problema de la inestabilidad del método SAHN. Los puntos circulares representan 3 clusters de distinta densidad. El punto en forma de X representa un nuevo caso a añadir incrementalmente. La Figura 3.2 muestra dos dendrogramas construidos a partir del algoritmo *average-linkage*. El de la izquierda corresponde a los puntos circulares, mientras que el de la derecha incluye también el punto en forma de X.

Se observa con claridad cómo la inclusión del punto X conlleva la división del cluster central (número 2). Por consiguiente, una parte del cluster central se une a un cluster y la otra parte a otro. La consecuencia es que las dos jerarquías son notoriamente diferentes.

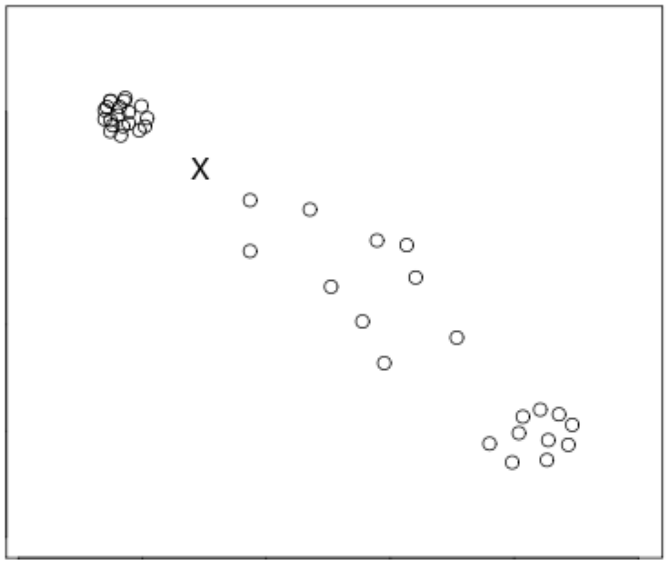

Figura 3.1: Una base de datos de tres clusters (puntos circulares) y un caso a añadir incrementalmente (punto X).

Aunque el ejemplo ha sido diseñado para provocar esta inestabilidad, es significativo el hecho de que la inclusión de un único punto pueda provocar un cambio tan importante. En un caso real en el que se añade un importante número de casos, incluyendo ruido, es previsible que ocurran este tipo de situaciones.

Aunque pueda existir algún entorno en el que es adecuado forzar un cambio de estructura para adaptarse a nuevos datos, en términos generales parece lógico tratar de mantener la estructura ya existente. Supongamos un usuario que hace uso de una jerarquía de clusters generada a través de un algoritmo SAHN. Una de las ventajas de una jerarquía es que nos informa de la estructura de los datos a distintos niveles, por lo que podemos suponer que este usuario ha estudiado la jerarquía y se ha familiarizado con los clusters propuestos. La adición de nuevos casos a esta jerarquía puede provocar una reestructuración de ésta, haciendo que el usuario tenga que reaprender parte de la estructura. Pero el problema principal no es ése, sino que el usuario perderá confianza en el algoritmo ya que casos que antes parecían estar estrechamente relacionados pueden no estarlo después.

El problema de la inestabilidad es un problema más estudiado en el ámbito de la clasificación supervisada y se considera una propiedad fundamental

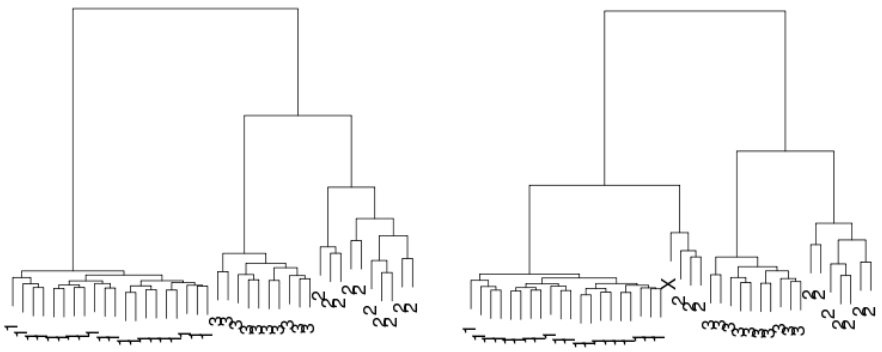

Figura 3.2: Jerarquía original y modificada tras la llegada de un nuevo caso.

de la explicabilidad (Pérez et al., 2007; Turney, 1995). En el ámbito de los árboles de clasificación Turney (1995) expresaba el inconveniente arriba expuesto de la siguiente manera: *"the engineers are disturbed when different batches of data from the same process result in radically different decision trees. The engineers lose confidence in the decision trees, even when we can demonstrate that the trees have high predictive accuracy"*.

Como ya se ha mencionado en la Sección 2.3.3 el problema de la inestabilidad se ha estudiado en el ámbito del clustering como un método de validación. La idea fundamental es que un algoritmo deberá encontrar particiones similares en muestras de datos similares. De todos modos este tipo de análisis se ha reducido típicamente a algoritmos particionales.

3.3.2. SIHC

En esta sección se describe un nuevo algoritmo incremental, llamado SIHC (*Stable Incremental Hierarchical Clustering*), basado en el método SAHN. El método es similar al de Ribert et al. (1999) ya que propone un algoritmo para añadir casos incrementalmente a una jerarquía construida con un algoritmo SAHN. La diferencia principal entre ambos estriba en que el objetivo de SIHC es la estabilidad. Esto provoca que las jerarquías actualizadas con SIHC no sean idénticas a las que generaría SAHN con los mismos datos, pero garantiza que la jerarquía actualizada mantiene una estructura muy similar a la que ya tenía.

SIHC imita a SAHN en una de sus principales características: es sencillo e intuitivo. SIHC añade los nuevos casos uno a uno siguiendo una idea sencilla:

si la distancia del nuevo caso al nodo raíz es mayor que la altura del nodo raíz[1], sitúa el nuevo caso como hermano del nodo raíz; si no, repite el proceso con el nodo hijo más cercano. Se puede ver claramente que el algoritmo no reestructura la jerarquía a actualizar, sino que simplemente añade el nuevo caso en el punto que considera más adecuado. El algoritmo se describe más detalladamente en el Algoritmo 3.1.

Algoritmo 3.1 SIHC

```
1:  d_nodo ← d(nuevoCaso, nodo)
2:  if nodo.altura ≤ d_nodo then
3:      singleton ← nuevo nodo hoja
4:      singleton.asignaCaso(nuevoCaso)
5:      nuevoNodo ← nuevo nodo interno
6:      nuevoNodo.asignaHijos(nodo, singleton)
7:      nuevoNodo.altura ← d_nodo
8:      reemplazar nodo con nuevoNodo
9:  else
10:     cercano ← arg mín_{hijo∈nodo}{d(nuevoCaso, hijo)}
11:     actualizaAltura(nodo)
12:     SIHC(nuevoCaso, cercano)
13: end if
```

En primer lugar se calcula la distancia del nuevo caso al nodo actual (siendo el primer nodo actual el nodo raíz). En este punto podemos utilizar cualquiera de las medidas de proximidad utilizadas con el método SAHN para calcular la distancia entre dos clusters. Por lo tanto, el algoritmo SIHC se adapta a cualquier variante de SAHN.

En segundo lugar, si la distancia calculada es mayor o igual que la altura del nodo actual, se crea un cluster con el nuevo caso y se establece como hermano del nodo actual. Para ello, se crea un nuevo nodo que sea padre del nodo actual y del nuevo caso, de forma que el dendrograma sigue siendo binario. La altura del nuevo nodo es la distancia calculada en la línea 1.

En caso de que la distancia entre el nuevo caso y el nodo actual sea menor que la altura del nodo actual, el algoritmo se ejecuta recursivamente sobre el nodo hijo más cercano al nuevo caso (línea 12). Pero antes de ello, es necesario actualizar la altura del nodo atravesado, ya que la distancia entre sus hijos varía en el momento en que añadimos el nuevo caso a uno de sus hijos. Podemos calcular esta distancia fácilmente comparando todos los

[1]La altura de un nodo indica la distancia entre sus nodos hijo.

casos implicados en los nodos hijo, pero podemos tratar de hacerlo de forma más eficiente estudiando la medida de proximidad utilizada. A continuación mostramos cómo calcular de forma eficiente la altura de un nodo atravesado por un nuevo caso en el método SIHC para varias medidas de proximidad:

- *Single-linkage*: $\min(h, d_{lejano})$
- *Average-linkage*: $(h \times n_{cercano} + d_{lejano})/(1 + n_{cercano})$
- *Complete-linkage*: $\max(h, d_{lejano})$

Donde h es la altura del nodo atravesado, $n_{cercano}$ es el tamaño del hijo más cercano y d_{lejano} es la distancia entre el nuevo caso y el nodo hijo más lejano.

El modo en que la altura se actualiza en los nodos atravesados y se asigna a los nodos creados garantiza que los dendrogramas actualizados mediante el método SIHC mantienen la semántica habitual de la altura del nodo en un dendrograma.

El coste computacional del algoritmo SIHC es inferior a SAHN. Cada vez que se añade un caso, éste se compara como máximo con una rama entera, cuya longitud media será de orden $O(\log N)$. En cada nodo se compara el nuevo caso con los casos de ese nodo, $O(N)$. Por ello, el coste de añadir N casos será $O(N^2 \log N)$, menor que el del algoritmo SAHN, $O(N^3)$. Además, aplicando las mismas técnicas de optimización del algoritmo SAHN también se puede reducir el coste computacional de SIHC.

Aunque el algoritmo SIHC no garantiza que la jerarquía obtenida sea la misma que generaría el método SAHN, esto no implica que la jerarquía obtenida sea "peor", en el sentido de que la estructura obtenida sea menos fiel a los datos. Por ello, se ha realizado una experimentación para comparar ambos métodos.

3.4. Experimentación

Con el objeto de comparar los métodos SIHC y SAHN se ha diseñado una experimentación de dos partes. En la primera se pretende comparar las jerarquías de ambos métodos con el objeto de cuantificar su similitud. De este modo se verificará hasta qué punto son distintos SIHC y SAHN. El objetivo de la segunda parte es analizar la adecuación a los datos de las jerarquías obtenidas tanto mediante SAHN como SIHC; es decir, medir la bondad o calidad de ambos métodos.

3.4.1. Bases de datos y algoritmos

Para las dos partes del trabajo experimental se han utilizado 11 bases de datos sintéticas y 6 reales. 9 de las 11 bases de datos sintéticas han sido generadas siguiendo el mismo patrón y han sido nombradas como Normal_d_s. d se refiere al número de dimensiones. En cada una de estas bases de datos hay un cluster por dimensión, compuesto por 50 casos generados a partir de una distribución normal multivariada. La media de la distribución del cluster i es e_i y la matriz de covarianza es $s \times I_d$, donde I_d es la matriz identidad d-dimensional. e_i es el vector que tiene todos los valores a 0 excepto el de la dimensión i, cuyo valor es 1. Los valores asignados a d son 2, 4 y 6 mientras que los valores de s han sido fijados a 3, 5 y 10. La variación del parámetro s permite definir clusters con diferente nivel de solapamiento.

Las otras dos bases de datos sintéticas se han generado manualmente en un espacio bidimensional y representan dos casos típicos de clusters bien separados, pero no separables linealmente. La Figura 3.3 muestra las dos bases de datos, llamadas *Concéntrico* y *T&U*.

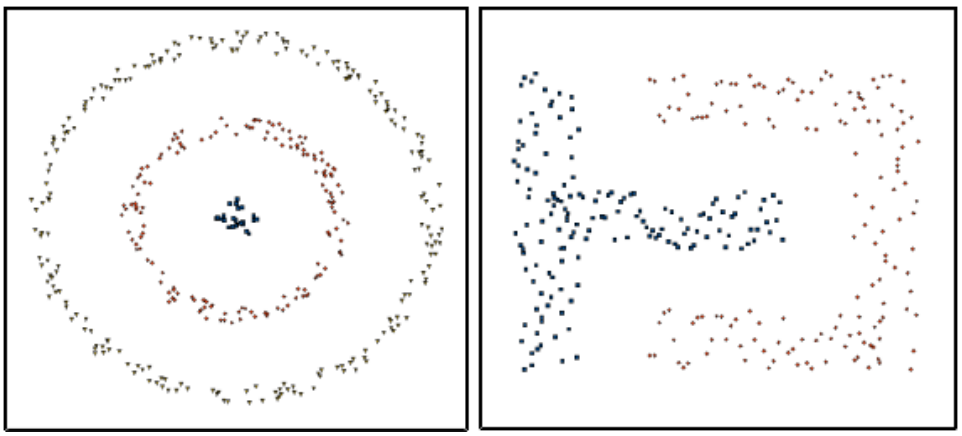

Figura 3.3: Bases de datos *Concéntrico* y *T&U*.

Las 6 bases de datos reales han sido obtenidas del repositorio UCI de la Universidad de California, Irvine (Asuncion y Newman, 2007). La Tabla 3.1 muestra las características principales de todas las bases de datos.

Para cada base de datos se ha construido un conjunto de jerarquías con distinto nivel de influencia de SAHN y SIHC. A continuación se describe el proceso de construcción de dichas jerarquías para una base de datos. En el resto de bases de datos se repetirá el mismo proceso. En la primera fase, a la que llamamos fase SAHN, se divide la base de datos en 20 partes del

Base de datos	Dimensiones	Clusters	Casos
Iris	4	3	150
Glass	9	7	214
Wine	13	3	178
Ecoli	8	8	336
Haberman	3	2	306
Ionosphere	34	2	351
Normal_2_3	2	2	100
Normal_4_3	4	4	200
Normal_6_3	6	6	300
Normal_2_5	2	2	100
Normal_4_5	4	4	200
Normal_6_5	6	6	300
Normal_2_10	2	2	100
Normal_4_10	4	4	200
Normal_6_10	6	6	300
Concéntrico	2	3	400
T&U	2	2	300

Tabla 3.1: Características de las bases de datos.

mismo tamaño y se crea una jerarquía SAHN con m partes. Posteriormente, en la fase llamada fase SIHC, se añaden las $20 - m$ partes restantes utilizando el método SIHC. El valor de m varía entre 1 y 19, de forma que nos permite cuantificar el efecto producido por la cantidad de casos añadidos incrementalmente.

Dado que el orden de los datos afecta al método SIHC, el proceso descrito en el párrafo anterior se ha repetido 25 veces. Por un lado se ha repetido la fase SAHN 5 veces por cada valor de m, seleccionando para ello distintas partes de la base de datos de forma aleatoria. Por otro lado, para cada repetición de la fase SAHN se ha repetido la fase SIHC otras 5 veces añadiendo las partes restantes en orden aleatorio.

Finalmente, con el objeto de tener una referencia base, se ha construido una jerarquía SAHN con toda la base de datos ($m = 20$) a la que denominamos jerarquía 100 % SAHN.

El proceso descrito se ha repetido con las 17 bases de datos y se han utilizado 3 algoritmos basados en SAHN (*singe-linkage*, *average-linkage* y *complete-linkage*). En consecuencia, el número total de jerarquías construidas asciende a 24.276. Resta por aclarar que en la fase SIHC se ha utilizado

siempre la misma variante que en la fase SAHN (*singe-linkage*, *average-linkage* o *complete-linkage*).

El procedimiento descrito nos permite tener jerarquías con distinto nivel de incidencia de cada método. Desde jerarquías donde el 95 % de los datos se ha añadido incrementalmente ($m = 1$) hasta jerarquías donde el número de casos añadidos incrementalmente es muy bajo o nulo. Las mismas jerarquías han sido utilizadas en las dos partes de la experimentación mencionadas al inicio de esta sección.

3.4.2. Método de validación

En la Sección 2.3.1 se ha hecho referencia a la dificultad de evaluar jerarquías de clustering. Dicha dificultad hace que la forma principal de evaluar jerarquías pase por obtener una matriz a partir de la jerarquía (típicamente la matriz cofenética) y que la evaluación se reduzca a una evaluación interna. En esta experimentación se sigue, principalmente, la misma aproximación y la comparación y evaluación de jerarquías se realiza mediante comparación de matrices.

Para dar mayor robustez a la experimentación realizada, las jerarquías se han evaluado utilizando dos tipos de matrices: las habituales matrices cofenéticas y las matrices PMD (Podani, 2000). Para más detalle véase la página 40 donde se justifica el uso de las matrices PMD ya que las matrices cofenéticas de dos jerarquías pueden ser muy distintas aunque topológicamente sean idénticas. Para la comparación de matrices se ha utilizado el coeficiente de correlación de Pearson descrito en la página 39.

Además de ello, en la segunda parte de la experimentación se ha realizado una validación externa utilizando para ello una variante del método utilizado por Widyantoro et al. (2002). Este método, descrito en la Sección 2.3.1, evalúa una jerarquía de clustering contra una partición que es dada como la correcta. Aunque la idea es intuitiva la formulación no es válida ya que asigna la máxima puntuación a jerarquías notoriamente incorrectas. A continuación se recuerda el índice, se muestra su inconveniente principal y se propone una variante más adecuada.

Nuevo método de validación externa de jerarquías

El método de validación de Widyantoro et al. (2002) consiste en sumar, por cada cluster, el número de casos que comparte con su nodo más parecido en la jerarquía. El resultado se normaliza dividiéndolo por el número de casos total.

$$V(H) = \frac{\sum_{C_i \in P^*} |C_i \cap \arg\ \text{máx}_{C_j \in T_H} \text{sim}(C_i, C_j)|}{N}$$

donde $\text{sim}(C_i, C_j) = \frac{|C_i \cap C_j|}{|C_i \cup C_j|}$.

Está claro que un valor grande de la medida estará asociado a valores grandes de $|C_i \cap \arg\ \text{máx}_{C_j \in T_H} \text{sim}_(C_i, C_j)|$. Esto hace que el método favorezca a aquellas jerarquías donde los nodos C_j seleccionados, los más similares, sean nodos grandes, ya que para el valor final se tiene en cuenta el valor de la intersección de cada grupo natural con su nodo más parecido, sin normalizarlo respecto a su tamaño.

Supongamos una jerarquía tan inadecuada que el nodo más parecido sea el nodo raíz para todos los clusters. En ese caso la jerarquía obtendría la mayor puntuación posible ya que $C_i \cap X = C_i$. En la Figura 3.4 se puede ver un ejemplo de una base de datos ilustrativa de 2 clusters de 4 casos y un dendrograma asociado a esos datos. La jerarquía une en primer lugar cada caso de un cluster con otro del cluster contrario. Evidentemente la jerarquía no es fiel a los datos, pero el valor asignado por el índice de validación es un 1.

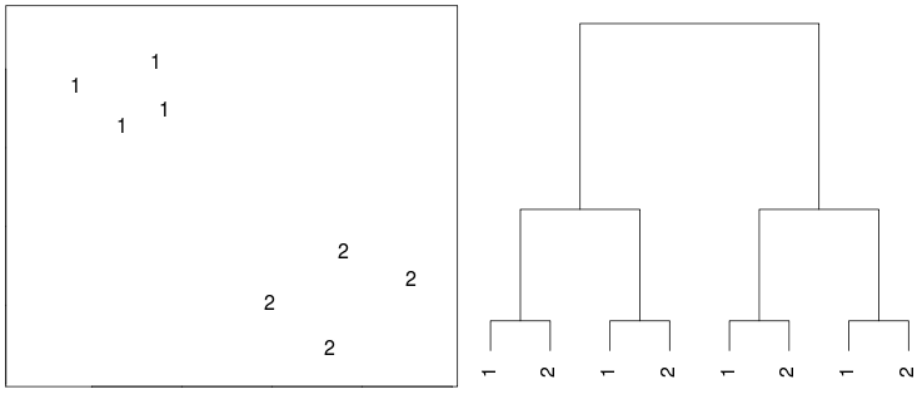

Figura 3.4: Una base de datos sencilla y un dendrograma notoriamente incorrecto.

La variante que proponemos utiliza la misma función para encontrar el nodo más parecido a cada grupo natural, pero utiliza el valor calculado por dicha función como puntuación para la jerarquía respecto a cada grupo. La puntuación final de la variante propuesta se obtiene a través de la media de la puntuación de todos los grupos ponderada por el tamaño de cada uno.

$$V'(H) = \frac{\sum_{C_i \in P^*} |C_i| \, \text{máx}_{C_j \in T_H} \, \text{sim}(C_i, C_j)}{N}$$

Nótese que el índice propuesto asigna un valor de 0,5 al dendrograma de la Figura 3.4 ya que $\text{sim}(C_i, X) = 4/8$. Es sencillo concluir que con una base de datos de K clusters del mismo tamaño el índice asignaría un valor de $1/K$ a una jerarquía equivalente. Este caso supone un límite inferior para el índice por lo que se puede definir un índice normalizado entre 0 y 1 de la siguiente manera: $V'_K(H) = \frac{V'(H)K-1}{K-1}$.

3.4.3. Similitud entre SAHN y SIHC

La primera parte de la experimentación tiene como objetivo cuantificar la diferencia entre las jerarquías construidas en su totalidad mediante el método SAHN y aquellas otras actualizadas mediante el método SIHC. De este modo, se puede conocer la desviación que genera el uso de SIHC respecto al uso exclusivo de SAHN.

Para ello, se han comparado las jerarquías construidas con valores de m entre 1 y 19 con la jerarquía construida exclusivamente con SAHN ($m = 20$). Como se ha mencionado antes, la comparación entre jerarquías se ha realizado obteniendo tanto la matriz cofenética como la matriz PMD de cada una de ellas.

La Figura 3.5 muestra los resultados obtenidos utilizando matrices cofenéticas. Los símbolos huecos muestran los resultados para las bases de datos sintéticas mientras que los sólidos describen los resultados para las bases de datos reales. Conviene recordar que para cada base de datos, algoritmo y valor de m se ha repetido el proceso 25 veces, por lo que cada punto de la figura corresponde a la media de 25 valores.

Como era de esperar las jerarquías se parecen más a la jerarquía 100 % SAHN cuanto mayor sea la proporción de casos utilizados en la fase SAHN. También se puede apreciar con claridad que la similitud de las jerarquías depende del algoritmo utilizado. El algoritmo *single-linkage* es el que obtiene mayor nivel de similitud mientras que *complete-linkage* es el que da los valores más bajos. Para finalizar, se puede observar que las jerarquías correspondientes a las bases de datos reales muestran valores sensiblemente más altos que las correspondientes a las sintéticas.

La Figura 3.6 muestra, de forma similar, los resultados obtenidos utilizando matrices PMD. Las conclusiones obtenidas no varían. La diferencia principal estriba en que los valores son algo más bajos que los anteriores, en

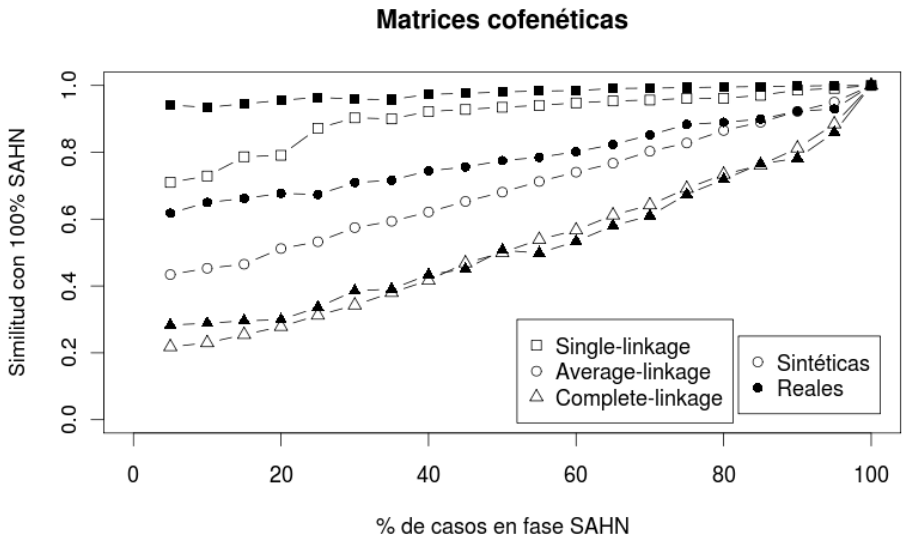

Figura 3.5: Resultados de similitud basados en matrices cofenéticas. La abscisa representa el porcentaje de la base de datos utilizada en la generación de la jerarquía SAHN ($5 \times m$).

particular los correspondientes a las bases de datos reales. Esto provoca que las diferencias entre tipos de bases de datos sean menores en este caso.

Los resultados parecen indicar que algunas variantes de SAHN son más adecuadas que otras para ser adaptadas a una versión incremental. El hecho de que la estructura de los clusters tenga más importancia en los algoritmos *average-linkage* y *complete-linkage* que en *single-linkage* hace que esta última sea más adaptable a una variante incremental.

Como conclusión, se puede ver con claridad que en general las jerarquías obtenidas mediante SIHC son distintas a las obtenidas mediante SAHN. Existe una única excepción: el algoritmo *single-linkage* donde a partir de valores medios de m las jerarquías son prácticamente iguales a las de SAHN. Una vez aclarado que la adición de casos mediante SIHC hace que las jerarquías difieran de las que han sido creadas totalmente mediante SAHN, la sección siguiente trata de medir la calidad o bondad de cada una de las jerarquías.

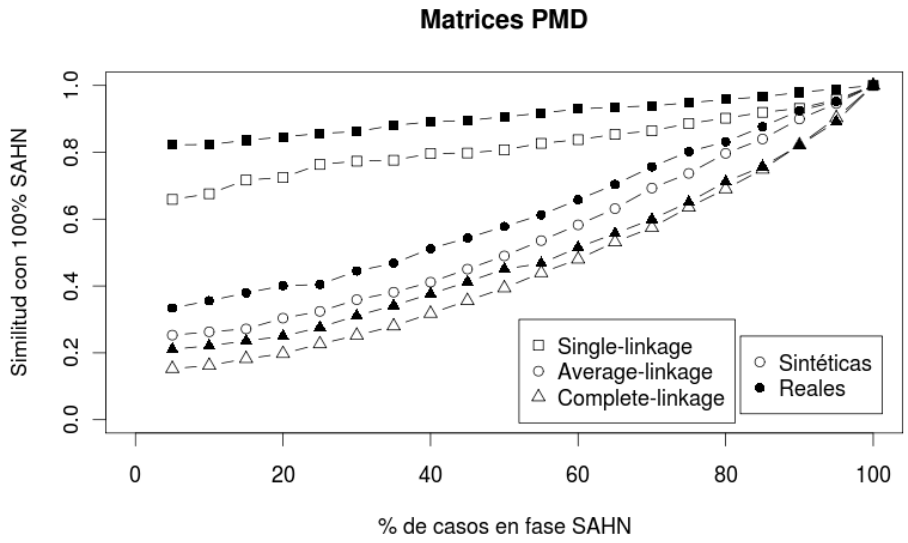

Figura 3.6: Resultados de similitud basados en matrices PMD. La abscisa representa el porcentaje de la base de datos utilizada en la generación de la jerarquía SAHN ($5 \times m$).

3.4.4. Adecuación de la jerarquía a los datos

En esta sección se cuantifica la bondad de las jerarquías construidas mediante el método SIHC en relación a las construidas mediante SAHN. Para ello, como es habitual al evaluar jerarquías, se utiliza una validación interna. En concreto, se compara, mediante el índice de correlación de Pearson, una matriz obtenida de la jerarquía con la matriz de distancias correspondiente a los datos utilizados para construir dicha jerarquía. En esta sección, al igual que en la anterior, se han utilizado tanto la matriz cofenética como la matriz PMD.

Posteriormente, dadas la limitaciones que sufren las técnicas de validación internas, se ha realizado una validación externa utilizando el índice propuesto en la Sección 3.4.2.

Validación interna

Dado que el objetivo es cuantificar la bondad de las jerarquías SIHC respecto a las jerarquías SAHN, se ha procedido de la siguiente manera. Para cada base de datos y algoritmo se calcula como valor de referencia la

bondad de la jerarquía 100 % SAHN. Posteriormente se calcula la bondad de cada jerarquía SIHC (25 jerarquías por cada algoritmo, base de datos y valor de m). El valor representado en la figura es la media de estos valores, normalizado respecto al valor de referencia (la bondad de SAHN). Esto significa que un valor superior a 1 implica que el método SIHC supera al método SAHN y un valor de 1 implica un comportamiento equivalente de los dos métodos.

Dadas las limitaciones impuestas a SIHC (no se conocen todos los datos de inicio, se prima la estabilidad respecto a la calidad . . .) no son de esperar muchos valores mayores que 1. De hecho, la Figura 3.7 muestra que nunca se supera el valor 1.

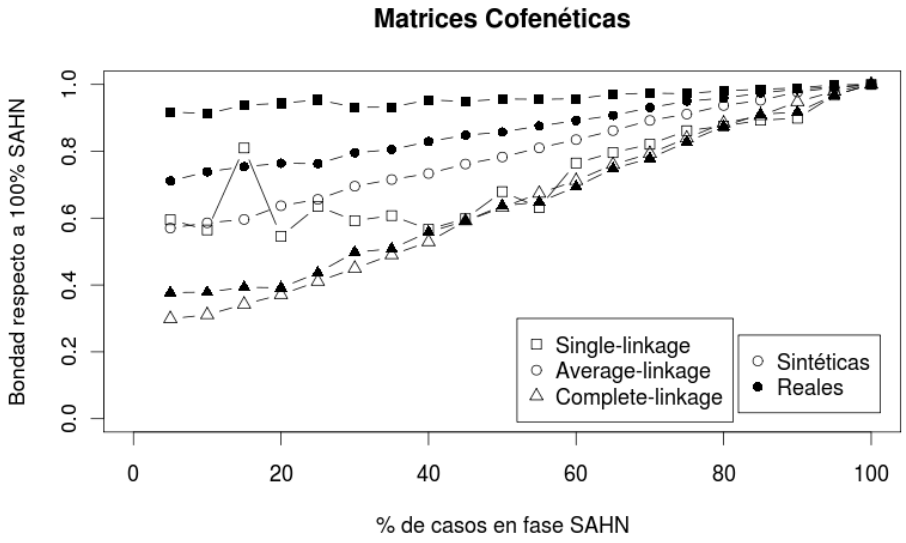

Figura 3.7: Resultados de bondad basados en matrices cofenéticas. La abscisa representa el porcentaje de la base de datos utilizada en la generación de la jerarquía SAHN ($5 \times m$).

En este caso, al igual que con la similitud, los valores crecen a medida que el número de casos añadidos incrementalmente disminuye. Esto hace que, en términos generales, el método SIHC no sea adecuado para entornos donde el número de casos a añadir supere con creces los casos conocidos en la fase inicial (fase SAHN). De todos modos, hay una excepción para las bases de datos reales y el algoritmo *single-linkage*. En este caso los dendrogramas SIHC eran similares a SAHN sin que la cantidad de datos utilizados en

cada fase afectara de forma significativa. Por consiguiente, SIHC obtiene resultados muy similares a SAHN. En el caso de *average-linkage* ocurre algo similar, pero la pendiente es algo más acusada. De todos modos se obtienen resultados que superan el 70 % incluso cuando la cantidad de datos añadidos con SIHC es 19 veces mayor que los utilizados en la fase SAHN ($m = 1$).

En el caso de las bases de datos sintéticas la degradación creada por la adición incremental de los casos es más acusada. Sorprende el hecho de que por primera vez el algoritmo *single-linkage* no supera al resto. Un análisis más detallado muestra que los resultados obtenidos para la base de datos *Concéntrico* son engañosos dado que el valor de bondad de la jerarquía 100 % SAHN es de 0,02. El problema surge porque el modo de medir la bondad no es adecuado para esta base de datos y no por la jerarquía en sí, ya que ésta detecta los 3 anillos de la base de datos de forma correcta. La medición de la bondad mediante la matriz PMD soluciona en parte dicho problema.

La Figura 3.8 muestra los resultados correspondientes a las matrices PMD. A diferencia de lo ocurrido al medir la similitud de los métodos, en este caso sí que hay diferencias significativas entre los dos modos de evaluación utilizados. Vemos que en este caso se consiguen valores superiores a 1 en más de un caso.

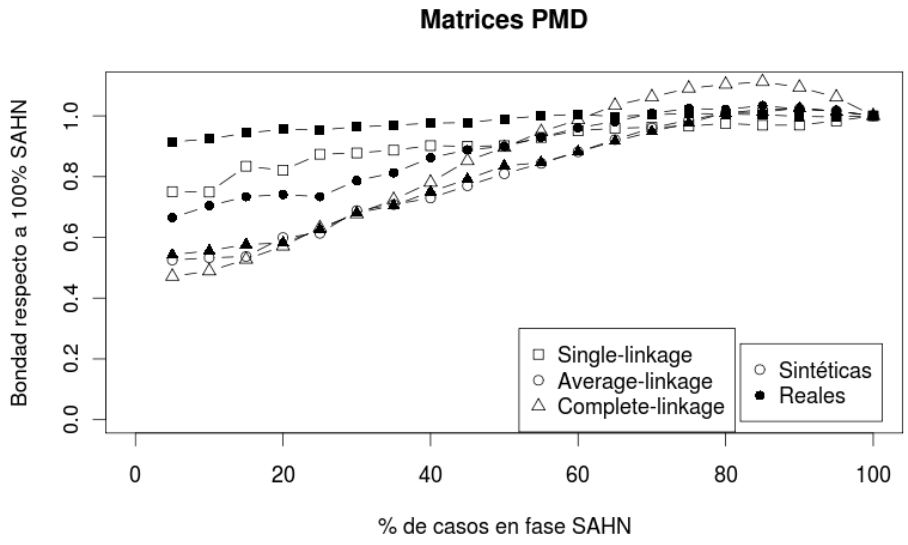

Figura 3.8: Resultados de bondad basados en matrices PMD. La abscisa representa el porcentaje de la base de datos utilizada en la generación de la jerarquía SAHN ($5 \times m$).

Empezando por las bases de datos reales se puede ver que los resultados son similares aunque para valores bajos de m los valores de *single-linkage* y *average-linkage* son algo más bajos y algo más altos los de *complete-linkage*. De todos modos, dado que la pendiente es más acusada, alcanzan valores cercanos e incluso superiores a 1 para valores altos de m. Esto significa que, según este método de validación, el uso moderado del método SIHC no sólo no reduce la calidad de las jerarquías sino que puede incluso mejorarla. La mejora en las bases de datos sintéticas es mucho más acusada y confirma las conclusiones obtenidas mediante las bases de datos reales.

Por otro lado, puede sorprender el hecho de que los resultados obtenidos para bases de datos reales sean mejores que los obtenidos para bases de datos sintéticas. De todos modos hay que recordar que los resultados son relativos a los valores obtenidos por la jerarquía 100 % SAHN. Esto significa que SIHC se comporta mejor, en relación a SAHN, en bases de datos reales que en bases de datos sintéticas. Los valores de bondad absolutos demuestran que los resultados son algo mejores para las bases de datos sintéticas, exceptuando las dos especiales: *Concéntrico* y *T&U*. Por lo tanto, parece que SAHN hace valer la ventaja de conocer todos los casos desde el principio en bases de datos mejor definidas mientras que para bases de datos más complejas la ventaja pasa a ser menos decisiva.

Vistos los resultados parece que, como regla general, se puede aplicar el método SIHC a una jerarquía sin excesiva pérdida de calidad siempre que el número de los casos añadidos incrementalmente no supere el de los casos iniciales. En el caso del algoritmo *single-linkage* el número de casos añadidos incrementalmente puede ser incluso mayor.

Validación externa

Tal y como se menciona en este mismo capítulo, se ha utilizado una medida de validación externa para evaluar la bondad de las jerarquías construidas. En esta sección se describen los resultados obtenidos en dicha validación. Al igual que en la validación interna, los resultados se han normalizado respecto al valor obtenido mediante la jerarquía 100 % SAHN, por lo que un valor superior a 1 indica un mejor resultado del método SIHC.

Como se puede observar en la Figura 3.9, los resultados confirman las conclusiones obtenidas hasta ahora. Por un lado, la calidad de las jerarquías se degrada cuando el número de casos añadidos incrementalmente crece. De todos modos, esta degradación es sensiblemente menor para el algoritmo *single-linkage*.

Los resultados también confirman el mejor comportamiento del método

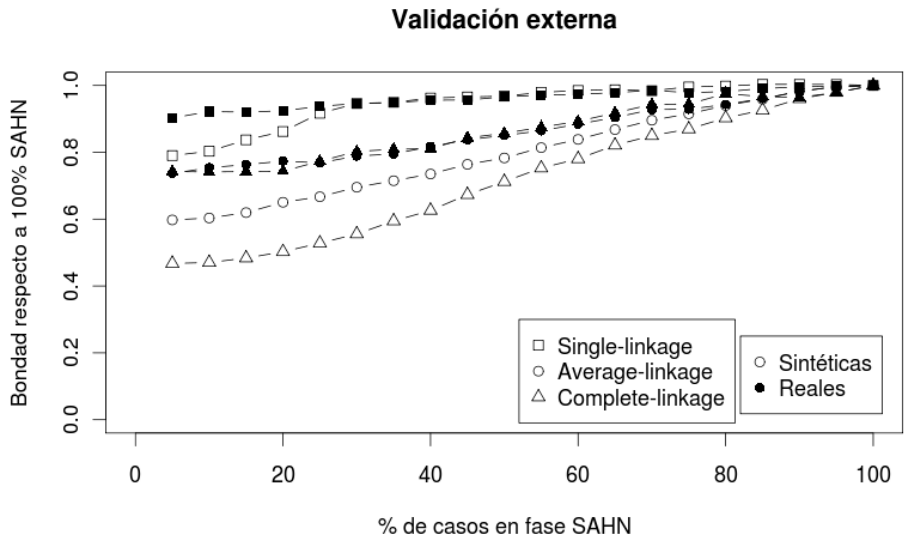

Figura 3.9: Resultados de bondad mediante validación externa. La abscisa representa el porcentaje de la base de datos utilizada en la generación de la jerarquía SAHN ($5 \times m$).

SIHC en las bases de datos reales que en las sintéticas. En estas bases de datos el algoritmo *single-linkage* obtiene, incluso para un valor de $m = 1$, una calidad del 90 % de la que obtiene la jerarquía SAHN. Asimismo, el comportamiento de las jerarquías SIHC es también superior al 80 % a partir de valores medios de m para los algoritmos *average-linkage* y *complete-linkage*.

3.5. Conclusiones

En este capítulo se ha propuesto un nuevo método, denominado SIHC, para añadir casos incrementalmente a una jerarquía de clusters construida mediante el método SAHN. A diferencia de otras aproximaciones el objetivo de SIHC no es ser fiel al método SAHN, sino añadir los nuevos casos modificando lo menos posible la estructura existente. Dicho en otras palabras, el objetivo de SIHC es la estabilidad, parte fundamental de la explicabilidad. Además, una jerarquía actualizada con SIHC sigue representándose mediante un dendrograma binario donde la altura de un nodo significa la distancia entre sus nodos hijo. Por lo tanto, cualquier método de post-proceso de

dendrogramas SAHN funciona con dendrogramas SIHC.

Una experimentación sobre casi 25.000 dendrogramas corrobora que las jerarquías actualizadas mediante SIHC difieren de las creadas exclusivamente con SAHN. Como era de esperar también queda claro que las jerarquías actualizadas mediante SIHC difieren más de las construidas exclusivamente con SAHN cuanto mayor es el número de casos añadidos incrementalmente. Los resultados han demostrado que la diferencia es mínima cuando el algoritmo utilizado es *single-linkage*.

De todos modos, los resultados sobre la calidad de los dendrogramas sugieren que la diferencia generada por SIHC no siempre implica una pérdida de calidad. Aunque un uso excesivo del método incremental implica casi siempre una pérdida de calidad considerable (*single-linkage* es una excepción en este sentido) su uso moderado no sólo no reduce significativamente la calidad sino que la puede aumentar. Aunque no es posible encontrar una regla general parece que una cantidad de casos añadidos incrementalmente igual al número de casos del dendrograma original puede ser un límite razonable.

Los resultados en torno a la calidad han sido corroborados por dos métodos de validación interna y uno de validación externa, por lo que pueden considerarse de una robustez notable. Precisamente, el método de validación externa utilizado es otra de las aportaciones de este capítulo. En concreto se ha propuesto una variación de un método de validación externa de jerarquías de clustering después de haber demostrado los problemas del método original. El método ha corroborado los resultados obtenidos mediante la validación interna.

Parte del trabajo descrito en este capítulo ha sido presentado en un congreso internacional (Gurrutxaga et al., 2009).

Capítulo 4

SEP, un algoritmo para buscar la mejor partición de una jerarquía de forma eficiente y COP, un nuevo índice de validación de clusters

4.1. Introducción

En este capítulo se afronta otro aspecto del clustering jerárquico. En particular se propone un novedoso método de post-proceso de jerarquías que permite encontrar la partición óptima en una jerarquía.

En la literatura científica es habitual encontrar trabajos en los que una jerarquía de clusters se reduce a una única partición (Bezdek y Pal, 1998; Maulik y Bandyopadhyay, 2002). A veces una jerarquía puede ser excesivamente compleja y se reduce con el objetivo de simplificarla. En otras ocasiones la jerarquía no es más que un medio para llegar a una única partición. Ciertamente, un dendrograma es habitualmente demasiado complejo como para analizar todos sus nodos y, de hecho, la mayoría de los nodos no suelen aportar mucha información. Sólo algunos, los que representan los grupos naturales, son realmente interesantes.

La solución ideal a este problema debería poder procesar la jerarquía

para eliminar los nodos poco informativos y reducir la jerarquía original a una jerarquía más simple. De todos modos, la aproximación habitual es más drástica y reduce la jerarquía a una única partición, perdiendo, por tanto, la posibilidad de definir distintos niveles de agrupación.

En este capítulo se demuestra que el método generalmente utilizado para reducir una jerarquía a una partición, al que llamaremos método tradicional, no es satisfactorio en determinados contextos. Por consiguiente, se propone una modificación del método que, mediante un nuevo algoritmo llamado SEP, se comporta satisfactoriamente también en dichos contextos. Además, este nuevo algoritmo permite una adaptación a través de la cual el resultado del post-proceso puede ser una jerarquía simplificada en lugar de una única partición. De este modo se reduce la complejidad del dendrograma sin perder las ventajas del clustering jerárquico.

Asimismo, en este capítulo se propone un nuevo índice de validación de clustering, COP, ya que los índices habituales no cumplen los requisitos necesarios para ser utilizados con el algoritmo SEP.

4.2. Método tradicional para reducir una jerarquía a una partición

El método tradicional para obtener una partición a partir de una jerarquía de clusters es conceptualmente sencillo. Ya se ha mencionado anteriormente que el método SAHN comienza con una partición donde cada caso se considera un cluster distinto y termina con la partición donde todos los casos forman un único cluster. En el proceso intermedio se van uniendo los clusters más cercanos obteniendo en cada paso una partición con un cluster menos que en el anterior. Esto nos lleva a que en el proceso de construcción de una jerarquía SAHN se definan N particiones, siendo N el número de casos de la base de datos procesada. A este conjunto de particiones lo denominaremos conjunto de particiones de una jerarquía, o simplemente, una jerarquía.

Una vez definida la jerarquía es sencillo definir el método tradicional de post-proceso de una jerarquía para reducirla a una partición: se selecciona un índice de validación de clusters o CVI (véase la página 40) y se evalúan las particiones de la jerarquía; la mejor partición, según el CVI utilizado, será la seleccionada.

Lo habitual es que la última partición (todos los casos en un cluster) se ignore ya que la mayoría de los CVI tienen problemas con ella. En realidad, lo más común suele ser reducir la búsqueda a un subconjunto de las particiones

de la jerarquía. Dado que muchos CVI están sesgados a valorar positivamente las particiones con un elevado número de clusters el subconjunto analizado suele estar compuesto por las particiones que tienen entre 2 y K_{max} clusters. Para la elección de K_{max} suelen utilizarse dos estrategias: asignarle un valor fijo, típicamente 15 o 20, o un valor en función de N, típicamente $\sqrt{N}$.

Dado que normalmente el número de clusters es mucho menor que el número de casos, esta búsqueda reducida no suele generar problemas. Por otra parte, al reducir el espacio de búsqueda, también se consigue reducir el coste computacional. De todas maneras, en aplicaciones con muchos clusters, la partición correcta podría quedar fuera del espacio de búsqueda.

En resumen, el método tradicional para reducir una jerarquía a una partición consiste en seleccionar la mejor partición (según un CVI concreto) del conjunto de particiones correspondiente a la jerarquía. En la práctica se analizan sólo las particiones con un número reducido de clusters.

Siguiendo la notación definida en el Capítulo 2, podemos definir formalmente el método general como una función de búsqueda $\mathrm{S}(\mathrm{V}, H) = \arg\,\mathrm{mín}_{P \in H}\, \mathrm{V}(P)$. De todos modos, como se acaba de mencionar, la búsqueda se restringe habitualmente a una parte de la jerarquía: $H' = \{P : P \in H, \quad 1 < |P| \leq K_{max}\}$.

En algunos casos la selección de la mejor partición se realiza en el mismo proceso de construcción de la jerarquía. En estos casos el proceso de construcción se suele detener cuando una partición es claramente peor que la anterior, devolviendo como resultado la penúltima partición. Una de las ventajas de esta variante es que podemos acortar el tiempo de construcción de la jerarquía ya que una vez seleccionada la mejor partición el proceso de construcción se finaliza. La desventaja más evidente de este procedimiento es que las particiones con un reducido número de clusters quedan sin analizar y por lo tanto el resultado puede ser un óptimo local. En este contexto los CVI suelen denominarse criterios de parada o reglas de parada (Milligan y Cooper, 1985).

Una variante de este tipo de algoritmos es aquélla donde los criterios de parada deciden parar la construcción de un subárbol y continúan con el resto. Dicho de otro modo, una vez se decide que uno de los clusters forma un grupo natural, éste se "congela" y se continúa con el proceso ignorando el cluster "congelado" (Fred y Leitão, 2003).

Una forma más visual de ver el método tradicional es a través de la representación gráfica más común de una jerarquía de clusters: el dendrograma. Cualquier partición del conjunto de particiones de una jerarquía puede ser representada mediante una recta horizontal en el dendrograma correspondiente. Cuanto más arriba se sitúe la recta, ésta representará una partición

con menos clusters. Por contra, cuanto más abajo esté, representará una partición con más clusters. En la figura 4.1 se puede ver como ejemplo una base de datos y su dendrograma correspondiente construido mediante el algoritmo *average-linkage*. Las dos rectas añadidas al dendrograma definen dos particiones de 3 y 14 clusters.

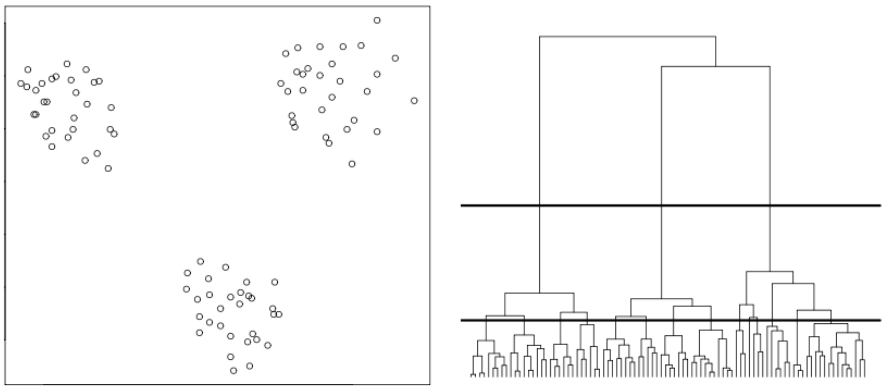

Figura 4.1: Base de datos y dendrograma correspondiente con líneas rectas indicando particiones de 3 y 14 clusters.

Por lo tanto, el método tradicional se puede ver como un método que corta un dendrograma con rectas horizontales en varios puntos de la parte superior y decide cuál es la mejor partición en base a un CVI.

El procedimiento descrito es simple e intuitivo y su coste computacional es asequible. No obstante, tiene el inconveniente de cortar el dendrograma con una recta. Es decir, para que el método funcione todos los clusters tienen que tener un nivel de densidad similar. Téngase en cuenta que los clusters compactos se definen en la parte baja del dendrograma mientras que los clusters más dispersos no se completan hasta llegar a los nodos más altos. En la Figura 4.2 se muestra una base de datos bidimensional de tres clusters y un dendrograma construido mediante el algoritmo *average-linkage*. Se puede ver con claridad que la partición correcta de tres clusters no se puede obtener mediante una recta horizontal. Kettenring (2006), en una reciente revisión, ya menciona el problema y lo ilustra con un ejemplo similar. Además, reivindica la necesidad de un método más sofisticado de extracción de particiones de una jerarquía: *"Because of the popularity of HCA [Hierachical Cluster Analysis], more sophisticated tools for extracting clusters from the dendrogram would be very beneficial."*.

Se podría considerar que el problema subyace en el algoritmo de cons-

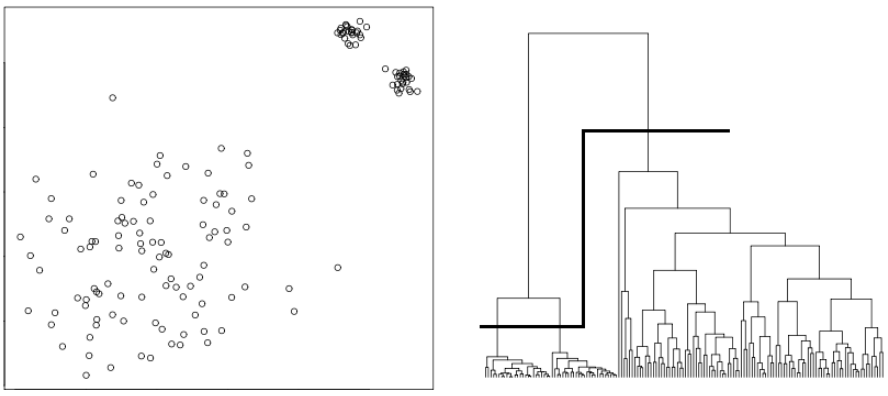

Figura 4.2: Base de datos y dendrograma correspondiente con una línea quebrada representado la partición correcta.

trucción de la jerarquía. Al fin y al cabo es el algoritmo de construcción quien decide unir los dos clusters compactos antes de terminar con el cluster más disperso, haciendo que la partición correcta no se encuentre en el conjunto de particiones de la jerarquía. Pero también es verdad que la jerarquía define la partición correcta de forma implícita, como se ha podido comprobar en la Figura 4.2.

Por lo tanto, una búsqueda que incluya las particiones que se representan en un dendrograma con cualquier tipo de línea dará mejores resultados que la búsqueda habitual. En la próxima sección se describe un algoritmo que realiza dicha búsqueda de forma eficiente. Definimos como Conjunto Extendido de Particiones o *Extended Partition Set* (EPS) de una jerarquía de clusters al conjunto de particiones que se pueden definir mediante cualquier tipo de línea (sea recta o quebrada) en el dendrograma correspondiente a la jerarquía.

A continuación se define el EPS más formalmente: $E_H = \{P : P \subseteq T_H, \bigcup_{C \in P} C = X \text{ y } \forall C_k, C_l \in P \quad C_k \cap C_l = \emptyset\}$ donde $T_H = \bigcup_{P \in H} \bigcup_{C \in P} C$ y X es el conjunto de datos. En otras palabras, EPS es el conjunto de particiones que pueden ser construidos combinando los diferentes nodos del dendrograma.

4.3. SEP: Búsqueda en el conjunto extendido de particiones

En esta sección se propone modificar el método de extracción de una partición a partir de una jerarquía de clusters. La propuesta no modifica la idea básica del método tradicional, simplemente modifica el espacio de búsqueda. En lugar de buscar en un subconjunto de particiones de la jerarquía se propone buscar en el EPS completo: $\mathrm{S}(\mathrm{V}, E_H)$.

Es evidente que una búsqueda exhaustiva en el EPS es inviable ya que el número de particiones en el EPS de cualquier jerarquía no trivial es inmensa. A modo de ejemplo hemos calculado el número de particiones en los EPS de los dendrogramas de las Figuras 4.1 y 4.2. A pesar de que las bases de datos utilizadas tienen tan sólo 90 y 150 casos, el tamaño de los EPS es de $5,67 \times 10^{14}$ y $4,90 \times 10^{23}$ respectivamente.

La propuesta fundamental de este capítulo es un nuevo algoritmo de búsqueda que permite realizar una búsqueda en un EPS de forma eficiente siempre que el CVI utilizado cumpla una serie condiciones. A continuación se describe el algoritmo, al que hemos llamado SEP (*Searching the Extended Partition set*), y se describen las requisitos necesarios para que un CVI pueda ser utilizado junto con SEP.

4.3.1. El algoritmo SEP

Para explicar la idea fundamental del algoritmo SEP es conveniente recordar el concepto de partición parcial explicado en el Capítulo 2. Una partición parcial es muy similar a una partición clásica, o partición total. La diferencia estriba en que la unión de todos los clusters de una partición parcial no tiene por qué ser la base de datos completa; puede ser un subconjunto de ella.

La descripción del algoritmo SEP, que se aplica a jerarquías de clusters, es más sencilla si se explica en base a dendrogramas, tal y como se hace en el Algoritmo 4.1. Desde este punto de vista, SEP recorre todo el dendrograma, desde los nodos hoja hasta el nodo raíz, decidiendo la mejor partición parcial para cada nodo. El caso más sencillo es el de los nodos hoja, donde SEP decide que la mejor partición parcial es la que contiene un único cluster con los casos (normalmente sólo uno) de ese nodo (línea 3 del Algoritmo 4.1).

En el resto de los nodos SEP compara dos particiones parciales (línea 9). La primera de ellas es la que contiene un único cluster, con todos los casos correspondientes al nodo en cuestión. La segunda consiste en la unión de las mejores particiones parciales de los nodos hijos del nodo analizado.

Algoritmo 4.1 SEP(V, $Nodo$)

```
1:  C ← cluster en Nodo
2:  if Nodo es un nodo hoja then
3:     return {C}
4:  else
5:     Union ← ∅
6:     for all Hijo ∈ Nodo do
7:        Union ← Union ∪ SEP(V, Hijo)
8:     end for
9:     if V({C}) ≤ V(Union) then
10:       return {C}
11:    else
12:       return Union
13:    end if
14: end if
```

La mejor de las dos particiones, según un CVI pasado como parámetro, será considerada como la mejor para ese nodo. El algoritmo ha de recorrer el dendrograma de abajo hacia arriba ya que para calcular la mejor partición de cada nodo es necesario conocer las mejores particiones de sus hijos. Esto se consigue mediante la llamada recursiva de la línea 7. Aunque hemos utilizado el algoritmo con dendrogramas SAHN puede ser utilizado con otro tipo de dendrogramas ya que no realiza ninguna suposición (número de hijos por nodo, tamaño de los nodos hoja...).

SEP hace posible la búsqueda de la mejor partición en el EPS de la jerarquía analizada con un coste computacional razonable. De hecho, SEP necesita calcular un CVI dos veces por cada nodo del dendrograma. De todos modos, uno de los cálculos puede ser simplificado si es posible calcular el valor del CVI para la unión de varias particiones parciales, en base a los valores del CVI para dichas particiones. En ese caso, el CVI tendría que ser calculado una vez por cada nodo. Por lo tanto, en el peor de los casos (un dendrograma binario), el CVI se calcularía $2N - 1$ veces.

4.3.2. Análisis de las propiedades requeridas al índice de validación utilizado con el algoritmo SEP

Uno de los parámetros del algoritmo SEP es el índice de validación de clusters a utilizar para medir la bondad de las particiones analizadas. Desafortunadamente, el CVI a utilizar en combinación con SEP ha de cumplir

una serie de requisitos que la mayoría de los CVI no cumplen. En esta sección se definen los requisitos que un CVI debe cumplir para poder ser utilizado con SEP y se explica por qué la mayoría de los CVI no los cumplen.

Propiedades requeridas por SEP

SEP requiere que el CVI utilizado cumpla dos requisitos. El primero de ellos es indispensable para que el algoritmo funcione, mientras que el segundo es necesario para que el algoritmo garantice que se encuentra la mejor partición. Antes de describir los requisitos se realiza una reflexión acerca del algoritmo SEP que ayuda a explicar por qué son necesarios.

El algoritmo SEP analiza dos particiones parciales en cada nodo. Una de ellas es la partición parcial correspondiente a un único cluster con todos los objetos del nodo analizado. Esto requiere que el índice de validación de clusters sea capaz de analizar una partición con un único cluster. Dado que los CVI normalmente manejan el concepto de separabilidad entre clusters, no son capaces de analizar particiones con un único cluster. De todos modos, SEP trabaja con particiones parciales, por lo que requiere que el CVI analice una partición parcial con un único cluster, pero permite medir la separabilidad a partir de los objetos no incluidos en dicha partición parcial (la única excepción la encontramos en el nodo raíz).

De todos modos, una de las claves de SEP estriba en calcular la mejor partición parcial de un nodo del dendrograma de forma independiente a la parte del dendrograma que no cuelga del nodo. Por lo tanto, el CVI en cuestión debe calcular la bondad de una partición parcial conociendo dicha partición y toda la base de datos, pero sin presuponer ninguna estructura en los datos no correspondientes a la partición parcial.

Por otra parte, en cada paso del algoritmo SEP se unen las mejores particiones parciales de los hijos del nodo analizado, para formar uno de los dos candidatos a mejor partición parcial para este nodo (el otro candidato es el que tiene un único cluster). Por lo tanto, el CVI debe garantizar que la mejor partición para el subconjunto de datos Y sigue siendo la mejor en un subconjunto con más datos, $Y \cup Z$.

De los párrafos anteriores se infieren las propiedades que un CVI debe cumplir para que pueda ser utilizado correctamente junto con el algoritmo SEP:

- *Parcialidad*: El CVI debe ser capaz de calcular la bondad de una partición parcial, aunque ésta contenga un único cluster.
- *Optimalidad independiente de contexto*: El CVI debe garantizar que

si una partición parcial es mejor que otra, lo seguirá siendo en otro contexto con más datos. Describimos formalmente dicha propiedad como sigue:

$$\mathrm{V}(P_1^Y) < \mathrm{V}(P_2^Y) \Leftrightarrow \mathrm{V}(P_1^Y \cup P^Z) < \mathrm{V}(P_2^Y \cup P^Z), Y \cap Z = \emptyset$$

Relación entre las propiedades requeridas por SEP y los CVI habituales

A continuación se describen los problemas que los CVI habituales tienen con las propiedades de Parcialidad y Optimalidad independiente de contexto. Aunque los CVI están diseñados para evaluar particiones totales, es sencillo emplearlos para evaluar particiones parciales: basta con tratar la partición parcial como si fuera total, ignorando todos los casos no incluidos en la partición parcial. Dicho de otro modo, si la forma habitual de un CVI es $\mathrm{V}(P^X, X)$ y nosotros queremos un CVI que calcule $\mathrm{V}(P^Y, X)$ siendo $Y \subseteq X$, basta con calcular $\mathrm{V}(P^Y, Y)$. Es decir, ignorar los objetos correspondientes a X/Y.

Aunque esta solución permite utilizar los CVI habituales para evaluar particiones parciales, no permite evaluar particiones parciales con un único cluster dado que generalmente los CVI no son capaces de evaluar una partición que incluya toda la base de datos en un cluster. Por lo tanto, se necesita un CVI que pueda evaluar particiones parciales de forma natural, incluyendo los datos que no están en la propia partición parcial.

A continuación se demuestra que ninguno de los CVI descritos en la Sección 2.3.2 cumple el requisito de Parcialidad debido a las razones descritas en los párrafos anteriores.

Calinski-Harabasz contiene el término $|P| - 1$ en el denominador, por lo que en el caso de un único cluster el valor del índice es infinito. En el caso de C-Index los valores de S, S_{min} y S_{max} serían iguales provocando una indeterminación de tipo 0/0. Lo mismo ocurriría con Gamma ya que la suma de todos los $\mathrm{dl}(x_i, x_j)$ sería 0 y $n_w = N(N-1)/2$. En el caso de G+ no habría ninguna cuádrupla inconsistente por lo que el valor del índice sería siempre 0. Por otra parte, al no poder calcular la distancia entre clusters no es posible el cálculo del índice McClain-Rao. Finalmente, tanto Davies-Bouldin como Dunn requieren, por definición, la comparación de al menos dos clusters por lo que tampoco cumplirían la propiedad de Parcialidad. En definitiva, el análisis de una partición con un único cluster es un problema que los índices de validación de clusters suelen obviar y por tanto, no serán utilizables en el algoritmo SEP.

Por otra parte, la propiedad de Optimalidad independiente de contexto no es una propiedad masivamente incumplida. De todos modos hay algunos índices que la incumplen. Un caso habitual es el del índice que asigna el mismo peso o importancia a cada cluster (caso habitual al evaluar cada cluster por separado y combinar los resultados mediante la media). En ese caso, la mejor partición parcial para unos datos puede depender de cómo se estructuren los datos fuera de la partición parcial (más clusters darán más peso a la partición).

El índice Davies-Bouldin puede servir de ejemplo. Dicho índice calcula un valor para cada cluster de la partición y finalmente calcula la media de todos los valores. Supongamos una partición parcial, P_1^Y, con dos clusters, cuyo valor de Davies-Bouldin es de $0,5$. Por otra parte, supongamos otra partición parcial de los mismos datos, P_2^Y, pero con 10 clusters y un valor de Davies-Bouldin de $0,6$. Esta claro que la mejor partición según Davies-Bouldin es P_1^Y, ya que obtiene un valor menor.

Supongamos ahora que unimos las dos particiones parciales con otra, P^Z, que tiene dos clusters y un valor de Davies-Bouldin de $0,8$. El valor que Davies-Bouldin asigna a la unión de dos particiones es el siguiente:

$$\mathrm{V}(P^Y \cup P^Z) = \frac{|P^Y| \times \mathrm{V}(P^Y) + |P^Z| \times \mathrm{V}(P^Z)}{|P^Y| + |P^Z|}$$

Por lo tanto, los valores asignados a $P_1^Y \cup P^Z$ y $P_2^Y \cup P^Z$ son:

$$\mathrm{V}(P_1^Y \cup P^Z) = \frac{2 \times 0,5 + 2 \times 0,8}{2+2} = 0,65$$

$$\mathrm{V}(P_2^Y \cup P^Z) = \frac{10 \times 0,6 + 2 \times 0,8}{10+2} = 0,63$$

El valor alto de la partición parcial P^Z "arrastra" más a P_1^Y que a P_2^Y, ya que tiene menos clusters. Por consiguiente, la partición parcial P_2^Y pasa a ser mejor, con lo que la propiedad de Optimalidad independiente de contexto queda incumplida.

Aunque una demostración formal del incumplimiento de esta propiedad es compleja, es posible encontrar contraejemplos que demuestran que la mayoría de los CVI descritos en la Sección 2.3.2 no lo cumplen.

En resumen, dadas las propiedades requeridas por el algoritmo SEP, parece más adecuado diseñar un nuevo CVI específico para él.

4.4. COP: Un índice que cumple las propiedades de Optimalidad independiente de contexto y Parcialidad.

En esta sección se describe un nuevo índice de validación de clustering al que llamamos COP (*Context-independent Optimality and Partiality*) y que cumple las propiedades descritas en la sección anterior. En primer lugar se describe la idea básica del índice, orientada al cumplimiento de los requerimientos del algoritmo SEP. Posteriormente se realiza una descripción formal de COP y finalmente se describen una serie de propiedades del índice.

COP es un índice de validación de clusters que como muchos otros se basa en el concepto de cohesión de los clusters (varianza intra) y separabilidad de los clusters (varianza inter). COP calcula un nivel de bondad o calidad para cada cluster analizado y posteriormente combina todos los niveles mediante una media ponderada por el tamaño de los clusters. El nivel de bondad de cada cluster es un ratio entre el nivel de varianza intra y el nivel de varianza inter del cluster, por lo que COP es un índice de tipo ratio (Kim y Ramakrishna, 2005).

COP estima el nivel de varianza intra de un cluster mediante un estimador típico para el nivel de cohesión: la distancia media entre el centroide y los casos del cluster. En cambio, el nivel inter se calcula como la distancia entre el cluster y el punto más cercano no perteneciente al cluster, siguiendo los criterios de *complete-linkage*. Es decir, para cada punto fuera del cluster se calcula la distancia al punto más lejano dentro del cluster. La menor de estas distancias será el valor del nivel inter.

Hay que subrayar que el punto fuera del cluster puede incluso estar fuera de la partición parcial. Por lo tanto, COP evalúa la separabilidad de cada cluster analizando los objetos fuera de éste, independientemente de si están en la partición parcial o no. Además, no necesita presuponer ningún tipo de estructura fuera de la partición parcial ya que examina cada objeto de forma independiente.

La definición formal del índice COP es la siguiente:

$$\mathrm{COP}(P^Y, X) = \frac{1}{|Y|} \sum_{C \in P^Y} |C| \frac{\mathrm{intra}_{COP}(C)}{\mathrm{inter}_{COP}(C)}$$

$\mathrm{intra}_{COP}(C) = \frac{1}{|C|} \sum_{x \in C} \mathrm{d}(x, \bar{C})$, $\mathrm{inter}_{COP}(C) = \mathrm{mín}_{x_i \in X \setminus C} \mathrm{máx}_{x_j \in C} \mathrm{d}(x_i, x_j)$.

A continuación se describen una serie de propiedades del índice COP. Por un lado se demuestra que COP está acotado entre 0 y 1. Por otra parte, se muestra que COP cumple las propiedades de Parcialidad y Optimalidad

independiente de contexto. Finalmente, se demuestra que el valor de COP para una partición parcial puede ser calculado a partir de los valores de las particiones parciales de las que se compone. De este modo, se garantiza que SEP necesitará calcular directamente el valor de COP una única vez por cada nodo (véase la Sección 4.3.1).

Se puede ver claramente que COP ≥ 0 ya que es un ratio entre valores que sólo incluyen valores positivos (cardinalidades de conjuntos y distancias euclidianas). También se puede demostrar que COP ≤ 1 ya que el denominador es siempre mayor o igual que el numerador, $\text{inter}_{COP}(C) \geq \text{intra}_{COP}(C)$.

Demostración. Supongamos que es posible que $\text{inter}_{COP}(C) < \text{intra}_{COP}(C)$. Por lo tanto, $\exists x_k \in X \setminus C : \text{máx}_{x_i \in C}\, \text{d}(x_i, x_k) < \text{intra}_{\text{COP}}(C)$. Y dado que la media nunca pude ser mayor que el máximo, $\exists x_k \in X \setminus C : \frac{1}{|C|} \sum_{x_i \in C} \text{d}(x_i, x_k) < \text{intra}_{\text{COP}}(C)$. Dado que d representa la distancia euclidiana, desarrollando $\text{intra}_{\text{COP}}(C)$ reescribimos la inecuación anterior como $\frac{1}{|C|} \sum_{x_i \in C} \sqrt{\sum_{f=1}^{F} (x_{if} - x_{kf})^2} < \frac{1}{|C|} \sum_{x_i \in C} \sqrt{\sum_{f=1}^{F} (x_{if} - \bar{C}_{.f})^2}$. Pero eso no es posible ya que la media (el centroide) es el punto que minimiza la suma de los cuadrados de las diferencias de los valores del conjunto a un punto cualquiera. Por lo tanto, no puede existir un punto x_k con las características mencionadas. Por consiguiente, la suposición inicial es incorrecta e $\text{intra}_{COP}(C) \leq \text{inter}_{COP}(C)$. □

COP alcanza su valor máximo en el improbable caso en el que el objeto más cercano al cluster (y no perteneciente a él) esté en el centroide del cluster y que todos los puntos del cluster estén a la misma distancia respecto a él.

Por otra parte, es obvio que COP puede evaluar particiones parciales, incluso las que sólo tienen un cluster. Esto se consigue al medir la separabilidad de los clusters con la base de datos completa en lugar de utilizar sólo los casos implicados en la partición parcial. El único problema surge cuando la partición de un único cluster es total. En este caso, como los CVI habituales, COP no tiene criterio para evaluar la separabilidad de los clusters. Algo similar ocurre con los clusters con un único objeto, dado que COP no puede evaluar la cohesión del cluster. Para evitar este problema la solución que se ha adoptado es asignar un valor de 1 a este tipo de clusters.

A continuación se demuestra que COP cumple la propiedad de Optimalidad independiente de contexto. Para ello, primero se describe cómo calcular COP para la unión de dos particiones parciales:

$$\text{COP}(P^Y \cup P^Z, X) =$$

$$= \frac{1}{|Y|+|Z|}\left(\sum_{C\in P^Y}|C|\frac{\text{intra}_{\text{COP}}(C)}{\text{inter}_{\text{COP}}(C)}+\sum_{C\in P^Z}|C|\frac{\text{intra}_{\text{COP}}(C)}{\text{inter}_{\text{COP}}(C)}\right)=$$

$$= \frac{1}{|Y|+|Z|}(|Y|\,\text{COP}(P^Y)+|Z|\,\text{COP}(P^Z)) \tag{4.1}$$

Ahora, se puede demostrar que $\text{COP}(P_1^Y, X) < \text{COP}(P_2^Y, X) \Leftrightarrow \text{COP}(P_1^Y \cup P^Z, X) < \text{COP}(P_2^Y \cup P^Z, X)$.

Demostración. $\text{COP}(P_1^Y \cup P^Z, X) < \text{COP}(P_2^Y \cup P^Z, X)$ puede ser reescrito, siguiendo la Ecuación 4.1, como $\frac{1}{|Y|+|Z|}(|Y|\,\text{COP}(P_1^Y) + |Z|\,\text{COP}(P^Z)) < \frac{1}{|Y|+|Z|}(|Y|\,\text{COP}(P_2^Y)+|Z|\,\text{COP}(P^Z))$. Finalmente, eliminando los términos iguales de cada lado se obtiene $\text{COP}(P_1^Y) < \text{COP}(P_2^Y)$, de forma que la equivalencia de ambas inecuaciones queda probada. □

Para terminar el análisis de las propiedades de COP a continuación se generaliza la Ecuación 4.1 para el caso de la unión de un conjunto de particiones parciales, U.

$$\text{COP}(\bigcup_{P\in U} P, X) = \frac{1}{\sum_{P\in U}\text{n}(P)}\sum_{P\in U}\text{n}(P)\,\text{COP}(P)$$

donde $\text{n}(P) = \sum_{C\in P}|C|$ es el número de objetos en una partición y todas las particiones en U son disjuntas.

Esta fórmula permite calcular el valor de COP correspondiente a la unión de un conjunto de particiones parciales basándose en los valores que COP asigna a cada una de las particiones unidas. Por lo tanto, el algoritmo SEP ha de calcular el índice COP directamente una única vez por cada nodo del dendrograma analizado.

4.5. Experimentación

En esta sección se describe la experimentación realizada para validar el comportamiento del algoritmo SEP junto con el índice de validación de clusters COP. Para ello hemos construido una serie de jerarquías a partir de un conjunto de bases de datos (sintéticas y reales) utilizando 3 algoritmos basados en el método SAHN. Estas jerarquías han sido post-procesadas siguiendo el método tradicional para extraer una partición utilizando para ello 5 CVI habituales en la literatura científica. Posteriormente hemos utilizado el algoritmo SEP junto con el índice COP para extraer una partición

de las mismas jerarquías y finalmente se ha medido la bondad de todas las particiones obtenidas mediante técnicas de validación externa (medidas de similitud de particiones).

A continuación se describe en detalle la experimentación realizada y los resultados obtenidos.

4.5.1. Metodología experimental

Para la evaluación del algoritmo SEP hemos utilizado 80 bases de datos sintéticas y 10 reales. A continuación se describe el proceso general de construcción de las bases de datos.

El proceso principal de generación de bases de datos sintéticas ha sido similar para las 80 bases de datos. Cada base de datos tiene 10 clusters. Para la ubicación de cada cluster, C_k, se ha seleccionado aleatoriamente un punto de referencia en un espacio bidimensional entre (0,0) y (50,50). Posteriormente, se ha generado un conjunto de puntos con una distribución normal con media en el punto de referencia y matriz de covarianza $\sigma_k \times I_2$, donde I_2 es la matriz identidad de 2 dimensiones. El número de casos asignados a cada cluster se ha seleccionado aleatoriamente entre 25 y 50 siguiendo una distribución uniforme. Por otra parte, σ_k es un valor aleatorio entre 0,1 y 3 extraído siguiendo una distribución similar a una campana gaussiana invertida. De este modo se favorecen las varianzas extremas. Hemos repetido este proceso 10 veces para obtener 10 bases de datos.

Con el objeto de evaluar el comportamiento de SEP y COP con distintos niveles de ruido hemos añadido ruido a las 10 bases de datos generadas en el párrafo anterior. En concreto, hemos añadido nuevos objetos siguiendo una distribución uniforme. De esta manera hemos creado otros 3 grupos de 10 bases de datos con niveles de ruido de 5 %, 10 % y 20 %.

Uno de los problemas de estas bases de datos es que algunos clusters pueden estar solapados con otros. Aunque éste es un problema que se da en las bases de datos reales, tiene el inconveniente de que el etiquetado de los casos puede ser confuso. En algunas ocasiones dos o más clusters pueden estar completamente solapados y en ese caso el concepto de cluster o grupo natural puede ser discutible. Éste es un problema inherente al clustering ya que en muchos casos la definición de cluster o grupo natural es una decisión subjetiva. Este problema se discute en la Sección 2.2.5.

Para evitar este tipo de debate a la hora de decidir el etiquetado hemos creado otro conjunto de 40 bases de datos de forma similar a la anterior. La diferencia está en que hemos forzado que los clusters no se solapen de forma significativa. Para ello, a la hora de crear los puntos de referencia de

cada cluster hemos descartado el cluster C_k y lo hemos sustituido por otro si $\mathrm{d}(C_k, C_l) < 3 \times (\sigma_k + \sigma_l) \forall l \neq k$. De este modo, sabemos que los resultados obtenidos para estas bases de datos no están afectados por etiquetados ambiguos o subjetivos.

En resumen, el conjunto de 80 bases de datos sintéticas se puede clasificar siguiendo dos criterios: nivel de solapamiento de los clusters (con solapamiento y sin solapamiento) y el nivel de ruido (0 %, 5 %, 10 % y 20 % de ruido).

Las bases de datos reales han sido extraídas del repositorio UCI de la Universidad de California, Irvine (Asuncion y Newman, 2007). Se han seleccionado bases de datos con distintas características (número de dimensiones, número de clusters y número de casos) que se resumen en la Tabla 4.1.

Base de datos	Dimensiones	Clusters	Casos
Ecoli	8	8	336
Glass	9	7	214
Haberman	3	2	306
Ionosphere	34	2	351
Iris	4	3	150
Page-blocks	10	5	5473
Segmentation	19	7	2310
Vehicle	18	4	846
Wine	13	3	178
Yeast	8	10	1484

Tabla 4.1: Características de las bases de datos reales.

Para cada una de las 90 bases de datos se han creado 3 jerarquías utilizando 3 algoritmos basados en el método SAHN: *single-linkage*, *average-linkage* y *complete-linkage*. Por consiguiente, los resultados obtenidos en esta experimentación se basan en 270 jerarquías. El solapamiento de los clusters y el ruido añadido provocan que la mayoría de las jerarquías no contengan la partición correcta. Por lo tanto, hemos calculado la partición del conjunto de particiones de la jerarquía que más se parece a la partición correcta. La partición de la jerarquía que más se parece a la correcta la denotaremos como $P' = \arg\,\mathrm{máx}_{P \in H}\, \mathrm{sim}(P, P^*)$ donde sim es una medida de similitud de particiones y P^* es la partición correcta de la base de datos. La partición P' establece un límite superior para el método tradicional de extraer una partición de una jerarquía, ya que se trata de la "mejor" partición de todo el conjunto de particiones de la jerarquía. Por lo tanto, se considera la par-

tición objetivo del método tradicional. Dado que SEP realiza la búsqueda en un espacio distinto, existe la posibilidad de que encuentre una partición aún mejor, es decir, más similar a la partición correcta.

No existe una medida universal para comparar particiones y en la literatura se pueden encontrar multitud de aproximaciones (Albatineh et al., 2006; Pfitzner et al., 2009). Por esta razón, en esta experimentación hemos medido la similitud entre particiones utilizando dos medidas de similitud distintas. La primera de ellas, Adjusted Rand (AR), es una medida ampliamente utilizada que casi se ha convertido en un estándar *de facto* (Dasgupta y Ng, 2009; Mucha, 2006; Silva et al., 2006; Yeung et al., 2001). Esta medida es una normalización de la medida Rand de forma que la similitud entre particiones aleatorias sea cercana a 0. La segunda medida, Variation of Information (VI), es una medida basada en la teoría de la información y su utilidad ha sido demostrada teóricamente por Meilă (2003).

En resumen, para cada una de las 270 jerarquías hemos calculado la mejor partición en base a dos medidas de similitud. Posteriormente, hemos utilizado el método tradicional para extraer una partición de una jerarquía en base a 5 índices de validación y hemos medido la similitud de las particiones extraídas respecto a la partición correcta en cada caso. En este punto también hemos utilizado las dos medidas de similitud: AR y VI.

Para terminar, hemos utilizado SEP junto con COP para buscar la mejor partición del EPS de una jerarquía y hemos medido su similitud respecto a la partición correcta utilizando las medidas de similitud AR y VI.

4.5.2. Resultados

En este apartado se muestran los resultados obtenidos en la experimentación llevada a cabo para evaluar el comportamiento del algoritmo SEP junto con el índice de validación COP. En primer lugar se muestran los resultados para las bases de datos sintéticas y a continuación se muestran los resultados correspondientes a las bases de datos reales.

Bases de datos sintéticas

Para cada una de las 240 jerarquías correspondientes a las bases de datos sintéticas calculamos: a) la similitud entre la partición correcta y las particiones extraídas siguiendo el método tradicional en base a 5 CVI, y b) la similitud entre la partición correcta y la partición extraída siguiendo el algoritmo SEP en base a COP. En cada caso la similitud se ha medido mediante 2 medidas de similitud.

Debido a la gran cantidad de datos hemos agrupado los resultados correspondientes a las bases de datos construidas siguiendo el mismo proceso. Por lo tanto hemos agrupado los resultados en 8 grupos de 10 bases de datos cada uno: 4 niveles de ruido y 2 niveles de solapamiento. Para cada grupo hemos contado las veces que cada CVI ha sido capaz de encontrar la mejor partición de la jerarquía. Por otra parte, hemos contado las veces que SEP y COP han sido capaces de encontrar una jerarquía tan buena o mejor que la mejor partición de la jerarquía. Los resultados se resumen en cuatro tablas (Tablas 4.2-4.5), una por cada nivel de ruido. El superíndice de la fila SEP/COP muestra el número de veces que SEP y COP han sido capaces de encontrar una partición mejor que "la mejor" partición de la jerarquía.

Los algoritmos *single-linkage*, *average-linkage* y *complete-linkage* se indican como sl, al y cl respectivamente. En términos generales los resultados para los tres algoritmos son similares aunque existen excepciones. No obstante, no se ha encontrado ningún patrón característico en ninguno de los algoritmos por lo que los comentarios realizados corresponden a un resultado medio.

Solap.:	Con solapamiento						Sin solapamiento					
Similitud:	AR			VI			AR			VI		
Algoritmo:	sl	al	cl	sl	al	cl	sl	al	cl	sl	al	cl
CH	4	3	3	6	6	2	6	7	6	8	8	6
C-Index	3	1	1	3	2	3	3	3	5	4	4	6
Gamma	3	1	1	3	2	3	2	3	5	3	4	6
DB	2	1	2	4	2	3	4	5	6	6	6	7
Dunn	0	0	0	0	1	1	1	1	3	3	2	4
SEP/COP	3^1	2^1	3^2	5^1	3^1	3^1	9^5	9^5	10^6	10^5	10^5	10^6

Tabla 4.2: Número de veces que cada CVI encuentra la mejor partición en las bases de datos sin ruido.

En la Tabla 4.2 se pueden ver los resultados para las bases de datos sin ruido. El índice Calinski-Harabasz (CH) obtiene claramente los mejores resultados para el método tradicional y bases de datos con clusters solapados. El resto demuestran un comportamiento similar excepto Dunn, que muestra un comportamiento netamente inferior. Los resultados para las bases de datos sin clusters solapados son, como era esperable, mejores. De todos modos, la tendencia relativa sigue siendo la misma.

Sin embargo, los resultados para SEP/COP no siguen la misma tendencia en las bases de datos con distinto nivel de solapamiento. En los casos

donde existe solapamiento entre clusters SEP/COP muestra un comportamiento mejor que la mayoría de los CVI con el método tradicional, pero no supera el nivel de Calinski-Harabasz. No obstante, en las bases de datos sin solapamiento entre clusters SEP/COP exhibe un comportamiento casi perfecto, mejorando los resultados del método tradicional en todos los casos.

Solap.:	Con solapamiento						Sin solapamiento					
Similitud:	AR			VI			AR			VI		
Algoritmo:	sl	al	cl	sl	al	cl	sl	al	cl	sl	al	cl
CH	1	3	1	2	7	4	3	3	7	5	4	6
C-Index	1	2	3	2	3	3	3	4	6	4	5	7
Gamma	1	3	2	2	4	2	3	4	6	4	5	7
DB	1	2	1	2	2	2	3	2	3	4	3	3
Dunn	0	0	0	1	1	0	1	1	4	2	2	4
SEP/COP	2^1	2^1	3^2	5^2	3^1	3^3	8^5	7^4	9^5	9^5	8^4	9^5

Tabla 4.3: Número de veces que cada CVI encuentra la mejor partición en las bases de datos con 5 % de ruido.

Añadir un poco de ruido a las bases de datos no genera mayores diferencias para el método tradicional en términos generales. Como se puede observar en la Tabla 4.3 parece que Calinski-Harabasz es quien más sufre el ruido y baja ligeramente a un nivel similar al resto de los CVI. Dunn sigue manteniendo unos niveles de éxito muy bajos.

Los resultados para SEP/COP apenas varían para las bases de datos con solapamiento entre clusters. En el resto de las bases de datos los resultados empeoran ligeramente, pero siguen siendo mejores que los del método tradicional. Por lo tanto, el ruido añadido ha provocado que SEP/COP muestre, en general, el mejor comportamiento.

Los resultados correspondientes a un nivel de ruido del 10 % (Tabla 4.4) muestran un declive generalizado para todos los CVI del método tradicional. Si exceptuamos a Dunn, que sigue mostrando los peores resultados, las diferencias entre el resto de CVI no son significativas. De todos modos, parece que Davies-Bouldin sufre las consecuencias del ruido más que los demás.

Los resultados para SEP/COP son totalmente distintos dado que muestran unos resultados similares, e incluso alguna pequeña mejoría, respecto a los resultados con 5 % de ruido. Estos resultados parecen indicar que el ruido afecta seriamente al método tradicional, pero que el algoritmo SEP es robusto frente a este tipo de perturbaciones.

Para poder confirmar las conclusiones relacionadas con el efecto del ruido

Solap.:	Con solapamiento						Sin solapamiento					
Similitud:	AR			VI			AR			VI		
Algoritmo:	sl	al	cl	sl	al	cl	sl	al	cl	sl	al	cl
CH	0	1	5	1	2	8	0	3	5	1	4	5
C-Index	0	1	3	1	2	5	0	3	6	1	3	6
Gamma	0	1	2	1	2	4	0	3	6	1	3	6
DB	0	0	1	0	1	2	0	2	4	1	2	4
Dunn	0	0	1	0	1	1	0	1	1	1	1	1
SEP/COP	3^1	2^1	4^2	6^2	3^1	6^3	9^6	7^5	10^9	10^6	8^5	9^8

Tabla 4.4: Número de veces que cada CVI encuentra la mejor partición en las bases de datos con 10 % de ruido.

Solap.:	Con solapamiento						Sin solapamiento					
Similitud:	AR			VI			AR			VI		
Algoritmo:	sl	al	cl	sl	al	cl	sl	al	cl	sl	al	cl
CH	0	1	2	0	3	6	0	3	6	0	3	6
C-Index	0	1	2	0	3	5	0	1	1	0	1	1
Gamma	0	1	2	0	3	5	0	1	1	0	1	1
DB	0	1	0	0	2	0	0	0	3	0	1	3
Dunn	0	0	0	0	1	2	0	3	2	0	4	2
SEP/COP	2^1	2^1	4^3	5^2	4^1	4^2	9^6	8^5	8^7	10^6	8^5	8^7

Tabla 4.5: Número de veces que cada CVI encuentra la mejor partición en las bases de datos con 20 % de ruido.

en el comportamiento de ambos métodos, la Tabla 4.5 muestra los resultados para las bases de datos con un nivel de ruido elevado (20 %). Una vez más, se puede observar el efecto negativo que el incremento de ruido provoca en el método tradicional. Los resultados confirman que el efecto negativo afecta a todos los CVI, pero sobre todo a Davies-Bouldin. Sorprendentemente, el índice Dunn vuelve a obtener valores similares a los obtenidos para bases de datos sin ruido. De todas maneras, son resultados muy pobres que confirman que este índice es el menos adecuado según la experimentación realizada.

Por otra parte, los resultados correspondientes a SEP/COP confirman que dicho algoritmo es prácticamente insensible al ruido ya que los niveles de éxito siguen siendo parecidos y alcanzan el 85 % en el caso de las bases de datos sin solapamiento.

En resumen, los resultados confirman que para las bases de datos sintéti-

cas el algoritmo SEP junto con el índice COP supera con claridad al método tradicional, independientemente del índice utilizado en éste. Las diferencias son más acusadas en las bases de datos sin clusters solapados y con altos niveles de ruido.

Es importante recalcar que la mejoría conseguida por SEP/COP es mayor de lo que se puede ver en una primera lectura. Hay que recordar que para el método tradicional cada unidad en la tabla significa haber encontrado la mejor partición de la jerarquía; pero para SEP/COP significa haber encontrado la mejor partición de la jerarquía o una mejor. De hecho, aproximadamente el 60 % de la veces que SEP encuentra una partición tan buena o mejor como la mejor de la jerarquía, es en realidad mejor. Además, este porcentaje aumenta con el ruido siendo de 51 % en las bases de datos sin ruido y de 64 % en las bases de datos con 20 % de ruido. Por lo tanto, la mejoría no es sólo cuantitativa, sino también cualitativa.

Estos resultados sugieren que el incremento de ruido hace que la probabilidad de que las mejores particiones estén en la jerarquía producida por SAHN disminuya.

Bases de datos reales

En este apartado se describen los resultados obtenidos para las bases de datos reales. Los resultados mostrados corresponden a la medida de similitud VI ya que la medida AR ha demostrado tener serios problemas con las jerarquías basadas en bases de datos reales. La búsqueda de la mejor partición de cada jerarquía ha arrojado resultados sorprendentes para AR ya que en muchos casos la mejor partición obtenía valores de similitud que no superaban el 0,2. Éste es un valor muy bajo si tenemos en cuenta que AR es un índice que da valores de similitud cercanos a 0 en el caso de particiones aleatorias.

Un análisis más detallado de estos resultados ha confirmado el extraño comportamiento de AR ya que muchas de las particiones seleccionadas como mejores tenían una gran cantidad de clusters (superando incluso los 600 clusters en el caso de la base de datos Yeast). En resumen, el valor medio de AR para las mejores particiones de las jerarquías correspondientes a las bases de datos reales es de 0,33 y el número medio de clusters es de 75. Por consiguiente, este apartado se centra en los resultados basados en el índice VI dado que consideramos que los resultados de AR no son fiables en este caso.

Dado que en este caso las bases de datos son totalmente independientes se muestran los resultados sin agrupar. La Tabla 4.6 muestra los niveles de

similitud entre la partición correcta y la mejor partición según los distintos métodos e índices. La columna Lím. muestra el nivel de similitud entre la partición correcta y la más parecida de la jerarquía, por lo que establece un límite superior para el método tradicional. La última línea muestra en cuántas ocasiones se ha alcanzado ese límite superior.

		Lím.	CH	C-Index	Gamma	DB	Dunn	SEP/COP
	sl	2,076	2,112	2,120	2,171	2,171	2,171	2,171
Ecoli	al	1,151	1,611	1,488	1,488	1,283	2,089	1,244
	cl	1,301	1,301	1,311	1,546	1,687	2,295	1,301
	al	2,058	2,175	2,249	2,204	2,204	2,175	2,204
Glass	al	1,988	2,063	2,112	2,274	2,175	2,175	2,175
	cl	2,070	2,132	2,147	2,147	2,445	2,147	2,345
	sl	0,853	0,853	0,853	0,853	0,853	0,853	0,853
Haber.	al	0,930	2,435	3,496	0,930	0,932	0,930	0,932
	cl	1,645	1,645	3,646	3,646	1,709	3,776	1,645
	sl	0,962	0,962	1,270	0,962	0,962	0,962	0,962
Ionos.	al	0,962	1,242	1,358	0,962	0,962	0,962	0,962
	cl	1,412	2,306	2,804	2,804	2,804	1,829	3,175
	sl	0,667	0,667	0,667	0,667	0,667	0,667	0,667
Iris	al	0,609	0,609	1,512	0,667	0,667	0,667	0,667
	cl	0,854	1,067	1,785	1,067	0,854	1,785	0,854
	sl	0,634	0,636	0,636	0,636	0,636	0,636	0,636
Page.	al	0,635	2,093	2,572	0,635	0,854	0,635	**0,634**
	cl	0,635	3,305	3,459	0,635	0,764	0,635	**0,633**
	sl	2,218	2,814	2,814	2,814	2,816	2,814	2,814
Segm.	al	2,150	2,237	2,814	2,814	2,814	2,814	2,814
	cl	2,267	3,588	2,814	2,814	2,814	2,814	2,814
	sl	2,018	2,135	2,135	2,135	2,260	2,135	2,018
Vehicle	al	2,501	2,526	4,537	4,537	2,526	4,521	2,526
	cl	2,523	3,214	4,800	4,800	2,523	4,985	2,523
	sl	1,564	1,954	1,954	1,564	1,599	1,599	1,599
Wine	al	1,378	3,144	3,173	3,173	2,141	3,144	1,378
	cl	1,328	3,270	3,270	3,270	2,618	3,270	1,328
	sl	2,505	2,505	2,555	2,527	2,586	2,514	2,511
Yeast	al	2,495	3,300	2,505	2,505	2,505	2,495	2,505
	cl	2,495	3,768	5,220	2,505	2,505	2,495	2,505
Total			7	2	8	6	9	13^2

Tabla 4.6: Similitud, según VI, entre la partición correcta y la seleccionada por cada método. La columna Lím. muestra el límite superior para el método tradicional.

Los resultados muestran que el método tradicional se comporta de manera similar para casi todos los índices excepto C-Index, cuyo comportamiento es netamente inferior. Sorprende que el resumen de la última fila sitúe a Dunn como al mejor CVI, ya que es el único índice que ha destacado por sus malos resultados en la experimentación con las bases de datos sintéticas. Dado que Dunn apuntaba una leve mejoría con altos niveles de ruido puede ser que este índice se adapte mejor a contextos complejos.

Por otra parte, los resultados del algoritmo SEP son significativamente mejores que los del método tradicional. No sólo es el método que claramente encuentra más veces la mejor partición de la jerarquía (o incluso dos veces otra aún mejor) sino que cuando no lo hace encuentra una partición aceptable. Si calculamos la diferencia de similitud entre la partición correcta y la encontrada por cada método podemos ver que Davies-Bouldin y SEP/COP son los más estables. En media SEP/COP es el que menos se aleja del límite superior (con un valor de 0,15) seguido por Davies-Bouldin (0,25). Si nos fijamos en el peor caso para cada método, Davies-Bouldin es el que menos se aleja del límite superior, seguido de SEP/COP. Por lo tanto SEP/COP es la opción que más probabilidades tiene de encontrar la partición correcta mientras que SEP/COP y Davies-Bouldin son los que menos probabilidades tienen de dar un resultado muy alejado del correcto.

En resumen, SEP/COP muestra los mejores resultados también para bases de datos reales. Para el método tradicional todos los índices estudiados excepto C-Index muestran resultados similares aunque parece que Davies-Bouldin es más robusto que los demás.

4.6. SEP como método para reducir una jerarquía a otra jerarquía menor

En la introducción de este capítulo se hace referencia al hecho de que la reducción idónea de una jerarquía daría como resultado otra jerarquía más sencilla. Es decir, si la base de datos utilizada contiene una estructura jerárquica el dendrograma original puede ser excesivamente complejo, pero reducirla a una partición hace perder la información sobre la estructura jerárquica de los datos reflejada por el dendrograma.

La Figura 4.3 muestra una base de datos bidimensional sencilla con estructura jerárquica y su dendrograma correspondiente al algoritmo *average-linkage*. En este caso se pueden definir particiones a distinto nivel de detalle, con un número de clusters variable entre 3 y 9. La idoneidad de cada partición dependerá del nivel de detalle requerido por cada usuario o aplicación,

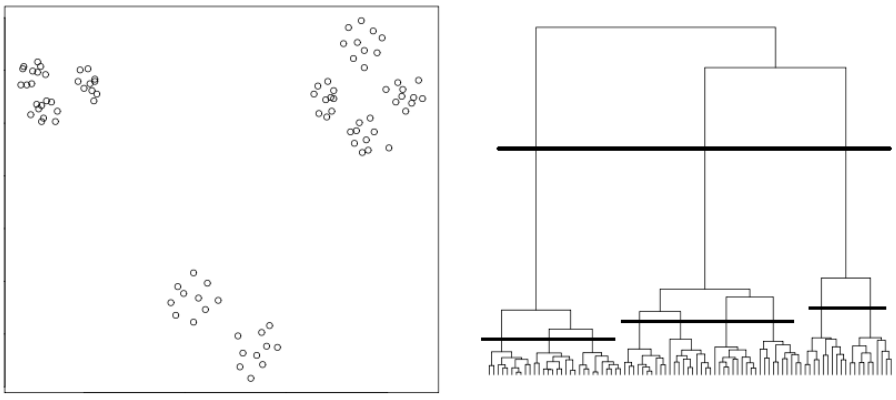

Figura 4.3: Base de datos con estructura jerárquica y dendrograma correspondiente con diferentes niveles de corte.

por lo que centrarse únicamente en una de ellas provoca pérdida de información relevante. Está claro que la reducción del dendrograma de la figura a una única partición obliga a ignorar uno de los niveles.

El algoritmo SEP puede ser utilizado para buscar, no únicamente la mejor, sino las mejores particiones de un dendrograma. Para ello el índice utilizado ha de cumplir la propiedad de Aditividad, es decir, el valor del índice para una partición ha de ser un valor calculado a partir de la suma de valores parciales para cada cluster. Es decir, el índice debe poder definirse de la siguiente manera: $\mathrm{V}(P) = \sum_{C \in P} \mathrm{v}(C)$. El índice COP cumple dicha propiedad ya que puede ser escrito como

$$\mathrm{COP}(P^Y, X) = \sum_{C \in P^Y} \frac{|C|}{|Y|} \frac{\mathrm{intra}_{COP}(C)}{\mathrm{inter}_{COP}(C)}$$

Se puede demostrar de forma sencilla que si un índice cumple la propiedad de Aditividad cumplirá la propiedad de Optimalidad independiente de contexto por lo que la variante de SEP descrita en esta sección requiere un índice que cumpla las propiedades de Parcialidad y Aditividad.

Demostración. Por la definición de la propiedad de Aditividad podemos calcular el valor de un índice para la unión de dos particiones parciales disjuntas, siempre que el índice cumpla dicha propiedad. $\mathrm{V}(P^Y \cup P^Z) = \sum_{C \in P^Y} \mathrm{v}(C) + \sum_{C \in P^Z} \mathrm{v}(C) = \mathrm{V}(P^Y) + \mathrm{V}(P^Z)$. Por lo tanto, la propiedad de Optimalidad independiente de contexto queda como $\mathrm{V}(P_1^Y) < \mathrm{V}(P_2^Y) \Leftrightarrow \mathrm{V}(P_1^Y) + \mathrm{V}(P^Z) < \mathrm{V}(P_2^Y) + \mathrm{V}(P^Z)$. □

A continuación se describe la idea general de la utilización de SEP para reducir un dendrograma a otro más sencillo. Para ello, se analiza el EPS de la jerarquía con el objetivo de encontrar las mejores particiones. Estas particiones serán variaciones de la mejor partición, bien la división de algún cluster o la unión de otros, por lo que es sencillo representar las modificaciones en el dendrograma. El resultado final será un dendrograma en el que se muestren sólo las partes correspondientes a las particiones seleccionadas.

En la Sección 4.3.1 se describe cómo el algoritmo SEP calcula en cada nodo la bondad de dos particiones según un CVI, en este caso COP. Podemos calcular la diferencia en la bondad de estas dos particiones y ordenarlas de menor a mayor, $M_1 = \{m_{11}, m_{12}, \ldots, m_{1N}\}$. La modificación correspondiente a la decisión tomada en el nodo asociado al primer valor de la lista será la que menos empeora la partición seleccionada, con lo que esta modificación generará la segunda mejor partición del EPS.

Se puede continuar el proceso para buscar la tercera mejor partición. La siguiente modificación en la lista, m_{12}, puede generar la partición buscada, pero no siempre será así ya que habrá otro candidato. Hay que recalcular los nuevos valores de COP suponiendo que se ha realizado la primera modificación (sólo variarán los nodos ascendientes del nodo modificado) y obtener una segunda lista ordenada, $M_2 = \{m_{21}, m_{22}, \ldots, m_{2N}\}$. Finalmente, habrá que seleccionar el menor valor entre m_{12} y $m_{11} + m_{21}$. En otras palabras, habrá que decidir si es mejor hacer la segunda menor modificación (m_{12}) o la menor modificación seguida de otra modificación ($m_{11} + m_{21}$) cuya suma no supere a la anterior.

Del mismo modo, se puede seguir con el proceso hasta un número de modificaciones determinado. Un criterio de parada en la búsqueda podría ser el empeoramiento abrupto del valor de COP entre dos modificaciones consecutivas. Idealmente, en un contexto como el representado en la Figura 4.3, el proceso debería terminar con un conjunto de particiones representable como el dendrograma de la Figura 4.4.

4.7. Conclusiones

En este capítulo hemos argumentado que el método tradicional para extraer una partición de una jerarquía de clusters está limitado a ciertos contextos. Hemos ilustrado y formulado el problema y hemos propuesto como solución un nuevo método. Por otra parte, dado que los índices de validación de clusters habituales no son válidos para el algoritmo propuesto hemos propuesto un nuevo índice.

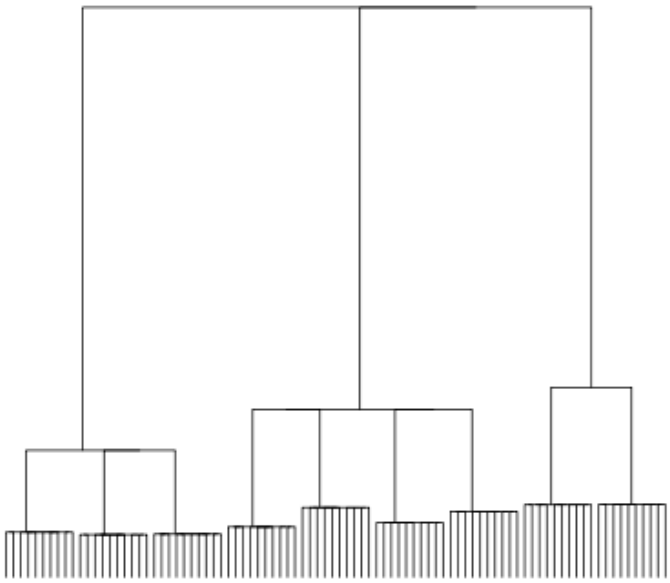

Figura 4.4: Dendrograma simplificado representando sólo grupos naturales.

Los resultados sobre 3 algoritmos jerárquicos y 80 bases de datos sintéticas y 10 reales han demostrado que el método propuesto mejora al método tradicional. En esta experimentación el algoritmo SEP superaría el 36 % de las veces al método tradicional incluso con un CVI que propusiese siempre la mejor partición de la jerarquía. Además, en otro 25 % de las veces el resultado sería igual. De todos modos, comparado con el método tradicional usando varios CVI conocidos, el algoritmo SEP lo ha superado en casi todas las ocasiones.

Aunque el método propuesto ha demostrado su superioridad en casi todas las situaciones estudiadas, los mejores resultados han sido obtenidos en situaciones con ruido y bajo nivel de solapamiento entre clusters. De hecho, el método parece ser insensible al ruido a niveles de hasta 20 %.

En el caso de las bases de datos reales el método propuesto vuelve a ser superior al método tradicional. Además, demuestra obtener resultados cercanos al límite superior del método tradicional en todos los casos. De todas maneras, en este sentido, el método tradicional junto con el índice Davies-Bouldin también obtiene buenos resultados.

Finalmente, se ha ilustrado cómo una variación del método puede post-

procesar la jerarquía de un modo más interesante. En concreto, el objetivo es extraer a partir de una jerarquía otra jerarquía de menor tamaño. De este modo se podría simplificar el dendrograma asociado sin perder la estructura jerárquica extraída de los datos.

Un artículo que recoge el trabajo descrito en este capítulo ha sido aceptado para publicación en una revista internacional (Gurrutxaga et al., 2010a).

Capítulo 5

Una nueva metodología para evaluar índices de validación de clusters

5.1. Introducción

La validación de los resultados es un paso fundamental del proceso de minería de datos. En el caso del clustering existen diversas técnicas de validación que ya han sido comentadas en la Sección 2.3.

Al igual que en otros ámbitos del aprendizaje automático, en el clustering hay multitud de caminos posibles desde la recolección de los datos hasta la obtención de unos resultados; normalmente en forma de una partición o jerarquía de particiones. La forma de obtener los datos, el modo de preprocesarlos, el algoritmo de clustering utilizado, los parámetros de dicho algoritmo... son ejemplos de las decisiones a tomar. Todas estas decisiones llevan a obtener un resultado u otro, por lo que la validación de dicho resultado es de capital importancia.

No obstante, el mayor esfuerzo en investigación se ha realizado en el proceso de aprendizaje, es decir, en mejorar y parametrizar los algoritmos de aprendizaje o crear otros nuevos. Aún así, el clustering ha sido un área donde la validación ha tenido un papel importante, por una sencilla razón: la dificultad en estimar el número correcto de clusters.

Este problema ha sido, y es, uno de los retos principales del clustering. Existen multitud de propuestas de algoritmos específicos o variaciones de algoritmos conocidos, para evitar que el usuario o un experto tenga que especificar el número de clusters que un algoritmo de clustering debe buscar.

Sin embargo, hoy en día sigue siendo un problema abierto.

Una aproximación habitual a este problema se basa en la utilización de los ya mencionados índices de validación de clusters o CVI, es decir, indicadores de la calidad o bondad de una partición. El procedimiento es sencillo: se ejecutan uno o varios algoritmos de clustering de forma que se generen varias particiones (modificando los valores de algún parámetro, seleccionando varias particiones de una jerarquía...). Posteriormente se selecciona la partición que mayor calidad tenga según un CVI determinado y se da como bueno el número de clusters de dicha partición.

El uso de los CVI para evitar la necesidad del conocimiento a priori del número de clusters de una base de datos ha llevado al desarrollo de una multitud de índices diferentes. Desafortunadamente, no hay un único índice adecuado para todas las situaciones. Ocurre lo mismo que con los algoritmos de aprendizaje, donde no existe el algoritmo ideal para todos los contextos.

Este hecho ha llevado a comparar grupos de CVI en distintos contextos para tratar de evaluar su comportamiento y su bondad. Este tipo de comparación, o evaluación de varios CVI, se realiza tradicionalmente en dos situaciones diferentes. La primera ocurre cuando un nuevo CVI se compara con otros bien conocidos, que sirven como referencia, con el objeto de demostrar la bondad del nuevo índice. El problema en este tipo de trabajos estriba en que cada investigador realiza la comparación en un contexto distinto y con unos pocos índices de referencia. Por lo tanto, es complicado contrastar la información de varios trabajos.

Este problema nos lleva a la segunda situación habitual para comparar los CVI. En este caso se suele realizar una comparación de un gran número de CVI en un contexto determinado, con el objetivo de realizar una evaluación justa que sirva como base para profundizar en el conocimiento de los CVI. Desafortunadamente, este tipo de trabajos es escaso y su diseño muy dispar, por lo que es difícil comparar los datos de distintos trabajos.

El problema de la falta de criterios unificados hace que las evaluaciones de índices de validación de clusters pierdan fiabilidad. Los criterios seleccionados en cada evaluación tienen el peligro de estar bajo sospecha de ser seleccionados *ad hoc* para llegar a los objetivos propuestos. Además, la imposibilidad de combinar la información de distintas evaluaciones y el hecho de que los resultados puedan ser contradictorios generan desconfianza en los potenciales usuarios de los algoritmos de clustering.

Por lo tanto, la evaluación de índices de validación de clusters requiere una serie de directrices comunes, recomendaciones o estándares para que los investigadores tengan criterios sólidos en los que basarse. Obviamente, para que dichas directrices sean aceptadas deben basarse en un estudio profundo

del tema y, por lo tanto, es necesario un extenso estudio teórico sobre la validación de clustering.

En este capítulo realizamos una serie de aportaciones en relación a la estandarización de la evaluación de índices de validación de clusters. Por un lado, definimos una metodología de evaluación que enmarca la mayoría de las evaluaciones conocidas. Por otra parte, argumentamos que uno de los pasos de dicha metodología no es adecuado ya que se basa en un supuesto que en muchas ocasiones no se cumple. En consecuencia, proponemos una modificación que sustituye ese paso por otro que no se basa en el mencionado supuesto y, por lo tanto, es adecuado en entornos donde la metodología tradicional no lo es.

Asimismo, se realiza una reflexión sobre las ventajas e inconvenientes de las distintas posibilidades que existen a la hora de abordar cada paso de la metodología con el objeto de limitar el número de opciones y acercarnos cada vez más a la estandarización del proceso. Aunque un estudio teórico profundo de la problemática de la evaluación de los CVI es necesario, realizamos una serie de recomendaciones que consideramos interesantes y pueden servir de punto de partida.

Finalmente, se describen los resultados de una experimentación donde se comparan varios CVI utilizando para ello dos metodologías: la metodología tradicional y la variante propuesta.

5.2. Metodología tradicional de evaluación de índices de validación de clusters

Aunque cada investigador diseña sus experimentos de evaluación de índices de forma particular, sin ningún tipo de estándar o directriz común, parece que hay una serie de elementos que se repiten en la mayoría de ellos. En base a esa serie de elementos comunes podemos describir lo que llamaremos la metodología tradicional de evaluación de índices de validación de clusters:

1. Seleccionar una serie de bases de datos cuya partición correcta sea conocida. Normalmente cada trabajo presenta sus propias bases de datos, la mayoría creadas sintéticamente. No hay ningún criterio establecido a la hora de generar las bases de datos sintéticas ni seleccionar las reales.

2. Obtener un conjunto de particiones para cada una de las bases de datos seleccionadas en el paso anterior. Se suele utilizar un algoritmo de clustering que nos permite especificar el número de clusters de la

partición de salida para obtener un conjunto de particiones con distinto número de clusters, $S = \{P_1, P_2, \ldots, P_m\}$ donde $|P_r| \neq |P_s|$, $1 \leq r, s \leq m$ y $r \neq s$. Suele ser habitual utilizar el algoritmo K-means (o alguna de sus variantes) o alguno de los algoritmos basados en el método SAHN.

3. Evaluar las particiones de salida con todos los índices de validación de clusters a evaluar, $\mathrm{V}(P_r), \quad 1 \leq r \leq m$.

4. Contar el número de aciertos para cada CVI. Para ello se selecciona, para cada CVI, la partición que ha obtenido la mejor evaluación, $\hat{P} = \arg\,\mín_{P \in S} \mathrm{V}(P)$, suponiendo que un valor menor de V denota una partición mejor. Se considera un acierto si la partición seleccionada es la que tiene el mismo número de clusters que la partición correcta, $|\hat{P}| = |P^*|$. Dicho de otra manera, $\hat{P} = P^{nc}$, donde P^{nc} es la partición en S que contiene el mismo número de clusters que la partición correcta, $P^{nc} = P \in S : |P| = |P^*|$.

5. Análisis del resultado. Normalmente el análisis se suele reducir a contar el número de aciertos de cada CVI. Sin embargo, en algunos trabajos se analizan también los fallos; por ejemplo, se trata de establecer si un CVI tiende a fallar por exceso o por defecto, por poco o por mucho...

Como se puede observar la metodología tradicional evalúa los CVI enfrentándolos al problema de determinar el número de clusters en una base de datos. La mayoría de las evaluaciones encontradas en la literatura se pueden englobar dentro de esta metodología, pero no hemos encontrado una justificación formal que la avale. De hecho, ni siquiera hemos encontrado una reflexión que describa los pasos de la metodología y discuta las distintas posibilidades de abordar cada uno de esos pasos.

5.3. El supuesto fundamental de la metodología tradicional

En esta sección se argumenta que la metodología tradicional se fundamenta en un supuesto que llamamos el supuesto de corrección del algoritmo. Asimismo, se argumenta e ilustra con varios ejemplos que dicho supuesto no se cumple en multitud de ocasiones. Posteriormente se discute acerca de la atención que ha recibido el supuesto de corrección del algoritmo en trabajos previos.

5.3.1. Descripción del supuesto de corrección del algoritmo

El mencionado supuesto afecta al paso 4 de la metodología tradicional. En concreto, el problema consiste en la definición que se hace de un acierto en dicho paso: el hecho de que la partición seleccionada por el CVI tenga el número correcto de clusters. Para que esta definición sea útil, ha de cumplirse que la mejor partición en S sea la que tiene el mismo número de clusters que la partición correcta.

Si el proceso de generación de particiones del paso 2 es capaz de crear la partición correcta, entonces es obvio que el supuesto se cumple; no hay mejor partición que la correcta y ésta tiene el número correcto de clusters. En caso contrario, dado que la partición correcta no está entre las analizadas, no se garantiza que la que tenga el número correcto de clusters sea "la mejor". Aunque la decisión de cuál es la mejor partición pueda ser discutible, parece coherente considerar como tal a la que más se parece a la partición correcta.

En resumen, la metodología tradicional no es correcta si no se cumple el supuesto de que la partición con el número correcto de clusters sea la más similar a la partición correcta. Más formalmente, el supuesto se cumple si $P^{nc} = P^{\text{sim}}$, donde $P^{\text{sim}} = \arg\,\text{máx}_{P \in S}\, \text{sim}(P, P^*)$ y sim es una función que mide la similitud entre dos particiones. Esto nos lleva a pensar que el supuesto se cumple con mayor probabilidad en bases de datos con clusters bien definidos y que se incumple frecuentemente en bases de datos con alto nivel de ruido y/o solapamiento de clusters.

A continuación, se ilustra con 3 ejemplos que el mencionado supuesto se incumple en multitud de casos, incluyendo bases de datos con clusters bien definidos y algoritmos de clustering bien conocidos. La conclusión es que la metodología tradicional de evaluación no es fiable en la mayoría de los casos.

La Figura 5.1 muestra una base de datos con 4 clusters bien definidos y algo de ruido. Si ejecutamos el algoritmo *single-linkage* sobre estos datos la partición de 4 clusters de la jerarquía corresponderá a la mostrada en la parte izquierda de la figura. Es decir, la partición tendrá dos clusters con un único punto de ruido cada uno (marcados con círculos), otro cluster con 3 grupos naturales y un cluster con un grupo natural. La partición es notoriamente incorrecta.

No obstante, si cortamos el dendrograma algo más abajo, encontraremos la partición de 10 clusters mostrada en la parte derecha de la figura. Los cuatro grupos naturales están divididos de forma correcta en cuatro clusters. Los 6 clusters restantes son clusters con un único caso de ruido cada uno. Es obvio que la partición de la derecha es mejor que la partición de la izquierda. De hecho, muchas particiones de 10 o más clusters son mejores que la de

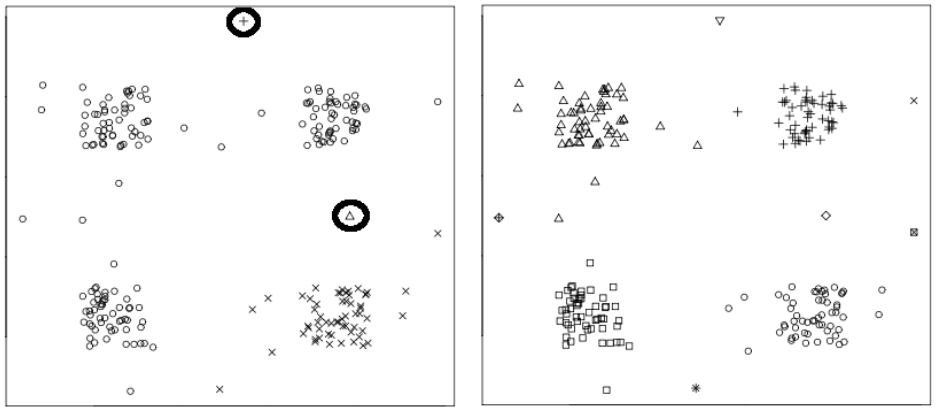

Figura 5.1: Base de datos con 4 clusters bien definidos y ruido añadido. A la izquierda una partición correspondiente a 4 clusters, a la derecha una correspondiente a 10.

4. Por tanto, un buen CVI no debería primar la partición de 4 clusters, ni nosotros deberíamos evaluarlo según ese criterio. Este ejemplo ilustra cómo en un entorno sencillo, clusters bien definidos, algo de ruido y el conocido algoritmo *single-linkage*, no se cumple el supuesto necesario para el correcto funcionamiento de la metodología tradicional de evaluación de clusters.

El segundo ejemplo se basa en una base de datos sin ruido con dos clusters bien definidos (Figura 5.2). En este caso el algoritmo utilizado ha sido el K-means. Tal y como se ve en la figura, el algoritmo ejecutado con valor $K = 2$ no ha sido capaz de definir la partición correcta dado que K-means tiene problemas para separar clusters no convexos. Debido a que la partición obtenida por K-means depende de la selección inicial de centroides, hemos ejecutado 10 veces el algoritmo con $K = 2$ y otras 10 con $K = 3$. Nueve de las particiones de 3 clusters eran más similares a la correcta que cualquiera de las 10 particiones de 2 clusters. Otra vez, descubrimos un entorno donde el supuesto de corrección del algoritmo no se cumple ya que en una base de datos con 2 grupos naturales las mejores particiones obtenidas tienen 3 clusters.

Para finalizar con los ejemplos, la Figura 5.3 muestra una base de datos muy sencilla: 4 clusters bien separados, del mismo tamaño y sin ruido añadido. En este caso, dadas las características del algoritmo y de la base de datos, es de esperar que K-means obtenga el resultado correcto. Sin embargo, en 2 de 10 ejecuciones del algoritmo K-means la partición obtenida fue incorrecta, tal y como muestra la Figura 5.3. En esos dos casos, una par-

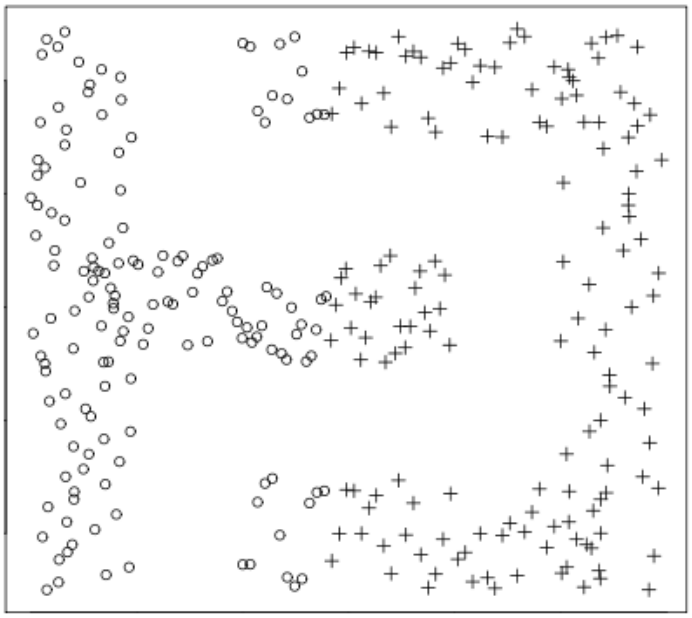

Figura 5.2: Base de datos con dos clusters no convexos bien separados. Partición obtenida con K-means ($K = 2$).

tición similar de 5 clusters que separe los dos grupos de la parte izquierda sería sin duda una mejor solución.

En resumen, con estos ejemplos queda claramente ilustrado el hecho de que confiar en que el supuesto de corrección del algoritmo se cumple es arriesgado en muchos casos. Pequeñas complicaciones, como algo de ruido o clusters no convexos, pueden hacer que la mejor partición encontrada por el algoritmo de clustering no tenga el mismo número de clusters que la partición correcta. De hecho, al contrario de lo que se podría esperar, este hecho ocurre incluso con bases de datos sencillas de particionar.

5.3.2. Tratamiento del supuesto de corrección del algoritmo en trabajos previos

Un análisis de la bibliografía relacionada descubre que varios trabajos previos aluden al supuesto de corrección del algoritmo. Sin embargo, cada uno de los trabajos actúa frente a él según sus propios criterios y ninguno trata de encontrar una solución global al problema.

A continuación se citan una serie de trabajos de evaluación de índices de

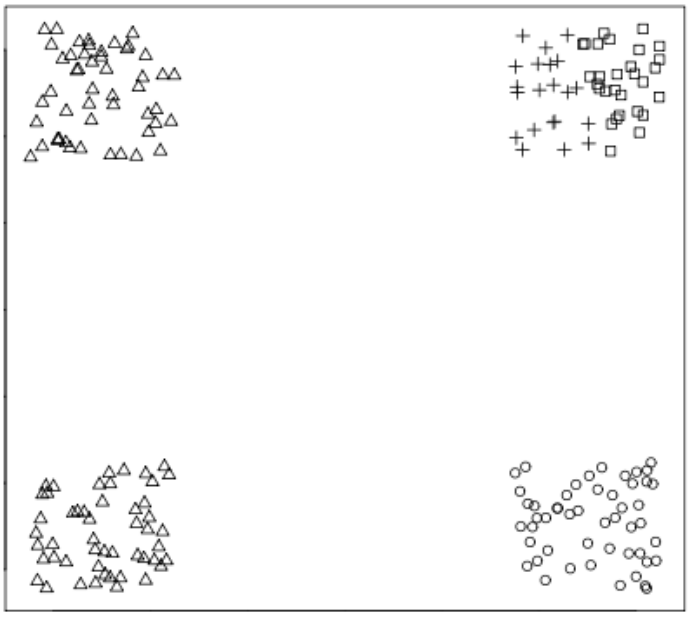

Figura 5.3: Base de datos con 4 clusters bien definidos. Partición incorrecta obtenida con K-means ($K = 4$).

validación de clusters. Todos ellos trabajan bajo el supuesto de corrección del algoritmo, pero sólo algunos de ellos lo mencionan. En este apartado se analizan dichos trabajos haciendo hincapié en su comportamiento frente al posible incumplimiento del supuesto.

Hay multitud de trabajos donde el supuesto no se tiene en cuenta (Günter y Bunke, 2003; Kim y Ramakrishna, 2005; Maulik y Bandyopadhyay, 2002; Pal y Biswas, 1997), pero los que nos interesan son aquellos que sí lo mencionan. Por ejemplo, Hardy (1996) comprueba manualmente si el supuesto se cumple o no para cada base de datos que utiliza. En el caso de que no se cumpla informa al lector de ello. Además, menciona en su trabajo el inconveniente de que en la metodología tradicional se pueda predecir el número correcto de clusters basándose en particiones incorrectas. Es decir, se puede "acertar" por azar el número correcto de clusters aunque la partición correspondiente sea claramente incorrecta. Este efecto puede desvirtuar los resultados hasta el punto de hacerlos inservibles siempre que no se tenga en cuenta el supuesto de corrección del algoritmo.

En otro trabajo (Dimitriadou et al., 2002) los autores muestran ser conscientes del problema y, por consiguiente, utilizan nuevos criterios para eva-

luar los CVI. Desafortunadamente, el trabajo se limita a bases de datos con atributos binarios y los nuevos criterios propuestos parecen servir sólo para este tipo de datos.

Por otra parte, Chou et al. (2004) utilizan algoritmos sensibles a la inicialización y ejecutan cada algoritmo varias veces por cada valor del parámetro K (número de clusters). Se calcula el valor de cada CVI para cada partición y se selecciona como número de clusters propuesto por cada CVI aquél que ha sido el más votado. Aunque este procedimiento aporta robustez y cierta independencia respecto al supuesto, no es una solución definitiva. Por ejemplo, el método funcionará con alta probabilidad en un caso como el mostrado en la Figura 5.3 donde el supuesto de corrección del algoritmo se cumple la mayoría de las veces. Pero en casos como el de la Figura 5.2 la solución no es válida ya que la mayoría de las veces, si no todas, el supuesto no se cumple.

Milligan y Cooper (1985) evalúan 30 índices en un trabajo de evaluación de índices de validación de clusters ampliamente citado. Los autores tienen en cuenta el supuesto y una vez construidas las particiones correspondientes al paso 2 de la metodología comprueban si las particiones con el número correcto de clusters son las más parecidas a la partición correcta. Para ello utilizan 2 medidas de similitud: Jaccard (Jaccard, 1908) y una versión de la medida Rand ajustada por Morey y Agresti (1984). La comprobación arrojó el resultado de que en cerca del 95 % de los casos el supuesto de corrección del algoritmo se cumplía. No obstante, consideramos que esta tasa de cumplimiento del supuesto no representa la realidad ya que los mismos autores indican que las bases de datos utilizadas en la experimentación están compuestas por grupos de datos *"internamente cohesionados y bien separados"*.

Aunque Milligan y Cooper (1985) detectaron alrededor de un 5 % de casos en los que el supuesto no se cumplía, incluyeron en su análisis los resultados obtenidos en estos casos. Creemos que hubiera sido mejor ignorarlos, ya que se basan en un supuesto incumplido y premian a los CVI que no priman la partición más similar a la correcta. De todos modos, dado el bajo número de casos donde el supuesto no se cumplía, no es esperable que en este caso los resultados hubieran variado significativamente.

En resumen, aunque muchos trabajos no tienen en cuenta el supuesto de corrección del algoritmo hay otros que sí lo hacen. De todos modos ninguno propone una solución general como la que se propone en la próxima sección. Consideramos que los trabajos que no tienen en cuenta el mencionado supuesto pueden ser correctos, pero que su validez es dudosa hasta que se realice una comprobación similar a la de Milligan y Cooper (1985).

5.4. Una nueva metodología de evaluación de índices de validación de clusters

En esta sección se presenta una modificación a la metodología tradicional de evaluación de índices de validación. Esta modificación hace que la nueva metodología no dependa del supuesto de corrección del algoritmo descrito en la Sección 5.3.1. Posteriormente se realiza un análisis de las distintas posibilidades existentes a la hora de abordar cada paso de la metodología y se enumeran las ventajas e inconvenientes de cada una. Finalmente, se realizan una serie de recomendaciones que pueden servir de punto de partida para un debate constructivo sobre la unificación de criterios en la evaluación de los CVI.

La nueva metodología modifica el paso 4 de la metodología tradicional. En particular, modifica el concepto de "acierto". Recordemos que según la metodología tradicional se considera un acierto de un CVI si la partición que ha obtenido el mejor valor para dicho CVI contiene el mismo número de clusters que la partición correcta, $\hat{P} = P^{nc}$. Según la nueva metodología se considera un acierto de un CVI si la partición que ha obtenido el mejor valor para dicho CVI es la más similar a la partición correcta, $\hat{P} = P^{\text{sim}}$.

Por lo tanto, se puede decir que la nueva metodología modifica la partición objetivo en S. Tradicionalmente la partición objetivo ha sido P^{nc} mientras que en nuestra nueva propuesta será P^{sim}. Dicho en otras palabras, según la nueva metodología el mejor CVI es aquel que con mayor frecuencia encuentra la partición más similar a la partición correcta. Por lo tanto, la metodología propuesta no depende del supuesto de corrección del algoritmo. De hecho, si el supuesto se cumple ambas metodologías son equivalentes, dado que el supuesto implica que $P^{nc} = P^{\text{sim}}$. En ese caso la partición objetivo para ambas metodologías es la misma.

Por otra parte, si sabemos que la partición correcta está entre las analizadas, $P^* \in S$, podemos asegurar que el supuesto se cumple ya que en ese caso $P^{nc} = P^*$ por definición y $P^{\text{sim}} = P^*$ para toda medida de similitud aceptable.

De todas formas, es importante recalcar que, a diferencia de la metodología tradicional, la nueva metodología no requiere que todas las particiones en S tengan un número diferente de clusters. En un contexto de este tipo las igualdades mencionadas anteriormente no son válidas ya que la metodología tradicional no es aplicable.

Antes de finalizar cabe mencionar que la nueva metodología tiene un inconveniente: es necesario un modo de decidir cuál es la partición más si-

milar a la correcta en cada caso. Esta cuestión se discute a continuación, junto con las distintas posibilidades de cada paso y subrayando las ventajas e inconvenientes de cada una. Posteriormente se enumeran una serie de recomendaciones en base a las reflexiones realizadas.

5.4.1. Reflexiones sobre la nueva metodología de evaluación

En este apartado se discuten los 5 pasos de la metodología propuesta con el objetivo de describir las distintas opciones posibles en cada uno de ellos y sus ventajas y desventajas. Dado que la mayoría de pasos de la nueva metodología coinciden con la metodología tradicional la parte fundamental del análisis es válida para ésta. Sin embargo, la reflexión está centrada en la nueva metodología y por lo tanto, algún elemento del análisis no será aplicable a la metodología tradicional.

1. *Seleccionar una serie de bases de datos cuya partición correcta sea conocida.*

   Aunque el clustering, al ser parte de la clasificación no supervisada, trabaja con datos sin etiquetar, la evaluación de clustering se beneficia enormemente del conocimiento de la partición correcta de las bases de datos con las que trabaja. La evaluación de los CVI se fundamenta en este conocimiento.

   Es por ello que la generación de bases de datos sintéticas con particiones conocidas ha sido el modo más habitual de instanciar este punto. De todos modos, la cada vez más accesible oferta de bases de datos reales etiquetadas ha hecho que hoy en día la mayoría de trabajos de validación de clustering se complementen con este tipo de datos.

   Desde nuestro punto de vista ambas opciones tienen sus ventajas e inconvenientes. Por un lado, las bases de datos sintéticas permiten trabajar en un entorno controlado y permiten adaptar las bases de datos a nuestro criterio. Permiten definir multitud de tipos de bases de datos (número de clusters, número de dimensiones, número de objetos, morfología de los clusters, nivel de ruido...) y evaluar cómo se comporta cada CVI en cada uno de ellos. Sin embargo, la fiabilidad de los resultados y su extrapolación a la realidad puede quedar en entredicho ya que la percepción de que una base de datos sintética no refleja la realidad está muy arraigada.

   Por otro lado, las bases de datos reales carecen de este inconveniente, ya que proceden de aplicaciones reales. Pero sufren un serio problema

que muchas veces es obviado: el etiquetado. El etiquetado en este tipo de bases de datos se realiza generalmente en base al conocimiento de expertos o en base a observaciones. Este hecho no garantiza que el etiquetado represente los grupos naturales de la base de datos en base a las características definidas. Es decir, la partición correcta derivada del etiquetado puede no reflejarse en las características almacenadas en la base de datos.

Un ejemplo que ilustra este problema se puede encontrar en la base de datos Iris. Ésta es una base de datos muy utilizada en clustering y que puede encontrarse en el repositorio de datos de UCI (Asuncion y Newman, 2007). Iris es una base de datos de 150 flores de tres tipos representadas por cuatro características. A pesar de que el etiquetado establece 3 clusters (una por cada tipo de flor) varios autores argumentan que en realidad dos de ellos son no separables y que por lo tanto 2 es el número de clusters correcto (Bezdek y Pal, 1998; Kim y Ramakrishna, 2005; Wu et al., 2009). Desde nuestro punto de vista el objetivo de un algoritmo de clustering debe ser encontrar grupos naturales en los datos proporcionados y un CVI debe premiar las particiones de este tipo.

Esta reflexión nos lleva a los primeros pasos de un proceso de clasificación, en concreto a la recolección y preproceso de datos, ya que es aquí donde se definen los datos proporcionados a los pasos siguientes. De todos modos, creemos que una evaluación como la que se propone en este capítulo no debe entrar en este punto ya que añadiría una variable más a la comparación con lo que se dificultaría la interpretación de los resultados. Además, generaría mayor variabilidad en los diversos trabajos dificultando aún más su comparación. Lo ideal sería tener un conjunto de bases de datos reales preprocesadas de manera que los datos formen grupos naturales que coinciden con el etiquetado.

Por otra parte, éste no es un problema exclusivo de las bases de datos reales ya que también puede afectar a las bases de datos sintéticas. En casos donde los clusters tienen un alto grado de solapamiento la definición de partición correcta pasa a tener un alto grado de subjetividad. En este tipo de contextos, aunque dos clusters hayan sido creados siguiendo procesos distintos, si su solapamiento es muy alto consideramos que es discutible si deberían ser etiquetados con la misma etiqueta o no. En la Sección 2.2.5 se ha discutido más detalladamente acerca de este tema.

2. *Obtener un conjunto de particiones para cada una de las bases de datos seleccionadas en el paso anterior.*

   La aproximación habitual en este punto consiste en ejecutar varias veces un algoritmo de clustering con distintos valores para alguno de los parámetros. Lo más común es ejecutar el algoritmo K-means (o alguna de sus variantes) con distintos valores de K. Otra opción típica es la de ejecutar un algoritmo jerárquico y extraer varias de sus particiones. Sin embargo, con esta aproximación no es posible el control de las particiones obtenidas. Por lo tanto, existe la necesidad de un método para averiguar cuál es la partición que más se parece a la correcta.

   No obstante, existe otra posibilidad que carece de este inconveniente: realizar modificaciones controladas sobre la partición correcta. El conjunto S estaría compuesto por la partición correcta y varias de las modificaciones de la partición correcta. De este modo, es posible cuantificar de forma razonable el nivel de similitud entre la partición correcta y el resto de las particiones en S. El inconveniente es que si las particiones diseñadas no tienen parecido con las particiones que generan los algoritmos de clustering, los CVI se evalúan sobre un entorno poco realista. Este hecho nos puede llevar a que las conclusiones obtenidas no sean extrapolables a la realidad. De algún modo es un inconveniente similar al mencionado para las bases de datos sintéticas.

   Tal y como hemos mencionado anteriormente, en el caso de la metodología propuesta no es un requisito necesario el que las particiones en S tengan distinto número de clusters. Por otra parte, aunque no es esperable, podría ocurrir que varias particiones tuvieran el mismo nivel de bondad y fueran además las mejores. En ese caso parece lo más lógico considerar a todas ellas como partición objetivo y valorar como acierto la selección de cualquiera de ellas.

3. *Evaluar las particiones de salida con todos los CVI a evaluar.*

   En este punto la única discusión puede proceder de los CVI a evaluar. Lo ideal sería poder evaluar la mayoría de los CVI propuestos en la literatura científica relacionada. En caso de tener que limitar el número de los CVI evaluados sería interesante conocer la relación entre ellos para descartar tan sólo índices que sean similares a otros. De este modo se puede asegurar que se cubra un amplio abanico aunque sería necesario un extenso análisis para relacionar los distintos CVI existentes.

Un problema en la selección de los CVI surge con los índices que necesitan la definición de algún parámetro, siempre que no exista método fiable para encontrar un valor adecuado. Otros índices problemáticos son aquéllos cuyos resultados son subjetivos. En este grupo entrarían aquéllos que para especificar la mejor partición requieren representar los valores gráficamente y buscar un cambio acusado de pendiente que no siempre suele ser claro.

4. *Contar el número de aciertos para cada CVI.*

   El mayor problema en este punto consiste en decidir la bondad de cada partición de forma objetiva. Si el modo de obtener el conjunto de particiones S ha sido la de modificar "manualmente" la partición correcta (paso 2), entonces es sencillo cuantificar la similitud de la partición correcta y cada una de las particiones de S.

   Sin embargo, si la opción utilizada ha sido la habitual de ejecutar un algoritmo de clustering, el proceso no es tan sencillo. En este caso la solución pasa por comparar la partición correcta con las distintas particiones en S. Para ello existen multitud de medidas de similitud, pero al igual que ocurre con los algoritmos de aprendizaje o los índices de validación de clusters, ninguno de ellos es mejor en todos los contextos. Por tanto, lo ideal sería utilizar la medida de similitud que mejor se adecúe a cada contexto particular.

   Desafortunadamente, no se conocen criterios claros sobre la medida de similitud adecuada para cada caso. En la literatura científica podemos encontrar comparaciones entre ellas con el objeto de agrupar las que muestran un comportamiento similar o equivalente. También existe alguna comparativa con el objetivo de medir la bondad de muchas de ellas (Pfitzner et al., 2009). Sin embargo, no parece haber una solución definitiva ya que, como argumenta Meilă (2005), no hay manera de definir una medida de similitud que sea perfecta.

5. *Análisis del resultado.*

   El análisis de resultados más habitual se reduce al recuento de aciertos de cada índice. En algunos casos se analizan también las características de los fallos, como por ejemplo la diferencia en el número de clusters entre la partición correcta y la partición predicha.

   Es posible realizar un análisis más completo y analizar los valores asignados a cada partición por cada CVI. Lo ideal sería que estos valores

tuvieran cierta correlación con los niveles de similitud de las particiones, es decir, a mayor nivel de similitud mejor valor del CVI. Suponiendo que el nivel de similitud de cada partición en S es conocido, el problema se resolvería mediante alguna técnica de correlación de secuencias numéricas como el coeficiente de correlación de Pearson.

De todas maneras, dado que habitualmente los CVI se utilizan para seleccionar únicamente la mejor partición de un conjunto, el recuento de aciertos es un método aceptable. No obstante, un estudio completo debería ser capaz de desglosar los aciertos y errores según los distintos contextos simulados en la experimentación.

5.4.2. Discusión

A continuación se realizan una serie de propuestas o recomendaciones en base a las reflexiones realizadas en el apartado anterior.

1. *Seleccionar una serie de bases de datos cuya partición correcta sea conocida.*

   Para el primer paso de la metodología parece interesante la opción habitual de combinar bases de datos sintéticas y reales. Sería interesante definir una serie de características a estudiar (número de clusters, número de dimensiones...) y combinarlas para estudiar el comportamiento de cada CVI en distintos contextos controlados. Aunque la combinación de todas las posibles variaciones lleva a un número de bases de datos impracticable, se podría definir una base de datos de referencia y crear diferentes variaciones de ésta.

   En el caso de querer evaluar varios CVI para utilizar el mejor de ellos en un contexto conocido, lo ideal sería poder crear bases de datos con las características particulares de dicho entorno. Desafortunadamente, no es probable que el contexto sea conocido ya que en la práctica habitual la necesidad de recurrir al clustering parte precisamente del desconocimiento de un entorno que nos interesa conocer.

   El problema del etiquetado subjetivo en casos de solapamiento es difícil de resolver. La solución de utilizar bases de datos sin clusters solapados no parece válida ya que es interesante conocer el comportamiento de los algoritmos de clustering y los CVI en entornos con distinto nivel de solapamiento. Una posible solución intermedia podría ser la de aplicar las técnicas de evaluación a los datos que claramente pertenecen a un cluster e ignorar los ambiguos.

En cuanto a las bases de datos reales es fundamental la creación de un repositorio de datos con bases de datos preprocesadas y etiquetadas de forma que la partición correcta represente los grupos naturales en los datos. Probablemente esto implique definir una serie de bases de datos exclusivas para la clasificación no supervisada y desterrar la costumbre de utilizar para la validación de clustering bases de datos típicas de clasificación supervisada.

2. *Obtener un conjunto de particiones para cada una de las bases de datos seleccionadas en el paso anterior.*

   Aunque la opción correspondiente a las variaciones controladas de la partición correcta permita medir la similitud entre la partición correcta y las particiones creadas con un nivel de precisión muy alto, parece que esta opción puede provocar cierta desconfianza. El hecho de que las particiones creadas no se basen en algoritmos de clustering puede generar dudas acerca de los resultados.

   Por consiguiente, nuestra recomendación es utilizar algoritmos de clustering para crear las particiones del conjunto a evaluar. Sin embargo, a diferencia de lo habitual, consideramos que es preferible utilizar no uno sino varios algoritmos, a poder ser de distinto tipo (jerárquico, basados en grafos, basados en densidad, de clustering de subespacios, basados en la teoría de la información...).

3. *Evaluar las particiones de salida con todos los CVI a comparar.*

   Para dar solidez a una evaluación es recomendable incluir la mayor cantidad posible de índices de validación. En la mayoría de los casos no existe una solución definitiva para los índices que requieren la definición de algún parámetro o aquéllos cuyo resultado puede depender del criterio del experimentador. Por lo tanto estos índices deberían ser los primeros candidatos a descartar salvo que se establezca un criterio claro que evite la toma de decisiones arbitrarias.

   Los trabajos de evaluación previos pueden servir como guía a la hora de seleccionar los CVI. Es interesante evaluar aquéllos que han obtenido buenos resultados en trabajos previos, pero no conviene olvidarse del resto y también deberían ser seleccionados, aunque sea en menor medida.

4. *Contar el número de aciertos para cada CVI.*

   El mayor problema de este punto radica en la forma de asignar un nivel de bondad objetivo a las particiones de S, utilizando como referencia

la partición correcta. Lo más recomendable es utilizar medidas de similitud entre particiones, pero como se ha mencionado anteriormente la selección de una única medida no es sencilla.

Nuestra recomendación es seleccionar varias de ellas tratando de abarcar el mayor número de tipos o familias de medidas de similitud (Albatineh et al., 2006; Pfitzner et al., 2009). La combinación de los resultados puede realizarse de diversas formas (medias, rankings...) aunque sería recomendable seleccionar una que sirviera de práctica común.

5. *Análisis del resultado.*

   Dado que la mayoría de las veces el objetivo de un CVI es encontrar la mejor partición en un conjunto, parece razonable analizar los resultados en base al recuento de aciertos de cada CVI. De todos modos, un análisis de los valores asignados a cada partición por cada CVI puede ayudar a profundizar en el conocimiento de sus virtudes y defectos. En este sentido, parece que lo ideal sería combinar ambos resultados y utilizar el recuento de aciertos como un indicador de calidad y el análisis completo como estudio del comportamiento del CVI.

   En cualquier caso, parece que lo realmente interesante sería desglosar los resultados en base a los distintos contextos representados en la experimentación y tratar de encontrar asociaciones entre los CVI y las distintas características de bases de datos y algoritmos utilizados.

Antes de finalizar esta sección es fundamental recordar que las reflexiones y recomendaciones previamente realizadas se basan en nuestra experiencia y tienen como objetivo crear un debate acerca de las características más importantes de una evaluación de índices de validación de clusters. El objetivo final es establecer una serie de recomendaciones que sirvan como estándar y aporten robustez a la validación de clustering. Por supuesto, la profundización en el conocimiento teórico de los distintos aspectos de una evaluación de este tipo ayudaría enormemente al establecimiento de criterios para justificar y argumentar dichas recomendaciones.

5.5. Experimentación

La aportación fundamental de este capítulo es la nueva definición que se le da al concepto de "acierto" en una evaluación de índices de validación de clusters. Por lo tanto, para comparar las diferencias provocadas por el

concepto tradicional y el propuesto se ha realizado una experimentación en dos partes.

En la primera parte se han evaluado varios CVI siguiendo la metodología tradicional. Posteriormente, se han evaluado los mismos CVI siguiendo la nueva metodología propuesta. Los pasos seguidos en ambas partes de la experimentación han sido idénticos, salvo en lo que respecta a la definición de acierto de un CVI. En el segundo caso se ha utilizado un índice de similitud para decidir cuál es la partición más parecida a la correcta.

Dadas las características de la aportación realizada en este capítulo su validación no resulta sencilla. La experimentación realizada genera dos evaluaciones, pero dado que no se conoce la evaluación óptima, no es posible establecer cuál de las dos evaluaciones es mejor.

Por consiguiente, el objetivo principal de la experimentación es demostrar que las conclusiones obtenidas a partir de la metodología propuesta difieren de las obtenidas a partir de la metodología tradicional. Esto sirve para confirmar que el incumplimiento del supuesto en el que se basa la metodología tradicional afecta a los resultados de una evaluación.

No obstante, la experimentación tiene otros dos objetivos. La segunda parte de la experimentación favorece a aquellos CVI que evalúan positivamente las particiones que se parecen a la partición correcta; en otras palabras, las que mejor particionan los datos en base a los grupos naturales. En cambio, la primera parte favorece a aquellos índices que evalúan positivamente las particiones con el mismo número de clusters que la partición correcta, sin tener en cuenta el modo de particionar los datos.

Dado que el objetivo principal de los CVI es evaluar positivamente aquellas particiones que más se parezcan a la partición correcta, es de suponer que la segunda parte de la experimentación obtenga un recuento de aciertos superior al primero. El segundo de los objetivos de la experimentación será comprobar este hecho y confirmar que la metodología propuesta se ajusta mejor a la realidad que la tradicional.

Para finalizar, el último objetivo será la confirmación de la importancia del índice de similitud seleccionado en el paso 4 de la nueva metodología. Se ha replicado la segunda parte de la experimentación utilizando varios índices de similitud y se ha tratado de contrastar cualitativamente los resultados.

Es importante observar que el objetivo real de esta experimentación no es la de evaluar un conjunto de índices de validación, sino comparar dos metodologías diseñadas para realizar dicha evaluación. Por lo tanto, en este caso las observaciones y recomendaciones de la Sección 5.4 no son aplicables. Al contrario, hemos diseñado una experimentación sencilla para demostrar que la metodología tradicional y la propuesta difieren incluso en contextos

con datos sencillos de agrupar y algoritmos de clustering habituales.

5.5.1. Metodología experimental

A continuación se describe la metodología experimental tomando como referencia las metodologías de evaluación de índices de validación de clusters descritas en este capítulo:

1. Se han utilizado 7 bases de datos sintéticas y 3 reales. Las bases de datos sintéticas han sido creadas en un espacio bidimensional y se ha evitado el solapamiento entre clusters para evitar subjetividades a la hora de definir la partición correcta. La primera base de datos, llamada *2x2*, se ha mostrado ya en la Figura 5.3 de la página 104. Está compuesta por 4 clusters bien separados de 50 objetos cada uno. *Ruido* es una variante de la anterior, donde se han añadido 25 puntos de ruido aleatoriamente. La base de datos se ha mostrado ya en la Figura 5.1 de la página 102. *Densidad* y *Tamaño* son también variaciones de *2x2*. La primera de ellas reduce el espacio en el que se reparten los objetos de 2 de los clusters, de forma que se obtienen clusters de distinta densidad. La segunda de ellas no modifica el espacio en el que se reparten, pero modifica el número de casos de dos de los clusters a 100 y de los otros dos a 25. Ambas se muestran en la Figura 5.4.

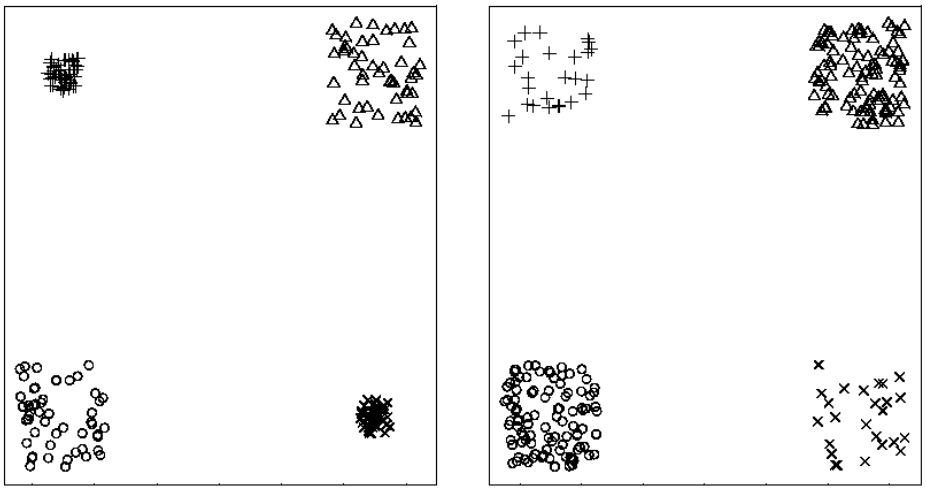

Figura 5.4: Bases de datos *Densidad* y *Tamaño*.

La base de datos *3x3* sigue la misma estructura que *2x2*, pero en este

caso hay 9 clusters bien definidos de 25 objetos cada uno distribuidos en una malla de 3 filas y 3 columnas. La base de datos *Arcos* está compuesta por 4 clusters en forma de arco dispuestos como muestra la Figura 5.5. Cada cluster se compone de 26 objetos. La última base de datos sintética es *T&U* y ya se ha mostrado en la Figura 5.2 de la página 103. Está compuesta por dos clusters, uno en forma de T y el otro en forma de U. El tronco del cluster en forma de T está dentro de la parte cóncava del cluster en forma de U.

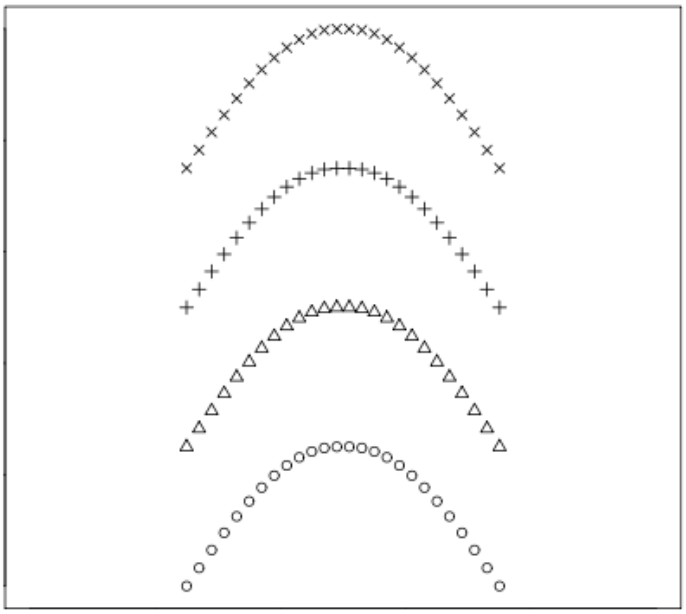

Figura 5.5: Base de datos *Arcos*.

Las 3 bases de datos reales han sido obtenidas del repositorio UCI de la Universidad de California, Irvine (Asuncion y Newman, 2007). Las 3 (*Iris*, *Glass* y *Ecoli*) han sido descritas en los Capítulos 3 y 4.

2. Para obtener un conjunto de particiones de cada base de datos se ha utilizado el algoritmo K-means y se ha variado el parámetro K desde 2 hasta $\sqrt{N}$, donde N es el número de casos de la base de datos. Para cada base de datos se han generado 10 conjuntos de particiones utilizando para ello distintas inicializaciones aleatorias del algoritmo K-means.

3. Los CVI evaluados han sido descritos en el Capítulo 2 y son los si-

guientes: Calinski-Harabasz, C-Index, Gamma, G(+), Davies-Bouldin y McClain-Rao. También hemos añadido el índice COP propuesto y descrito en el Capítulo 4.

4. En la primera parte de la experimentación se ha considerado como acierto la definición de la metodología tradicional. Para la segunda parte se ha utilizado la nueva definición propuesta. La similitud se ha medido con la medida de similitud Variation of Information. Para medir la variabilidad añadida por la elección de la medida de similitud se ha replicado la experimentación 4 veces utilizando las medidas de similitud Rand, Adjusted Rand, Jaccard y Fowlkes-Mallows. Todas estas medidas han sido descritas en la Sección 2.3.2.

5. Los resultados se muestran como un recuento de aciertos desglosados por base de datos.

5.5.2. Resultados

En este apartado se muestran los resultados correspondientes a la experimentación realizada. En primer lugar, en la Tabla 5.1, se muestran los resultados correspondientes a la metodología tradicional. Dado que para cada base de datos y valor de K se ha ejecutado el algoritmo K-means 10 veces, el valor máximo de cada celda será de 10.

	CH	C-I	Gam.	G(+)	DB	McC-R	Dunn	COP	Tot.
2x2	8	8	8	8	8	0	8	8	56
Ruido	5	5	5	5	6	0	5	5	36
Dens.	2	5	5	5	1	0	5	5	28
Tam.	9	9	9	9	0	0	9	9	54
3x3	2	2	2	2	0	0	2	3	13
Arcos	0	0	0	0	0	0	0	0	0
T&U	7	0	0	0	0	0	0	0	7
Iris	6	0	0	0	0	0	0	0	6
Glass	0	0	0	0	0	0	0	0	0
Ecoli	0	0	0	5	0	0	0	0	5
Total	39	29	29	34	15	0	29	30	205

Tabla 5.1: Número de aciertos de cada CVI siguiendo el criterio de la metodología tradicional.

Estos resultados muestran un resultado idéntico para C-Index, Gamma

y Dunn. Un análisis más detallado ha mostrado que el comportamiento de estos tres CVI ha coincidido hasta el punto de que sus aciertos han correspondido a las mismas ejecuciones del algoritmo de clustering. Además, los índices COP y G(+) muestran un comportamiento muy similar a los 3 anteriores y tan sólo difieren en una base de datos. Calinski-Harabasz también podría formar parte de este grupo aunque sus diferencias son algo mayores. Finalmente, los resultados de Davies-Bouldin son claramente inferiores al resto y McClain-Rao no consigue ningún acierto.

Por lo tanto, según la metodología habitual, 6 de los 8 CVI evaluados obtienen resultados similares. Calinski-Harabasz destaca porque es el único capaz de conseguir algún acierto en las bases de datos *T&U* e *Iris*. Este hecho hace que globalmente sea el índice con más aciertos a pesar de que tenga ciertos problemas con la base de datos *Densidad*. La segunda posición en el ranking que ocupa G(+) también se basa en ser el único en conseguir aciertos en una de las bases de datos reales, en este caso *Ecoli*.

Respecto a Davies-Bouldin, este índice parece mantener un comportamiento aceptable tan sólo en la base de datos más sencilla, aunque parece que el ruido no le afecta en este caso. Pequeñas variaciones en el tamaño o densidad de los clusters tienen efectos catastróficos. El mal comportamiento atribuido a McClain-Rao (Milligan y Cooper, 1985) se confirma en esta experimentación.

Desde el punto de vista de las bases de datos, la más sencilla, *2x2*, es la que más aciertos ha proporcionado aunque parece que la variación del número de objetos en los clusters no influye negativamente a los CVI. Por contra, la variación de densidad o el ruido influyen moderadamente. Un aumento de los clusters también influye negativamente, más de lo que cabría esperar en un principio. Finalmente, es indudable que las bases de datos más sencillas no marcan muchas diferencias entre los CVI mientras que aquéllas más complejas (sintéticas con clusters no convexos o reales) generan mayores diferencias y peores resultados.

En general, dado que hay 10 ejecuciones por cada una de las 10 bases de datos, el límite superior de aciertos por cada CVI es de 100. Eso nos lleva al hecho de que según la metodología tradicional ningún CVI ha alcanzado un nivel de 40 % de aciertos en esta experimentación. Si nos limitamos a las bases de datos más sencillas, las 5 primeras de la tabla, el nivel de acierto máximo alcanza el 60 %.

A continuación se describen los resultados de la segunda parte de la experimentación, es decir, los correspondientes a la metodología de evaluación propuesta. Antes de ello, se ha cuantificado la variación que puede haber entre ambas metodologías. Dado que cuando el supuesto de corrección del

algoritmo se cumple ambas metodologías son idénticas, hemos contabilizado el número de veces que se da esta condición. Es decir, hemos contado el número de veces que para cada base de datos $P^{nc} = P^{\text{sim}}$. La medida de similitud utilizada es Variation of Information de Meilă (2005). Los resultados se muestran en la Tabla 5.2.

2x2	*Ruido*	*Dens.*	*Tam.*	*3x3*	*Arcos*	*T&U*	*Iris*	*Glass*	*Ecoli*
8	6	5	9	3	1	1	0	0	0

Tabla 5.2: Número de ejecuciones del algoritmo K-means en las cuales el supuesto de corrección del algoritmo se cumple para la medida de similitud VI.

Estos resultados nos aseguran que los resultados para bases de datos como *2x2* o *Tamaño* serán muy parecidos en ambas metodologías de evaluación. Sin embargo, los resultados para el resto pueden variar considerablemente. Por otra parte, los resultados vuelven a confirmar el bajo nivel de cumplimiento del supuesto de corrección del algoritmo ya que en esta experimentación se cumple tan sólo un tercio de las veces. Cabe destacar que el supuesto nunca se cumple en las bases de datos reales analizadas.

En la Tabla 5.3 se muestran los resultados según la nueva metodología propuesta y la medida de similitud VI. Las diferencias respecto a los resultados de la metodología tradicional son evidentes.

	CH	C-I	Gam.	G(+)	DB	McC-R	Dunn	COP	Tot.
2x2	10	10	10	9	8	0	8	10	65
Ruido	9	9	10	8	5	0	5	9	55
Den.	4	8	10	8	3	0	5	9	47
Tam.	10	10	10	10	0	0	10	10	60
3x3	7	8	7	7	4	0	4	5	42
Arcos	8	0	0	0	0	0	1	2	11
T&U	1	0	0	0	0	0	0	0	1
Iris	4	9	10	0	10	0	10	10	53
Glass	5	7	7	0	5	0	7	6	37
Ecoli	0	5	1	0	0	0	8	4	18
Total	58	66	65	42	35	0	58	65	389

Tabla 5.3: Número de aciertos de cada CVI siguiendo el criterio de la nueva metodología propuesta y según la medida de similitud VI.

Según la metodología tradicional el comportamiento de los índices C-

Index, Gamma y Dunn era idéntico. Según la nueva metodología no. C-Index y Gamma siguen mostrando un comportamiento similar aunque ahora hay variaciones en 5 bases de datos. La mayor diferencia surge en la base de datos *Ecoli* donde C-Index muestra ser claramente superior. De todos modos, en el cómputo global ambos índices son muy similares. Estos dos CVI obtienen muy buenos resultados en las bases de datos más sencillas y en dos de las reales. En cambio fallan, como prácticamente todos los índices, en las sintéticas con clusters no convexos.

El comportamiento del índice Dunn es diferente. Sólo demuestra un alto nivel de acierto en dos de las bases de datos sintéticas y en las tres reales. De hecho, es indudablemente el mejor índice para las bases de datos reales en esta experimentación. Curiosamente, Dunn también obtenía los mejores resultados en las bases de datos reales en la experimentación del Capítulo 4.

La metodología tradicional situaba a COP y a G(+) como índices muy similares al grupo de tres mencionado en los párrafos previos (C-Index, Gamma y Dunn). La nueva metodología confirma que COP es similar a C-Index y Gamma. Aunque presenten ligeras diferencias el comportamiento global es similar. En cambio G(+) muestra una característica muy particular. Su comportamiento es bueno, y sólo ligeramente peor que el de C-Index, Gamma y COP para las bases de datos sintéticas con clusters convexos. Sin embargo, no obtiene ningún acierto en el resto de las bases de datos. Los resultados sugieren que G(+) sólo es adecuado para bases de datos sencillas.

Si nos centramos en el índice Calinski-Harabasz podemos ver que se diferencia de los mencionados anteriormente. Aunque según la metodología tradicional era el mejor CVI los resultados para la nueva metodología lo sitúan en cuarto lugar junto con Dunn. De todos modos, Calinski-Harabasz es más irregular. Por ejemplo, mientras que en cuanto a las bases de datos reales los resultados sólo superan a G(+) y McClain-Rao, los resultados correspondientes a las sintéticas están entre los mejores (excepto para *Densidad*). Destaca el resultado casi óptimo para la base de datos *Arcos*.

Por otra parte, el resultado de Calinski-Harabasz para *T&U* también destaca, pero esta vez debido a su acusado descenso. Este hecho demuestra que los 7 aciertos obtenidos según la metodología tradicional correspondían a un afortunado "error" en el que Calinski-Harabasz seleccionaba una partición con el número correcto de clusters a pesar de que hubiera mejores particiones en S. Hardy (1996) ya menciona esta posibilidad, tal y como se indica en la Sección 5.3.2. La nueva metodología "desenmascara" estos aciertos fortuitos.

El caso de G(+) y la base de datos *Ecoli* es otro acierto del mismo tipo: los 5 son fortuitos. Curiosamente, según la metodología tradicional Calinski-

Harabasz y G(+) destacan sobre el resto de los CVI gracias a estos aciertos. En el resto de la experimentación hay otros dos casos de este tipo, pero son de menor importancia.

Para finalizar, Davies-Bouldin y McClain-Rao vuelven a mostrar los peores resultados. Davies-Bouldin logra una mejoría apreciable sólo para las bases de datos *Iris* y *Glass*, mientras que McClain-Rao no consigue ningún acierto.

Si observamos los resultados respecto a cada base de datos el ordenamiento obtenido según el número de aciertos es similar al de la metodología tradicional. Es destacable el incremento de aciertos para todas las bases de datos excepto *T&U*. En este sentido el incremento de aciertos en las bases de datos *Iris* y *Glass* es lo más significativo, ya que les hace subir 3 posiciones en el ranking.

El análisis realizado demuestra que el objetivo principal de la experimentación se cumple. Es decir, los resultados obtenidos por la nueva metodología difieren de los obtenidos por la metodología tradicional tanto cuantitativamente como cualitativamente. Además, la mayoría de los índices muestran un patrón similar según la metodología tradicional mientras que según la nueva metodología los índices muestran un abanico más amplio de comportamientos. Los resultados sugieren que la nueva metodología es más adecuada para captar los pequeños detalles particulares de cada CVI en cada base de datos.

Por otra parte, el segundo objetivo de la experimentación también queda claramente confirmado. Es decir, el número de aciertos de la nueva metodología casi duplica a los de la metodología tradicional. Mientras que el mejor CVI no alcanza el 40 % de aciertos según la metodología tradicional, 5 de los CVI superan este porcentaje según la nueva metodología. Los mejores índices según la nueva metodología rondan el 65 % de aciertos. Además, el nivel de acierto máximo para las 5 bases de datos más sencillas pasa de 60 % a 95 %. Por lo tanto, los resultados confirman que el criterio utilizado para definir un acierto en la nueva metodología se adecúa mejor al comportamiento real de los CVI.

Finalmente, el tercer objetivo era cuantificar la importancia de la medida de similitud utilizada para comparar particiones en la nueva metodología. La comparación de los resultados obtenidos mediante las 5 medidas de similitud mencionadas previamente (VI, Rand, Adjusted Rand, Jaccard y Fowlkes-Mallows) muestra que la variabilidad no es excesiva. Sin embargo, existen algunos resultados que dependen de la medida utilizada. A continuación se describen brevemente los resultados obtenidos, mientras que las tablas con los resultados completos se muestran en el Apéndice A.

Los resultados para Adjusted Rand, Jaccard y Fowlkes-Mallows muestran unos resultados muy similares y coinciden a su vez con el resultado medio. Este resultado medio sitúa a Gamma y C-Index como los mejores índices (con una tasa de acierto de 60 %), ligeramente por delante de Calinski-Harabasz y COP (55 %). G(+) y Dunn muestran unos niveles de acierto algo inferiores (45 %), pero muy por encima de Davies-Bouldin (25 %) y McClain-Rao (2 %). La medida de similitud que más se desvía de esta clasificación es RAND que intercambia las posiciones de G(+) y COP. VI también muestra un resultado atípico ya que sitúa a Dunn entre los mejores. De todos modos, sea cual sea la medida utilizada, los resultados difieren sensiblemente de los obtenidos por la metodología tradicional con lo que todas ellas confirman el objetivo principal de la experimentación.

Por otro lado, es importante recalcar que la tasa de acierto media de la metodología propuesta está entre el 41 % y el 49 %, dependiendo de la medida de similitud utilizada. No obstante, para la metodología tradicional esta tasa de acierto es del 26 %. Por lo tanto, las 5 medidas utilizadas también confirman el segundo objetivo de la experimentación

5.6. Conclusiones

En este capítulo hemos argumentado que, aunque la mayoría de investigadores siguen un esquema similar a la hora de comparar índices de validación de clusters, no existe un método unificado. Además, hemos ilustrado que la metodología tradicional de evaluación se basa en un supuesto que no se cumple en numerosas ocasiones y, por consiguiente, los resultados son incorrectos en muchos casos. Como respuesta hemos definido un nuevo criterio que no se basa en dicho supuesto.

Además, hemos tratado de recoger las distintas opciones que podemos encontrar en la bibliografía dentro de la metodología tradicional de evaluación. Hemos realizado una reflexión acerca de las ventajas e inconvenientes de las distintas opciones y hemos tratado de realizar una serie de recomendaciones basadas en nuestra experiencia. De todos modos, creemos que para dar pasos hacia la unificación de criterios es necesario un trabajo teórico más profundo.

Finalmente hemos realizado una experimentación para comparar la metodología tradicional de evaluación con la nueva metodología propuesta. Los resultados demuestran que las conclusiones que se obtienen a partir de las diferentes metodologías son significativamente diferentes. Asimismo, el número de aciertos de los CVI es mayor según la nueva metodología con lo que los

resultados sugieren que ésta se adapta mejor al objetivo real de los CVI.

La experimentación realizada también confirma que la medida de similitud utilizada junto con la nueva metodología influye en los resultados finales. Sin embargo, la diferencia en los resultados es mucho menor que la diferencia respecto a los resultados correspondientes a la metodología tradicional. Es posible que en una experimentación más extensa la variabilidad sea menor y que la influencia de la medida utilizada se minimice. Mientras tanto parece que lo ideal es combinar los resultados de varias medidas.

Para este trabajo en particular la medida de similitud utilizada no es de gran importancia ya que las 5 medidas utilizadas confirman las mencionadas diferencias de la nueva metodología respecto a la tradicional.

El trabajo descrito en este capítulo ha sido aceptado para publicación con cambios menores en una revista internacional (Gurrutxaga et al., 2010b).

Parte III

Aplicaciones

Capítulo 6

MALBEC: Una aplicación de técnicas de clustering para agrupar código malicioso

6.1. Introducción

En este capítulo se presenta una aplicación de técnicas de clustering, y de algunas de las propuestas presentadas en los capítulos anteriores, al contexto de la seguridad informática. En concreto se presenta una plataforma denominada MALBEC (*MALware BEhaviour Clustering*) en la que se utilizará el conocimiento adquirido en las experimentaciones mostradas en capítulos anteriores con el fin de implementar una plataforma capaz de generar automáticamente familias de código malicioso en base a su comportamiento.

Muchos de los problemas de seguridad derivados del acceso a Internet están relacionados con programas y herramientas con código malicioso. Código malicioso, ampliamente referenciado como *malware*, es software diseñado para infiltrarse y/o dañar un sistema informático sin el consentimiento ni conocimiento del propietario. En la actualidad, dado el gran número de transacciones realizadas a través de las redes (compra-ventas, transferencias bancarias, trámites administrativos...) el daño que puede causar el *malware* va más allá de la simple eliminación de datos.

El *malware* es generalmente el origen de la mayor parte de tipos de ataques existentes actualmente como, por ejemplo, *spam*, *phishing*, denegación de servicio (*DoS*), etc. Cada día aparece nuevo código *malware* o variaciones del existente, y no sólo para plataformas tipo PC basadas en Windows, ya

que últimamente han aparecido virus y gusanos para MacOS y troyanos para plataformas móviles basadas en J2ME. Estas últimas han sido diseñadas para sustraer dinero de las cuentas de los usuarios de móviles.

La tendencia en la evolución del *malware* continúa y el número de nuevos programas maliciosos diseñados para producir daños financieros sigue incrementando. Esto hace que el *malware* suponga una gran pérdida económica que hay que evitar en la medida de lo posible.

El *malware* moderno es muy complejo; la aparición diaria de muchas variantes del mismo virus hace que la detección sea muy difícil y el análisis tedioso. Comprender cómo los atacantes o *hackers* utilizan funcionalidades *malware* como, por ejemplo, puertas traseras, *keyloggers* o *password stealers*, se está convirtiendo en una tarea cada vez más necesaria y compleja.

Por todo esto, la clasificación manual del nuevo *malware* que implica memorización, búsqueda de librerías y sus características o recolección de muestras, hacen que sea una tarea, por un lado, subjetiva, y por otro, prácticamente imposible de realizar desde el punto de vista de la inversión de tiempo que supone (Lee y Mody, 2006). Como consecuencia, se hace necesario disponer de herramientas automáticas y robustas que permitan comprender el comportamiento del *malware*. Estas herramientas deben permitir representar y almacenar la información de una manera adecuada y ser capaces de aprender de esos datos.

Debido a que históricamente el tipo de *malware* más conocido ha sido el virus, hoy en día los programas maliciosos, sean del tipo que sean (troyanos, gusanos, virus...), son habitualmente llamados virus. De la misma manera, los sistemas de detección de *malware*, extensiones de los antiguos antivirus, siguen manteniendo aún hoy el nombre de antivirus. En este capítulo seguiremos esta costumbre y utilizaremos el término virus para refererirnos a todo tipo de *malware*.

La mayoría de los antivirus utilizan información extraída de los ejecutables de los virus para detectar los ficheros infectados. Esta información normalmente consiste en secuencias de bytes extraídas de cada virus y que lo caracterizan de forma particular. Esta información se conoce normalmente como firma. Estas firmas son cada vez menos eficientes para la detección de *malware* debido a la proliferación de técnicas como el polimorfismo o metamorfismo y técnicas de ofuscación de código y datos (Bailey et al., 2007). Además, muchos virus copian trozos de otros para utilizar alguna de sus funcionalidades en particular, pudiendo copiar la firma característica del virus. En este caso el antivirus confunde los dos virus a pesar de que su comportamiento pueda ser completamente distinto.

Estos problemas plantean la necesidad de utilizar datos extraídos de la

ejecución del *malware*, principalmente de la secuencia de llamadas al sistema que realiza el virus (escritura en ficheros, creación de procesos, etc.). Este tipo de información se caracteriza por ser más útil a la hora de detectar y prevenir ataques de *malware* ya que, por un lado, permite estimar de una manera más certera el riesgo potencial que supone para el sistema el virus detectado y actuar en consecuencia; y por otro lado, permite la detección y clasificación de nuevo *malware* en base a un comportamiento similar a alguno de los previamente analizados.

La idea de utilizar información extraída del comportamiento dinámico ha sido utilizada en otras áreas de la seguridad informática. Trabajos como los de Brugger (2004); Christodorescu et al. (2007); Gao et al. (2005); Yeung y Yuxin (2003) han aplicada esta idea y también se puede ver en sistemas comerciales como *Norman Solutions* y *CWSandbox*. Sin embargo, pocos trabajos se han basado en el comportamiento dinámico de *malware*. En esta línea se han encontrado trabajos como los de Bailey et al. (2007) en el que agrupan *malware* y evalúan los resultados en base a la salida de 5 antivirus. A su vez, Lee y Mody (2006) agrupan *malware* utilizando la distancia de edición y el algoritmo de clustering *K-medoids*. Sin embargo, no presentan una comparación de diferentes estrategias.

En este capítulo, se plantea la aplicación de técnicas de clustering utilizando información extraída del comportamiento dinámico del *malware* con el objetivo de superar las limitaciones que tienen las herramientas automáticas de clasificación de *malware* existentes. Servirá para poder analizar el nuevo *malware* que diariamente aparece y asignarlo a grupos o clusters existentes con comportamientos similares, lo que permitirá detectar de una manera eficiente variaciones de virus anteriores o nuevos virus que se asemejen, en comportamiento, a una familia preexistente. Dado que se estima que la mayoría de los nuevos virus son simples variantes de otros, este sistema permitirá una drástica reducción en el tiempo requerido para determinar el comportamiento de cada uno de ellos. Asimismo, la estructura generada por el proceso de clustering podría ser utilizada para definir una nomenclatura estándar basada en el comportamiento real del *malware*, y evitar la gran cantidad de denominaciones que se pueden encontrar para un mismo virus.

Para poder realizar la agrupación de *malware* necesitaremos determinar los datos que se deben recolectar y establecer una representación para los mismos, perdiendo la menor cantidad de información posible. También habrá que determinar qué función de distancia o métrica vamos a utilizar para comparar patrones. Finalmente, tendremos que decidir qué algoritmo de clustering se va a utilizar sobre los datos recolectados. En todo momento habrá que tener en cuenta el coste computacional ya que puede condicionar

la representación, la métrica o el algoritmo utilizado. En este capítulo se presenta el estudio realizado en este sentido y se muestra una breve descripción de la plataforma MALBEC que sirve de interfaz con el usuario y que permite trabajar de una manera cómoda con las diferentes familias de *malware* generadas por el algoritmo de clustering.

6.2. Adquisición y representación de datos

Como se ha comentado anteriormente para poder aplicar técnicas de clustering hay que determinar la información que se va a extraer del comportamiento dinámico de los programas y determinar cómo se va a representar dicha información. En este caso, la información utilizada ha sido proporcionada por una de las principales empresas en seguridad informática, S21sec. Para la generación de los datos se han utilizado máquinas reales en un entorno controlado obteniendo resultados más fiables que a través de la práctica habitual de utilizar máquinas virtuales. El código *malware* ha sido ejecutado en máquinas monitorizadas, en concreto, en Pentium IV a 3,20 GHz con 1 GB de memoria RAM y Windows XP SP2 como sistema operativo.

Cada virus ha sido ejecutado durante un minuto en el entorno controlado y se han registrado todas las llamadas al sistema efectuadas durante la ejecución. Posteriormente, los expertos en seguridad de la empresa S21sec han filtrado algunas de las llamadas que no aportan información y han clasificado el resto de las llamadas en 45 categorías que vamos a denominar eventos. Los eventos están agrupados en 6 grupos: Ficheros, Registro, Procesos, Drivers, Red y Otros. En la Tabla 6.1 se puede ver el listado completo de los eventos y sus correspondientes códigos.

La información almacenada para cada ejecución es una secuencia de códigos correspondientes a los eventos detectados durante el primer minuto de ejecución monitorizada del programa. Este tipo de representación, que vamos a denominar representación en secuencia, compacta bastante la información asociada al comportamiento dinámico de los programas y mantiene información sobre el orden y la frecuencia en la que se han ido produciendo los eventos. Sin embargo, este tipo de representación tiene un inconveniente a la hora de aplicar técnicas de clustering, y es que la mayoría de las técnicas de aprendizaje automático están diseñadas para trabajar con datos en forma de vectores de tamaño fijo. Además, normalmente estos vectores serán más cortos que las secuencias y por tanto, el coste computacional de analizarlos será menor.

Código	Evento	Código	Evento
1	File_superseded	24	Proc_thread_create
2	File_opened	25	Proc_thread_open
3	File_created	26	Proc_thread_resume
4	File_created_open	27	Proc_thread_term
5	File_created_overwrite	28	Proc_create
6	File_created_superseded	29	Proc_create_ex
7	File_overwritted	30	Proc_open
8	File_dir_opened	31	Proc_terminate
9	File_dir_created	32	Drv_load
10	File_dir_created_open	33	Drv_unload
11	File_failed	34	Drv_set_sys_info
12	File_delete	35	Drv_ioctl_file
13	File_set_info	36	Other_write_virt_mem
14	Reg_query_key	37	Other_section_open
15	Reg_open_key	38	Other_mutant_create
16	Reg_close_key	39	Other_setwinhookex
17	Reg_create_key	40	Network_connect
18	Reg_enumerate_key	41	Network_send
19	Reg_delete_key	42	Network_listen
20	Reg_query_value	43	Network_accept
21	Reg_set_value	44	Network_disconnect
22	Reg_enumerate_value	45	Network_receive
23	Reg_delete_value		

Tabla 6.1: Listado de eventos y sus correspondientes códigos.

Por ello, también se ha experimentado con una versión de los datos representados de forma vectorial. En este caso, habrá que analizar hasta qué punto son determinantes las ventajas asociadas a la representación vectorial ya que el cambio de representación implicará una pérdida de información. La representación particular utilizada pierde el orden en que se han realizado los eventos ya que se basa en un recuento de eventos. En concreto cada secuencia se representa como un vector de 45 dimensiones en el que se contabiliza el número de veces que cada virus ha efectuado cada una de las llamadas al sistema.

Utilizaremos, por tanto, representaciones vectoriales y en secuencias, lo que condicionará el tipo de métricas o medidas de similitud que habrá que utilizar en el proceso de clustering. En el siguiente apartado se presentan las

alternativas que se han evaluado en este trabajo.

6.3. Aplicación del clustering

Como se ha mostrado en el apartado anterior, la información extraída de la ejecución de los programas se puede representar vectorialmente o mediante secuencias, y el tipo de representación que se utilice condiciona la distancia o métrica a utilizar en el proceso de clustering.

Para el caso de la representación vectorial se ha utilizado la distancia euclidiana y en el ámbito de la representación en secuencias se ha trabajado con diversas distancias: la distancia de edición, dos distancias que se basan en analizar la longitud de las subsecuencias comunes más largas y una distancia basada en técnicas de compresión. En los siguientes subapartados se comentan brevemente las distancias utilizadas y las normalizaciones aplicadas.

6.3.1. Medidas de similitud para comparar secuencias

Para el caso de la representación vectorial de las secuencias (Count) se ha utilizado la distancia euclidiana como métrica para comparar los vectores de frecuencias de los eventos de cada una de las secuencias dado que es una de las métricas más utilizadas en la bibliografía.

Para el caso de la representación en secuencias se han evaluado cuatro métricas diferentes. La primera de las distancias utilizadas es la conocida distancia de edición o *Edit Distance* (ED). La distancia de edición entre dos secuencias es el mínimo número de operaciones de edición sobre elementos individuales necesarios para transformar una secuencia en la otra (Gusfield, 1997). Las operaciones de edición pueden ser inserción, borrado y substitución (o reemplazo) de un elemento.

Las distancias *Longest Common Substring* (LCSst) y *Longest Common Subsequence* (LCSse) se calculan en base a la longitud del *substring* o *subsequence* común más largo a las dos secuencias que se comparan. Para el caso del *substring* los caracteres que lo componen deben de ser contiguos en ambas secuencias, sin embargo, para el caso de la *subsequence* no es necesario que los caracteres estén contiguos en las secuencias que se comparan, aunque, eso sí, deben de aparecer en el mismo orden. Como se puede comprobar, estas dos métricas son medidas de similitud y no distancias, así que habrá que transformarlas para poder aplicarlas en los algoritmos de clustering. Más adelante se comenta cómo se ha realizado esta transformación.

Por último, también se ha utilizado una distancia basada en técnicas de compresión denominada *Normalized Compression Distance* (NCD) (Li et al., 2004; Wehner, 2007). Esta distancia se basa en la complejidad de Kolmogorov, pero dadas las dificultades para calcularla se aproxima mediante algoritmos de compresión. La distancia NCD, distancia de compresión normalizada, se define de la siguiente manera:

$$\mathrm{d_{NCD}}(x, y) = \frac{\mathrm{C}(x, y) - \mathrm{mín}(\mathrm{C}(x), \mathrm{C}(y))}{\mathrm{máx}(\mathrm{C}(x), \mathrm{C}(y))}$$

donde $\mathrm{C}(x)$ es la longitud del resultado obtenido tras comprimir x, y $\mathrm{C}(x, y)$ es la longitud del resultado obtenido tras la compresión de la concatenación de los strings x e y.

6.3.2. Normalizaciones y casos de referencia

Algunas de las distancias planteadas, como ED o Count, son muy dependientes de la longitud de las secuencias que se comparan. Para atenuar este efecto se ha aplicado un proceso de normalización. Para la representación vectorial Count se han aplicado dos tipos de normalizaciones. Una se basa en dividir las frecuencias absolutas por la longitud de la secuencia, obteniendo así la frecuencia relativa. A la representación obtenida con esta normalización la hemos denominado CountL. El otro tipo de normalización realizado se basa en tipificar todas las variables de forma que su media sea 0 y la desviación típica 1. A esta representación la hemos denominado CountM.

Para el caso de las distancias basadas en secuencias se han implementado dos variaciones para cada una de las tres primeras distancias (la distancia NCD está normalizada en su propia definición). Las variaciones se basan en realizar una normalización dividiendo la distancia obtenida por la longitud de la secuencia más corta que se compara (EDS, LCSstS y LCSseS), o por la de la secuencia más larga (EDL, LCSstL y LCSseL). Además, como ya se ha comentado previamente, para el caso de las métricas de tipo LCS se ha tenido que realizar una transformación para convertirlas en distancias. Dicha transformación se ha realizado tras la normalización, de tal manera que si la medida de similitud proporciona un valor s, el valor de la distancia será $d = 1 - s$.

Además de las opciones comentadas en los párrafos anteriores (distancias, normalizaciones, etc.) se han utilizado dos distancias simples para poder descartar ciertas sospechas o para ser utilizadas como referencia base:

- *Length*: cada secuencia se ha representado mediante su longitud y se ha usado la distancia euclidiana como métrica. Esta opción servirá para

comparar los resultados con las opciones no normalizadas de ED y Count y comprobar si el comportamiento es similar o no. De este modo comprobaremos si efectivamente la longitud de las secuencias comparadas influye excesivamente en las distancias no normalizadas.

- *Random*: Para cada par de secuencias comparadas se ha generado una distancia aleatoria. Esta opción servirá para comprobar si el resto de las distancias aportan alguna información o no, es decir, servirá para contrastar la estructura encontrada en los datos reales frente a un caso en el que no existe ningún tipo de estructura.

6.4. Experimentación

A continuación se describe la experimentación realizada. Por un lado se evalúan las distintas distancias descritas y por otro se analiza qué algoritmo de clustering se adecúa mejor a los datos.

6.4.1. Metodología experimental

La empresa S21sec proporcionó una base de datos correspondiente a 500 ejecuciones de código malicioso. Por tanto, los datos que se han utilizado para esta investigación han sido las 500 secuencias de eventos extraídas de las correspondientes ejecuciones. Como se ha comentado anteriormente, expertos de la empresa filtraron las llamadas al sistema dejando las más relevantes. Este proceso se ha realizado aplicando unos filtros que eliminan patrones de llamadas fijos asociados a la ejecución de rutinas del sistema operativo, etc. Dado que el número de llamadas al sistema relevantes durante el primer minuto es diferente para cada programa la longitud de las secuencias es variable. La longitud de las secuencias varía desde 1 hasta 12.944 eventos con una media de 1.008 y una desviación estándar de 2.173.

La base de datos se ha dividido aleatoriamente en 5 partes iguales de 100 secuencias cada una y la experimentación se ha realizado sobre cada una de las partes. Para cada una de las secuencias se conoce el tipo de virus, diagnosticado mediante el antivirus Kaspersky, información que no se utilizará en el proceso de aprendizaje.

A la hora de seleccionar el método de clustering a utilizar se ha tenido en cuenta que el *malware* está estructurado de forma jerárquica (familias, subfamilias, variantes, tipos...). Por ello, de los diferentes métodos de clustering que existen hemos utilizado los que de forma natural se adaptan a esta estructura jerárquica, es decir, los métodos de clustering jerárquicos.

Estos métodos nos van a permitir ir más allá de la generación de ciertas familias de *malware*, pudiendo representar una jerarquía completa. En este caso se han probado tres algoritmos de clustering jerárquico: *single-linkage*, *complete-linkage* y *average-linkage*.

La experimentación planteada pretende analizar, por un lado, en qué medida las distancias utilizadas son capaces de detectar las diferencias entre las diferentes familias de *malware*. A este respecto, se ha incluido una fase de aprendizaje para tratar de establecer las relaciones entre las diferentes distancias utilizadas. Por otro lado, también se pretende analizar en qué medida los diferentes algoritmos de clustering probados son capaces de detectar la estructura (familias) proporcionada por las distancias. No obstante, la validación del sistema no será sencilla ya que no se conoce la jerarquía ideal.

Para analizar el comportamiento de las 13 distancias hemos calculado la matriz de distancias correspondiente a cada una de ellas (Count, CountL, CountM, ED, EDS, EDL, LCSstS, LCSstL, LCSseS, LCSseL, Length, NCD y Random). Con el objeto de compararlas y analizar la intensidad y la dirección de la relaciones entre ellas hemos calculado la correlación entre las matrices de distancias. Para ello se ha utilizado el coeficiente de correlación de Pearson descrito en la Sección 2.3.1.

El segundo objetivo es evaluar en qué medida las tres opciones de clustering jerárquico analizadas son capaces de identificar la estructura representada por las distancias. El primer paso ha consistido en construir los dendrogramas correspondientes a cada una de las matrices de distancias aplicando los tres algoritmos. Para cuantificar en qué medida cada uno de los algoritmos es capaz de detectar la estructura subyacente en los datos se ha calculado el coeficiente de correlación entre la matriz de distancias y la matriz cofenética correspondiente a cada dendrograma, es decir el CPCC.

Un aspecto a tener en cuenta a la hora de evaluar el sistema es el coste computacional de las distancias. Ya que a diario aparece nuevo *malware* que habrá que ir incluyendo en familias, es necesario mantener el coste computacional del sistema en unos niveles razonables desde el punto de vista práctico.

A continuación se describen los resultados obtenidos. Aunque se han realizado múltiples ejecuciones sobre diferentes muestras con 100, 200, 300, 400 y 500 secuencias las conclusiones generales han sido similares en todas las ejecuciones. Por lo tanto, se van a mostrar los resultados de la primera ejecución realizada sobre una muestra de 100 secuencias. Cuando analicemos el coste computacional se mostrarán los tiempos de ejecución de muestras ampliadas incrementalmente desde 100 hasta 500.

6.4.2. Evaluación de las distancias

Se ha realizado una comparación por pares para determinar la similitud entre las estructuras reflejadas por cada distancia. La Tabla 6.2 muestra los valores de correlación entre todos los posibles pares de matrices de distancias (13×13). Los valores están multiplicados por 100 y sólo se muestra la matriz triangular inferior, ya que la matriz es simétrica.

	Count	CountL	CountM	ED	EDS	EDL	LCSstS	LCSstL	LCSseS	LCSseL	Length	NCD
CountL	47											
CountM	59	39										
ED	99	44	60									
EDS	40	29	20	41								
EDL	43	69	41	43	23							
LCSstS	18	2	30	17	-31	-6						
LCSstL	34	49	40	34	16	75	30					
LCSseS	21	34	41	17	-12	12	56	14				
LCSseL	44	67	40	43	23	98	-12	72	4			
Length	97	40	55	99	43	42	13	33	9	43		
NCD	50	57	57	52	29	75	7	76	2	75	50	
Random	0	2	-2	0	-1	1	0	1	-1	1	0	1

Tabla 6.2: Valores de correlación entre las 13 matrices de distancias evaluadas en la experimentación. Por claridad, los valores están multiplicados por 100.

El primer aspecto a comentar es que, como se esperaba, los valores de la matriz muestran que existe cierta similitud entre la estructura capturada por todas las distancias excepto para el caso Random. Los valores de la última fila de la matriz, donde se compara la opción Random con todas las demás distancias, son cercanos a 0, mientras que el resto de los valores de la matriz son sensiblemente mayores. Esto sugiere que las distancias planteadas son capaces de extraer cierta estructura de los datos.

El resto de valores podrían ser analizados uno a uno para decidir en qué medida las distancias están correladas entre sí. Las distancias con un valor de correlación alto serían distancias con un comportamiento similar. Los valores negativos que aparecen en la tabla indican que algunas distancias están inversamente correladas. Analizando la tabla se puede ver que, además del caso Random, los valores negativos están asociados a las distancias en las que la normalización se ha realizado respecto a la secuencia más corta

(sufijo S). Este hecho parece indicar que este tipo de normalización genera comportamientos inesperados.

Con el objeto de obtener resultados de un nivel semántico más elevado se ha construido un dendrograma con la matriz de correlación de la Tabla 6.2. Este proceso no se puede realizar directamente con los valores de la tabla porque los valores de correlación miden similitud en lugar de distancia (o disimilitud) que es lo que se precisa para construir el dendrograma. Por ello, se ha realizado la siguiente transformación a los datos de la matriz de correlación:

$$\mathrm{d}(i,j) = \begin{cases} 1 & \text{si} \quad \mathrm{corr}(i,j) < 0 \\ 1 - \mathrm{corr}(i,j) & \text{en caso contrario} \end{cases}$$

Este proceso se ha realizado con tres algoritmos (*single-linkage*, *complete-linkage* y *average-linkage*) y la estructura obtenida ha sido la misma en los tres casos. En la Figura 6.1 se muestra dicha estructura con la altura de los nodos correspondiente a *average-linkage*.

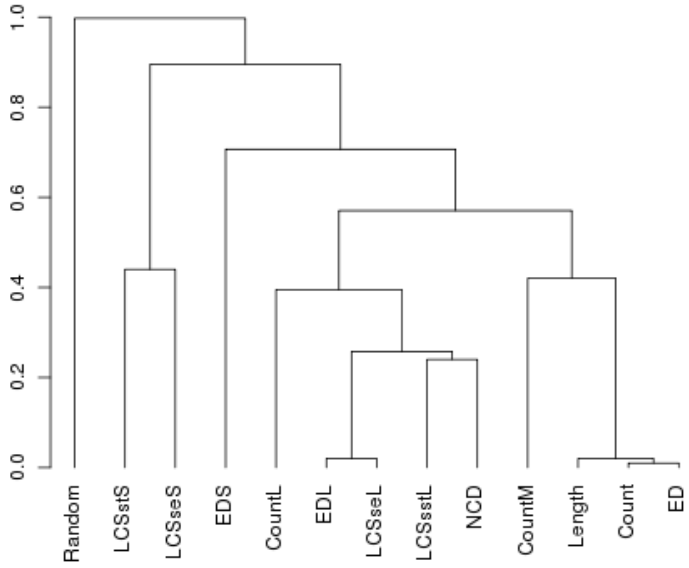

Figura 6.1: Dendrograma que muestra la relación de las distintas distancias utilizadas.

La primera conclusión que podemos extraer analizando el dendrograma es, al igual que se reflejaba en la Tabla 6.2, que cualquiera de las distancias tienen más parecido entre sí que la similitud que tienen respecto al caso

Random. Como consecuencia podemos afirmar que, a pesar de la diversidad de representaciones y distancias comparadas, todas son capaces de encontrar algún tipo de estructura común en los datos.

El dendrograma de la Figura 6.1 puede ayudarnos a encontrar familias de distancias, es decir, distancias que tienen comportamientos similares. Por un lado, como se sospechaba, las distancias ED y Count tienen un comportamiento muy similar al caso Length. Es decir, son muy dependientes de la longitud de la secuencia y no añaden prácticamente ninguna mejora a la opción Length. Ésta, a pesar de ser mucho más simple de calcular, no parece ser una opción a tener en cuenta ya que ignora los eventos particulares asociados a las secuencias. La opción CountM parece relacionada con este grupo por lo que parece que no aporta demasiado. Por lo tanto, descartamos las distancias asociadas a este subárbol del dendrograma.

El dendrograma también muestra que la estructura capturada por las distancias EDL y LCSseL es prácticamente la misma y que, aún habiendo algunas diferencias, estas dos distancias junto con LCSstL y NCD forman un grupo compacto. Aunque en menor medida CountL también se parece a estas cuatro distancias.

Finalmente, parece que las distancias normalizadas respecto a la longitud de la secuencia más corta no son capaces de capturar demasiado bien la estructura subyacente en los datos, dado que no se parecen especialmente entre ellas ni a ninguna de las otras distancias. Esto confirma, en cierta medida, las sospechas que han surgido al analizar la Tabla 6.2.

En definitiva, dado que las muestras no están etiquetadas no tenemos información para determinar cuál de las distancias es más adecuada. Sin embargo, el análisis realizado sugiere que el grupo formado por EDL, LCSseL, LCSstL, NCD y CountL es el más prometedor en este sentido.

6.4.3. Evaluación de los algoritmos

En este apartado vamos a tratar de determinar cuál de los 3 algoritmos analizados se adecúa mejor a los datos. El método utilizado para la evaluación de los algoritmos se ha descrito en la Sección 2.3.1 y se resume a continuación. La idea es que cuanto más parecida sea la matriz cofenética a la matriz de distancias, mejor se ajusta la jerarquía a los datos. El procedimiento de evaluación ha sido dividido en tres pasos para cada uno de los algoritmos evaluados:

1. Utilizando la matriz de distancias correspondiente se ha construido un dendrograma para cada una de las 13 distancias.

2. Se ha calculado la matriz cofenética para cada uno de los dendrogramas construidos.

3. Se ha calculado el valor de CPCC, es decir, la correlación entre la matriz de distancias y el correspondiente dendrograma (matriz cofenética obtenida a partir de él).

La Tabla 6.3 nos ayuda a identificar la bondad del comportamiento de los tres algoritmos respecto a cada una de las distancias analizadas.

	Average-linkage	*Single-linkage*	*Complete-linkage*
Count	0,99	0,97	0,99
CountL	0,88	0,62	0,76
CountM	0,98	0,97	0,95
ED	0,99	0,98	0,99
EDS	0,34	0,33	0,34
EDL	0,95	0,82	0,90
LCSstS	0,50	0,34	0,33
LCSstL	0,97	0,89	0,90
LCSseS	0,48	0,00	0,36
LCSseL	0,95	0,81	0,91
Length	0,97	0,95	0,98
NCD	0,92	0,83	0,81
Random	0,25	0,07	0,22
Media	0,83	0,71	0,77

Tabla 6.3: Valores de correlación entre la matrices de distancias y la matrices cofenéticas obtenidas mediante tres algoritmos de clustering. Los valores del caso Random no han sido incluidos en el cálculo de la media.

Parece claro que, independientemente de la distancia utilizada, los mejores resultados los obtiene el algoritmo *average-linkage*. Es el algoritmo que consigue los valores de correlación más altos, no sólo en media, sino que para todas las distancias evaluadas excepto Length. También se puede confirmar que la distancia Random es diferente a todas las demás ya que ninguno de los algoritmos es capaz de obtener un dendrograma representativo.

Además del análisis realizado sobre los algoritmos, la información de la Tabla 6.3 muestra ciertas diferencias entre las distancias. Los valores de la tabla muestran que los algoritmos jerárquicos aglomerativos tienen serias dificultades para poder extraer estructura mediante las distancias normalizadas en base a la secuencia más corta: las distancias EDS, LCSstS y LCSseS

obtienen los valores más bajos para los tres algoritmos, y nunca superan el 0,5. Además, estas distancias han mostrado una mayor inestabilidad en las diferentes ejecuciones llevadas a cabo.

Por lo demás, podemos decir que para el resto de las distancias los algoritmos de clustering son capaces de extraer la estructura de la matriz de distancias y además, si la opción utilizada es *average-linkage*, la correlación entre la matriz de distancias y la matriz cofenética es, en media, de 0,96.

6.4.4. Coste computacional

Como se ha mencionado, la detección de *malware* necesita ser realizada de una manera eficiente ya que un gran número de nuevo *malware* aparece todos los días. Por lo tanto, otro aspecto importante a tener en cuenta cuando se está diseñando una herramienta de detección de *malware* es el tiempo de ejecución.

Obviamente, la velocidad será un factor importante para discriminar distancias que presenten un comportamiento similar y adecuado (aunque Random y Length sean las opciones más rápidas no parece sensato seleccionarlas y se han incluido en el estudio previo simplemente como valores de referencia).

La Tabla 6.4 muestra el tiempo requerido para construir las matrices de distancias en milisegundos. Las ejecuciones se han realizado en un Pentium IV, 3.2 Ghz, 1 GB de RAM. No se muestran los tiempos de las distancias normalizadas porque son muy similares a los de las originales. Se puede observar en la tabla que hay grandes diferencias entre los tiempos requeridos por las diferentes métricas.

Distancia	Tiempo (ms)	Factor de aceleración
ED	1.102.673	1
LCSse	63.648	17
LCSst	33.782	33
NCD	2.716	406
Count	347	3.178

Tabla 6.4: Tiempo de ejecución (milisegundos) para la generación de las matrices de distancias. La última columna muestra el factor de aceleración de cada distancia respecto a la distancia *Edit-Distance* (ED)

Como era de esperar, los valores de la tabla muestran que, en general, las comparaciones entre secuencias requieren más tiempo que las aproximaciones vectoriales. Sin embargo, conviene recordar que la representación en

secuencias tiene un nivel semántico mayor que la vectorial. En cualquier caso, las diferencias entre las distancias que utilizan la representación en secuencias también son muy grandes. La más costosa es ED y el tiempo que requiere es significativamente mayor que el que requiere la siguiente pareja: LCSse y LCSst. La distancia NCD se encuentra a un orden de magnitud por debajo de estas dos últimas distancias, pero a un orden de magnitud por encima de la aproximación vectorial Count combinada con la distancia euclidiana.

Para realizar un análisis incremental del tiempo de ejecución requerido hemos utilizado las 5 muestras de 100 ejecutables mencionadas en la Sección 6.4. La Tabla 6.5 muestra los tiempos requeridos para obtener las matrices de distancias variando el tamaño de las muestras desde 100 hasta 500, de 100 en 100. A la hora de analizar estos resultados hay que tener en cuenta que aunque el número de secuencias en cada muestra aumenta gradualmente, el tamaño de las mismas es variable y, por tanto, el número de eventos a procesar no tiene por qué crecer de la misma manera, lo que puede influir en los resultados. La segunda fila de la Tabla 6.5 muestra el número total de eventos de las muestras utilizadas para la experimentación, es decir, la suma de las longitudes de todas las secuencias que pertenecen a la muestra correspondiente.

Se puede observar que cuanto más costoso computacionalmente es el cálculo de una distancia, el número de secuencias utilizadas afecta más al tiempo necesario para procesarla. El mejor resultado es claramente para Count. En el caso de las distancias sobre secuencias el caso de la distancia NCD es sin ninguna duda la que mejores resultados da en este sentido.

Sec.	100	200	300	400	500
Eventos	85.602	228.501	341.418	434.764	504.223
ED	1.102.673	8.298.168	19.195.181	31.030.384	42.197.967
LCSse	63.648	463.660	1.033.027	1.695.709	2.258.664
LCSst	33.782	246.621	563.651	912.161	1.231.900
NCD	2.716	11.728	26.078	45.652	64.564
Count	347	760	884	936	1.308

Tabla 6.5: Tiempo de ejecución (milisegundos) para la generación de las matrices de distancias para muestras de 100, 200, 300, 400 y 500 secuencias.

6.5. Discusión

Teniendo en cuenta la experimentación realizada, y contando con la colaboración de los expertos de la empresa S21sec, hemos analizado los resultados de cara a determinar los métodos a implementar en una plataforma que sirva como herramienta para analizar la estructura en familias del *malware* en base a su comportamiento dinámico.

En relación a la elección del método de clustering, tal y como se ha mencionado al comienzo de este capítulo, se ha utilizado una aproximación jerárquica aglomerativa. Los resultados han dejado claro que, entre los analizados, el algoritmo con mejor comportamiento para los datos disponibles es el *average-linkage*.

En cuanto a las distancias la decisión no está tan clara. La decisión pasa por seleccionar alguna de las distancias del grupo formado por NCD, LCSstL, LCSseL y CountL, ya que parece ser el grupo capaz de extraer una estructura más estable. EDL se puede descartar porque es muy similar a LCSseL y su coste computacional es excesivo. Dentro de este grupo hay que destacar la distancia CountL por su bajo coste computacional. Hay que recordar que se trata de una proyección en vector de las frecuencias de uso de cada llamada relevante al sistema por lo que esta métrica tiene el inconveniente de que supone una pérdida de información importante, como por ejemplo, el orden en que se han ido efectuando las llamadas.

Por este motivo, se planteó implementar la plataforma con la posibilidad de utilizar tanto la distancia CountL como la distancia NCD, ya que esta última era la distancia computacionalmente menos costosa de las aproximaciones basadas en secuencias. Tras analizar el comportamiento de ambas distancias mediante la plataforma en diferentes zonas del espacio de clasificación los expertos de la empresa de seguridad ratificaron que la distancia NCD es la más adecuada para la aplicación.

6.6. Plataforma MALBEC

La plataforma que brevemente se presenta en este apartado es la herramienta desarrollada para que los expertos en seguridad de la empresa S21sec puedan hacer un análisis de las estructuras y familias de *malware* que se van detectando mediante la aplicación de las técnicas mencionadas en las secciones anteriores. Debido a compromisos de confidencialidad contraídos con la empresa, la profundidad con la que se describe la plataforma es limitada.

La plataforma debía de proporcionar una jerarquía de *malware* con ca-

pacidad de aprendizaje incremental y cierta estabilidad, es decir, que no variara sustancialmente ante el aprendizaje de casos nuevos. Esta necesidad ha sido cubierta mediante la aplicación del nuevo algoritmo propuesto en el Capítulo 3, SIHC.

Otras de las condiciones que debía de cumplir la plataforma están relacionadas con la manejabilidad o usabilidad por parte del usuario. En este sentido la plataforma debía de cumplir las siguientes condiciones:

1. Enfoque modular. Era necesario diseñar la plataforma mediante un enfoque modular que permitiera hacer manejable el tratamiento de las numerosas familias de *malware* existentes.

2. Interfaz gráfica ergonómica. Para que el usuario pueda ir analizando la jerarquía que se va generando es necesario disponer de una interfaz gráfica adecuada que permita visualizar y manipular la información referente a las familias.

3. Posibilidad de centrar el proceso en zonas determinadas de la estructura. Gracias a esta característica el usuario puede hacer que el proceso de aprendizaje incremental se centre en las zonas de la jerarquía que más interés tengan desde el punto de vista de la seguridad informática.

4. Coste computacional. Para que el uso de la plataforma sea satisfactorio es necesario mantener el coste computacional dentro de unos márgenes razonables. Dado el elevado coste computacional de los algoritmos de clustering jerárquicos basados en secuencias, se han introducido optimizaciones en la plataforma que permiten reducir dicho coste. Por ejemplo, se ha implementado un proceso de simplificación de los nodos (se ejecuta automáticamente o manualmente) que permite reducir sustancialmente las necesidades de cálculo y almacenamiento. Por otro lado, el enfoque modular comentado anteriormente también permite atenuar el efecto del coste computacional. Finalmente, en caso de necesidad, se podría utilizar una de las implementaciones paralelas existentes del algoritmo de clustering utilizado (Olson, 1995).

Tal y como muestra la Figura 6.2 la interfaz gráfica de la plataforma tiene una pestaña con cuatro zonas o paneles para cada una de las partes de la jerarquía que el usuario tenga abiertas. El panel visualizador principal (el más extenso) es en el que se representa el estado actual de la jerarquía. En él se encuentra representado el dendrograma correspondiente. En el panel que está justo debajo del principal se visualiza una vista satélite de la jerarquía

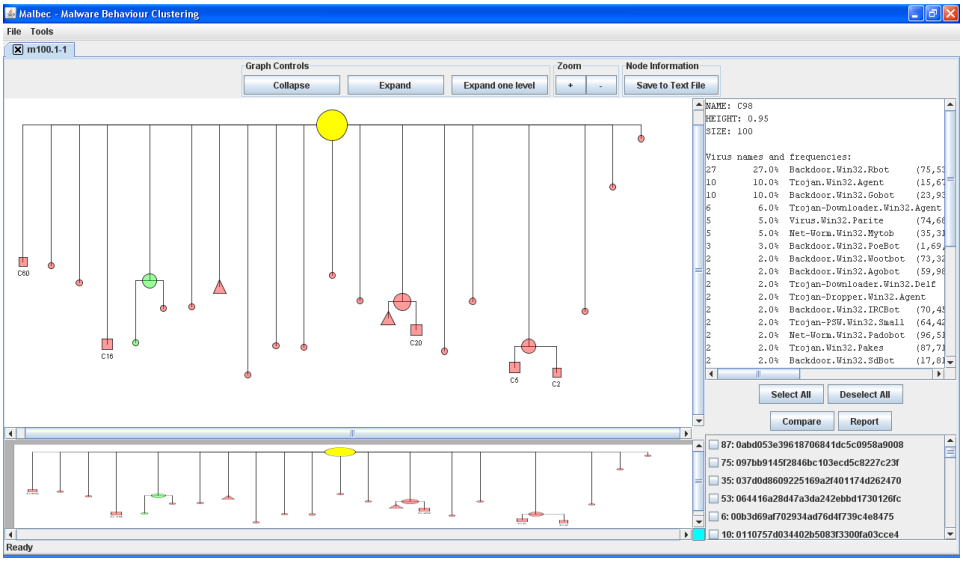

Figura 6.2: Pantalla principal de la plataforma MALBEC.

(para los casos en los que toda la jerarquía no entra en el panel principal). La parte derecha de la pantalla está dividida en dos zonas: en la parte de arriba se proporciona información del nodo seleccionado en el panel principal (nombre, tamaño, virus que contiene, etc.) mientras que en la parte de abajo se presenta una lista de los virus contenidos en el nodo seleccionado con sus correspondientes *checkbox*. Estos *checkbox* sirven para poder seleccionar los virus y comparar sus secuencias visualmente mediante la opción *Compare* o ver un informe sobre ellos con la opción *Report*.

En la Tabla 6.6 se presenta un resumen de las funcionalidades principales de la plataforma. Las opciones de la plataforma están divididas en dos menús en la barra de la aplicación (*File* y *Tools*) y un menú accesible mediante el botón derecho del ratón (*Explore*). Algunas funcionalidades tienen un botón propio entre la barra de menú y el panel principal. Todos los menús son sensibles al contexto, es decir, las opciones varían según el estado de la aplicación.

El menú *File* contiene toda la funcionalidad asociada con el tratamiento de la jerarquía (representada en forma de árbol/dendrograma): lectura, escritura, generación de nuevos árboles, impresión, etc.

El menú *Tools* es el que se encarga de la funcionalidad asociada a la clasificación de nuevos casos, la posibilidad del aprendizaje incremental, etc. También se han añadido algunas funcionalidades como la comparación de

Menú *File*	Menú *Tools*	Menú *Explore*
Open	Classify	Node Information
Build	Learn	Collapse Node
Save	Compare	Expand Node
Close	Report	Expand Node One Level
Export to jpg	Search Cluster	Toogle Interesting
Print		Change Cluster's Label
Exit		Prune Cluster
		New Subtree
		Open Subtree
		Open Parent Tree
		Rebuild Subtree

Tabla 6.6: Opciones de los menús de la plataforma MALBEC

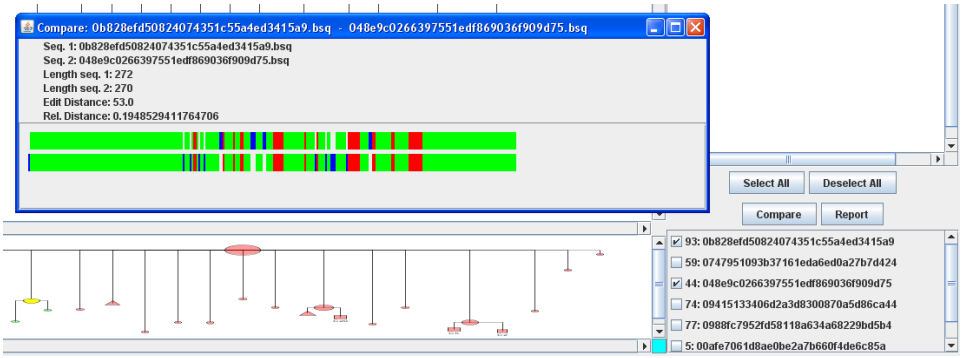

Figura 6.3: Ejemplo de comparación de las secuencias de llamadas al sistema de dos virus.

dos secuencias (ver Figura 6.3), la generación de un informe específico para la empresa S21sec, etc.

Finalmente, el menú *Explore* es el encargado de proporcionar toda la funcionalidad a la gestión y exploración de la jerarquía. Permite gestionar todo lo relacionado con los nodos (colapsarlos visualmente, expandirlos, obtener su información...) y todo lo relacionado con los subárboles (*subtrees*), es decir, partes de la jerarquía que se han modularizado para poder tener una visualización más adecuada. También tiene funciones para reducir el coste computacional en base a simplificar los representantes de cada nodo. Por último, también gestiona la posibilidad de seleccionar las zonas de la jerarquía que se desea tener activas a la hora de participar en el aprendizaje

incremental.

La aplicación ha funcionado satisfactoriamente para las muestras de datos proporcionadas por la empresa de seguridad. Asimismo, una vez realizada la plataforma la empresa ha proporcionado una nueva muestra de 2.100 virus donde se ha probado la plataforma de forma satisfactoria.

Parte del trabajo descrito en este capítulo ha sido presentado en un congreso internacional (Gurrutxaga et al., 2008). En concreto, la parte presentada corresponde a la comparativa de las diferentes medidas de distancia y los tres algoritmos utilizados, quedando fuera la parte más práctica relacionada con la plataforma.

Capítulo 7

Aplicación de técnicas de clustering a la detección de intrusos

7.1. Introducción

El clustering se puede utilizar para solucionar problemas de diversa índole de la vida real. En este capítulo nos centraremos en la aplicación de clustering a la seguridad de las redes informáticas, uno de los contextos con mayor repercusión en el buen funcionamiento de los sistemas informáticos.

En las últimas décadas el uso de las redes de ordenadores ha crecido sustancialmente. Este hecho amplía el espectro de los agresores de redes e incrementa el daño producido por los ataques. Los ataques a las redes de ordenadores afectan tanto a la seguridad de la información almacenada en las máquinas conectadas a la red como a la estabilidad de la propia red. Por consiguiente, es fundamental construir sistemas capaces de detectar ataques antes de que éstos causen ningún daño. Por ello, cualquier paquete completo de seguridad incluye un sistema de detección de intrusos o IDS (*Intrusion Detection System*).

Los sistemas de detección de intrusos se clasifican típicamente en 2 tipos, dependiendo de la naturaleza de los datos analizados. Los IDS de red o *network* IDS (nIDS) analizan el tráfico de datos transmitido por una red con el objetivo de detectar paquetes correspondientes a un ataque. Los IDS de ordenador o *host* IDS (hIDS) analizan el comportamiento de una máquina para tratar de determinar comportamientos sospechosos. Suelen monitorizar las llamadas al sistema, logs de aplicaciones, modificaciones en

ficheros... Algunos hIDS especializados en una aplicación particular suelen denominarse IDS de aplicación o *application* IDS (aIDS). En este capítulo nos centraremos exclusivamente en los IDS basados en red.

La detección de ataques en la red se puede hacer manualmente, mediante un análisis humano o automáticamente, mediante técnicas de aprendizaje automático. La detección hecha por el experto en seguridad requiere memorización, análisis de librerías de descripciones de ataques, búsqueda de ejemplos, etc. Evidentemente este método no resulta efectivo ya que es demasiado lento y subjetivo. Como consecuencia, para hacer frente al problema con éxito los sistemas de seguridad requieren unos nIDS automáticos y robustos.

En este contexto, las técnicas de aprendizaje automático pueden ser de gran ayuda en el análisis de los datos relativos a la red. La información obtenida puede servir para entrenar sistemas que sean capaces de detectar futuros ataques y evitar que éstos perjudiquen el sistema.

Las estrategias principales de aplicación del aprendizaje automático para crear nIDS automáticos se han clasificado tradicionalmente en dos grupos: detección de uso indebido (*misuse detection*) (Lee et al., 1999a) y detección de anomalías (*anomaly detection*) (Warrender et al., 1999). Tratando de evitar algunos de los inconvenientes de ambas estrategias ha surgido una tercera que se conoce como detección de anomalías no supervisada (*unsupervised anomaly detection*) (Portnoy et al., 2001).

Dada la gran cantidad de servicios o aplicaciones de red existentes en la actualidad, en las comunicaciones se transportan datos de muy variada naturaleza y formato. Por ello, los nIDS suelen centrarse en el análisis de las cabeceras de cada conexión[1], es decir, en la información relativa al funcionamiento de la propia red. En el caso de Internet eso implica que la mayoría los nIDS se centran exclusivamente en la información relativa a los protocolos de red como TCP, UDP, IP, ICMP etc. En general, nos referiremos a esta información como información TCP/IP o cabeceras TCP/IP.

No obstante, existen algunos nIDS que también analizan la parte correspondiente a los datos de aplicación (información conocida como *payload*). De todos modos, dado que cada aplicación tiene sus propios protocolos y formatos, los nIDS suelen especializarse en algunos de los servicios (típicamente web, correo electrónico, DNS...) realizando un análisis del *payload* en función del servicio que lo ha generado.

[1] Aunque algunos protocolos de red como UDP o ICMP no utilizan conexiones propiamente dichas utilizaremos el término conexión para referirnos a todo tipo de comunicaciones.

En este capítulo se utilizan técnicas de aprendizaje automático no supervisado para la detección de intrusos en red analizando tanto las cabeceras TCP/IP como el *payload*. El objetivo principal es demostrar que un análisis automático y genérico del *payload*, sin tener en cuenta el servicio correspondiente, mediante técnicas de clustering puede proporcionar unos resultados satisfactorios. Para ello se ha experimentado con un sistema de detección de intrusos basado en la detección de anomalías no supervisada y distintas aproximaciones para analizar el *payload*. Se ha cuantificado la capacidad de detección de ataques de las distintas aproximaciones y los resultados demuestran que una combinación de distintas técnicas ofrece resultados positivos.

7.2. Detección de intrusos en red basada en el aprendizaje automático

En esta sección se realiza una introducción a la detección de intrusos y más concretamente a la detección de intrusos en red basada en el aprendizaje automático. En primer lugar se describen las tres estrategias principales utilizadas en este campo. Posteriormente se define una clasificación de los distintos ataques que se pueden encontrar en la red y sus principales características. Finalmente se realiza un breve resumen de algunos de los nIDS que aparte de la información TCP/IP también trabajan con información obtenida del *payload*.

7.2.1. Estrategias principales para la detección de intrusos en red

En este apartado se describen las 3 estrategias principales basadas en el aprendizaje automático para la detección de intrusos en red. Tal y como se ha mencionado en la introducción éstas son la detección de uso indebido, la detección de anomalías y la detección de anomalías no supervisada.

Detección de uso indebido

En la primera aproximación, la de detección de uso indebido, se parte de una muestra de entrenamiento etiquetada a la que se le aplican técnicas de aprendizaje automático supervisado: el clasificador aprende a partir de un conjunto de conexiones etiquetadas, entre las que hay tráfico normal y ataques. Por lo tanto, aprende a reconocer esos ataques, o variantes similares. Esta aproximación se usa en sistemas como MADAM/ID (Lee et al., 1999b).

Este tipo de métodos tiene dos problemas principales. Por un lado es muy difícil obtener una muestra de tráfico de red completamente etiquetada. El número de paquetes transmitidos en una red es excesivamente alto para analizarlo manualmente. Además, un análisis de este tipo es tedioso y propenso a errores. Por lo tanto, obtener una muestra representativa y etiquetada con garantías es una tarea casi imposible.

Por otro lado, los sistemas basados en estos métodos sólo son capaces de detectar ataques existentes en la muestra de entrenamiento. Es decir, no son capaces de detectar ataques nuevos por lo que requieren una continua actualización. De todos modos, debido a que cada día se inventan nuevos ataques, estos sistemas se usan como filtro para detectar ataques conocidos, pero no para detectar nuevos tipos de ataques. Esto contradice el objetivo principal de los IDS que debería ser la detección de las primeras apariciones de nuevas intrusiones y evitar que éstas afecten a las víctimas.

En resumen, los nIDS basados en la detección de uso indebido basan su funcionamiento en aprender los ataques conocidos y emitir alarmas cuando los detectan. En este sentido, este tipo de nIDS se pueden equiparar a los antivirus clásicos, que almacenan firmas de virus conocidos a través de las cuales pueden reconocer ficheros infectados. De hecho, muchos nIDS de detección de uso indebido se basan también en firmas que sirven para detectar conexiones relacionadas con un ataque.

Detección de anomalías

La segunda aproximación, la detección de anomalías, la propuso por primera vez Denning (1987). La idea que subyace en esta aproximación es la de modelar el comportamiento del tráfico normal y considerar como intrusiones las conexiones que se desvíen del modelo construido. Por lo tanto, los clasificadores aprenden el comportamiento del tráfico normal y las conexiones anómalas se consideran ataques.

Las ventajas principales respecto a la detección de uso indebido son dos. Por un lado no es necesario conseguir una base de datos etiquetada sino que basta con tener una base de datos con tráfico exclusivamente normal. Por otro lado, esta estrategia es capaz de detectar ataques nuevos ya que todo aquello que se desvíe del tráfico normal será considerado un ataque.

Desafortunadamente, estas dos ventajas no están libres de problemas. Por un lado, obtener una muestra de datos representativa del tráfico real en la que no haya ningún ataque no es trivial. Hay que garantizar que en la red monitorizada no haya habido ningún ataque, o bien es necesario ser capaz de filtrarlos. El primer caso es inviable en entornos reales, mientras

que el segundo nos exige un filtrado muy difícil de realizar. Hay que tener en cuenta que este punto es fundamental ya que cualquier ataque presente en los datos de entrenamiento sería asimilado como tráfico normal.

Por otro lado, este tipo de estrategias genera una gran cantidad de falsas alarmas (falsos positivos). Las causas principales son que el tráfico puede variar con el tiempo y que no es sencillo obtener datos representativos de todos los tipos de usos legítimos. Por ello, toda actividad normal no presente en los datos de entrenamiento será considerada un ataque.

Aunque hace ya muchos años que aparecieron los primeros sistemas basados en detección de anomalías, ésta es todavía una aproximación que se utiliza mucho. Por ejemplo, Dagorn (2008) presentó recientemente un sistema de detección de intrusos basado en anomalías para aplicaciones web y Rehák et al. (2008) presentaron una manera de disminuir el error en sistemas basados en anomalías utilizando lo que denominan un modelado de confianza colectiva (algo similar a la combinación de soluciones que propondremos nosotros en este capítulo). Para una comparativa reciente de nIDS basados en detección de anomalías se puede acudir al trabajo de Ashfaq et al. (2008) donde comparan 8 sistemas desde el punto de vista del error, escalabilidad, complejidad y retardo de detección. En esta comparativa se utilizan dos bases de datos recolectadas independientemente (ambas incluyen únicamente información TCP/IP puesto que los 8 sistemas evaluados se basan en este tipo de información).

Detección de anomalías no supervisada

Debido a los problemas de las dos aproximaciones mencionadas, una tercera aproximación, la detección de anomalías no supervisada (Portnoy et al., 2001), está tomando protagonismo desde hace algunos años. Esta tercera opción se basa en dos premisas:

- El volumen de tráfico normal que encontramos en la red es mucho mayor que el volumen de tráfico correspondiente a ataques. Este hecho se cumple como norma general aunque es cierto que algunos ataques de envío masivo de paquetes generan una ingente cantidad de tráfico.
- El comportamiento o estructura de las intrusiones es distinto al del tráfico normal. Este supuesto es necesario para cualquier sistema de detección ya que si los ataques fueran indistinguibles del tráfico normal poco podrían hacer los IDS.

Si estos supuestos se cumplen, el problema de la detección de intrusos se puede afrontar como un problema de detección de casos atípicos (*outlier*

detection). Ciertamente, bajo los mencionados supuestos, los ataques se pueden ver como unos pocos casos distintos a la mayoría.

La ventaja principal de la detección de anomalías no supervisada es que trabaja sobre datos sin etiquetar que incluye tráfico normal y ataques. Es decir, se basan en un tipo de bases de datos que se puede conseguir simplemente monitorizando una red con lo que se facilita mucho el proceso de obtención de la muestra. Los métodos de detección de anomalías no supervisados normalmente construyen modelos probabilísticos de los datos que ayudan a decidir si una conexión es anómala o atípica, es decir corresponde a un ataque, o no.

Existen diversas técnicas para la detección de casos atípicos entre los que se incluyen los algoritmos de clustering. En este caso el procedimiento exige aplicar el algoritmo a los datos de entrada. Posteriormente se analiza la partición obtenida en busca de clusters con menor número de casos (Portnoy et al., 2001). Se puede considerar el número de casos de un cluster como un indicador de "normalidad" de los objetos de ese cluster (cuantos más objetos, más normal y cuantos menos objetos, más anómalo).

7.2.2. Tipos de ataques y sus características

Tanto las técnicas de seguridad informática como los posibles ataques evolucionan de una manera vertiginosa. Por consiguiente, no es sencillo realizar una clasificación general de los distintos tipos de ataques. De todas maneras, siguiendo a Lee et al. (1999a) vamos a clasificar los ataques en cuatro grupos principales:

- Denegación de servicio o *Denial of Service* (DoS): El objetivo de este tipo de ataques es conseguir que un servidor no pueda ofrecer su servicio a un usuario legítimo. La técnica habitual de este tipo de ataques se basa en el envío masivo de paquetes o técnica de inundación (*flood*) aunque en algunos casos basta con un único paquete que explote una vulnerabilidad conocida. Aunque este tipo de ataques puede generar grandes molestias su peligrosidad es moderada ya que no son peligrosos para la integridad de los sistemas informáticas.

- Escaneo o *Probing*: Los ataques de este tipo se dedican a rastrear la red para obtener la mayor cantidad de información posible. Estos ataques suelen preceder a otros más dañinos. Es habitual que este tipo de ataques generen una gran cantidad de paquetes en poco tiempo, por lo que se consideran del tipo inundación, aunque hay versiones "pacientes" que realizan escaneos espaciados en el tiempo. La peligrosidad de

estos ataques es baja, pero es interesante detectarlos porque pueden servir como aviso de ataques más peligrosos.

- De remoto a local o *Remote to Local* (R2L): Este grupo de ataques engloba a todos aquellos que tienen como objetivo la obtención de acceso a un sistema para el que el atacante no tiene permisos de acceso. En general, a diferencia de los anteriores, estos ataques no generan un tráfico elevado. Es habitual que estos ataques precedan a los ataques del siguiente grupo (U2R). La peligrosidad de estos ataques es alta ya que conlleva que usuarios ilegítimos accedan al sistema.

- Usuario a superusuario o *User to Root* (U2R): El objetivo de estos ataques es el de obtener permisos de superusuario, *root* o administrador a partir de un usuario "normal". El acceso al usuario normal puede ser legítimo o no. Al igual que los ataques R2L, éstos no requieren un envío masivo de paquetes. Su peligrosidad es extrema ya que si el ataque tiene éxito el intruso obtiene el control total del sistema.

Debido a sus características particulares los nIDS basados en la detección de anomalías no supervisada no son adecuados para los ataques que generan una gran cantidad de tráfico. En estos casos el primer supuesto no se cumple y por lo tanto los ataques forman grupos de gran tamaño que impiden catalogarlos como casos atípicos. Una posible solución es la de clasificar como ataques los casos más aislados y los que se agrupan en un número especialmente alto. Sin embargo, se corre el peligro de clasificar como ataques un elevado número de conexiones normales.

De todos modos, los ataques tipo *flood* suelen ser los menos peligrosos (DoS y *Probing*) y los más fáciles de detectar. La mayoría de ataques *flood* se pueden detectar analizando la información TCP/IP de las conexiones realizadas en una breve ventana de tiempo.

Por lo tanto, parece adecuado centrar los nIDS basados en detección de anomalías no supervisada en la detección de ataques que no sean del tipo *flood*. Esto conlleva a que la detección de muchos de los ataques se beneficie de (o incluso requiera) un análisis más extenso que el de la información TCP/IP, es decir, del análisis de los datos de aplicación o *payload*.

7.2.3. Análisis del *payload*

Como ya se ha mencionado en la introducción a este capítulo las características del *payload* varían dependiendo del servicio o aplicación de red correspondiente a la conexión. Como consecuencia, existen pocos sistemas

de detección de intrusos que utilizan esta información y la mayoría de los nIDS que se pueden encontrar en la bibliografía y que analizan el *payload* son específicos para un reducido número de servicios. Veamos tres ejemplos:

- Krügel et al. (2002) presentan un trabajo que se centra en ataques tipo R2L y utiliza información específica de los protocolos de aplicación para incrementar el ratio de detección de intrusos. El prototipo que han implementado puede procesar tráfico de tipo HTTP y DNS pero sólo presentan resultados para DNS.

- Wang y Stolfo (2004) basan su trabajo en distribuciones de perfiles de frecuencias de bytes. Durante la fase de entrenamiento calculan la desviación estándar del *payload* que llega a un host y puerto determinado (en situaciones normales el puerto identificará el servicio utilizado). El sistema crea un perfil para cada host, puerto y longitud de *payload* y durante la fase de detección utilizan la distancia de Mahalanobis para comparar las nuevas conexiones con los perfiles almacenados. Si la distancia supera cierto umbral el sistema genera una alarma. Evidentemente este modelo es específico para un host y puerto concreto y también está condicionado por la longitud del *payload*.

- La ampliamente usada base de datos Kddcup99 (Lee, 1999) puede servir de ejemplo de otro tipo de análisis de *payload*. Las particularidades de esta base de datos se describen en la Sección 7.4.1. Para la generación de esta base de datos se analizaron millones de conexiones. El análisis del *payload* se realizó en base a unas reglas generadas por expertos específicamente para esos datos.

  Este tipo de análisis tiene serios inconvenientes. Por un lado requiere el conocimiento de expertos que generen las reglas adecuadas para esos datos. Por lo tanto, la utilización del sistema en otra red puede requerir la modificación de las reglas por expertos en la nueva red. Por otro lado, las reglas requieren el conocimiento de las aplicaciones de red y por ello se suelen reducir a las más utilizadas. Aunque las aplicaciones más conocidas engloben la inmensa mayoría de las conexiones es posible que los ataques se basen precisamente en aquéllas menos utilizadas. Finalmente, el sistema suele ser estático, es decir, siempre se aplican las mismas reglas a no ser que los expertos las actualicen. En un mundo con tanta variabilidad es necesario un sistema inteligente y adaptativo que sea capaz de adaptar su funcionamiento a los datos.

Como se puede observar, todos estos métodos son específicos, dependientes de servicios. Además, dado que el servicio se suele identificar mediante el puerto utilizado puede ocurrir que esta identificación sea errónea, ya que, tal y como advierten Zanero y Savaresi (2004), el puerto utilizado para un servicio no tiene por qué ser siempre el estándar.

7.2.4. Combinación de nIDS

Las ventajas y desventajas de los distintos tipos de nIDS hacen que la mejor solución disponible hoy en día sea la de combinar varios nIDS. De hecho, lo mejor es combinar distintos tipos de sistemas de seguridad como cortafuegos, antivirus, IDS de host, IDS de red...

Centrándonos en los nIDS se ha podido observar que los nIDS basados en detección de uso indebido son muy fiables a la hora de detectar ataques conocidos y que su tasa de falsas alarmas es muy baja, pero son incapaces de detectar ataques nuevos. En cambio, los que se basan en la detección de anomalías, sufren de un alto número de falsas alarmas, pero son capaces de detectar ataques desconocidos. Una combinación de distintos nIDS puede conseguir un gran aumento en el rendimiento.

Una posible combinación consiste en ejecutar paralelamente los distintos sistemas y utilizar alguna técnica de correlación de alarmas. Otra posible combinación consiste en ejecutar los sistemas en serie, filtrando los ataques detectados en cada caso. (Debar y Wespi, 2001; Porras et al., 2002)

Este capítulo se centrará en el análisis de un sistema de detección de intrusos en red basado en la detección de anomalías no supervisada que es capaz de analizar el *payload* independientemente del servicio utilizado. De todos modos, el sistema no se presenta como un sistema independiente sino que se propone como parte de un sistema más amplio. Dado que este tipo de nIDS no es adecuado para ataques *flood* nuestra propuesta podría ser combinada con un filtro de tipo cortafuegos que se ocupara de detectar los ataques *flood*. Además, los resultados mejorarían si los datos fueran filtrados por un nIDS de detección de uso indebido para eliminar los ataques conocidos y potenciar el primer supuesto de los nIDS basados en detección de anomalías no supervisada.

7.3. Un nuevo sistema para la detección de intrusos en red

En esta sección se describe un nuevo sistema de detección de intrusos en red basado en la detección de anomalías no supervisada. La característica más novedosa de este sistema es que analiza el *payload* de las conexiones de forma genérica, sin tener en cuenta ningún parámetro como el servicio asociado, la cantidad de datos... y de forma automática, mediante técnicas de aprendizaje automático. De esta manera, el sistema es genérico y aplicable en cualquier tipo de red.

El objetivo principal de este sistema es demostrar que un sistema automático basado en el clustering, y que analice el *payload* de forma genérica, puede obtener resultados satisfactorios. Por ello, el sistema de detección de anomalías o casos atípicos es sencillo con lo que queda margen de mejora para trabajos futuros.

En los siguientes apartados se describe el sistema propuesto. Primero se describe la técnica utilizada para detectar casos atípicos en los datos. Posteriormente se describen las distintas técnicas de análisis de *payload* que hemos utilizado, todas ellas genéricas. Finalmente se describe un esquema general que permite combinar las distintas aproximaciones propuestas para conseguir un único sistema combinado.

7.3.1. Detección de anomalías basado en clustering

Como hemos mencionado anteriormente, la detección de anomalías no supervisada se afronta utilizando las mismas estrategias que se utilizan para resolver cualquier problema de detección de casos atípicos (Hodge y Austin, 2004): se construyen modelos probabilísticos de los datos que ayudarán a decidir si las conexiones examinadas corresponden a anomalías (ataques) o no (normales).

La herramienta utilizada en el sistema propuesto es el clustering, tal y como lo hacen Eskin et al. (2002) o Leung y Leckie (2005). Para ello, hemos utilizado un algoritmo de clustering sencillo y escalable conocido como *Fixed-Width* (Eskin et al., 2002) o *Leaders* (Vijaya et al., 2004) descrito en el Capítulo 2 (pág. 25).

La sencillez del algoritmo hace que sea eficiente aunque también hace que el algoritmo no sea adecuado en muchos contextos. De todos modos, la mayoría de los problemas de este algoritmo afloran en los clusters de gran tamaño ya que los sobreparticiona. Sin embargo, como para la detección de casos atípicos nos interesan los clusters con pocos casos, el algoritmo resulta

válido.

Una vez ejecutado el algoritmo sobre una base de datos de conexiones de red se asigna un valor o puntuación a cada conexión. Para ello, en primer lugar se calcula para cada cluster el número de conexiones que están en su hiperesfera correspondiente. Ese valor será la puntuación de todas las conexiones asignadas a ese cluster. Hay que subrayar que aunque una conexión se asigna a un único cluster es posible que esté en la hiperesfera correspondiente a varios, dado el solapamiento de los clusters producidos por *Fixed-Width*.

Por lo tanto, un valor más alto representa un mayor nivel de "normalidad" de la conexión y los objetos con valores más bajos serán considerados como ataques. El resultado es un ranking de conexiones y es tarea del administrador de seguridad establecer las conexiones que serán consideradas ataques. Esto se puede realizar definiendo una puntuación umbral o un número máximo de alertas a tratar.

7.3.2. Propuestas para el procesamiento del *payload*

Como hemos mencionado en la Sección 7.2.3, debido a cuestiones prácticas, los sistemas de detección de intrusos en red se centran principalmente en la información de las cabeceras TCP/IP de los paquetes detectados en la red. El formato y semántica de estas cabeceras es conocido y los datos se pueden procesar fácilmente. El problema es que en muchos casos esta información no es suficiente para detectar intrusos y por lo tanto surge la necesidad de analizar el *payload*.

En este apartado se describen las diversas aproximaciones que hemos propuesto y analizado para procesar el *payload* de las conexiones sin tener en cuenta ningún conocimiento a priori. Es decir, principalmente, sin presuponer que corresponde a ningún servicio en particular. Partiendo desde esta premisa la única opción posible es la de analizar el *payload* como una secuencia de elementos sin estructura. Dado que no podemos realizar ninguna suposición, lo más lógico parece que estos elementos mínimos sean bytes. Por lo tanto, tal y como lo planteamos, el análisis de conexiones en base a su *payload* requiere la combinación del algoritmo de clustering *Fixed-Width* con métodos adecuados para comparar secuencias de bytes.

A continuación se describen varias técnicas para la comparación de *payloads*. Obviamente, el objetivo principal de estas técnicas es la de asignar valores bajos a los *payloads* similares y altos a los diferentes, de forma que ayuden a distinguir las conexiones normales de los ataques. Sin embargo hay otra serie de características que son deseables:

- Ser automáticos, es decir, que no requieran intervención humana.
- Ser genéricos, es decir, que sean independientes de los servicios o puertos concretos y como consecuencia utilizables en distintos entornos y adaptables a situaciones cambiantes.
- Ser computacionalmente eficientes para que puedan operar en tiempo real y en entornos con grandes anchos de banda.

No es fácil construir un método que aglutine todas estas características por lo que se han analizado 6 aproximaciones, agrupadas en 3 grupos, que se describen a continuación.

Análisis de secuencias

Tal y como se acaba de mencionar, un análisis genérico del *payload* requiere tratar éste como una secuencia de bytes. La aproximación más directa para comparar secuencias de bytes es la de utilizar distancias de comparación de secuencias como por ejemplo la distancia de edición o *edit distance* (Gusfield, 1997). Aunque existen múltiples variantes, la distancia básica calcula el número mínimo de ediciones necesarias para convertir una secuencia en otra. Su mayor inconveniente es su alto coste computacional por lo que no lo consideramos práctico para el fin propuesto ya que en el tráfico habitual de una red normal podemos encontrar un gran número de conexiones y *payloads* de gran tamaño. Además, es un método que tiene dificultades al comparar secuencias de longitudes muy heterogéneas, cosa habitual en el tráfico de red.

Con el objetivo de resolver dichos inconvenientes nuestra propuesta es la de utilizar una distancia mucho más eficiente como es la distancia de compresión normalizada o NCD (*Normalized Compression Distance*) (Li et al., 2004; Wehner, 2007). Esta distancia, basada en la complejidad de Kolmogorov, se aproxima mediante algoritmos de compresión tal y como se ha descrito en el Capítulo 6.

Análisis de histogramas

Otra posibilidad para representar la información de secuencias es la de procesarlas para extraer cierta información más sencilla de analizar. Un análisis clásico de secuencias es el del análisis de n-gramas. Sin embargo, dado el tamaño del alfabeto en este caso (256 símbolos), el análisis de n-gramas sólo es práctico para $n = 1$.

En esta segunda aproximación proponemos calcular la frecuencia de aparición de cada uno de los 256 posibles n-gramas en cada uno de los *payloads*. Esta representación permite utilizar las distintas técnicas existentes para el análisis de histogramas tal y como lo han hecho otros autores (Wang y Stolfo, 2004).

Dentro de las distancias entre histogramas podemos encontrar, por ejemplo, una distancia basada en la proximidad de posiciones y la forma de los picos de los histogramas comparados (Strelkov, 2008). La idea es la de desplazar uno de los histogramas sobre el otro para tratar de sobreponerlos de la mejor forma posible. No obstante, esta distancia considera que los distintos símbolos del alfabeto están ordenados y que por lo tanto hay distintos niveles de similitud entre ellos. No hemos utilizado esta distancia dado que, para garantizar que el análisis del *payload* es genérico, no podemos realizar ninguna suposición y, por lo tanto, no podemos presuponer ningún tipo de relación entre símbolos (bytes).

La propuesta de Serratosa y Sanfeliu (2006) permite tratar a todos los símbolos por igual, sin tener en cuenta el orden, por lo que es adecuada para el caso que nos ocupa. Los autores proponen el uso de una representación compacta de los histogramas a la que llaman firma, con lo que consiguen reducir el espacio requerido para almacenar los histogramas. Además, proponen un modo para comparar las firmas en base a diferentes distancias equivalentes a la comparación de los histogramas completos.

El concepto de firma de Serratosa y Sanfeliu consiste en un vector que contiene los símbolos del alfabeto con frecuencias mayores que cero. Junto con cada símbolo se almacena también su frecuencia. Para poder comparar firmas proponen el concepto de firma extendida que incluye los símbolos necesarios con frecuencia 0 de forma que las dos firmas comparadas tengan la misma longitud y los elementos en la misma posición en las dos firmas correspondan al mismo símbolo.

Para definir las distancias entre firmas que hemos utilizado definimos w_i^A como el símbolo correspondiente al elemento i-ésimo de la firma extendida de la secuencia A y m_i^A como su frecuencia correspondiente. La longitud de la firma extendida será denotada como z. Siguiendo la nomenclatura de Serratosa y Sanfeliu (2006) a continuación describimos la distancia nominal que coincide con la distancia Manhattan:

$$\mathrm{d}_{\mathrm{nom}}(A,B) = \sum_{i=1}^{z} \left| m_i^A - m_i^B \right|$$

De forma similar, podemos definir la distancia euclidiana entre dos fir-

mas:

$$\mathrm{d}_{\mathrm{euc}}(A,B) = \sqrt{\sum_{i=1}^{z}(m_i^A - m_i^B)^2}$$

Finalmente, describimos la distancia ordinal que representa el número mínimo de movimientos de elementos que son necesarios para convertir un histograma en otro:

$$\mathrm{d}_{\mathrm{ord}}(A,B) = \sum_{i=1}^{z-1}(w_{i+1}^A - w_i^A)\left|\sum_{j=1}^{i}(m_j^A - m_j^B)\right|$$

Análisis de elementos más frecuentes

A continuación presentamos un tercer tipo de distancias para comparar *payloads*. En este caso, al igual que en el análisis de histogramas del apartado anterior, se procesa el *payload* para calcular la frecuencia de aparición de cada byte. Asimismo, se transforma este recuento de bytes para que se pueda almacenar en un espacio más reducido. La diferencia principal del preproceso estriba en que la representación obtenida no almacena toda la información del histograma.

El análisis propuesto en este apartado se basa en la idea de que en los *payload* similares los bytes más frecuentes serán similares. Por ello, se analizan sólo los m bytes más frecuentes de cada *payload*. Con ello se pretende reducir el tamaño de la información almacenada e ignorar el posible ruido que se puede generar con tan elevado número de símbolos.

En base a esta representación hemos definido dos distancias para comparar dos secuencias de m símbolos:

- La primera de las distancias es un recuento de posiciones en las cuales los símbolos son diferentes, es decir, la distancia Hamming.
- Dado que con esta representación de los datos parece sensato tener en cuenta la posición de cada símbolo, hemos definido otra distancia.

$$\mathrm{d}(A,B) = \frac{\sum_{i=1}^{m}(\mathrm{d}'(i,A,B))}{m^2}$$

donde

$$\mathrm{d}'(i,A,B) = \begin{cases} |i-j| & \text{si} \quad \exists j : A_i = B_j, 1 \le j \le m \\ m & \text{en caso contrario} \end{cases}$$

Esta distancia calcula la suma de las diferencias de las posiciones del mismo símbolo en ambas secuencias. Si un símbolo aparece en tan sólo una de las secuencias se penaliza sumando m. Finalmente, se normaliza el valor para que esté entre 0 y 1.

7.3.3. Combinación de los sistemas propuestos

En la Sección 7.2.4 ya se menciona la utilidad de combinar distintos tipos de sistemas de detección de intrusos. El objetivo de analizar el *payload* es utilizar dicho conocimiento junto con el que aporta la información TCP/IP. Por ello, en este caso es indispensable diseñar un método que permita combinar distintos sistemas. A continuación se describe un método para combinar de forma efectiva distintas técnicas que puntúen las conexiones en base a su nivel de "normalidad". Esto permite combinar el análisis TCP/IP con el análisis del *payload* o incluso combinar varias técnicas de análisis de *payload*.

El sistema para la detección de anomalías no supervisada descrito en la Sección 7.3.1 produce un resultado donde se asocia una puntuación a cada una de las conexiones analizadas. Por consiguiente, un método para combinar el análisis de los datos realizado mediante las diferentes técnicas propuestas ha de combinar las diferentes puntuaciones asignadas a cada conexión.

El problema principal para poder realizar esta combinación es que las puntuaciones no están normalizadas. Dado que la puntuación de una conexión corresponde al número de objetos de su cluster, la distribución de las puntuaciones depende de la partición generada por el algoritmo de clustering. Además, dado que el algoritmo *Fixed-Width* genera clusters solapados, el nivel de solapamiento también influirá en dicha distribución. En resumen, una combinación de puntuaciones requiere una normalización de las puntuaciones.

La aproximación más sencilla consistente en normalizar las puntuaciones entre 0 y 1 de forma lineal no parece suficiente y varias pruebas experimentales lo han confirmado. Por ello, hemos definido una normalización basada en el rango. En este caso todas las conexiones se ordenan en función de su puntuación y se asigna a cada conexión una nueva puntuación que corresponde a la posición que ocupa en la lista ordenada. En caso de que varias conexiones tengan la misma puntuación se les asigna a todos la misma puntuación normalizada, correspondiente a la media de las posiciones de todas las conexiones implicadas.

Una vez normalizadas las puntuaciones éstas se pueden combinar de diversas formas. Nuestra propuesta es utilizar la media de todas las pun-

tuaciones normalizadas. Aunque también hemos realizado algunas pruebas con el mínimo y el máximo de los valores parece que la media obtiene los mejores resultados.

Para terminar, a continuación se describe un breve resumen del proceso propuesto para la detección de intrusos en red basado en la detección de anomalías no supervisada y un análisis genérico del *payload*. El proceso completo se representa en la Figura 7.1.

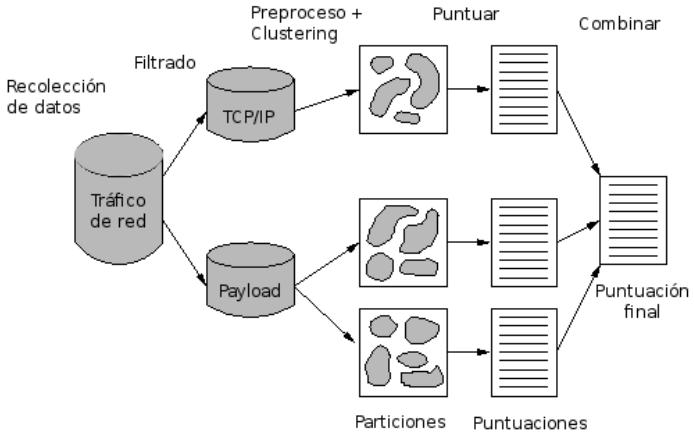

Figura 7.1: Esquema del sistema de detección de intrusos propuesto.

La primera fase del proceso consiste en la recolección de datos de la red. Una vez recolectada dicha información se divide en dos partes: por un lado la información perteneciente a las cabeceras TCP/IP de las conexiones y por otro lado, los datos del nivel de aplicación o *payload*. La información TCP/IP, al ser estructurada, es generalmente sencilla de procesar y transformar en un vector. De esta manera se puede procesar de forma sencilla mediante cualquier algoritmo de clustering. Los *payloads* obtenidos se pueden agrupar utilizando varias de las distancias definidas en la sección anterior. En cualquier caso, al final de este paso tendremos varias particiones correspondientes a las mismas conexiones. A cada una de las conexiones se le asigna una puntuación en base a cada una de las particiones y se normalizan en base al rango. La puntuación final para cada conexión será la media de todas esas puntuaciones normalizadas. Las conexiones con puntuaciones bajas serán consideradas como ataques con mayor probabilidad que las conexiones con puntuaciones altas.

7.4. Experimentación

Para evaluar el sistema de detección de intrusos propuesto y las distintas técnicas posibles para analizar el *payload* hemos definido dos objetivos. El primer objetivo es el de medir si un sistema sencillo de detección de casos atípicos como el descrito en la Sección 7.3.1 es capaz de conseguir tasas de detección aceptables basándose sólo en el *payload* de las conexiones de forma genérica. Como referencia utilizaremos las tasas de detección obtenidas por un análisis similar mediante información TCP/IP o el análisis de *payload* realizado por expertos.

El segundo objetivo de la experimentación será cuantificar las tasas de detección de las distintas combinaciones posibles del análisis de información TCP/IP y los distintos tipos de análisis de *payload*. De este modo conseguiremos saber si las distintas técnicas son complementarias o no.

7.4.1. Generación de datos

La obtención de una base de datos adecuada para la experimentación ha sido una tarea compleja. Tal y como hemos mencionado anteriormente, no es fácil obtener datos etiquetados sobre tráfico de red. Aunque la detección de anomalías no supervisada no requiere datos etiquetados, la evaluación de uno de estos sistemas sí que lo requiere. Por ello, y para utilizar unos datos utilizados por otros investigadores, hemos utilizado una versión de la base de datos Kddcup99 que se puede obtener en el repositorio de UCI (Asuncion y Newman, 2007).

La base de datos Kddcup99 se generó procesando la base de datos DARPA98; base de datos generada por el grupo de tecnologías de sistemas de información (IST) del Lincoln laboratory del MIT (Masachussets Institute of Technology) con la colaboración de DARPA (Defense Advanced Research Projects Agency) y AFRL (Air Force Research Laboratory) (Lippmann et al., 2000). Para la obtención de datos montaron una red de ordenadores que simulaba una situación real de tráfico de red que contiene tráfico normal y ataques. Utilizaron *Tcpdump* (Jacobson et al., 1989) para monitorizar la red y almacenar todos los paquetes correspondientes al tráfico de red en un fichero *tcpdump*. El experimento se prolongó durante 7 semanas de 5 días.

Lee (1999) obtuvo mediante técnicas semi-automáticas una versión vectorizada de la base de datos DARPA98. Esta versión, al formatear los datos en forma de tabla, es directamente analizable mediante técnicas estándar de minería de datos. Dado que esta base de datos fue utilizada en la competición Kddcup del año 1999 se conoce como Kddcup99.

Cada entrada de la base de datos Kddcup99 contiene información de 41 variables independientes sobre una conexión, además de una variable dependiente indicando si es una conexión normal o corresponde a un ataque. Las 41 variables se dividen de la siguiente manera:

- Variables intrínsecas (9): aquéllas que se obtienen analizando la información de las cabeceras TCP/IP de los paquetes como protocolo, estado, bit de urgencia... o características básicas de la conexión como duración, bytes enviados y recibidos...

- Variables de tráfico (19): toman en cuenta información de las cabeceras de conexiones anteriores dentro de una ventana concreta. El tamaño de la ventana puede definirse por tiempo o por número de conexiones.

- Variables de contenido (13): aquéllas que se obtienen examinando el *payload* de algunos servicios concretos. Algunas variables de este tipo son el número de logins fallidos, el número de operaciones generadas como root, la cantidad de ficheros generados... El análisis se hizo mediante reglas generadas por expertos.

El problema de esta base de datos es que no contiene los *payloads* asociados a cada conexión, sólo una serie de variables calculadas a partir de ellos (variables de contenido). Para poder realizar el análisis del *payload* propuesto en este capítulo procesamos la base de datos original DARPA98 en base a la descripción publicada por Lee (1999) y utilizando la herramienta *Bro* (Paxson, 1999). De esta manera obtuvimos una base de datos similar a Kddcup99, pero incluyendo los *payloads* asociados a cada conexión (Perona et al., 2008a).

Por otra parte, dado que en la base de datos Kddcup99 dominan las conexiones de unos pocos tipos de ataques (ataques *flood*) y éstos no son adecuados para la detección de anomalías no supervisada, eliminamos todos los ataques de este tipo. Este filtrado no supone ningún inconveniente ya que los ataques *flood* son sencillos de detectar y se pueden filtrar mediante un reducido número de reglas.

Finalmente, dado el gran número de conexiones resultantes hemos filtrado algunas de las conexiones para obtener una base de datos más manejable. Esta reducción es muy común en los trabajos relacionados con Kddcup99 dado su gran tamaño. Sin embargo, dado que hemos filtrado los ataques *flood* el número de ataques del que partimos nosotros es menor que el del Kddcup99 por lo que únicamente hemos eliminado las conexiones normales. El resultado es una base datos con 178.810 conexiones de las cuales 3.937

son ataques de 27 tipos diferentes. Por lo tanto, la proporción de ataques es del 2,2 %. La información correspondiente a los tipos de ataques utilizados para la experimentación y su frecuencia se resume en la Tabla 7.1.

Ataques	Cantidad	Porcentaje
anomaly	9	0,23
dict	879	22,33
dict_simple	1	0,03
eject	11	0,28
ejectfail	1	0,03
ffb	10	0,25
ffb_clear	1	0,03
format	6	0,15
format_clear	1	0,03
format_fail	1	0,03
ftp_write	8	0,20
guest	50	1,27
imap	7	0,18
land	35	0,89
load_clear	1	0,03
loadmodule	8	0,20
multihop	9	0,23
perl_clear	1	0,03
perlmagic	4	0,10
phf	5	0,13
rootkit	29	0,74
spy	2	0,05
syslog	4	0,10
teardrop	1085	27,56
warez	1	0,03
warezclient	1749	44,42
warezmaster	19	0,48
Total ataques	3937	100

Tabla 7.1: Nombres y frecuencias de los ataques que aparecen en la base de datos utilizada para la experimentación.

7.4.2. Metodología experimental

En primer lugar hemos cuantificado el nivel de detección de ataques de las distintas posibilidades para analizar el *payload* descritas en este capítulo. Para ello, hemos utilizado el algoritmo de clustering *Fixed-Width* descrito en el Capítulo 2. Como referencia hemos utilizado el mismo algoritmo junto con la distancia euclidiana para analizar las variables de contenido de la base de datos Kddcup99. De esta forma, podemos comparar nuestras propuestas de análisis genérico de *payload* con un método basado en expertos y centrado en sólo algunos servicios. Por último, hemos utilizado el mismo algoritmo junto con las variables intrínsecas y de tráfico de la base de datos Kddcup99. Este último análisis sirve de referencia como resultado obtenido mediante análisis clásicos de tráfico de red, es decir, análisis de lo que hemos definido como información TCP/IP.

En resumen, hemos realizado siete análisis en base al *payload* y 1 análisis en base a la información TCP/IP de la base de datos de 178.810 conexiones descrita en el apartado anterior. A continuación se enumeran las características de los 8 análisis:

- Variables intrínsecas y de tráfico de Kddcup99 (IT).
- Variables específicas de contenido de Kddcup99 (C).
- Comparación de secuencias de bytes mediante la distancia NCD (NCD).
- Comparación de histogramas mediante la distancia euclidiana (HE).
- Comparación de histogramas mediante la distancia nominal o Manhattan (HM).
- Comparación de histogramas mediante la distancia ordinal (HO).
- Comparación de elementos más frecuentes mediante la distancia Hamming: Most Frequent Nominal (MFN).
- Comparación de elementos más frecuentes mediante la distancia propuesta: Most Frequent Ordinal (MFO).

El número de elementos más frecuentes utilizado para las dos últimas opciones ha sido de 30. Se ha comprobado experimentalmente que es un valor adecuado para un compromiso entre el nivel de detección y el coste computacional. Una vez analizada la capacidad de cada técnica por separado hemos comparado varias combinaciones de ellas utilizando el método descrito en la Sección 7.3.3.

El análisis de los resultados de esta experimentación no es tarea sencilla. Por un lado, el sistema requiere la definición de un umbral que indique la puntuación a partir de la cual las conexiones se consideran normales. Por otro lado, las distintas frecuencias de ataques hacen que los resultados de unos enmascaren los resultados de otros. Por ejemplo, en la base de datos Kddcup99 original los ataques *Neptune* y *Smurf* representan el 98,8 % de los ataques. Un IDS que sólo detectara estos dos ataques lograría una tasa de detección muy alta, pero su bondad es claramente discutible.

Dado que la especificación del umbral depende de la técnica utilizada y varía dependiendo del número de falsos positivos admisibles hemos mostrado unos resultados independientes de este umbral. Modificando el valor del umbral desde el mínimo al máximo podemos obtener distintas tasas de aciertos y errores que representados como falsos positivos y verdaderos positivos generan una curva ROC (Fawcett, 2004). Los resultados mostrados corresponderán al área bajo la curva (AUC) por lo que un nivel de 1 significa que todos los ataques tienen una puntuación menor que todas las conexiones normales. Por contra, un valor de 0,5 implica un resultado comparable a un IDS que asigne las puntuaciones aleatoriamente.

Para evitar los problemas surgidos por el desbalanceo en los tipos de ataques mostraremos los resultados correspondientes a los 27 tipos de ataques de la base de datos. Dado que las curvas ROC básicas se calculan en base a dos clases (positivo/negativo) a la hora de calcular las curvas ROC para un tipo de ataque en particular hemos ignorado el resto de los ataques. Como resumen también mostramos una media de los valores para todos los ataques ponderada por el número de ataques. Finalmente, para conocer la robustez de cada método calculamos el mínimo valor entre todos los ataques ya que un valor mínimo alto nos asegura que dicho método se comporta de manera aceptable para todos los tipos de ataques estudiados.

El valor del parámetro radio del algoritmo *Fixed-Width* se se ha establecido en base a los resultados experimentales. A pesar de que esto nos da un resultado optimista todas las aproximaciones se analizan en igualdad de condiciones.

7.4.3. Resultados

Los resultados obtenidos se muestran en la Tabla 7.2 donde las filas corresponden a distintos tipos de ataques y las columnas a las distintas aproximaciones evaluadas. La segunda columna muestra como referencia los valores de AUC obtenidos para cada tipo de ataque utilizando la información TCP/IP de las conexiones. Finalmente, las 7 últimas columnas muestran los

resultados obtenidos con cada una de las metodologías evaluadas basadas únicamente en el *payload*. Las dos últimas filas muestran el AUC mínimo y la media ponderada del AUC obtenido para cada método.

Ataques	IT	C	NCD	MFN	MFO	HM	HE	HO
anomaly	0,76	1,00	0,88	0,35	0,74	0,98	0,64	1,00
dict	0,76	0,99	0,82	0,64	0,83	0,93	0,92	1,00
dict_simple	0,65	1,00	0,81	0,69	0,83	0,98	0,99	1,00
eject	0,76	0,98	0,82	0,80	0,80	0,99	0,35	0,98
eject-fail	0,99	0,80	0,48	0,99	0,58	0,95	0,95	0,97
ffb	0,80	0,85	0,88	0,72	0,93	0,95	0,65	0,96
ffb_clear	0,65	1,00	0,81	0,71	0,67	0,99	0,97	0,96
format	0,79	0,75	0,93	0,81	0,95	0,93	0,82	0,96
format_clear	0,52	1,00	0,81	0,88	0,83	0,99	0,21	0,95
format-fail	0,98	1,00	0,81	0,80	0,67	0,99	0,75	0,95
ftp-write	0,88	0,73	0,88	0,76	0,56	0,80	0,87	0,90
guest	0,77	1,00	0,85	0,81	0,83	0,93	0,92	0,89
imap	0,90	0,80	0,97	0,97	0,68	0,97	0,86	0,89
land	0,92	0,80	0,48	0,99	0,58	0,95	0,95	0,88
load_clear	0,65	1,00	0,81	0,12	0,14	0,93	0,99	0,87
loadmodule	0,70	0,84	0,71	0,69	0,68	0,97	0,77	0,87
multihop	0,72	0,74	0,78	0,63	0,71	0,93	0,94	0,84
perl_clear	0,95	1,00	0,81	0,52	0,87	0,99	0,99	0,81
perlmagic	0,66	1,00	0,83	0,86	0,86	0,99	0,99	0,77
phf	0,90	0,50	0,71	0,99	0,72	0,98	0,98	0,77
rootkit	0,88	0,81	0,77	0,86	0,77	0,94	0,96	0,76
spy	0,71	0,80	0,81	0,66	0,52	0,99	1,00	0,76
syslog	0,82	0,80	0,48	0,97	0,58	1,00	0,90	0,75
teardrop	0,82	0,65	0,48	0,76	0,58	0,89	0,48	0,69
warez	0,96	0,31	1,00	0,12	0,85	0,93	0,97	0,69
warezclient	0,81	0,68	0,86	0,86	0,86	0,94	0,92	0,69
warezmaster	0,94	0,75	0,87	0,96	0,88	0,82	0,87	0,69
Mínimo	0,52	0,31	0,48	0,12	0,14	0,80	0,21	0,69
Media	0,85	0,80	0,74	0,63	0,77	0,93	0,92	0,85

Tabla 7.2: Nivel de detección de ataques para los 8 métodos analizados.

La primera conclusión que se puede obtener de estos resultados es que, aunque se utilice un sistema de detección de anomalías sencillo, las seis opciones para modelar el *payload* de una forma general (NCD, MFN, MFN,

HE, HM, HO) son capaces de diferenciar, en menor o mayor medida, entre tráfico normal e intrusiones. Además, las tres opciones que utilizan representaciones basadas en histogramas (HM, HE y HO) son las que han obtenido mayor valor para el AUC medio. Estos tres métodos han obtenido mejores valores medios que el modelo construido a partir de datos obtenidos utilizando información específica de contexto para el procesamiento del *payload* (C) o la información correspondiente a las cabeceras TCP/IP (IT).

Si nos fijamos en la fila en la que mostramos los valores mínimos de AUC dos parecen ser más interesantes: HM y HO. Los valores de AUC obtenidos para cualquier tipo de ataques son de 0,69 o mayores, lo que significa que estas opciones tienen tasas de detección aceptables para todos los tipos de ataques. El resto de las opciones basadas en procesamiento de *payload* obtienen valores mínimos por debajo de 0,5, lo que significa que para algunos tipos de ataques obtienen peores resultados que los que obtendría un clasificador aleatorio.

Otra de las conclusiones que podemos obtener de los resultados es que las distintas opciones propuestas se especializan en la detección de determinados ataques. Como ejemplo podemos observar en la Tabla 7.2 que C y HO obtienen muy buenos resultados para los tres primeros ataques (*anomaly*, *dict* y *dict_simple*) mientras que HM es la mejor opción para *syslog*. Este resultado sugiere que una combinación adecuada de distintas opciones podría llevar a la obtención de un sistema con mayores tasas de detección de ataques.

Para comprobar este hecho hemos experimentado con varias combinaciones. Por un lado, para tener una combinación que sirva de referencia, hemos combinado las opciones IT y C. Esta información corresponde a las variables de la base de datos Kddcup99. Por otro lado, hemos combinado el método IT con uno o varios métodos basados en el análisis genérico del *payload*. Dado que los 6 métodos permiten 63 combinaciones distintas hemos experimentado con sólo algunas de ellas. En la Tabla 7.3 se muestran los resultados correspondientes a la mejor combinación (IT+HM+HO) junto con la combinación de referencia (IT+C). Como se puede observar, los mejores resultados se obtienen combinando las dos opciones que, teniendo en cuenta valores medios y mínimos, han obtenido los mejores resultados individuales.

Como se puede observar en la Tabla 7.3 las combinaciones han contribuido a la mejora de los resultados. Los valores de AUC tanto mínimos como medios han incrementado en las dos combinaciones mostradas. La combinación de IT, HM y HO supera los resultados del método de referencia con un AUC medio de 0,96 y valor mínimo de AUC de 0,89. Estos resultados muestran con claridad que un análisis genérico del *payload* y una detección de

Ataques	IT+C	IT+HM+HO
anomaly	1,00	0,96
dict	0,95	0,95
dict_simple	1,00	0,96
eject	0,98	0,96
eject-fail	1,00	0,96
ffb	0,93	0,95
ffb_clear	1,00	0,95
format	0,89	0,93
format_clear	1,00	0,92
format-fail	1,00	1,00
ftp-write	0,87	0,89
guest	0,94	0,96
imap	0,92	0,94
land	0,94	0,94
load_clear	1,00	0,92
loadmodule	0,87	0,89
multihop	0,83	0,98
perl_clear	1,00	0,94
perlmagic	1,00	0,96
phf	0,88	0,93
rootkit	0,87	0,96
spy	0,86	0,98
syslog	0,85	0,94
teardrop	0,85	0,91
warez	0,98	0,96
warezclient	0,83	0,95
warezmaster	0,96	0,95
min	0,83	0,89
Average	0,91	0,96

Tabla 7.3: Nivel de detección de ataques para combinaciones de clasificadores.

anomalías basada en el clustering pueden proporcionar un nivel de detección altamente satisfactorio.

7.5. Conclusiones

En este capítulo hemos demostrado que un análisis genérico del *payload* de las conexiones correspondientes a tráfico de red puede ser útil para detectar ataques. La gran ventaja de este método es que, a diferencia de los métodos conocidos, no requiere conocimiento alguno de los formatos y protocolos de las distintas aplicaciones de red. Esto lo hace altamente versátil.

Hemos utilizado distintos métodos de análisis de *payload* en un ámbito de detección de anomalías no supervisada. Para ello, hemos utilizado un sencillo algoritmo de clustering y los resultados obtenidos muestran que a pesar de su sencillez se pueden obtener altos niveles de detección de ataques. Además, dado que nos hemos centrado exclusivamente en los ataques potencialmente más peligrosos (los que no son del tipo *flood*) los resultados son muy positivos. Asimismo, un análisis de los resultados realizado ataque por ataque produce un análisis más justo ya que evita que los ataques más frecuentes enmascaren los resultados de los menos frecuentes.

Gracias a este análisis podemos asegurar que dos de los métodos propuestos basados en la comparación de histogramas son adecuados para los 27 tipos de ataques analizados. Además, la combinación de estos dos métodos junto con la información basada en las cabeceras TCP/IP obtiene un AUC medio del 96 % y un mínimo de 89 %. Los resultados basados en la base de datos Kddcup99 original (que analiza el *payload* de forma específica por medio de expertos) son inferiores ya que el AUC medio es del 91 % con un mínimo de 83 %. Teniendo en cuenta que a la aproximación propuesta no se le ha proporcionado ningún tipo de información de expertos y que ignora cualquier tipo de conocimiento sobre formatos o protocolos de aplicación, la mejoría es significativa.

Finalmente, cabe destacar que el sistema propuesto es altamente escalable por lo que su aplicación a entornos reales se considera factible. Por un lado, el método de detección de casos atípicos, el algoritmo de clustering *Fixed-Width*, tiene un coste computacional bajo, $O(KN)$. Por otro lado, el análisis del *payload* en base a histogramas es eficiente ya que construir el histograma de una secuencia de m bytes tiene una complejidad de $O(m)$. El análisis del histograma tendrá un coste computacional constante ya que su complejidad es independiente de la longitud de los *payload* analizados.

El trabajo correspondiente a este capítulo se ha presentado en dos congresos internacionales (Perona et al., 2008b, 2009). Asimismo, un informe interno describe detalladamente el proceso seguido para obtener la base de datos utilizada en la experimentación a partir de la base de datos DARPA98 (Perona et al., 2008a).

Parte IV

Conclusiones

Capítulo 8

Conclusiones

8.1. Conclusiones y líneas futuras de trabajo

El trabajo descrito en esta memoria, a pesar de que surge de dos proyectos de seguridad informática, realiza sus principales aportaciones en el área del clustering. Este hecho puede deberse a las dificultades de fundamentar matemáticamente las bases del clustering (Kettenring, 2006) lo que da un amplio margen para la crítica, el debate y la mejora.

En efecto, la profundización realizada en el ámbito del clustering con el objetivo de resolver los problemas definidos en los proyectos de seguridad informática mencionados ha dejado clara la necesidad de unos criterios claros y unas directrices sólidamente argumentadas para abordar distintos problemas del área. Sin embargo, es evidente que la formalización teórica y matemática precisa del clustering no es tarea sencilla.

Este trabajo supone un grano de arena en el debate acerca de los criterios y directrices que deberían regir algunos de los aspectos del clustering. Asimismo, sirve como acicate para discutir la idoneidad de algunos procedimientos que se han ido estableciendo como procedimientos estándar debido a su uso generalizado. Finalmente, deseamos que el trabajo realizado sirva para provocar la reflexión crítica en el ámbito científico y abrir un debate constructivo acerca de estos temas.

A continuación se describen las conclusiones a las que conducen los distintos capítulos en los que se ha dividido este trabajo junto con sus líneas futuras de trabajo principales. Dado que cada uno de los capítulos finaliza con una sección dedicada a las conclusiones particulares del trabajo descrito en él, las conclusiones mostradas a continuación se limitan a ser un resumen de las más importantes.

El trabajo realizado en torno al algoritmo SIHC, un algoritmo de clustering jerárquico incremental y estable propuesto en el **Capítulo 3**, **muestra que es posible definir un algoritmo de actualización de jerarquías estable con una calidad aceptable**. A pesar de que el algoritmo **SIHC** garantiza que las estructuras de la jerarquía actualizada no sufrirán grandes variaciones, la calidad de las jerarquías se mantiene en unos niveles razonables, mejorando en algún caso los niveles de calidad obtenidos por SAHN.

Sin embargo, en la mayoría de los casos la calidad disminuye de forma sensible con el aumento del número de actualizaciones. Un análisis empírico sugiere que no es conveniente añadir incrementalmente más casos que los que la jerarquía tenía en un principio. El algoritmo *single-linkage* es una excepción a esta regla ya que muestra un nivel de calidad que prácticamente no disminuye con las actualizaciones.

Las características del algoritmo SIHC lo hacen particularmente interesante en entornos reales donde la estabilidad y la explicabilidad cobran una importancia equiparable a la efectividad de las herramientas utilizadas. Además, en la experimentación realizada, el algoritmo SIHC se comporta mejor, en términos relativos, en las bases de datos reales que en las sintéticas.

Cabe destacar que el algoritmo ha sido evaluado mediante una experimentación que combina varias técnicas de validación, incluida una validación externa en base a un nuevo índice propuesto en el mismo capítulo.

Como trabajo futuro se podría intentar modificar el algoritmo con el objetivo de obtener mejores resultados. Sin embargo, esta modificación conllevaría inevitablemente la pérdida de la sencillez del algoritmo, siendo precisamente esta sencillez la que lo hace similar al algoritmo original, SAHN. Parece interesante repetir la experimentación con otros algoritmos, como por ejemplo *Ward*, ya que se ha observado que el resultado obtenido por SIHC depende de este parámetro.

En el **Capítulo 4** se ha mostrado que **el inconveniente detectado al método habitual para extraer una partición de una jerarquía provoca que el resultado no sea satisfactorio en numerosas ocasiones**. Asimismo, se ha mostrado que **el algoritmo propuesto, SEP, junto con el índice de validación de clusters COP es capaz de evitar dicho inconveniente**, consiguiendo un resultado satisfactorio en muchas más ocasiones. Un valor añadido de este capítulo es que puede servir como aliciente para replantear el modo de post-procesar las jerarquías que se ha establecido como estándar. Este debate podría servir para provocar nuevas propuestas alternativas que mejoren la descrita en esta memoria.

Como ya se ha esbozado en el Capítulo 4, el algoritmo SEP podría ser

utilizado para reducir una jerarquía a otra jerarquía menor, con la ventaja de simplificar los resultados sin perder la estructura jerárquica. Por consiguiente, uno de los trabajos futuros en este área será la profundización en esta posible aplicación del algoritmo.

Por otra parte, es importante recalcar que el índice propuesto, COP, representa tan sólo uno de los múltiples índices compatibles con SEP. Es probable que otras propuestas consigan mejorar aún más los resultados. De todos modos, los resultados de la evaluación realizada en el Capítulo 5 sugieren que el comportamiento de COP es comparable a los mejores índices evaluados.

La metodología utilizada para dicha evaluación corresponde, precisamente, a otra de las aportaciones principales de este trabajo. Tras constatar que **la metodología tradicional para evaluar índices de validación de clusters falla en muchos casos**, definimos una metodología alternativa que se describe en el **Capítulo 5**.

Uno de los problemas principales de este capítulo ha sido la validación de la metodología propuesta. Para comparar dos metodologías es muy útil conocer el resultado que obtendría la metodología ideal, de forma que la evaluación se reduciría a observar cuál de las metodologías evaluadas se acerca más a ella. Sin embargo, el resultado ideal de una evaluación de índices de validación de clusters es desconocido, aunque existe algún que otro trabajo comparativo que puede dar algún indicio.

Sin embargo, en el Capítulo 5 el problema de la metodología tradicional queda claramente ilustrado y la bondad de la nueva metodología sólidamente argumentada. Además, los resultados de la experimentación realizada apoyan la hipótesis inicial de la forma más clara posible. Por ello, la conclusión es que **este capítulo da un paso importante hacia la estandarización de la metodología de evaluación de índices de validación de clusters**.

No obstante, ningún procedimiento o metodología es inmejorable, por lo que uno de los objetivos del trabajo correspondiente a este capítulo es, de forma análoga a lo ocurrido en el Capítulo 4 con el algoritmo SEP, provocar una reflexión crítica acerca de la metodología tradicional de evaluación de índices de clusters. Es de esperar que una reflexión de este tipo ayude a mejorar tanto la metodología tradicional como la alternativa propuesta en esta memoria.

El problema principal de la nueva metodología estriba en la necesidad de seleccionar una medida de similitud. Los resultados sugieren que combinar los resultados correspondientes a varias medidas permite obtener unos resultados más robustos. De momento, mientras no se consigan avances más

significativos en el área de la comparación de particiones, esta solución permite aplicar la metodología de forma satisfactoria.

El principal trabajo futuro relacionado con esta línea es obviamente la realización de una extensa evaluación de índices de validación de clusters que sirva como referencia para los investigadores en este área. Para poder considerar la experimentación como una referencia válida y robusta, se requiere la comparación de varias decenas de índices. Las bases de datos deben ser de distinto tipo (número, densidad y forma de los clusters, dimensiones...) y las particiones generadas en base a más de un algoritmo de distinto tipo. Finalmente es indispensable el uso de varias medidas de similitud de distinto tipo.

Un problema importante a tener en cuenta en este tipo de evaluaciones es que muchos índices de validación necesitan la definición de uno o varios parámetros. Los resultados obtenidos dependen de los valores asignados a los parámetros por lo que no resulta sencillo realizar una comparación justa. Por consiguiente, definir métodos automáticos para establecer los valores de dichos parámetros también es un trabajo futuro que puede relacionarse con el trabajo realizado en este capítulo.

En lo que respecta a las aplicación de técnicas de clustering a la seguridad informática, el **Capítulo 6** muestra que **es posible agrupar código malicioso en base a su comportamiento**. A pesar de que analizar el comportamiento del *malware* exige ejecutarlos en entornos controlados, el esfuerzo compensa con creces ya que el análisis estático tiene serias limitaciones.

En este caso la validación del sistema tampoco ha sido sencilla por carecer de un conjunto de datos etiquetado. Sin embargo, los expertos en seguridad de la empresa S21sec han podido ofrecer ayuda en el análisis de algunas jerarquías particulares. Este análisis, junto con la experimentación comparativa realizada, sugiere que una medida de comparación de secuencias como la distancia de compresión normalizada (NCD) puede ser una opción interesante para agrupar código malicioso en base al comportamiento. No obstante, el análisis efectuado demuestra que los resultados obtenidos mediante una representación vectorial de las secuencias puede obtener resultados no muy diferentes con un coste computacional sensiblemente menor.

Por otro lado, los resultados indican que, independientemente de la distancia utilizada para comparar secuencias de eventos correspondientes a la ejecución del *malware*, el algoritmo *average-linkage* es más adecuado que los algoritmos *single-linkage* y *complete-linkage*.

Finalmente, en el **Capítulo 7** se muestra que, a pesar de que la mayoría de los trabajos relacionados parecen sugerir lo contrario, **un análisis**

de los datos transmitidos a través de una red (*payload*), sin tener en cuenta la aplicación que los genera, puede servir para detectar intrusiones. Asimismo, dado que el sistema de detección de intrusos (IDS) propuesto se basa en la aproximación conocida como detección de anomalías no supervisada, el sistema funciona bajo la premisa de que no se conoce ninguna información acerca de la naturaleza de las distintas conexiones de red de la muestra de entrenamiento. Por tanto, consideramos la propuesta como una propuesta aplicable en entornos reales. Asimismo, el bajo coste computacional del algoritmo de clustering utilizado corrobora esta afirmación.

Una de las características novedosas de este trabajo es que se centra únicamente en los ataques que no requieren generar un elevado número de paquetes. Este hecho garantiza que el sistema se ha evaluado frente a ataques que son habitualmente más complejos de detectar ya que los ataques que inundan de paquetes la red se detectan fácilmente. Además, los ataques utilizados en la evaluación son generalmente los más peligrosos ya que suelen desembocar en un acceso ilegítimo al sistema.

A pesar de que en la base de datos utilizada en nuestra experimentación han sido filtrados los ataques de envío masivo de paquetes, sigue existiendo una desproporción evidente en el número de ataques de cada tipo. Por tanto, todos los resultados se han mostrado desglosados por tipo de ataque, de forma que se pueda analizar el comportamiento de los distintos sistemas propuestos para cada uno de los tipos de ataques analizados.

Este análisis de resultados ha servido para comprobar que los distintos sistemas de detección de intrusos se especializan en distintos tipos de ataques. En consecuencia, tal y como se ha demostrado, una combinación adecuada de distintos sistemas consigue mejoras en el nivel de detección.

El análisis del *payload* se ha realizado de diversas formas, pero los resultados han sido claros favoreciendo los métodos basados en comparaciones de histogramas. En concreto dos de los métodos basados en histogramas han conseguido los mejores resultados tanto de manera independiente como de manera combinada. Estos histogramas se han obtenido contando el número de apariciones de cada byte en los *payload* analizados.

Dado que este trabajo pretendía demostrar la viabilidad de un sistema de detección de intrusos con las mencionadas características, el sistema propuesto se basa en algoritmos y procesos sencillos. Por lo tanto, el sistema tiene grandes posibilidades de mejora. Para empezar, el método de detección de casos atípicos se basa en un sencillo algoritmo de clustering. A pesar de que esta sencillez permite un bajo coste computacional sería interesante probar otros algoritmos de detección de casos atípicos. En este sentido estamos

realizado algunas pruebas con árboles de sufijos probabilísticos (*probabilistic suffix trees*).

Asimismo, queda pendiente establecer una forma de definir el parámetro radio del algoritmo de clustering. Por lo tanto, uno de los trabajos futuros es el de tratar de calcular dicho parámetro de forma automática y efectiva. En este sentido estamos realizando una primera aproximación mediante índices de validación de clusters.

Otro aspecto donde existe margen de mejora es en el de la combinación de sistemas de detección de intrusos. En este trabajo hemos presentado un sistema sencillo de promedio de rangos que ha dado resultados satisfactorios, pero es de suponer que soluciones más sofisticadas consigan mejorar los resultados. Por ejemplo, se podrían utilizar métodos de aprendizaje automático para realizar la combinación de manera inteligente, en lo que supondría una fase de meta-aprendizaje.

Por otra parte, queda pendiente evaluar el sistema en un análisis *on-line*. El sistema propuesto se ha evaluado en una base de datos estática, es decir, realizando un análisis forense, pero es sencillo de aplicar en un entorno real ya que el algoritmo de clustering propuesto es inherentemente incremental. La aplicación a un entorno real aportaría solidez a la validación y probablemente pondría de manifiesto posibles debilidades del sistema. En este sentido se han realizado algunos contactos con el Centro de Informática de Docencia, Investigación y Red (CIDIR) de la Universidad del País Vasco/Euskal Herriko Unibertsitatea, pero el acuerdo no es sencillo debido a, principalmente, problemas legales de privacidad.

8.2. Publicaciones asociadas

Como ya se ha ido señalando durante toda la memoria, gran parte del trabajo asociado a esta tesis ha sido, o va a ser, publicado tanto en publicaciones internacionales de prestigio como en informes internos. A continuación se enumeran las referencias completas de dichas publicaciones, clasificadas por tipo.

- Revistas internacionales

  - Ibai Gurrutxaga, Iñaki Albisua, Olatz Arbelaitz, José I. Martín, Javier Muguerza, Jesús M. Pérez, Iñigo Perona (2010). "SEP/COP: An efficient method to find the best partition in hierarchical clustering based on a new cluster validity index". Pattern Recognition. Aceptado para publicación.

  - Ibai Gurrutxaga, Javier Muguerza, Olatz Arbelaitz, Jesús M. Pérez, José I. Martín (2010). "Towards a standard methodology to evaluate internal cluster validity indices". Pattern Recognition Letters. Aceptado para publicación con cambios menores.

- Congresos internacionales

  - I. Gurrutxaga, O. Arbelaitz, J.M. Pérez, J. Muguerza, J.I. Martín, I. Perona (2008). "Evaluation of Malware clustering based on its dynamic behaviour". Proceedings of the 7th Australasian Data Mining Conference (AusDM), Adelaide, Australia, 163–170.
  - I. Perona, I. Gurrutxaga, O. Arbelaitz, J.I. Martín, J. Muguerza, J.M. Pérez (2008). "Service-independent payload analysis to improve intrusion detection in network traffic". Proceedings of the 7th Australasian Data Mining Conference (AusDM), Adelaide, Australia, 171–178.
  - I. Gurrutxaga, O. Arbelaitz, J.I. Martín, J. Muguerza, J.M. Pérez, I. Perona (2009). "SIHC: A Stable Incremental Hierarchical Clustering Algorithm". Proceedings of the 11th International Conference on Enterprise Information Systems (ICEIS), Milán, Italia, 300–304.
  - I. Perona, I. Albisua, O. Arbelaitz, I. Gurrutxaga, J.I. Martín, J. Muguerza, J.M. Pérez (2009). "Histogram based payload processing for unsupervised anomaly detection systems in network intrusion". Proceedings of the 14th Portuguese Conference on Artificial Intelligence (EPIA), Aveiro, Portugal, 329–340.

- Informes internos

  - I. Perona, O. Arbelaitz, I. Gurrutxaga, J.I. Martín, J. Muguerza, J.M. Pérez (2008). "gureKddcup datu-basearen sorrera". Informe interno EHU-KAT-IK-08-08, Universidad del País Vasco/Euskal Herriko Unibertsitatea.

Asimismo, MALBEC, la plataforma de agrupación de *malware* se encuentra en régimen de explotación en la empresa de seguridad informática S21sec.

Bibliografía

Margareta Ackerman y Shai Ben-David. Measures of clustering quality: A working set of axioms for clustering. En *Proceedings of the 22nd Annual Conference on Neural Information Systems Processing*, pág. 121–128, 2008.

Margareta Ackerman y Shai Ben-David. Clusterability: A theoretical study. En *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics (AISTATS)*, vol. 5, pág. 1–8, 2009.

Charu C. Aggarwal, Joel L. Wolf, Philip S. Yu, Cecilia Procopiuc, y Jong Soo Park. Fast algorithms for projected clustering. En *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, pág. 61–72, ACM, Junio 1999.

Rakesh Agrawal y Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. En *Proceedings of 20th International Conference on Very Large Data Bases (VLDB)*, pág. 487–499. Morgan Kaufmann, Septiembre 1994.

Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, y Prabhakar Raghavan. Automatic subspace clustering of high dimensional data. *Data Mining and Knowledge Discovery*, 11(1):5–33, Julio 2005.

Ahmed N. Albatineh, Magdalena Niewiadomska-Bugaj, y Daniel Mihalko. On similarity indices and correction for chance agreement. *Journal of Classification*, 23:301–313, 2006.

Ramiz M. Aliguliyev. Performance evaluation of density-based clustering methods. *Information Sciences*, 179(20):3583–3602, 2009.

J. Allik y R. R. McCrae. Towards a geography of personality traits — pattern of profiles across 36 cultures. *Journal of Cross-Cultural Psychology*, 35: 13–28, 2004.

M. R. Anderberg. *Cluster Analysis for Applications.* Academic Press, Inc., 1973.

Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, y Jörg Sander. Optics: Ordering points to identify the clustering structure. En *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, pág. 49–60, ACM, Junio 1999.

Ayesha Binte Ashfaq, Maria Joseph Robert, Asma Mumtaz, Muhammad Qasim Ali, Ali Sajjad, y Syed Ali Khayam. A comparative evaluation of anomaly detectors under portscan attacks. En *Proceedings of the 11th International Symposium on Recent Advances in Intrusion Detection*, pág. 351–371, 2008.

A. Asuncion y D.J. Newman. UCI machine learning repository, 2007. URL `http://www.ics.uci.edu/~mlearn/MLRepository.html`. Irvine, CA: University of California, School of Information and Computer Science.

Michael Bailey, Jon Oberheide, Jon Andersen, Z. Morley Mao, Farnam Jahanian, y Jose Nazario. Automated classification and analysis of internet malware. Informe interno, Electrical Engineering and Computer Science Department University of Michigan, Abril 2007.

Frank B. Baker y Lawrence J. Hubert. Measuring the power of hierarchical cluster analysis. *Journal of the American Statistical Association*, 70:31–38, 1975.

S. Bandyopadhyay y S. Saha. A point symmetry-based clustering technique for automatic evolution of clusters. *IEEE Transactions on Knowledge and Data Engineering*, 20:1441–1457, 2008.

Daniel Barbará y Sushil Jajodia, editores. *Applications of Data Mining in Computer Security.* Kluwer Academic Publishers, Norwell, Massachusetts, 2002.

Vladimir Batagelj y Matevz Bren. Comparing resemblance measures. *Journal of Classification*, 12(1):73–90, Marzo 1995.

F.B. Baulieu. A classification of presence/absence based dissimilarity coefficients. *Journal of Classification*, 6(1):233–246, 1989.

A. Ben-Hur, A. Elisseeff, y I. Guyon. A stability based method for discovering structure in clustered data. En *Pacific Symposium on Biocomputing*, pág. 6–17. World Scientific, 2002.

James C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Kluwer Academic Publishers, Norwell, MA, 1981.

James C. Bezdek y Nikhil R. Pal. Some new indexes of cluster validity. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, 28(3):301–315, Junio 1998.

James C. Bezdek y Richard J. Hathaway. VAT: a tool for visual assessment of (cluster) tendency. En *Proceedings of the International Joint Conference of Neural Networks*, pág. 2225—-2230, Piscataway, NJ, 2002. IEEE Press.

David M. Blei, Andrew Y. Ng, y Michael I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

Mohamed Bouguessa, Shengrui Wang, y Haojun Sun. An objective approach to cluster validation. *Pattern Recognition Letters*, 27(13):1419–1430, 2006.

Paul S. Bradley, Usama Fayyad, y Cory Reina. Scaling clustering algorithms to large databases. En *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pág. 9–15. AAAI Press, 1998.

T. Brugger. Data mining methods for network intrusion detection. Informe interno, University of California, Davis, 2004.

John Burke, Dan Davison, y Winston Hide. d2_cluster: A validated method for clustering EST and full-length cDNA sequences. *Genome Research*, 9 (11):1135–1142, Noviembre 1999.

T. Calinski y J. Harabasz. A dendrite method for cluster analysis. *Communications in Statistics*, 3:1–27, 1974.

Olivier Chapelle, Bernhard Schölkopf, y Alexander Zien, editores. *Semi-supervised learning*. MIT press, Cambridge, MA, 2006.

Irène Charon, Lucile Denœud, Alain Guénoche, y Olivier Hudry. Maximum transfer distance between partitions. *Journal of Classification*, 23:103–121, 2006.

Chien-Hsing Chou, Mu-Chun Su, y Eugene Lai. A new cluster validity measure and its application to image compression. *Pattern Analysis and Applications*, 7(2):205–220, Julio 2004.

Mihai Christodorescu, Somesh Jha, y Christopher Kruegel. Mining specifications of malicious behavior. En *Proceedings of the 6th Joint Meeting of*

the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE), pág. 5–14, 2007.

Nathalie Dagorn. WebIDS: A cooperative bayesian anomaly-based intrusion detection system for web applications. En *Proceedings of the 11th International Symposium on Recent Advances in Intrusion Detection*, 2008.

Sajib Dasgupta y Vincent Ng. Topic-wise, sentiment-wise, or otherwise?: Identifying the hidden dimension for unsupervised text classification. En *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pág. 580–589, 2009.

D. L. Davies y D. W. Bouldin. A clustering separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1:224–227, 1979.

William H. E. Day y Herbert Edelsbrunner. Efficient algorithms for agglomerative hierarchical clustering methods. *Journal of Classification*, 1: 7–24, 1984.

H. Debar y A. Wespi. Aggregration and correlation of intrusion-detection alerts. En *Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection (RAID)*, pág. 85–103, 2001.

A. P. Dempster, N. M. Laird, y D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39:1–38, 1977.

Dorothy E. Denning. An intrusion-detection model. *IEEE Transactions on Software Engineering*, 13(2):222–232, Febrero 1987.

Evgenia Dimitriadou, Sara Dolňicar, y Andreas Weingessel. An examination of indexes for determining the number of clusters in binary data sets. *Psychometrika*, 67(1):137–159, Marzo 2002.

Byron E. Dom. An information-theoretic external cluster-validity measure. Informe interno RJ 10219, IBM, 2001.

J. C. Dunn. A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*, 3:32–57, 1973.

Yasser El-Sonbaty y M.A. Ismail. On-line hierarchichal clustering. *Pattern Recognition Letters*, 19:1285–1291, 1998.

Eleazar Eskin, Andrew Arnold, Michael Prerau, Leonid Portnoy, y Salvatore Stolfo. *Applications of Data Mining in Computer Security*, capítulo A Geometric Framework for Unsupervised Anomaly Detection: Detecting Intrusions in Unlabelled Data. Kluwer Academic Publishers, 2002.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, y Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases. En *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, pág. 226–231, Agosto 1996.

M. Falasconi, A. Gutierrez, M. Pardo, G. Sberveglieri, y S. Marco. A stability based validity method for fuzzy clustering. *Pattern Recognition*, 43(4): 1292–1305, 2009.

Tom Fawcett. ROC graphs: Notes and practical considerations for researchers. Informe interno HPL-2003-4, HP Labs., Palo Alto, CA, 2004.

Douglas. H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine learning*, 2:139–172, 1987.

Lloyd Fisher y John W. Van Ness. Admissible clustering procedures. *Biometrika*, 58:91–104, 1971.

E. B. Fowlkes y C. L. Mallows. A method for comparing two hierarchical clusterings. *Journal of the American Statistical Association*, 78(383):553–569, 1983.

Ana L. N. Fred y José M. N. Leitão. A new cluster isolation criterion based on dissimilarity increments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(8):944–958, 2003.

John F. Gantz. The diverse and exploding digital universe, Marzo 2008. URL `http://www.emc.com/collateral/analyst-reports/diverse-exploding-digital-universe.pdf`.

Debin Gao, Michael K. Reiter, y Dawn Xiaodong Song. Behavioral distance for intrusion detection. En *Recent Advances in Intrusion Detection, 8th International Symposium (RAID)*, pág. 63–81, 2005.

Olivier Gascuel y Andy McKenzie. Performance analysis of hierarchical clustering algorithms. *Journal of Classification*, 21(1):3–18, Marzo 2004.

X. Gong y M. B. Richman. On the application of cluster analysis to growing season precipitation data in North America east of the rockies. *Journal of climate*, 8(4):897–931, 1995.

Simon Günter y Horst Bunke. Validation indices for graph clustering. *Pattern Recognition Letters*, 24(8):1107–1113, Mayo 2003.

Valerie Guralnik y George Karypis. A scalable algorithm for clustering sequential data. En *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pág. 179–186, 2001.

Ibai Gurrutxaga, Olatz Arbelaitz, Jesús M. Pérez, Javier Muguerza, José I. Martín, e Iñigo Perona. Evaluation of malware clustering based on its dynamic behaviour. En *Proceedings of the 7th Australasian Data Mining Conference (AusDM)*, pág. 163–170, 2008.

Ibai Gurrutxaga, Olatz Arbelaitz, José I. Martín, Javier Muguerza, Jesús M. Pérez, e Iñigo Perona. SIHC: A stable incremental hierarchical clustering algorithm. En *Proceedings of the 11th International Conference on Enterprise Information Systems (ICEIS)*, pág. 300–304, 2009.

Ibai Gurrutxaga, Iñaki Albisua, Olatz Arbelaitz, José I. Martín, Javier Muguerza, Jesús M. Pérez, e Iñigo Perona. SEP/COP: An efficient method to find the best partition in hierarchical clustering based on a new cluster validity index. *Pattern Recognition*, Aceptado para publicación, 2010a.

Ibai Gurrutxaga, Javier Muguerza, Olatz Arbelaitz, Jesús M. Pérez, y José I. Martín. Towards a standard methodology to evaluate internal cluster validity indices. *Pattern Recognition Letters*, Aceptado para publicación con cambios menores, 2010b.

Dan Gusfield. *Algorithms on Strings, Trees, and Sequences: computer science and computational biology*. Cambridge Unibersity Press, Cambridge, 1997.

Lars Hagen y Andrew B. Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 11(9):1074–1085, 1992.

Maria Halkidi, Yannis Batistakis, y Michalis Vazirgiannis. On clustering validation techniques. *Journal of Intelligent Information Systems*, 17: 107–145, 2001.

M. E. Hall. Pottery production during tha Late Jomon period: Insights from the chemical analyses of Kasori B pottery. *Journal of Archaeological Science*, 31(10):1439–1450, 2004.

Jiawei Han y Micheline Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, segunda edición, 2006.

Julia Handl y Joshua Knowles. Evolutionary multiobjective clustering. En *Parallel Problem Solving from Nature - PPSN VIII*, vol. 3242 of *Lecture Notes in Computer Science*, pág. 1081–1091, 2004.

André Hardy. On the number of clusters. *Computational Statistics & Data Analysis*, 23(1):83–96, Noviembre 1996.

Richard J. Hathaway, James C. Bezdek, y Jacalyn M. Huband. Scalable visual assessment of cluster tendency for large data sets. *Pattern Recognition*, 39:1315–1324, 2006.

José Hernández, M. José Ramírez, y César Ferri. *Introducción a la Minería de Datos*. Pearson Educación, 2004.

A. Hinneburg y D. A. Keim. An efficient approach to clustering in large multimedia databases with noise. En *Proceedings of the 4th International Conference on Knowledge Discovery and Datamining (KDD)*, pág. 58–65, Septiembre 1998.

Victoria Hodge y Jim Austin. A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22(2):85–126, 2004.

Karl J. Holzinger y Harry H. Harman. *Factor Analysis*. University of Chicago Press, Chicago, 1941.

J.M. Huband, J.C. Bezdek, y R.J. Hathaway. Revised visual assessment of (cluster) tendency (reVAT). En *Proceedings of the North American Fuzzy Information Processing Society (NAFIPS)*, pág. 101–104, 2004.

J.M. Huband, J.C. Bezdek, y R.J. Hathaway. bigVAT: Visual assessment of cluster tendency for large data dets. *Pattern Recognition*, 38:1875–1886, 2005.

L. Hubert y P. Arabie. Comparing partitions. *Journal of Classification*, 2: 193–218, 1985.

L. J. Hubert y J. R. Levin. A general statistical framework for assessing categorical clustering in free recall. *Psychological Bulletin*, 83:1072–1080, 1976.

P. Jaccard. Nouvelles recherches sur la distribution florale. *Bulletin de la Societé Vaudoise de Sciences Naturelles*, 44:223–370, 1908.

V. Jacobson, C. Leres, y S. McCanne. *The TCP Dump Manual Page*. University of California, Berkeley, 1989. URL `http://ee.lbl.gov/`.

Anil K. Jain. Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, 31(8):651–666, 2009.

Anil K. Jain y Richard C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.

R. John, D. S. Kerby, y C. H. Hennessy. Patterns and impact of comorbidity and multimorbidity among community-resident american indian elders. *The Gerontologist*, 43(5):649–660, 2003.

K. Julisch. Clustering intrusion detection alarms to support root cause analysis. *ACM Transactions on Information and System Security*, 6:443–471, 2003.

Karin Kailing, Hans-Peter Kriegel, y Peer Kröger. Density-connected subspace clustering for high-dimensional data. En *Proceedings of the Fourth SIAM International Conference on Data Mining*, pág. 246–257, 2004.

L. Kaufman y P. J. Rouseew. *Finding Groups in Data. An Introduction to Cluster Analysis*. John Wiley & Sons, Inc., 1990.

Jon R. Kettenring. The practice of cluster analysis. *Journal of Classification*, 23:3–30, 2006.

Shehroz S. Khan y Amir Ahmad. Cluster center initialization algorithm for K-means clustering. *Pattern Recognition Letters*, 25(11):1293–1302, Agosto 2004.

C. Kim, H. Shin, y H. Choi. A phenetic analysis of *Typha* in Korea and far east Russia. *Aquatic Botany*, 75:33–43, 2003.

Minho Kim y R. S. Ramakrishna. New indices for cluster validity assessment. *Pattern Recognition Letters*, 26(15):2353–2363, Noviembre 2005.

Jon Kleinberg. An impossibility theorem for clustering. En *Advances in Neural Information Processing Systems 15*, pág. 446–453. MIT Press, Cambridge, MA, 2002.

Christopher Krügel, Thomas Toth, y Engin Kirda. Service specific anomaly detection for network intrusion detection. En *Proceedings of the 2002 ACM symposium on Applied computing*, pág. 201–208, 2002.

Luis F. Lago-Fernández y Fernando Corbacho. Normality-based validation for crisp clustering. *Pattern Recognition*, 43:782–795, 2010.

G. N. Lance y W. T. Williams. A general theory of classificatory sorting strategies: II. Clustering systems. *Computer Journal*, 10:271–277, 1967.

Tilman Lange, Volker Roth, Mikio L. Braun, y Joachim M. Buhmann. Stability-based validation of clustering solutions. *Neural Computation*, 16:1299–1323, 2004.

M. Laszlo y S. Mukherjee. A genetic algorithm that exchanges neighboring centers for K-means clustering. *Pattern Recognition Letters*, 28:2359–2366, 2007.

Tony Lee y Jigar J. Mody. Behavioral classification. En *Proceedings of EICAR Conference*, Abril 2006.

Wenke Lee. *A Data Mining Framework for Constructing Features and Models for Intrusion Detection Systems*. Tesis doctoral, Columbia University, 1999.

Wenke Lee, Salvatore J. Stolfo, y Kui W. Mok. Mining in a data-flow environment: Experience in network intrusion detection. En *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pág. 114–124, 1999a.

Wenke Lee, Salvatore J. Stolfo, y Kui W. Mok. A data mining framework for building intrusion detection models. En *Proceedings of the IEEE Symposium on Security and Privacy*, 1999b.

Kingsly Leung y Christopher Leckie. Unsupervised anomaly detection in network intrusion detection using clusters. En *Twenty-Eighth Australasian Computer Science Conference (ACSC)*, vol. 38, pág. 333–342, 2005.

E. Levine y E. Domany. Resampling method for unsupervised estimation of cluster validity. *Neural Computation*, 13:2573–2593, 2001.

Ming Li, Xin Chen, Xin Li, Bin Ma, y Paul M. B. Vitányi. The similarity metric. *IEEE Transactions on Information Theory*, 50(12):3250–3264, 2004.

Richard Lippmann, Joshua W. Haines, David J. Fried, Jonathan Korba, y Kumar Das. The 1999 DARPA off-line intrusion detection evaluation. *Computer Networks*, 34(4):579–595, 2000.

Bing Liu. *Web Data Mining. Exploring Hyperlinks, Contents, and Usage Data*. Springer, 2007.

Huan Liu y Hiroshi Motoda. *Feature Selection for Knowledge Discovery and Data Mining*. Kluwer Academic Publishers, 1998.

Aiyesha Ma y Ishwar K. Sethi. Distributed K-median clustering with application to image clustering. En *Proceedings of the 7th International Workshop on Pattern Recognition in Information Systems*, pág. 215–220, 2007.

Y. Maarek y A. Wecker. The librarian's assistant: Automatically assembling books into dynamic bookshelves. En *Proocedings of RIAO'94, Intelligent Multimedia, Information Retrieval Systems and Management*, 1994.

Kalyani Mali y Sushmita Mitra. Clustering and its validation in a symbolic framework. *Pattern Recognition Letters*, 24(14):2367–2376, Octubre 2003.

Marcus A. Maloof, editor. *Machine Learning and Data Mining for Computer Security*. Springer-Verlag, London, 2006.

Francesco M. Malvestuto. Statistical treatment of the information content of a database. *Information Systems*, 11(3):211–223, 1986.

Chris Manning y Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, Mayo 1999.

Jianchang Mao y Anil K. Jain. A self-organizing network for hyperellipsoidal clustering (HEC). *IEEE Transactions on Neural Networks*, 7:16–29, Enero 1996.

Louis Massey. Determination of clustering tendency with art neural networks. En *Proceedings of 4th International Conference on Recent Advances in Soft Computing*, 2002.

Ujjwal Maulik y Sanghamitra Bandyopadhyay. Performance evaluation of some clustering algorithms and validity indices. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(12):1650–1654, Diciembre 2002.

J. O. McClain y V. R. Rao. CLUSTISZ: A program to test for the quality of clustering of a set of objects. *Journal of marketing research*, 12:456–460, 1975.

Marina Meilă. Comparing clusterings by the variation of information. En *Proceedings of the Sixteenth Annual Conference ofn Computational Learning Theory (COLT)*, pág. 173–187, 2003.

Marina Meilă. Comparing clusterings – an axiomatic view. En *Proceedings of the Twenty-Second International Conference on Machine Learning (ICML)*, pág. 577–584, 2005.

Marina Meilă. The uniqueness of a good optimum for K-means. En *Proceedings of the 23rd international conference on Machine learning*, pág. 625–632, Pittsburgh, PA, 2006.

Marina Meilă y Jianbo Shi. A random walks view of spectral segmentation. En *Proceedings of the 8th International Workshop on Artificial Intelligence and Statistics*, 2001.

Glenn W. Milligan y Martha C. Cooper. An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 50(2): 159–179, Junio 1985.

Leslie C. Morey y Alan Agresti. The measurement of classification agreement: An adjustment to the Rand statistic for chance agreement. *Educational and Psychological Measurement*, 44:33–37, 1984.

Hans-Joachim Mucha. On validation of hierarchical clustering. En *Advances in Data Analysis. Proceedings of the 30th Annual Conference of the Gesellschaft für Klassifikation*, pág. 115–122, Marzo 2006.

Fionn Murtagh. The remarkable simplicity of very high dimensional data: Application of model-based clustering. *Journal of Classification*, 26:249–277, 2009.

Andrew Y. Ng, Michael I. Jordan, y Yair Weiss. On spectral clustering: Analysis and an algorithm. En *Advances in Neural Information Processing Systems 14*, pág. 849–856, 2002.

Clark F. Olson. Parallel algorithms for hierarchical clustering. *Parallel Computing*, 21(8):1313–1325, 1995.

Alberto Paccanaro, James A. Casbon, y Mansoor A. S. Saqi. Spectral clustering of protein sequences. *Nucleic Acid Research*, 34(5):1571–1580, 2006.

Nikhil R. Pal y J. Biswas. Cluster validation using graph theoretic concepts. *Pattern Recognition*, 30(6):847–857, Junio 1997.

Vern Paxson. Bro: a system for detecting network intruders in real-time. *Computer Networks*, 31(23–24):2435–2463, Diciembre 1999.

Dan Pelleg y Andrew Moore. Accelerating exact K-means algorithms with geometric reasoning. En *Proceedings of the Fifth International Conference on Knowledge Discovery in Databases*, pág. 277–281. AAAI Press, Agosto 1999.

Dan Pelleg y Andrew Moore. X-means: Extending K-means with efficient estimation of the number of clusters. En *Proceedings of the Seventeenth International Conference on Machine Learning*, pág. 727–734, Morgan Kaufmann, 2000.

Jesús M. Pérez, Javier Muguerza, Olatz Arbelaitz, Ibai Gurrutxaga, y José I. Martín. Combining multiple class distribution modified subsamples in a single tree. *Pattern Recognition Letters*, 28(4):414–422, 2007.

Iñigo Perona, Olatz Arbelaitz, Ibai Gurrutxaga, José I. Martín, J. Muguerza, y Jesús M. Pérez gurekddcup datu-basearen sorrera. Informe interno EHU-KAT-IK-08-08, The University of the Basque Country, 2008a.

Iñigo Perona, Ibai Gurrutxaga, Olatz Arbelaitz, José I. Martín, Javier Muguerza, y Jesús M. Pérez. Service-independent payload analysis to improve intrusion detection in network traffic. En *Proceedings of the 7th Australasian Data Mining Conference (AusDM)*, pág. 171–178, 2008b.

Iñigo Perona, Iñaki Albisua, Olatz Arbelaitz, Ibai Gurrutxaga, José I. Martín, Javier Muguerza, y Jesús M. Pérez. Histogram based payload processing for unsupervised anomaly detection systems in network intrusion. En *Proceedings of the 14th Portuguese Conference on Artificial Intelligence (EPIA)*, pág. 329–340, 2009.

Darius Pfitzner, Richard Leibbrandt, y David Powers. Characterization and evaluation of similarity measures for pairs of clusterings. *Knowledge and Information Systems*, 19(3):361–394, Junio 2009.

Paul J. Planet. Tree disagreement: Measuring and testing incongruence in phylogenies. *Journal of Biomedical Informatics*, 39:86–102, 2006.

János Podani. Simulation of random dendrograms and comparison tests: Some comments. *Journal of Classification*, 17:123–142, 2000.

Phillip A. Porras, Martin W. Fong, y Alfonso Valdes A mission-impact-based approach to INFOSEC alarm correlation. En *Proceedings of the 5th International Symposium on Recent Advances in Intrusion Detection (RAID)*, pág. 95–114, 2002.

Leonid Portnoy, Eleazar Eskin, y Sal Stolfo. Intrusion detection with unlabeled data using clustering. En *Proceedings of ACM CSS Workshop on Data Mining Applied to Security*, 2001.

Christian Posse. Hierarchical model-based clustering for large datasets. *Journal of Computational and Graphical Statistics*, 10(3):464–486, Septiembre 2001.

William M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of American Statistical Association*, 66:846–850, 1971.

Martin Rehák, Michal Pechoucek, Karel Bartos, Martin Grill, Pavel Celeda, y Vojtech Krmicek. Improving anomaly detection error rate by collective trust modeling. En *Proceedings of the 11th International Symposium on Recent Advances in Intrusion Detection*, 2008.

Arnaud Ribert, Abdel Ennaji, y Yves Lecourtier. An incremental hierarchical clustering. En *Vision Interface '99*, pág. 586–591, Mayo 1999.

Stephen J. Roberts, Christopher Holmes, y Dave Denison. Minimum-entropy data clustering using reversible jump Markov chain Monte Carlo. En *Artificial Neural Networks — ICANN 2001*, vol. 2130 of *Lecture Notes in Computer Science*. Springer, 2001.

F. James Rohlf. Methods of comparing classifications. *Annual Review of Ecology and Systematics*, 5:101–113, 1974.

Sudip Seal, Srikanth Komarina, y Srinivas Aluru. An optimal hierarchical clustering algorithm for gene expression data. *Information Processing Letters*, 93(3):143–147, Febrero 2005.

Francesc Serratosa y Alberto Sanfeliu. Signatures versus histograms: Definitions, distances and algorithms. *Pattern Recognition*, 39(5):921–934, 2006.

Gholamhosein Sheikholeslami, Surojit Chatterjee, y Aidong Zhang. Wavecluster: a wavelet-based clustering approach for spatial data in very large databases. *The VLDB Journal*, 8(3-4):289–304, 2000.

Jianbo Shi y Malik Jitendra. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8): 888–905, Agosto 2000.

Helena B. Silva, Paul Brito, y Joaquim P. da Costa. A partitional clustering algorithm validated by a clustering tendency index based on graph theory. *Pattern Recognition*, 39(5):776–788, Mayo 2006.

Noam Slonim y Naftali Tishby. Document clustering using word clusters via the information bottleneck method. En *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pág. 208–215, ACM, 2000.

Peter H. A. Sneath y Robert R. Sokal. *Numerical Taxonomy*. Books in biology. W. H. Freeman and Company, San Francisco, 1973.

T. Stefurak, G. G. Calhoun, y B. A. Glaser. Personality typologies of male juvenile offenders using a cluster analysis of the millon adolescent clinical inventory introduction. *International Journal of Offender Therapy and Comparative Criminology*, 48:96–110, 2004.

Alexander Strehl y Joydeep Ghosh. Cluster ensembles — a knowledge reuse framework for combining multiple partitions. *The Journal of Machine Learning Research*, 3:583–617, Marzo 2003.

V. V. Strelkov. A new similarity measure for histogram comparison and its application in time series analysis. *Pattern Recognition Letters*, 29(13): 1768–1774, 2008.

Ting Su y Jennifer Dy. A deterministic method for initializing K-means clustering. En *Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence*, pág. 784–786, 2004.

R. Tibshirani, G. Walther, y T. Hastie. Cluster validation by prediction strength. Informe interno, Statistic Department, Stanford University, Stanford, CA, 2001a.

Robert Tibshirani, Guenther Walther, y Trevor Hastie. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society, B*, 63(2):411–423, 2001b.

Naftali Tishby, Fernando C. Pereira, y William Bialek. The information bottleneck method. En *Proceedings of the 37th Allerton Conference on Communication, Control and Computing*, pág. 368–377, 1999.

Peter D. Turney. Technical note: Bias and the quantification of stability. *Machine Learning*, 20:23–33, 1995.

P. A. Vijaya, M. Narasimha Murty, y Devika K. Subramanian. Leaders-Subleaders: an efficient hierarchical clustering algorithm for large data sets. *Pattern Recognition Letters*, 25(4):505–513, 2004.

P. A. Vijaya, M. Narasimha Murty, y D. K. Subramanian. Efficient bottom-up hybrid hierarchical clustering techniques for protein sequence classification. *Pattern Recognition*, 39(12):2344–2355, Diciembre 2006.

Ke Wang y Salvatore Stolfo. Anomalous payload-based network intrusion detection. En *Proceedings of the 7th International Symposium on Recent Advances in Intrusion Detection*, 2004.

Wei Wang, Jiong Yang, y Richard R. Muntz. STING: A statistical information grid approach to spatial data mining. En *Proceedings of the 23rd International Conference on Very Large Data Bases*, pág. 186–195, Morgan Kaufmann, 1997.

Christina Warrender, Stephanie Forrest, y Barak Pearlmutter. Detecting intrusions using system calls: Alternative data models. En *Proceedings of the 1999 IEEE Symposium on Security and Privacy*, 1999.

Matthijs J. Warrens. On the equivalence of Cohen's Kappa and the Hubert-Arabie Adjusted Rand index. *Journal of Classification*, 25(2):177–183, Noviembre 2008a.

Matthijs J. Warrens. Bounds of resemblance measures for binary (presence/absence) variables. *Journal of Classification*, 25(2):195–208, Noviembre 2008b.

Stephanie Wehner. Analyzing worms and network traffic using compression. *Journal of Computer Security*, 15(3):303–320, 2007.

Dwi H. Widyantoro, Thomas R. Ioerger, y John Yen. An incremental approach to building a cluster hierarchy. En *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pág. 705–708. IEEE Computer Society, 2002.

W. T. Williams y G. N. Lance. *Statistical Methods for Digital Computers*, capítulo Hierarchical Classificatory Methods, pág. 269–295. John Wiley & Sons, Inc., New York, NY, 1977.

Kuo-Lung Wu, Miin-Shen Yang, y June-Nan Hsieh. Robust cluster validity indexes. *Pattern Recognition*, 42(11):2541–2550, Marzo 2009.

Xindong Wu, Kumar Vipin, J. Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J. McLachlan, Angus Ng, Bing Liu, Philip S. Yu, Zhi-Hua Zhou, Michael Steinbach, David J. Hand, y Dan Steinberg. Top 10 algorithms in data mining. *Knowledge and Information Systems*, 14:1–37, 2008.

D. Yeung y D. Yuxin. Host-based intrusion detection using dynamic and static behavioral models. *Pattern Recognition*, 36:229–243, 2003.

K. Y. Yeung, D. R. Haynor, y W. L. Ruzzo. Validating clustering for gene expression data. *Bioinformatics*, 17(4):309–318, 2001.

Stella Yu y Jianbo Shi. Multiclass spectral clustering. En *Proceedings of the International Conference on Computer Vision*, pág. 313–319, 2003.

Stefano Zanero y Sergio M. Savaresi. Unsupervised learning techniques for an intrusion detection system. En *Proceedings of the 2004 ACM Symposium on Applied Computing*, pág. 412–419, 2004.

Wei Zhong, G. Altun, R. Harrison, P.C. Tai, y Yi Pan. Improved K-means clustering algorithm for exploring local protein sequence motifs representing common structural property. *NanoBioscience, IEEE Transactions on*, 4(3):255–265, Septiembre 2005.

Apéndice A

Resultados adicionales de la evaluación de índices de validación de clusters

En este apéndice se muestran unos resultados adicionales a la evaluación de índices de validación de clusters del Capítulo 5. En concreto las siguientes tablas muestran los resultados correspondientes a las medidas de similitud Rand (Tabla A.1), Adjusted Rand (Tabla A.2), Jaccard (Tabla A.3) y Fowlkes-Mallows (Tabla A.4). Finalmente, la Tabla A.5 muestra los resultados medios de las 5 medidas de similitud utilizadas en este trabajo.

	CH	C-I	Gam.	G(+)	DB	McC-R	Dunn	COP	Tot.
2x2	10	10	10	9	8	0	8	8	63
Ruido	9	9	10	8	5	0	5	7	53
Dens.	4	8	10	8	3	0	5	7	45
Tam.	10	10	10	10	0	0	10	8	58
3x3	10	9	10	10	4	0	2	4	49
Arcos	0	6	7	5	6	4	3	0	31
T&U	0	0	0	0	0	0	0	0	0
Iris	6	0	0	0	0	0	0	0	6
Glass	0	0	0	5	0	4	0	0	9
Ecoli	0	4	4	0	0	0	4	5	17
Total	49	56	61	55	26	8	37	39	331

Tabla A.1: Número de aciertos de cada CVI siguiendo el criterio de la nueva metodología propuesta y según la medida de similitud Rand.

	CH	C-I	Gam.	G(+)	DB	McC-R	Dunn	COP	Tot.
2x2	10	10	10	9	8	0	8	10	65
Ruido	9	9	10	8	5	0	5	9	55
Dens.	4	8	10	8	3	0	5	9	47
Tam.	10	10	10	10	0	0	10	10	60
3x3	9	10	9	9	4	0	3	6	50
Arcos	6	1	1	1	1	1	1	2	14
T&U	0	0	0	0	0	0	0	0	0
Iris	6	0	0	0	0	0	0	0	6
Glass	2	6	6	0	2	1	4	2	23
Ecoli	0	5	4	0	0	0	5	4	18
Total	56	59	60	45	23	2	41	52	338

Tabla A.2: Número de aciertos de cada CVI siguiendo el criterio de la nueva metodología propuesta y según la medida de similitud Adjusted Rand.

	CH	C-I	Gam.	G(+)	DB	McC-R	Dunn	COP	Tot.
2x2	10	10	10	9	8	0	8	10	65
Ruido	9	9	10	8	5	0	5	9	55
Dens.	4	8	10	8	3	0	5	9	47
Tam.	10	10	10	10	0	0	10	10	60
3x3	9	10	9	9	4	0	3	6	50
Arcos	9	0	0	0	0	0	1	3	13
T&U	0	0	0	0	0	0	0	0	0
Iris	6	0	0	0	0	0	0	0	6
Glass	2	10	10	0	2	0	8	4	36
Ecoli	0	6	3	0	0	0	6	5	20
Total	56	59	60	45	23	2	41	56	342

Tabla A.3: Número de aciertos de cada CVI siguiendo el criterio de la nueva metodología propuesta y según la medida de similitud Jaccard.

	CH	C-I	Gam.	G(+)	DB	McC-R	Dunn	COP	Tot.
2x2	10	10	10	9	8	0	8	10	65
Ruido	9	9	10	8	5	0	5	9	55
Dens.	4	8	10	8	3	0	5	9	47
Tam.	10	10	10	10	0	0	10	10	60
3x3	9	10	9	9	4	0	3	6	50
Arcos	9	0	0	0	0	0	1	3	13
T&U	0	0	0	0	0	0	0	0	0
Iris	6	0	0	0	0	0	0	0	6
Glass	3	9	9	0	3	0	7	5	36
Ecoli	0	5	1	0	0	0	8	4	18
Total	60	61	59	44	23	0	47	56	350

Tabla A.4: Número de aciertos de cada CVI siguiendo el criterio de la nueva metodología propuesta y según la medida de similitud Fowlkes-Mallows.

	CH	C-I	Gam.	G(+)	DB	McC-R	Dunn	COP	Tot.
2x2	10	10	10	9	8	0	8	10	65
Ruido	9	9	10	8	5	0	5	9	55
Dens.	4	8	10	8	3	0	5	9	47
Tam.	10	10	10	10	0	0	10	10	60
3x3	9	9	9	9	4	0	3	5	48
Arcos	6	1	2	1	1	1	1	2	16
T&U	0	0	0	0	0	0	0	0	0
Iris	6	2	2	0	2	0	2	2	15
Glass	2	6	6	1	2	1	5	3	28
Ecoli	0	5	3	0	0	0	6	4	18
Total	56	61	61	46	26	2	46	54	352

Tabla A.5: Número de aciertos medio de cada CVI según las 5 medidas de similitud utilizadas: VI, Rand, Adjusted Rand, Jaccard y Fowlkes-Mallows.