UNIVERSIDAD TECNOLÓGICA NACIONAL

Facultad Regional Santa Fe

Doctorado en Ingeniería

Mención Sistemas de Información

Tesis doctoral

«Ensamble de agrupamientos con aplicaciones en bioinformática»

Ing. Milton Pividori

Directora: Dra. Georgina Stegmayer

Codirector: Dr. Diego H. Milone

Santa Fe de la Vera Cruz, Argentina. Septiembre de 2016

Pividori, Milton Damián
Ensamble de agrupamientos con aplicaciones en bioinformática / Milton Damián Pividori. - 1a ed . - Avellaneda : Milton Damián Pividori, 2016.
Libro digital, PDF

Archivo Digital: descarga
ISBN 978-987-42-2289-3

1. Tesis Doctorales. 2. Minería de Datos. I. Título.
CDD 001.9

Fecha de catalogación: 05/10/2016

UNIVERSIDAD TECNOLÓGICA NACIONAL

Facultad Regional Santa Fe

Comisión de Posgrado

Se presenta esta tesis en cumplimiento de los requisitos exigidos por la Universidad Tecnológica Nacional para la obtención del grado académico de Doctor en Ingeniería, mención Sistemas de Información.

«Ensamble de agrupamientos con aplicaciones en bioinformática»

por

Ing. Milton Pividori

Directora: Dra. Georgina Stegmayer

Codirector: Dr. Diego H. Milone

Jurados de Tesis

Dra. Elisabeth Tapia

Dr. Ignacio Ponzoni

Dr. Ariel Bayá

Santa Fe de la Vera Cruz, Argentina. Septiembre de 2016

A Sofía.

Índice general

Índice de tablas

Índice de figuras

Resumen

Los algoritmos de clustering tienen como objetivo encontrar clusters o grupos de datos, donde cada cluster representa un conjunto de objetos que son similares, mientras que objetos de diferentes clusters no lo son. Si bien existe una gran cantidad de algoritmos de clustering en la literatura, cada uno tiene sus ventajas y desventajas, y poseen diferentes suposiciones sobre los datos y la forma de los clusters por encontrar. Esto hace que cada uno produzca soluciones muy diferentes sobre el mismo conjunto de datos. Las técnicas de ensambles de agrupamientos nacen para mitigar estos problemas en los enfoques clásicos de clustering. En lugar de tener que elegir un solo algoritmo y un solo conjunto de parámetros, estos novedosos métodos utilizan un ensamble de todas las soluciones existentes, y las combinan para generar una única solución consensuada, superior en calidad. Sin embargo, varios estudios recientes han mostrado que la función de consenso requiere que el ensamble de entrada tenga determinadas características para producir un buen resultado. Una de estas características fundamentales es la diversidad, es decir, qué tan diferentes son las particiones miembro. Aunque se han propuesto diferentes medidas para cuantificar la diversidad de los ensambles, los distintos estudios no han llegado a conclusiones sólidas acerca de cómo los diferentes niveles de diversidad afectan al consenso final. Esto se debe, principalmente, a que los trabajos actuales generan ensambles con una distribución de diversidades a partir de la que no es posible llegar a conclusiones acerca de cómo impacta en el consenso. En esta tesis doctoral se trabajó principalmente en un método novedoso que permite controlar la diversidad de forma gradual, induciendo cambios suaves en la función de consenso. A diferencia de todos los enfoques actuales, el método propuesto analiza la estructura del ensamble de entrada y, a partir de ahí realiza pequeñas modificaciones en las particiones miembro, eligiendo inteligentemente aquellas que producen cambios suaves en la diversidad. El método fue evaluado exhaustivamente sobre varios conjuntos de datos y fue comparado contra otros métodos del estado del arte. Los resultados obtenidos

muestran que el enfoque propuesto es un avance importante en el análisis de la diversidad de los ensambles y representa una base sólida para llevar a cabo mejores estudios sobre el impacto de la diversidad en el desempeño del consenso. Los métodos propuestos se aplicaron también de forma original en el área de bioinformática utilizando datos de fuentes heterogéneas y los resultados obtenidos superaron a los algoritmos tradicionales para este tipo de problemas. Finalmente, los avances para procesar fuentes de datos con una alta heterogeneidad fueron incorporados como un componente dentro de una arquitectura inicial basada en ensambles, que fue propuesta para abordar más integralmente este problema en el contexto de los Grandes Datos, los cuales además de la variedad, tienen la característica de poseer grandes volúmenes de información que se generan a una alta velocidad.

Agradecimientos

Mi primer agradecimiento es a toda mi familia. En primer lugar, quiero agradecer a Sofía, mi compañera de la vida y una gran mujer, quien me ha apoyado todo este tiempo. Sin su cariño, su dedicación y paciencia, este trabajo no se hubiera podido realizar. También quiero darles las gracias a mis padres, Mario y Doli, que me apoyaron en todo siempre, en todo sentido. A mis hermanas, Gisela y Daiana, por su compañía y amistad incondicional.

Le agradezco también a César, un gran compañero de facultad y amigo de la vida. Mi gratitud por su amistad y por todo lo compartido, y porque él también ha sido responsable de que yo haya tomado este rumbo profesional. También agradezco a mis amigos Sebastián, Luis y Martín, con quienes compartimos mucho tiempo como compañeros de facultad y me han hecho crecer personalmente. Gracias a todos mis amigos, quienes se alegran con cada paso en mi vida, y que fueron y son una valiosa compañía.

Mis directores, la Dra. Georgina Stegmayer y el Dr. Diego Milone, han sido una guía fundamental no solo durante todo el transcurso de la tesis, sino desde antes, ya que gracias a sus palabras, consejos y amor por la vida me han ayudado a descubrir mi vocación profesional.

Agradezco a todos los integrantes del Centro de Investigación y Desarrollo de Ingeniería en Sistemas de Información (CIDISI), de la Universidad Tecnológica Nacional, Facultad Regional Santa Fe. Gran parte de los investigadores y becarios de este centro fueron mis profesores en la carrera de grado, me han mostrado su pasión por lo que hacen y su compañía ha sido muy importante para mí.

También agradezco a todos y cada uno de los integrantes del Instituto de Investigación en Señales, Sistemas e Inteligencia Computacional (sinc(i)), de la Universidad Nacional del Litoral, Facultad de Ingeniería y Ciencias Hídricas. Estoy enormemente agradecido a estas personas que aman lo que hacen, por haber compartido siempre sus conocimientos y su valiosa experiencia.

Les doy las gracias a mis amigos Guillermo y David, con quienes hemos

hecho juntos todo este camino, compartido importantes experiencias y ayudado mutuamente. También quiero agradecer a Mariano y a Matías, quienes me han transmitido su experiencia profesional, que ha sido un gran apoyo para mí.

Mi agradecimiento también va dirigido al Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET), que me ha otorgado una beca para poder financiar mi doctorado y realizar este trabajo.

Finalmente, estoy infinitamente agradecido a Dios porque, por una razón que yo no alcanzo a entender, me ha dado el don inconmensurable de creer que *«todo aquello de lo que está hecho el mundo, un día se hizo Uno de nosotros»*, y esta certeza es la que sostiene y le da sentido a todo en mi vida, desde lo más aparentemente banal y obvio, hasta la realización de mi doctorado y el trabajo hecho en esta tesis.

CAPÍTULO 1

Introducción

No existe nada más absurdo que la respuesta a una pregunta que no se ha planteado.
— Reinhold Niebuhr

La minería de datos es un campo interdisciplinario que estudia la obtención de conocimiento útil a partir de repositorios con grandes cantidades de datos (Bishop, 2006). Estas técnicas han sido ampliamente utilizadas en la industria, en diferentes áreas científicas y de ingeniería, y en el sector gubernamental, debido a su capacidad de extraer de los datos información previamente desconocida y potencialmente útil (Witten y otros, 2011). Estos algoritmos se han empleado en una gran cantidad de problemas muy diversos: desde el análisis de imágenes satelitales y de cuerpos celestes en el universo, hasta el pronóstico de consumo de energía eléctrica y la identificación de genes funcionalmente similares en un genoma. Indudablemente las técnicas de minería de datos ofrecen una herramienta indispensable para afrontar problemas clave para la sociedad de hoy, donde la información se genera cada vez a mayor velocidad y en volúmenes crecientes (Murphy, 2012).

La minería de datos se puede describir formalmente como un proceso que cuenta con varios pasos intermedios (Xu y Wunsch, 2009). Este proceso general se puede observar en la Figura 1.1, donde cada paso individual genera una salida que es consumida por la tarea siguiente. Cualquier problema de este tipo comienza con un conjunto de datos en particular. Una vez que se tiene en claro el problema a resolver y debido a que los datos pueden tener orígenes muy diversos, se hace una *selección de las fuentes* de dichos datos, de acuerdo a los objetivos que se hayan planteado. Esta selección puede consistir en un muestreo o en la elección de determinadas fuentes de datos, descartando aquellas consideradas menos impor-

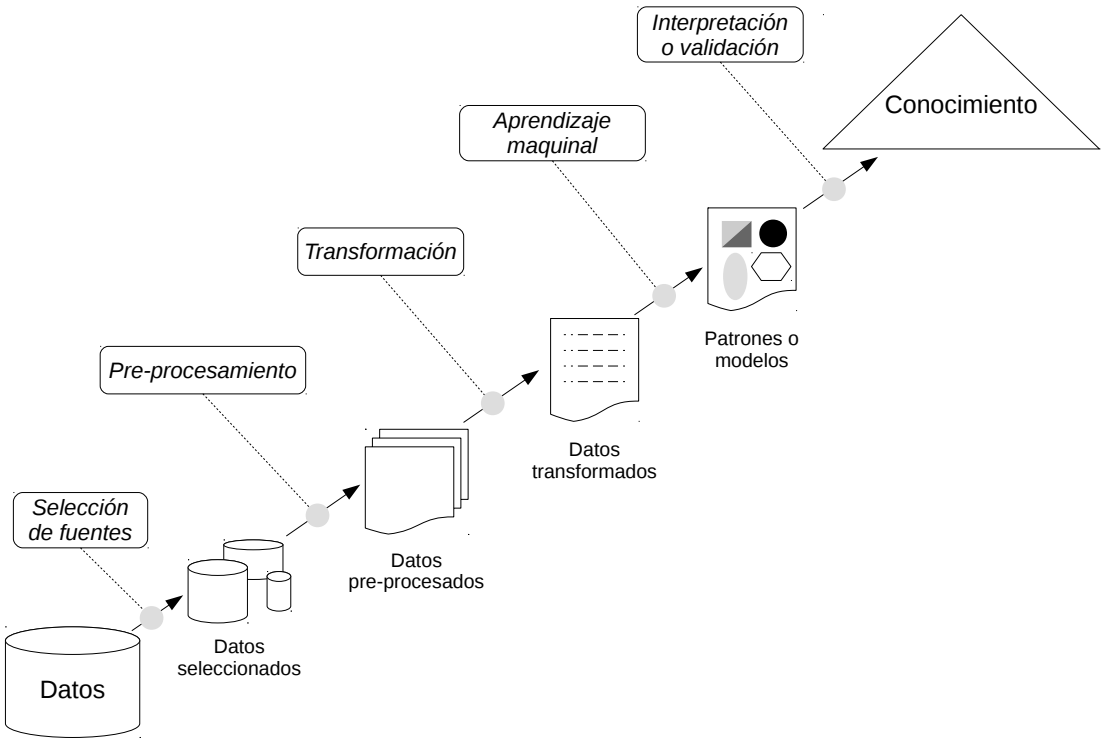


Figura 1.1. El proceso de minería de datos, modificado a partir de los pasos del *Proceso de Extracción de Conocimiento* (Han y otros, 2011).

tantes. Los *datos seleccionados* serán entonces aquellos datos que se consideren útiles para encarar la tarea de minería de datos según el objetivo propuesto. Debido a la posible presencia de ruido, de datos faltantes o de otras propiedades que puedan perjudicar la correcta ejecución de los pasos siguientes, es necesario realizar un *preprocesamiento* de los datos seleccionados (Gan y otros, 2007). Este paso puede consistir, por ejemplo, en apartar aquellos objetos considerados como atípicos, o en estandarizar las variables para que posean ciertas propiedades estadísticas deseables. Una vez preprocesada, toda la información puede requerir cierto tipo de *transformación*. Por citar algunos ejemplos, los datos pueden ser transformados si es necesario reducir su dimensionalidad, ya que de otro modo esta podría impedir la aplicación de los algoritmos siguientes. También puede ser necesario realizar operaciones de estandarización o normalización de los datos para lograr el correcto funcionamiento de los métodos de *aprendizaje maquinal*, que representan el siguiente paso de todo el proceso. Como se explicará en breve, existen diferentes algoritmos de aprendizaje pero todos ellos tienen en común la capacidad de aprender de los datos encontrando patrones o modelos que los expliquen (Bishop, 2006). Si bien estos algoritmos pueden ser evaluados internamente

utilizando diferentes métricas bien conocidas, es deseable que los resultados finales sean *interpretados* o *validados* por una persona experta en el dominio de aplicación, como por ejemplo un biólogo o un empresario. Dicho análisis es el que permite producir, finalmente, el nuevo conocimiento sobre los datos; ya sea por ejemplo el descubrimiento de nuevas relaciones entre genes o diferentes patrones de comportamiento de los clientes de una empresa.

Existen diferentes tipos de algoritmos de aprendizaje maquinal, donde cada uno tiene características particulares que lo hacen aplicable para cierto tipo de problema (Bishop, 2006). Uno de estos tipos corresponde al *aprendizaje supervisado*, donde quizás los más conocidos sean los algoritmos de *clasificación*. Para estos problemas se cuenta con casos u observaciones que tienen una clase o respuesta asociada. Este conjunto se observaciones, conocidas de antemano, son utilizada para entrenar y validar un algoritmo de clasificación. Una vez entrenado, el mismo se utiliza para clasificar casos no vistos antes. Un ejemplo puede ser la clasificación de correos electrónicos como spam o correo normal, o la clasificación de clientes riesgosos para una empresa de seguros.

Los algoritmos de clasificación funcionan para determinado tipo de problemas donde se poseen ejemplos etiquetados. Sin embargo, existen otras situaciones donde no se posee la etiqueta a la que pertenece cada ejemplo en particular (Murphy, 2012). En este tipo de problemas el objetivo no es entrenar un algoritmo a partir de ejemplos ya clasificados, sino encontrar las clases de los objetos o, mejor dicho, encontrar en los datos patrones o grupos no conocidos previamente y que pueden resultar interesantes para el usuario. Los algoritmos diseñados para este tipo de problemas caen en la categoría de *aprendizaje no supervisado*, y esta tesis trata sobre este tipo de métodos. Por ejemplo, ante los registros de ventas de un supermercado, un algoritmo de este tipo podría encontrar que existen diferentes clases o grupos de clientes (Kogan y otros, 2006). Esta información se podría utilizar para enviar publicidad personalizada a cada grupo.

Si bien existen diferentes técnicas que caen dentro de la categoría de aprendizaje no supervisado, los métodos de *clustering* (o agrupamiento) son los más conocidos. Los algoritmos de clustering tienen como objetivo encontrar *clusters* o grupos de objetos en los datos, donde cada cluster representa *"un conjunto de objetos o entidades que son similares, mientras que objetos o entidades de diferentes clusters no lo son"* (Xu y Wunsch, 2009). Existe una gran cantidad de diferen-

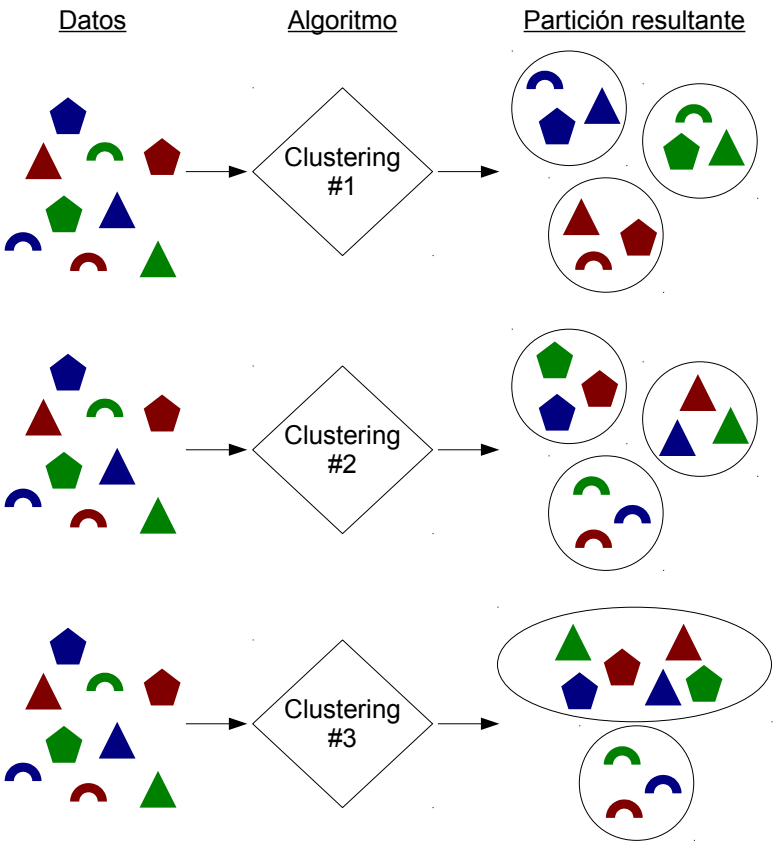


Figura 1.2. Ejemplo ilustrativo de la aplicación de tres algoritmos de clustering diferentes sobre un mismo conjunto datos, generando distintas soluciones.

tes tipos de algoritmos para resolver esta clase de problemas, y cada uno tiene sus propias particularidades. Algunos de los más populares son los algoritmos de agrupamiento jerárquico (Gan y otros, 2007; Sibson, 1973), k-medias (Jain, 2010; Jain y Dubes, 1988) y los mapas auto-organizativos (Kohonen, 1982).

Los algoritmos de clustering reciben como entrada un conjunto de datos y producen como salida una solución de clustering o *partición de los datos*, que consiste en un grupo de clusters (Gan y otros, 2007). El hecho de que existan diferentes técnicas de clustering, como se ha mencionado anteriormente, responde a la necesidad de que existen diferentes problemas de clustering. Esto se intenta mostrar de una manera sencilla en la Figura 1.2 donde, ante un grupo de formas geométricas (los datos, mostrados a la izquierda), se aplican tres técnicas de clustering diferentes que, como se puede ver, producen particiones muy distintas (como puede verse a la derecha). Cada partición tiene un número de clusters que agrupa ciertos objetos que poseen algún tipo de similitud. Aparece aquí un inconveniente: pueden existir diferentes formas de medir cuán similares son los objetos.

El primer algoritmo de clustering (Clustering #1) ha agrupado las entidades exclusivamente por su color. En cambio, el segundo algoritmo ha dado una mayor importancia a la forma de los objetos. Finalmente, los grupos generados por el tercer algoritmo poseen similitudes en cuanto a las características de sus lados. Todas estas soluciones, encontradas sobre los mismos datos, podrían ser interesantes de analizar, dado que en este tipo de problema no existe una referencia previa de lo que se debe agrupar.

Hasta aquí hemos introducido el tipo de problema que intentan resolver las técnicas de clustering, y hemos visto un ejemplo sencillo que muestra que no existe un único algoritmo para solucionar todos los problemas. De hecho, para un conjunto de datos determinado, pueden existir múltiples soluciones de clustering diferentes e igualmente válidas, y esta es la razón por la cual el problema de clustering o agrupamiento es considerado un problema mal condicionado (Kleinberg, 2002; Yi y otros, 2012).

1.1. Motivación de la tesis

Cada algoritmo de clustering está diseñado para resolver un problema específico (Jain, 2010). Cada técnica supone, por ejemplo, la forma que tendrán los clusters buscados y la distribución de los datos. De esta manera, el usuario de estos métodos, y particularmente el no experto, debe conocer con precisión las características de los algoritmos y de los datos a analizar, además del efecto que los diferentes parámetros del método puedan producir sobre los resultados.

En la Figura 1.3 se presentan dos ejemplos que ilustran casos donde no se cumplen las suposiciones que el algoritmo k-medias hace sobre las características de los clusters. En ambas subfiguras se muestran dos conjuntos de datos con diferentes propiedades, donde el color y el estilo de cada punto indican el cluster al que k-medias lo ha asignado. En la Figura 1.3a se han generado artificialmente 3 clusters "elongados". Al aplicar k-medias sobre estos datos, se han encontrado 3 clusters que difieren mucho de los originalmente generados. Esto es así porque k-medias supone que los clusters son hiperesféricos, y si esta suposición no se cumple, los resultados no serán los esperados, como puede verse en esa subfigura. Por otro lado, en la Figura 1.3b también se han generado 3 clusters, esta vez hiperesféricos pero con diferente varianza. Y aquí nuevamente aparece otra suposición

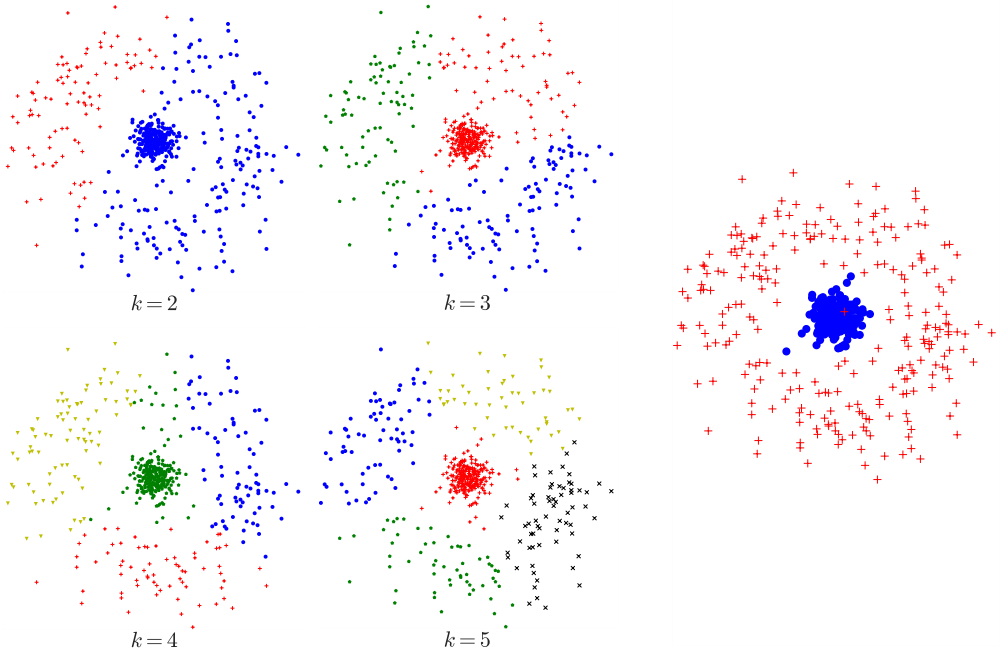
(a) Clusters con forma elongada. (b) Cluster con varianza desigual.

Figura 1.3. Diferentes propiedades de los clusters que no cumplen con las suposiciones de k-medias sobre los datos (modificado a partir de un ejemplo de scikit-learn[1]).

de k-medias que no es respetada: que la varianza es similar en todos los clusters. Con estos datos se vuelven a ver resultados de baja calidad, donde el algoritmo no es capaz de encontrar la estructura subyacente. Similares situaciones pueden presentarse con otros algoritmos, los cuales tienen sus propias suposiciones y no son capaces de resolver todos los problemas de clustering.

Las técnicas de ensambles de agrupamientos nacen justamente para mitigar estos problemas en los enfoques clásicos de clustering (Huang y otros, 2016; Iam-On y otros, 2011; Topchy y otros, 2005). En lugar de tener que elegir un solo algoritmo y un solo conjunto de parámetros, estos novedosos métodos utilizan todas las soluciones existentes, combinándolas para generar una única solución superior en calidad. Los ensambles de agrupamientos poseen básicamente dos pasos (Strehl y otros, 2002): I) se genera un *ensamble de agrupamientos*, es decir, un gran número de soluciones de clustering convencionales, y luego II) estas son combinadas mediante una *función de consenso*, produciendo una única solución consensuada. La posibilidad que brindan los ensambles de agrupamientos simplifica significativamente la labor del usuario no experto, ya que este puede emplear varios algoritmos y diferentes combinaciones de parámetros para explorar un conjunto de datos, sin preocuparse de tener que elegir el mejor enfoque. Luego, la función de consenso es la encargada de generar una única solución combinada a

[1]Fuente: `http://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_assumptions.html`



(a) Ensamble de agrupamientos con $k = 2, 3, 4$ y 5. (b) Solución consensuada.

Figura 1.4. Ejemplo de un ensamble de agrupamientos sobre el conjunto de datos Difficult Doughnut.

partir de aquel conjunto de soluciones heterogéneas de entrada.

En la Figura 1.4 se muestra un ejemplo de un ensamble de agrupamientos junto con la solución consensuada, usando un conjunto de datos artificial llamado Difficult Doughnut, ampliamente utilizado en el área de aprendizaje de máquina (Hadjitodorov y otros, 2006). Estos datos presentan dos grupos de objetos con formas muy difíciles de detectar por un algoritmo de clustering tradicional: el primero es un grupo denso ubicado en el centro, y el segundo un anillo que lo rodea. El ensamble que se muestra en la Figura 1.4a posee cuatro particiones generadas con k-medias usando diferentes números de clusters, los cuales son representados con diferentes colores. Se puede ver que ninguna de las particiones miembro del ensamble, creadas con k-medias, es capaz de detectar los patrones de este conjunto de datos. Sin embargo, cuando este ensamble es procesado por una función de consenso, la estructura compleja de Difficult Doughnut sí es recuperada, como puede verse en la Figura 1.4b, donde el cluster central está claramente separado del anillo exterior. Como se ha mostrado en varios artículos (Iam-On y otros, 2012; Strehl y otros, 2002; Topchy y otros, 2005), la partición consensuada final generalmente supera en calidad a las particiones miembro del ensamble.

El ejemplo mostrado revela varias ventajas de los ensambles de agrupamientos. Primero, que estas técnicas facilitan mucho el trabajo para un usuario no experto en clustering, ya que este puede utilizar una batería de algoritmos y parámetros para generar muchas soluciones, y utilizar la información de todas ellas en lugar de tener que elegir solo una. Luego, que la función de consenso es la que procesa ese conjunto de soluciones produciendo una única partición consensuada que supera en calidad al resto. Esto evita la desafiante tarea de tener que elegir un único algoritmo y determinar sus parámetros.

Otra ventaja importante es que los ensambles de agrupamientos permiten hacer computación distribuida (Strehl y otros, 2002; Wang y otros, 2008). Un ejemplo claro es una base de datos de clientes de una empresa distribuida en varias sucursales. Cada sucursal podría tener clientes solapados (los mismos clientes compran en varias de ellas) o registrar diferentes características sobre sus compras. En este caso no solo es posible distribuir el procesamiento en cada una de ellas, sino que también se logran superar eventuales barreras legales o de privacidad. Por ejemplo, es posible que, por cuestiones legales o restricciones gubernamentales, alguna de las sucursales no tenga permitido compartir ciertos datos de sus clientes, lo cual es un requisito para un enfoque tradicional de clustering. Sin embargo, dichas restricciones no son un problema para los enfoques de ensambles, ya que la función de consenso no necesita acceder a los datos originales, sino solo a las particiones.

Puede afirmarse que los ensambles de agrupamientos representan, hoy en día, un método superador de los enfoques clásicos y han sido utilizados con éxito en varios problemas (Fiori y otros, 2016; Huang y otros, 2011; Iam-On y otros, 2012). Sin embargo, algunos estudios recientes han mostrado situaciones en las que estas técnicas no logran superar siempre la calidad de las particiones del ensamble, es decir, no mejoran los resultados de los algoritmos clásicos como k-medias. Estos trabajos han mostrado que la función de consenso requiere que el ensamble de entrada tenga determinadas características para producir un buen resultado (Fern y Lin, 2008; Hu y otros, 2016; Iam-On y otros, 2011). Una de estas características fundamentales es la *diversidad* del ensamble (Gullo y otros, 2009; Hadjitodorov y otros, 2006), es decir, qué tan diferentes son las particiones miembro. Intuitivamente, el ensamble debe contener cierta diversidad para que la función de consenso logre mejores resultados (Strehl y otros, 2002), ya que

la homogeneidad total no permitiría encontrar nuevos y mejores clusters. En el ejemplo de la Figura 1.4, el ensamble de cuatro particiones evidentemente tiene la diversidad adecuada para que la función de consenso logre un resultado final superador.

Aunque se han propuesto diferentes medidas para cuantificar la diversidad de los ensambles (Azimi y Fern, 2009; Gullo y otros, 2009; Naldi y otros, 2013; Yu y otros, 2014), los distintos estudios no han llegado a conclusiones sólidas sobre cómo afectan al consenso final los diferentes niveles de diversidad. De hecho, existe actualmente una gran confusión sobre esta cuestión: algunos estudios afirman que deben elegirse los ensambles más diversos (Fern y Brodley, 2003; Iam-On y otros, 2011), mientras que otros sostienen que se debe buscar una diversidad media (Hadjitodorov y otros, 2006). Estas opiniones contradictorias se deben al comportamiento de la función de consenso al combinar ensambles diversos generados con los métodos actuales. Esto puede comprobarse si se realiza una gráfica de la calidad de las particiones consensuadas en función de la diversidad de los ensambles de donde estas se derivan. En este tipo de análisis se puede observar cómo ante ensambles con diversidades similares la función de consenso genera particiones con calidades sumamente diferentes, lo que hace imposible llegar a algún tipo de conclusión sobre cómo impacta la diversidad en el consenso. Estos resultados son indicativos del estado actual de la investigación sobre este tema, donde además existe gran variabilidad al utilizar conjuntos de datos distintos o emplear diferentes funciones de consenso (Hadjitodorov y otros, 2006; Hu y otros, 2016; Iam-On y otros, 2011). Esta situación arroja muchas dudas sobre los hallazgos actuales, donde si bien se sabe que la diversidad tiene un rol importante en el desempeño del consenso, aún no se conoce con precisión de qué manera esta lo afecta o condiciona.

La motivación central de esta tesis está inspirada en los problemas actuales que existen con respecto a la diversidad de los ensambles. Como se planteará en los objetivos y se mostrará en los capítulos siguientes, el abordaje de esta cuestión ha producido métodos útiles no solo para un tratamiento directo del problema de la diversidad, sino también para las aplicaciones en problemas de bioinformática con grandes datos.

1.2. Objetivos

El objetivo general de esta tesis es proveer nuevas herramientas para el análisis de los ensambles de agrupamientos y sus propiedades, a partir de las cuales se puedan desarrollar nuevos modelos para combinar los resultados de múltiples agrupamientos utilizando diferentes fuentes de datos biológicos.
Los objetivos específicos son los siguientes:

1. Analizar la relación entre la diversidad y el desempeño de los ensambles de agrupamientos en las soluciones consensuadas.

2. Proponer nuevos métodos para controlar el impacto de la diversidad en las soluciones consensuadas.

3. Diseñar y aplicar nuevos métodos de ensambles de agrupamientos para el tratamiento de datos biológicos provenientes de fuentes heterogéneas y con grandes volúmenes.

4. Validar la aplicación de los métodos desarrollados con expertos del dominio, al integrar diferentes tipos de datos, obtenidos a partir de fuentes heterogéneas del mismo material biológico.

5. Proponer nuevas metodologías que simplifiquen la reproducibilidad y accesibilidad de los resultados científicos en el dominio de aplicación.

1.3. Organización de la tesis

La presente tesis está organizada en una serie de capítulos en donde se describen todos los trabajos publicados. En el Capítulo 2, se explican los principales métodos utilizados de clustering y de ensambles de agrupamientos, así como también los diferentes índices para evaluar el desempeño. Luego, en el Capítulo 3, se presenta el aporte principal de esta tesis, donde se hace un análisis más profundo del problema de la diversidad en los ensambles, se formula una hipótesis sobre sus causas y se detalla un método propuesto para controlar la diversidad. En el Capítulo 4 se describe la aplicación de los métodos de ensambles a un conjunto de datos reales de origen biológico con fuentes heterogéneas. Luego, los

métodos utilizados en esta aplicación se extienden en una arquitectura general inicial para el tratamiento de grandes datos. Luego, en el Capítulo 5, se presenta una herramienta que permite mejorar de forma significativa la reproducibilidad y accesibilidad de los resultados de las investigaciones científicas. Finalmente, el Capítulo 6 resume las principales conclusiones de cada aporte de esta tesis.

1.4. Principales contribuciones

Durante el transcurso de esta tesis doctoral, se trabajó principalmente en el análisis del impacto de la diversidad de los ensambles en el desempeño final del consenso. Este trabajo produjo el desarrollo de un método inicial para explorar las características de los ensambles, el cual ha sido publicado en un congreso nacional (Pividori y otros, 2013b), y luego como trabajo extendido en una revista internacional con referato (Pividori y otros, 2014). Después de diferentes avances sobre estos trabajos iniciales, finalmente se desarrolló un método de control de diversidad, el cual fue exhaustivamente evaluado sobre varios conjuntos de datos y comparado/contrastado contra otros métodos del estado del arte, siendo el principal aporte de esta tesis. Los resultados obtenidos muestran que el método propuesto representa un avance importante en el estudio de la diversidad de los ensambles. Este trabajo ha sido publicado en una revista internacional con referato y factor de impacto 4.038 (Pividori y otros, 2016a).

Paralelamente al estudio de la diversidad de los ensambles, se propuso un método para el análisis integrado de diferentes medidas obtenidas sobre datos de tomate (*Solanum Lycopersicum*), las cuales provenían de fuentes heterogéneas. El enfoque propuesto utiliza diferentes métodos para generar diversidad en el ensamble de soluciones de clustering, el cual ha producido resultados satisfactorios en comparación con los algoritmos tradicionales. Este trabajo fue publicado en un congreso nacional sobre bioinformática y biología computacional (Pividori y otros, 2013a), recibiendo una distinción de honor.

Los ensambles de agrupamientos, como se ha mencionado anteriormente, permiten el análisis distribuido de los datos lo cual hace posible paralelizar todo el procesamiento, representando así una atractiva herramienta para encarar problemas relacionados con Grandes Datos. Este tipo de problemas involucra datos de gran volumen, una enorme variedad de formatos y fuentes, y una gran veloci-

dad de generación de información[2], características que están muy presentes en los datos biológicos, principal área de aplicación de esta tesis. El resultado ha sido una propuesta de arquitectura para procesar este tipo de información utilizando ensambles de agrupamientos. Este trabajo ha sido publicado en un congreso nacional, en un simposio específico de Grandes Datos (Pividori y otros, 2015). Un nuevo manuscrito, extendiendo la idea previamente presentada, se encuentra actualmente en preparación (Pividori y otros, 2016c)

Durante el desarrollo de la tesis se ha abordado un problema que es transversal a todas las cuestiones tratadas en este documento: la reproducibilidad de las investigaciones, particularmente relevante en el ámbito de la bioinformática. La tarea misma de investigación y publicaciones de los avances ha motivado el desarrollo de una herramienta para generar fácilmente aplicaciones web llamadas *web-demos*, que permiten no solo mejorar la reproducibilidad de los resultados, sino también su accesibilidad y visualización. Este trabajo fue publicado en una muy importante revista de bioinformática con factor de impacto 9.617 (Stegmayer y otros, 2016).

A modo de resumen, se listan a continuación las publicaciones que se han realizado en el desarrollo de esta tesis doctoral:

1. Pividori, M.; Stegmayer, G. y Milone, D. H. (2016a). «Diversity control for improving the analysis of consensus clustering». *Information Sciences*, **361**, pp. 120–134.

2. Stegmayer, G.; Pividori, M. y Milone, D. H. (2016). «A very simple and fast way to access and validate algorithms in reproducible research». *Briefings in Bioinformatics*, **17(1)**, pp. 180–183.

3. Pividori, M.; Stegmayer, G. y Milone, D. H. (2016b). «Multi-hierarchical clustering for exploring hierarchies networks on gene expression data». Manuscrito en preparación.

4. Pividori, M.; Stegmayer, G. y Milone, D. H. (2016c). «Scalable and flexible consensus clustering framework for big data mining problems». Manuscrito en preparación.

[2] `http://www.ibmbigdatahub.com/infographic/four-vs-big-data`

5. Pividori, M.; Stegmayer, G. y Milone, D. H. (2015). «Cluster Ensembles for Big Data Mining Problems». En: *Argentine Symposium on Big Data, 44 JAIIO (Argentine Conference on Informatics)*.

6. Pividori, M.; Stegmayer, G. y Milone, D. H. (2014). «A Method to Improve the Analysis of Cluster Ensembles». *Revista Iberoamericana de Inteligencia Artificial*, **17(53)**, pp. 46–56.

7. Pividori, M.; Stegmayer, G. y Milone, D. H. (2013b). «A Novel Method to Control the Diversity in Cluster Ensembles». En: *Argentine Symposium on Artificial Intelligence (ASAI 2013) - 42º JAIIO*pp. 121–132.

8. Pividori, M.; Stegmayer, G.; Carrari, F. y Milone, D. H. (2013a). «Consensus clustering from heterogeneous measures of S. Lycopersicum». En: *4to. Congreso Argentino de Bioinformática y Biología Computacional (4CAB2C)*.

CAPÍTULO 2

Marco teórico

Poca observación y mucho razonamiento llevan al error.
Mucha observación y menos razonamiento llevan a la verdad.
— Alexis Carrel

Yo investigo para saber algo, no para pensarlo.
— San Agustín

En este capítulo se hará una presentación más detallada de los fundamentos para los principales temas tratados en la tesis: clustering y ensambles de agrupamientos. Sobre el primero, se presentarán los conceptos más básicos de forma breve, como cuál es la entrada de un algoritmo de clustering y qué se espera como salida, las diferentes medidas de distancia y los problemas asociados, la clasificación de los algoritmos junto con ejemplos concretos utilizados durante el desarrollo de la tesis, y finalmente los índices que se utilizan para validar los resultados. Para los ensambles de agrupamientos, se introducirá la idea de estos métodos, los cuales superan a las técnicas de clustering convencionales. Se describirán los dos pasos principales: la generación de un ensamble, y la combinación del mismo en una solución consensuada. También se describirá el concepto de diversidad de un ensamble y las diferentes medidas existentes en la literatura para cuantificarla.

2.1. Clustering

Como ya se ha explicado en el Capítulo 1, entre los métodos de aprendizaje no supervisado se encuentran los algoritmos de clustering. El objetivo de estos algoritmos es encontrar clusters o grupos homogéneos de objetos entre los datos. Esto significa que dado un conjunto particular de datos y una definición precisa

de similaridad (Witten y otros, 2011), lo que hace un método de clustering es agrupar dichos datos de tal forma que aquellos que se encuentren en el mismo grupo sean similares, mientras que los pertenecientes a diferentes grupos no lo sean (Xu y Wunsch, 2009).

Al referirnos a los problemas de clustering, aparecen varios conceptos diferentes que muchas veces significan lo mismo (Gan y otros, 2007). Por ejemplo, dado un conjunto de datos en particular, decimos que el mismo contiene varios registros: los términos *objeto*, *caso* u *observación* se utilizan para referirse a un ítem de datos en particular (por ejemplo, un gen o una persona). Para cada objeto u observación en un espacio de alta dimensionalidad, usamos los términos *medida*, *característica* o *atributo* para referirnos a cierto componente del objeto (por ejemplo, la edad o el estado civil si se trata de una persona). Matemáticamente, decimos que un conjunto de datos posee n objetos, cada uno de los cuales está descrito por m características o atributos. El conjunto de datos completo se denota con $S = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, donde $\mathbf{x}_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,m})^\top$ es un vector que denota al i-ésimo objeto, y x_{ij} es un escalar que denota al j-ésimo atributo o característica de $\mathbf{x}_i$.

La entrada de un algoritmo de clustering puede ser básicamente de dos tipos (Murphy, 2012). En el primero, esta puede ser una matriz de distancias $M \in \mathbb{R}^{n \times n}$. Cada entrada de esta matriz contiene la distancia, medida de alguna forma en particular, entre los objetos i y j, $1 \leq i, j \leq n$. Por otro lado, la entrada puede ser también una matriz de características $Q \in \mathbb{R}^{n \times m}$. Estos dos tipos de entradas se pueden ilustrar con un ejemplo usando el conjunto de datos Iris, ampliamente utilizado en la comunidad de aprendizaje maquinal. El mismo posee mediciones sobre 3 especies diferentes de flores. Contiene 150 objetos (es decir, $n = 150$) y 4 características ($m = 4$): el ancho y largo del pétalo y el sépalo. Para este caso, poseemos una matriz de características $Q \in \mathbb{R}^{150 \times 4}$, que también podemos transformar en una matriz de distancias $M \in \mathbb{R}^{150 \times 150}$.

Una vez que el algoritmo de clustering procesa la entrada, ya sea esta una matriz de distancias o de características, el mismo produce esencialmente como salida una *partición* de los datos, que denotamos como U. Una partición U puede representarse como una matriz de $k \times n$, donde k es el número de clusters o grupos:

$$U = \begin{bmatrix} u_{11} & u_{12} & \dots & u_{1n} \\ u_{21} & u_{22} & \dots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ u_{k1} & u_{k2} & \dots & u_{kn} \end{bmatrix}$$

Cada valor u_{ji} indica la pertenencia o no del objeto i al cluster j. De acuerdo a la naturaleza de la pertenencia de cada objeto a un cluster, los algoritmos de clustering pueden clasificarse en *exclusivos* o *difusos*. Cuando el algoritmo es exclusivo, un objeto pertenece exclusivamente a un solo cluster, y la partición U satisface las siguientes restricciones (Gan y otros, 2007):

$$u_{ji} \in \{0, 1\}, \forall 1 \leq j \leq k, 1 \leq i \leq n, \tag{2.1}$$

$$\sum_{j=1}^{k} u_{ji} = 1, \forall 1 \leq i \leq n, \tag{2.2}$$

$$\sum_{i=1}^{n} u_{ji} \geq 0, \forall 1 \leq j \leq k. \tag{2.3}$$

Estas restricciones implican que cada objeto pertenece o no a un cluster (2.1); que puede pertenecer a un único cluster (2.2); y que cada cluster tiene al menos un objeto (2.3), es decir, no se permiten clusters vacíos. En el caso de los algoritmos difusos, estas restricciones se relajan y un objeto puede pertenecer a varios clusters con cierto grado de membresía. Durante el desarrollo de esta tesis doctoral se han utilizado únicamente los algoritmos exclusivos, que son los más comúnmente usados. Por lo tanto, al referirnos a una partición, nos referimos a una partición exclusiva y, dada las características de la matriz U para el caso de agrupamiento exclusivo, a esta la podemos representar como un vector de tamaño $1 \times n$, donde cada elemento puede tomar valores del intervalo $[1 \dots k]$, indicando así la pertenencia de cada objeto a un determinado cluster. Esta partición también se puede representar como un conjunto de clusters $\pi = \{\Omega_1, \Omega_2, \dots, \Omega_k\}$ donde el j-ésimo cluster es $\Omega_j = \{\omega_{j\ell}\}$, con $\omega_{j\ell} \in [1 \dots n]$ y $\bigcup_{j=1}^{k} \bigcup_{\ell=1}^{|\Omega_j|} \mathbf{x}_{\omega_{j\ell}} = S$.

Los algoritmos de clustering también se pueden clasificar en *particionales* y *jerárquicos*. Los algoritmos particionales tienen como salida una única partición de los datos, la cual posee un determinado número de clusters como la partición

U mostrada antes. En los algoritmos jerárquicos, en cambio, la salida no es una única partición, sino un conjunto de particiones donde sus clusters están anidados.

A continuación se darán más detalles sobre el proceso de clustering que se ha descrito hasta ahora.

2.1.1. Medidas de distancia y métodos de estandarización

Al dar la definición de un cluster, es decir, un grupo de objetos que son similares entre sí y a la vez diferentes a objetos de otros clusters, no se ha dado una definición precisa de estos términos. En esta sección se definen las medidas de distancia más utilizadas, así como también ciertos problemas que surgen de su uso en aplicaciones reales.

Quizá la medida de distancia más comúnmente empleada sea la distancia euclídea (Jain y Dubes, 1988), cuya definición está dada por

$$d_e(\mathbf{x}_i, \mathbf{x}_j) = \left[\sum_{\ell=1}^{m}(x_{i\ell} - x_{j\ell})^2\right]^{\frac{1}{2}} = \left[(\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^\top\right]^{\frac{1}{2}}, \tag{2.4}$$

donde $\mathbf{x}_i$ y $\mathbf{x}_j$ son objetos m-dimensionales. Otra medida, en este caso de similaridad, es el coeficiente de correlación de Pearson (Theodoridis y Koutroumbas, 2006), definido como

$$\rho(\mathbf{x}_i, \mathbf{x}_j) = \frac{\sum_\ell^m (x_{i\ell} - \bar{x}_i)(x_{j\ell} - \bar{x}_j)}{\sqrt{\sum_\ell^m (x_{i\ell} - \bar{x}_i)^2 \sum_\ell^m (x_{j\ell} - \bar{x}_j)^2}}, \tag{2.5}$$

el cual puede tomar valores entre -1 y 1. Una medida de distancia asociada a ρ puede definirse como

$$d_\rho(\mathbf{x}_i, \mathbf{x}_j) = \frac{1 - \rho(\mathbf{x}_i, \mathbf{x}_j)}{2}, \tag{2.6}$$

la cual toma valores en el rango $[0, 1]$, y ha sido utilizada para el análisis de datos de expresión de genes (Eisen y otros, 1998).

Un conjunto de datos, sin embargo, puede poseer atributos con unidades muy diferentes entre sí. En este caso, aquellos atributos con valores o varianzas mucho mayores, que dominan al resto de las características, pueden provocar distorsiones al medir, por ejemplo, la distancia euclídea entre los objetos (Xu y Wunsch, 2009). Para paliar estos problemas, uno de los métodos utilizados

es la estandarización de los datos, haciendo que cada característica contribuya igualmente en la distancia.

Existen varios métodos de estandarización de los datos y este debe elegirse adecuadamente según sus propiedades. Básicamente, para cada atributo, se resta una medida de posición y se divide por una medida de escala (Gan y otros, 2007). Es decir,

$$x_{ij} = \frac{x^*_{ij} - L_j}{E_j}, \tag{2.7}$$

donde x_{ij} representa el valor estandarizado, x^*_{ij} el valor original, L_j la medida de posición y E_j es la medida de escala. Se pueden obtener varios métodos de estandarización eligiendo diferentes criterios para L_j y E_j. Algunos de los valores más utilizados son la media y la desviación estándar o un rango entre máximos y mínimos, cuyas definiciones son

$$\begin{aligned}
\bar{x}^*_j &= \frac{1}{n}\sum_{i=1}^{n} x^*_{ij}, \\
\sigma^*_j &= \left[\frac{1}{n-1}\sum_{i=1}^{n}(x^*_{ij} - \bar{x}^*_j)^2\right]^{\frac{1}{2}}, \\
R^*_j &= \max_{1\le i\le n} x^*_{ij} - \min_{1\le i\le n} x^*_{ij}.
\end{aligned}$$

Algunas formas de estandarización comúnmente utilizadas son la de *unidad tipificada* (*z-score*, en inglés) donde $L_j = \bar{x}^*_j$ y $E_j = \sigma^*_j$. Este método hace que los datos estandarizados tengan media igual a 0 y varianza igual a 1, aunque al aplicarlo se pierde la información de posición y escala de los datos originales. Otro método conocido es el de rango donde $L_j = \min_{1\le i\le n} x^*_{ij}$ y $E_j = R^*_j$.

2.1.2. k-medias

El algoritmo k-medias es uno de los métodos de clustering más populares (Jain, 2010). Este método particional, comparado con otros tipos de algoritmos, es muy eficiente para hacer clustering sobre datos de gran volumen y de alta dimensionalidad (Gan y otros, 2007). Uno de sus parámetros más importantes, y que debe ser establecido *a priori*, es el número de clusters k que el usuario quiere encontrar en los datos.

Algoritmo 1: k-medias.

Entrada:
S: conjunto de datos de tamaño $n \times m$
k: cantidad de clusters a obtener
I: cantidad máxima de iteraciones

Salida:
π: partición de los datos S con k clusters

```
1  inicio
2      {μ_1 ... μ_k} ← elegir k centroides
3      {Ω_1, Ω_2, ..., Ω_k} ← crear k clusters vacíos
4      iteración ← 0
5      repetir
6          para cada x_i ∈ S hacer
7              d_ij ← distancia entre x_i y μ_j, ∀j
8              j* ← arg min_{1≤j≤k} d_ij
9              asignar el objeto x_i al cluster Ω_{j*}
10         {μ_1 ... μ_k} ← calcular centroides para cada cluster Ω_j
11         π ← {Ω_1, Ω_2, ..., Ω_k}
12         iteración ← iteración + 1
13     hasta que {Ω_1 ... Ω_k} no cambian o iteración > I
14 devolver π
```

Siendo $S = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ un conjunto de datos con n objetos, supongamos que $\{\Omega_1, \Omega_2, \ldots, \Omega_k\}$ sean k clusters disjuntos de S. La función objetivo que k-medias intenta minimizar es

$$E = \sum_{j=1}^{k} \sum_{\forall i \in \Omega_j} d(\mathbf{x}_i, \boldsymbol{\mu}_j), \tag{2.8}$$

donde $\boldsymbol{\mu}_j = \frac{1}{|\Omega_j|} \sum_{\forall i \in \Omega_j} \mathbf{x}_i$ es el centroide del cluster Ω_j, $d(\mathbf{x}_i, \boldsymbol{\mu}_j)$ denota la distancia entre $\mathbf{x}_i$ y $\boldsymbol{\mu}_j$, y por lo general se utiliza la distancia euclídea definida en (2.4).

Los pasos de k-medias se detallan en el Algoritmo 1. Como puede verse, el algoritmo se divide en dos etapas: la etapa de inicialización y la de iteración. En la etapa de inicialización, k-medias escoge de alguna manera cuáles serán los centroides $\boldsymbol{\mu}_j$ de los k clusters. La forma de inicializarlos puede ser aleatoria (es decir, eligiendo cualquier valor dentro de cierto rango posible; o escogiendo aleatoriamente k objetos como centroides) o utilizando algún otro método más complejo como en *k-means++* (Arthur y Vassilvitskii, 2007). En la fase de ite-

ración el algoritmo calcula la distancia entre cada objeto con todos los clusters, asociándolo al más cercano. Cuando estas asociaciones no cambian, o cuando se alcanza un número máximo de iteraciones, el algoritmo se detiene y devuelve la partición encontrada.
Este algoritmo tiene las siguientes características (Gan y otros, 2007):

- Es un método eficiente para grandes conjuntos de datos, ya que su complejidad computacional es lineal con n.
- Generalmente encuentra mínimos locales al problema de optimización de la función objetivo.
- Debido a la función objetivo que utiliza, los clusters que encuentra son convexos.
- Es muy dependiente de la inicialización de los centroides, ya que esta afecta de manera significativa las soluciones que el algoritmo produce.

2.1.3. Clustering jerárquico

Los algoritmos de clustering jerárquico, a diferencia de k-medias u otro método particional, producen una secuencia de particiones con clusters anidados (Everitt y otros, 2011). Existen dos tipos de algoritmos jerárquicos: los aglomerativos y los divisivos. Los primeros, que son los más utilizados, comienzan considerando a cada objeto del conjunto de datos como un cluster en sí mismo. Luego estos clusters se van fusionando con los clusters más cercanos, de acuerdo a un determinado criterio de enlace, hasta que todos los datos se encuentran agrupados en un solo cluster. Los métodos divisivos, en cambio, comienzan con todos los datos en un solo cluster y luego van realizando divisiones sucesivas. Los enfoques divisivos, por lo general, son más costosos computacionalmente, y por eso los métodos aglomerativos son más comúnmente utilizados. En este documento se describirán solo estos últimos, que son los que han sido empleados en la tesis.

La salida de estos algoritmos se representa con una estructura en forma de árbol binario llamada *dendrograma*, como el ejemplo que se muestra en la Figura 2.1. Aquí se ha procesado un conjunto de datos con 9 objetos, denotados en la parte inferior como $0, 1, \ldots, 8$, y que son los nodos inferiores o nodos hoja

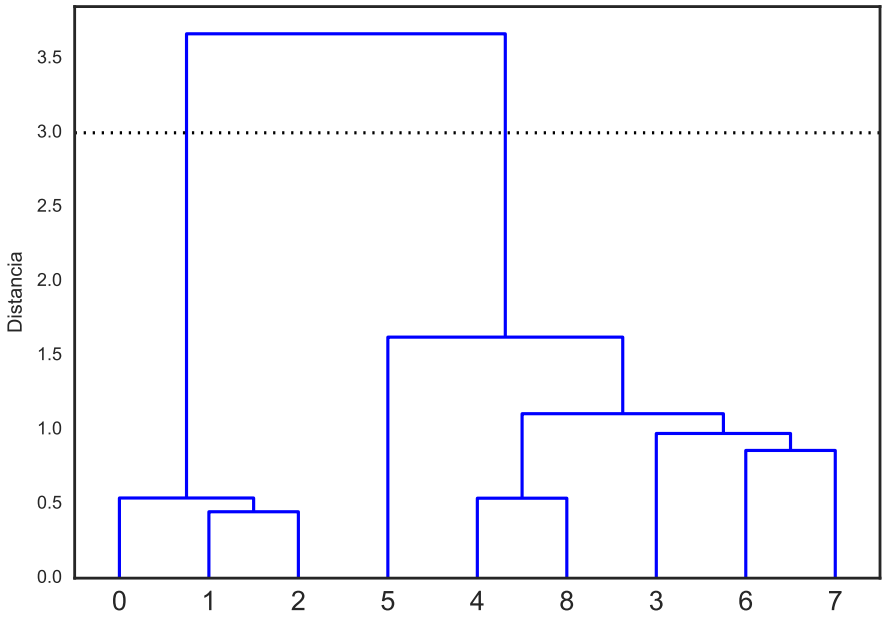


Figura 2.1. Ejemplo de un dendrograma generado con un enfoque aglomerativo.

del dendrograma. Los nodos intermedios, que en la figura se representan con la unión de otros dos nodos, poseen una determinada altura que en general expresa la distancia entre el par de nodos fusionados (ya sean nodos hoja u otros nodos intermedios) (Xu y Wunsch, 2009). Para obtener una partición definitiva de los datos se "corta" el dendrograma en diferentes alturas o niveles. En la figura se representa un corte con una línea punteada en una altura de 3.0, produciendo una partición final con 2 clusters (el primer cluster contiene los objetos $0, 1$ y 2 y el segundo el resto), en los cuales se anidan otros clusters intermedios.

2.1.3.1. Métodos aglomerativos

El procedimiento general de un método de clustering aglomerativo se describe en el Algoritmo 2. Los métodos aglomerativos comienzan considerando a los n objetos del conjunto de datos como clusters con un solo objeto. Luego comienza una serie de $n-1$ fusiones entre pares de clusters, hasta el punto en que todos los objetos forman uno solo. El algoritmo devuelve una matriz de enlace Z de tamaño $(n-1) \times 3$ que posee información sobre cómo los clusters fueron fusionados. Las primeras dos columnas de Z contienen los índices de los dos clusters que fueron combinados en cada paso, lo cual produjo la creación de un nuevo cluster formado por ambos. Los clusters de entrada (que coinciden con los objetos) se etiquetan como $0, 1, \ldots, n-1$, y los nuevos que se crean en cada paso tienen las

Algoritmo 2: Jerárquico aglomerativo.

Entrada:
M: matriz de distancias de tamaño $n \times n$
ce: criterio de enlace para calcular distancia d_{ce} entre clusters
Salida:
Z: matriz de enlace de tamaño $(n-1) \times 3$

```
inicio
    {Ω_1, Ω_2, ..., Ω_n} ← crear n clusters con un objeto cada uno
    Z ← crear matriz de enlace vacía
    repetir
        (i*, j*) ← arg min_{0≤i,j≤2n-2} d_ce(Ω_i, Ω_j)
        Z ← combinar los clusters Ω_{i*} y Ω_{j*} en el nuevo cluster Ω_{i*j*}
    hasta que quede un solo cluster de tamaño n
devolver Z
```

etiquetas $n, \ldots, 2n-2$. La tercer columna de Z representa la distancia entre los dos clusters al momento de la fusión (la cual se mide con un criterio de enlace, que se explicará a continuación). Por ejemplo, la matriz Z del dendrograma mostrado en la Figura 2.1 está dada por

$$Z = \begin{bmatrix} 1 & 2 & 0.447 \\ 4 & 8 & 0.538 \\ 0 & 9 & 0.539 \\ 6 & 7 & 0.860 \\ 3 & 12 & 0.974 \\ 10 & 13 & 1.107 \\ 5 & 14 & 1.623 \\ 11 & 15 & 3.666 \end{bmatrix}.$$

Esta matriz de enlace se utiliza luego para obtener una partición en particular con las características deseadas: por ejemplo, con un cierto número de clusters, o con los clusters que tengan una distancia o altura mínima determinada (como se ha mostrado con la línea punteada de la Figura 2.1).

2.1.3.2. Criterio de enlace

La fusión entre dos pares de clusters, formando uno nuevo, depende de la definición de una medida de distancia entre clusters, los cuales pueden poseer

Criterio de enlace	α_i	α_j	β	γ
Enlace simple	$1/2$	$1/2$	0	$-1/2$
Enlace completo	$1/2$	$1/2$	0	$1/2$
Enlace promedio (no pesado)	$\frac{n_i}{n_i+n_j}$	$\frac{n_j}{n_i+n_j}$	0	0
Enlace promedio (pesado)	$1/2$	$1/2$	0	0
Enlace basado en centroide (no pesado)	$\frac{n_i}{n_i+n_j}$	$\frac{n_j}{n_i+n_j}$	$\frac{-n_i n_j}{(n_i+n_j)^2}$	0
Enlace basado en centroide (pesado)	$1/2$	$1/2$	$-1/4$	0
Enlace de Ward	$\frac{n_i+n_\ell}{n_i+n_j+n_\ell}$	$\frac{n_j+n_\ell}{n_i+n_j+n_\ell}$	$\frac{-n_\ell}{n_i+n_j+n_\ell}$	0

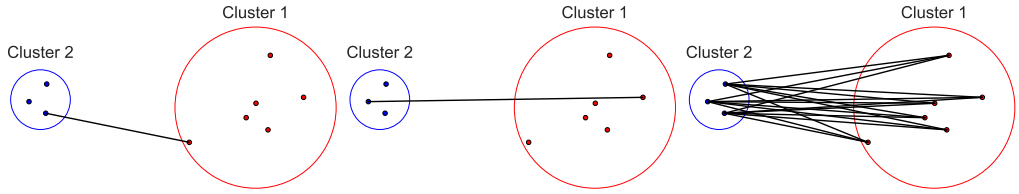
Tabla 2.1: Diferentes criterios de enlace y su distancia con la fórmula general recurrente de Lance y Williams. n_i, n_j y n_ℓ son la cantidad de objetos de los clusters Ω_i, Ω_j y Ω_ℓ, respectivamente.

más de un objeto. Estas medidas de distancia reciben el nombre de *criterios de enlace*, y son utilizadas en la línea 5 del Algoritmo 2. Existen varios criterios de enlace diferentes para medir la distancia entre un cluster Ω_ℓ y un nuevo cluster Ω_{ij} creado a partir de los clusters Ω_i y Ω_j. Estas distancias pueden generalizarse con la fórmula recurrente (Lance y Williams, 1967)

$$\begin{aligned} d(\Omega_\ell, (\Omega_i, \Omega_j)) =& \alpha_i d(\Omega_\ell, \Omega_i) + \alpha_j d(\Omega_\ell, \Omega_j) + \\ & \beta d(\Omega_i, \Omega_j) + \gamma \left| d(\Omega_\ell, \Omega_i) - d(\Omega_\ell, \Omega_j) \right|, \end{aligned} \tag{2.9}$$

donde d es una función de distancia y los parámetros $\alpha_i, \alpha_j, \beta$ y γ son coeficientes que toman diferentes valores de acuerdo al esquema que se utilice. Los esquemas más comúnmente utilizados para el criterio de enlace se listan en la Tabla 2.1 (Xu y Wunsch, 2009), y a continuación se da una descripción más detallada de cada uno.

Para el criterio de enlace simple, también conocido como "vecino más cercano", la distancia entre un par de clusters se determina por los dos objetos más cercanos, cada uno perteneciente a los diferentes clusters (Johnson, 1967). Esto puede verse gráficamente con un ejemplo en la Figura 2.2a. Siguiendo los pará-



(a) Enlace simple. (b) Enlace completo. (c) Enlace promedio.

Figura 2.2. Ejemplo del funcionamiento de los criterios de enlace simple, completo y promedio (no pesado).

metros de la Tabla 2.1 y a partir de (2.9) se obtiene

$$d(\Omega_\ell, (\Omega_i, \Omega_j)) = min(d(\Omega_\ell, \Omega_i), d(\Omega_\ell, \Omega_j)). \tag{2.10}$$

Este criterio de enlace tiene la particularidad de crear clusters "elongados" (Everitt y otros, 2011). Esto puede provocar la fusión de un par de clusters con propiedades muy diferentes, en particular bajo la presencia de ruido en los datos. Por lo tanto este esquema funciona bien siempre y cuando los clusters se encuentren suficientemente separados.

A diferencia del enlace simple, el criterio de enlace completo considera la distancia entre un par de clusters como la máxima distancia entre dos de sus puntos (Jain y Dubes, 1988), como muestra el ejemplo de la Figura 2.2b. Para este caso, (2.9) queda definida como

$$d(\Omega_\ell, (\Omega_i, \Omega_j)) = \text{máx}(d(\Omega_\ell, \Omega_i), d(\Omega_\ell, \Omega_j)). \tag{2.11}$$

El esquema de enlace promedio, también conocido como promedio no pesado de pares de grupo (UPGMA, por sus siglas en inglés), define la distancia entre dos clusters como el promedio de las distancias entre todos los pares de objetos de ambos clusters (Everitt y otros, 2011; Sokal y Michener, 1958). El funcionamiento del esquema se ilustra con un ejemplo en la Figura 2.2c. A partir de (2.9) se obtiene

$$d(\Omega_\ell, (\Omega_i, \Omega_j)) = \frac{1}{2}(d(\Omega_\ell, \Omega_i) + d(\Omega_\ell, \Omega_j)). \tag{2.12}$$

El criterio de enlace de promedio pesado es similar a UPGMA, ya que también utiliza el promedio de todas las distancias entre los puntos de ambos clusters. Sin embargo, la diferencia radica en que este método, al que también se conoce como WPGMA (por sus siglas en inglés), pesa las distancias de acuerdo

a la cantidad de objetos que haya en cada cluster fusionado (Jain y Dubes, 1988; McQuitty, 1966). El criterio se expresa como

$$d(\Omega_\ell, (\Omega_i, \Omega_j)) = \frac{n_i}{n_i + n_j} d(\Omega_\ell, \Omega_i) + \frac{n_j}{n_i + n_j} d(\Omega_\ell, \Omega_j). \tag{2.13}$$

El criterio de enlace basado en centroide no pesado se conoce también como centroide no pesado de pares de grupo (UPGMC, por sus siglas en inglés) (Everitt y otros, 2011; Sokal y Michener, 1958). El mismo tiene en cuenta el centroide de cada grupo, el cual se define como

$$\boldsymbol{\mu}_j = \frac{1}{n_j} \sum_{\forall i \in \Omega_j} \mathbf{x}_i, \tag{2.14}$$

donde n_j es la cantidad de objetos pertenecientes al cluster Ω_j. Para este caso, la ecuación queda definida como

$$\begin{aligned} d(\Omega_\ell, (\Omega_i, \Omega_j)) = & \frac{n_i}{n_i + n_j} d(\Omega_\ell, \Omega_i) + \frac{n_j}{n_i + n_j} d(\Omega_\ell, \Omega_j) - \\ & \frac{n_i n_j}{(n_i + n_j)^2} d(\Omega_i, \Omega_j). \end{aligned} \tag{2.15}$$

Por lo tanto, este método calcula la distancia entre dos clusters como la distancia euclídea al cuadrado entre sus centroides. Así, la definición anterior también puede expresarse como

$$d(\Omega_\ell, (\Omega_i, \Omega_j)) = \left\| \boldsymbol{\mu}_\ell - \boldsymbol{\mu}_{ij} \right\|^2. \tag{2.16}$$

El criterio de enlace basado en centroide pesado también se conoce como "enlace mediano" (*median linkage*, en inglés), y se refiere a él como WPGMC (por sus siglas en inglés) (Gower, 1967; Jain y Dubes, 1988). Es similar a UPGMC, con la diferencia de que se asigna el mismo peso a las distancias con los clusters que han sido fusionados. Se expresa formalmente como

$$d(\Omega_\ell, (\Omega_i, \Omega_j)) = \frac{1}{2} d(\Omega_\ell, \Omega_i) + \frac{1}{2} d(\Omega_\ell, \Omega_j) - \frac{1}{4} d(\Omega_i, \Omega_j). \tag{2.17}$$

El criterio de Ward también se conoce como el método de varianza mínima (Everitt y otros, 2011; Ward, 1963). La idea es minimizar la suma de los cuadrados de las diferencias,

$$E = \sum_{j=1}^{k} \sum_{\forall i \in \Omega_j} \left\| \mathbf{x}_i - \boldsymbol{\mu}_j \right\|^2, \tag{2.18}$$

donde k es la cantidad de clusters y $\boldsymbol{\mu}_j$ es el centroide del cluster Ω_j, tal como se definió en (2.14). Para este caso, (2.9) queda definida como

$$\begin{aligned} d(\Omega_\ell, (\Omega_i, \Omega_j)) = & \frac{n_i + n_\ell}{n_i + n_j + n_\ell} d(\Omega_\ell, \Omega_i) + \frac{n_j + n_\ell}{n_i + n_j + n_\ell} d(\Omega_\ell, \Omega_j) - \\ & \frac{n_\ell}{n_i + n_j + n_\ell} d(\Omega_i, \Omega_j). \end{aligned} \tag{2.19}$$

Los diferentes criterios de enlace tienen distintas propiedades (Xu y Wunsch, 2009). El enlace simple, completo y promedio (no pesado) consideran todos los puntos de un par de clusters al calcular su distancia, y son conocidos como *métodos de grafos*. Los otros métodos son llamados *geométricos* porque utilizan centros geométricos (los centroides) para representar a los clusters y medir sus distancias.

Entre las características de los algoritmos jerárquicos (particularmente los aglomerativos, que se han descripto), se pueden mencionar:

- Debido a la forma en que se agrupan los datos ("desde abajo hacia arriba"), los objetos agrupados incorrectamente en una instancia temprana no pueden ser reagrupados luego, lo que ocasiona a veces resultados de baja calidad.
- Al usarse diferentes criterios de enlace se obtienen diferentes resultados.
- Poseen una alta complejidad computacional, de al menos $O(n^2)$.

2.2. Medidas de validación

Una vez finalizada la etapa de aplicación del método de clustering y obtenidas las particiones de los datos, el siguiente paso consiste en validar los resultados. Esta tarea se realiza antes de que un experto del dominio de aplicación interprete los agrupamientos obtenidos. Debido a la naturaleza no supervisada de los métodos de clustering, al gran número de algoritmos que existen, y la combinación posible de sus parámetros, es necesario algún método de validación para escoger las mejores soluciones, según el criterio que se defina (Gordon, 1998; Halkidi y otros, 2001). Otro aspecto importante de la etapa de validación es la cantidad

de clusters en los datos, ya que encontrar el número óptimo es un problema no menor (Gordon, 1999; Jain y Dubes, 1988).

La literatura menciona dos tipos de criterios para evaluar las soluciones de clustering (Theodoridis y Koutroumbas, 2006; Xu y Wunsch, 2009): criterios externos e internos. Dado un conjunto de datos S y una solución de clustering π obtenida utilizando algún algoritmo aplicado a S, los *índices externos* comparan la solución π contra una estructura preespecificada, la cual representa información *a priori* sobre la estructura de los datos S. Un ejemplo es cuando ya se posee previamente una partición de referencia de los datos, como las tres clases de flores que forman el conjunto de datos Iris. Esta partición de referencia podría utilizarse para comparar las nuevas soluciones de clustering obtenidas. Los *índices internos*, a diferencia de los anteriores, evalúan la solución de clustering observando exclusivamente las propiedades de los datos y los grupos formados, sin depender de información externa sobre su estructura. Un ejemplo consiste en evaluar la varianza que existe dentro de los clusters, o cuán separados están de acuerdo a su centroide, entre otras.

Durante el desarrollo de este documento de tesis, nos referiremos también a estos índices como *medidas de clustering*, ya que son utilizados para cuantificar ciertas propiedades de la solución final de clustering (partición).

2.2.1. Índices externos

Un índice externo evalúa dos estructuras independientes de los datos $S = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$. La primera representa una partición de referencia $\pi' = \{\Omega'_1, \Omega'_2, \ldots, \Omega'_q\}$, la cual posee q clusters. La segunda estructura es la partición que queremos evaluar y que fue obtenida con un algoritmo de clustering: $\pi = \{\Omega_1, \Omega_2, \ldots, \Omega_k\}$, con k grupos. Así, un índice externo se aplica comparando las particiones π' y π. Generalmente, en la evaluación de un algoritmo de clustering se utilizan conjuntos de datos que proveen una partición de referencia contra la cual compararse.

Las diferentes medidas externas consideran el grado de solapamiento entre las dos particiones π' y π, el cual se resume en la matriz de contingencia $T = [n_{ij}]$ (Tabla 2.2), de tamaño $q \times k$. Aquí n_{ij} indica la cantidad objetos comunes en los clusters Ω'_i y Ω_j (Vinh y otros, 2010). A continuación se describirán los índices

$\pi' \setminus \pi$	Ω_1	Ω_2	$\dots$	Ω_k	Sumas
Ω'_1	n_{11}	n_{12}	$\dots$	n_{1k}	a_1
Ω'_2	n_{21}	n_{22}	$\dots$	n_{2k}	a_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
Ω'_q	n_{q1}	n_{q2}	$\dots$	n_{qk}	a_q
Sumas	b_1	b_2	$\dots$	b_k	$\sum_{ij} n_{ij} = n$

Tabla 2.2: Matriz de contingencia T, donde $n_{ij} = |\Omega'_i \cap \Omega_j|$.

más comúnmente utilizados: aquellos basados en el conteo de pares (Rand y Rand ajustado), y los basados en teoría de información (información mutua normalizada e información mutua ajustada).

Basados en conteo de pares

Este tipo de índices se basa en contar los pares de objetos en los que las particiones π' y π coinciden o no, de acuerdo a la matriz de contingencia (Tabla 2.2). De esta forma, surgen cuatro casos (Hubert y Arabie, 1985):

- N_{11}: el número de pares que están en el mismo cluster en ambas particiones π' y π.
- N_{00}: el número de pares que están en diferentes clusters en ambas particiones π' y π.
- N_{01}: el número de pares que están en el mismo cluster en π', pero en diferentes clusters en π.
- N_{10}: el número de pares que están en diferentes clusters en π', pero en el mismo cluster en π.

Índice de Rand. Este índice (Rand, 1971) se define como

$$RI(\pi', \pi) = \frac{N_{11} + N_{00}}{N_{11} + N_{00} + N_{01} + N_{10}} = \frac{N_{11} + N_{00}}{\binom{n}{2}}, \tag{2.20}$$

donde intuitivamente $N_{11} + N_{00}$ indica la cantidad de pares en los que π' y π coinciden, y $N_{01} + N_{10}$ la cantidad de pares en los que no coinciden. Este índice puede tener un valor entre 0 y 1, donde 0 indica que las particiones no coinciden en ningún par de puntos (en cuanto a cómo agruparlos), mientras que un valor de 1 indica que las particiones son exactamente iguales.

Índice ajustado de Rand. Este índice es una versión ajustada del índice de Rand (Hubert y Arabie, 1985). El hecho de que esté ajustado significa que el índice arroja un valor constante cuando se comparan dos particiones independientes, como por ejemplo dos soluciones de clustering elegidas totalmente al azar (Vinh y otros, 2010). Esta propiedad se conoce como ajuste para particiones aleatorias (*adjusted-for-chance*, en inglés), y no se cumple en el índice de Rand original. La forma general de un índice ajustado está dada por (Hubert y Arabie, 1985)

$$a(I) = \frac{I - E\{I\}}{max\{I\} - E\{I\}}, \tag{2.21}$$

donde $E\{I\}$ representa el valor esperado del índice para particiones aleatorias. El índice ajustado de Rand (ARI, por sus siglas en inglés) queda definido como

$$\Lambda(\pi', \pi) = \frac{2(N_{00}N_{11} - N_{01}N_{10})}{(N_{00} + N_{01})(N_{01} + N_{11}) + (N_{00} + N_{10})(N_{10} + N_{11})}. \tag{2.22}$$

Si bien el RI puede arrojar valores entre 0 y +1, esta versión ajustada puede devolver también valores negativos.

Basados en teoría de información

Este tipo de medidas utilizan conceptos fundamentales de teoría de información (Cover y Thomas, 2006). Dadas dos particiones π' y π, ambas con q y k clusters respectivamente, se definen sus entropías, entropías conjuntas, entropías condicionales y la información mutua de la siguiente forma

$$\begin{aligned}
H(\pi') &= -\sum_{i=1}^{q} \frac{a_i}{n} \log \frac{a_i}{n}, \\
H(\pi', \pi) &= -\sum_{i=1}^{q}\sum_{j=1}^{k} \frac{n_{ij}}{n} \log \frac{n_{ij}}{n}, \\
H(\pi' \mid \pi) &= -\sum_{i=1}^{q}\sum_{j=1}^{k} \frac{n_{ij}}{n} \log \frac{n_{ij}/n}{b_j/n}, \\
I(\pi', \pi) &= H(\pi') - H(\pi' \mid \pi) \\
&= H(\pi) - H(\pi \mid \pi') \\
&= \sum_{i=1}^{q}\sum_{j=1}^{k} \frac{n_{ij}}{n} \log \frac{n_{ij}/n}{a_i b_j / n^2},
\end{aligned}$$

donde n_{ij}, a_i y b_j son los valores que se muestran en la matriz de contingencia (Tabla 2.2).

Información mutua normalizada. La información mutua $I(\pi', \pi)$, la cual toma valores no negativos, puede utilizarse como la medida de similaridad más básica entre dos particiones. Esta medida, para diferentes precisiones, se puede acotar superiormente por distintos valores, y de ahí surgen diferentes variantes normalizadas (NMI, por sus siglas en inglés) (Meilă, 2005; Strehl y otros, 2002; Wu y otros, 2009). En esta tesis se ha utilizado una de las normalizaciones más difundidas, que está dada por

$$\Upsilon(\pi', \pi) = \frac{I(\pi', \pi)}{\sqrt{H(\pi')H(\pi)}}, \tag{2.23}$$

la cual toma valores posibles entre 0 y 1. Si las particiones comparadas son idénticas, el NMI alcanza un valor de 1.

Información mutua ajustada. De la misma forma que el ARI es la versión ajustada para el RI, existen diferentes versiones ajustadas para la información mutua (Vinh y otros, 2010). Para realizar el ajuste es necesario calcular el valor esperado de la información mutua entre las particiones π' y π, el cual está dado por (Vinh y otros, 2009)

$$E\{I(\pi', \pi)\} = \sum_{i=1}^{q}\sum_{j=1}^{k}\sum_{n_{ij}=max\{a_i+b_j-n,0\}}^{min(a_i,b_j)} \frac{n_{ij}}{n}\log(\frac{n\dot{n}_{ij}}{a_i b_j})\frac{a_i!b_j!(n-a_i)!(n-b_j)!}{n!n_{ij}!(a_i-n_{ij})!(b_j-n_{ij})!(n-a_i-b_j+n_{ij})!}. \tag{2.24}$$

De esta manera, utilizando la forma general dada en (2.21), se define la información mutua ajustada (AMI, por sus siglas en inglés) como (Vinh y otros, 2010)

$$\Psi(\pi', \pi) = \frac{I(\pi', \pi) - E\{I(\pi', \pi)\}}{max\{H(\pi'), H(\pi)\} - E\{I(\pi', \pi)\}}. \tag{2.25}$$

2.2.2. Índices internos

Compactitud. La compactitud es una medida que indica la homogeneidad de los clusters calculando la varianza interna de los mismos. La compactitud global para una partición está dada por

$$C = \frac{1}{k}\sum_{j}^{k} C_j = \frac{1}{k}\sum_{j}^{k}\left(\frac{1}{|\Omega_j|}\sum_{\forall i\in\Omega_j}\|\mathbf{x}_i - \boldsymbol{\mu}_j\|_2\right). \tag{2.26}$$

Valores de C cercanos a 0 indican clusters compactos.

Separación. Este índice interno cuantifica el grado de separación entre los clusters de una partición. La medida se define como

$$W = \frac{2}{k(k-1)} \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\|_2. \tag{2.27}$$

Valores altos de W indican una mayor separación de los clusters.

Davies-Bouldin. Esta medida combina la compactitud y la separación (Davies y Bouldin, 1979), y se define como

$$B = \frac{1}{k} \sum_{i=1}^{k} \max_{j \neq i} \left(\frac{C_i + C_j}{\|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\|_2} \right). \tag{2.28}$$

Este índice está en función de la relación entre la suma de las compactitudes de los clusters relativa a la separación entre los mismos. De esta forma, valores de B cercanos a 0 indican que la partición posee clusters compactos y separados.

Dunn. Este índice combina la disimilaridad entre los clusters y sus diámetros (Dunn, 1974), y está dado por

$$D = \frac{\min\limits_{0<m\neq n\leq k} \left\{ \min\limits_{\substack{\forall i \in \Omega_m \\ \forall j \in \Omega_n}} \{\|\mathbf{x}_i - \mathbf{x}_j\|_2\} \right\}}{\max\limits_{0<m\leq k} \max\limits_{\forall i,j \in \Omega_m} \{\|\mathbf{x}_i - \mathbf{x}_j\|_2\}}. \tag{2.29}$$

Este índice mide la relación entre el valor mínimo de separación entre clusters y la máxima varianza dentro de los mismos. Si la partición contiene clusters separados, la distancia mínima entre ellos es usualmente grande y es esperable que su máximo diámetro sea pequeño. De esta forma, valores altos de D indican mejores particiones.

2.3. Ensambles de agrupamientos

Los ensambles de agrupamientos han tenido un gran desarrollo en los últimos años (Fiori y otros, 2016; Fred y Jain, 2005; Iam-On y otros, 2012; Strehl y otros, 2002). Estas técnicas han nacido para mitigar los inconvenientes que surgen de los métodos tradicionales de clustering, como la dificultad para el usuario no experto de seleccionar un determinado algoritmo o un conjunto específico de

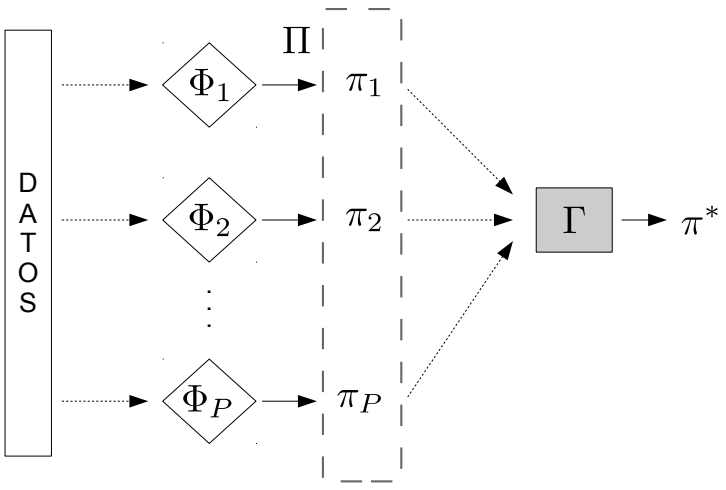


Figura 2.3. El proceso de ensambles de agrupamientos y sus componentes (basado en (Strehl y otros, 2002)).

parámetros. Este novedoso enfoque, conocido también como "clustering de consenso", tiene básicamente dos pasos (Iam-On y otros, 2011; Topchy y otros, 2005): primero se genera un *ensamble de agrupamientos*, es decir, un conjunto de soluciones de clustering generadas con algoritmos tradicionales; y luego este ensamble es procesado por una *función de consenso* que deriva de él una nueva partición consensuada, que generalmente supera la calidad de todas las soluciones miembro del ensamble de entrada (Huang y otros, 2016, 2011; Yu y otros, 2012b).

El proceso general de los ensambles de agrupamientos puede verse en la Figura 2.3, donde se muestran sus diferentes componentes, entradas y salidas. Con acceso directo a los datos (los objetos y sus características), los *clusterers* Φ_i generan soluciones de clustering π_i, y el conjunto de todas ellas es un ensamble Π de tamaño P. Un clusterer representa una instancia de un algoritmo de clustering con un conjunto específico de parámetros, como por ejemplo un k-medias con $k = 5$, un k-medias con $k = 2$, o un jerárquico aglomerativo con criterio de enlace simple y $k = 7$. El ensamble Π es la entrada de la función de consenso Γ, la cual genera una partición consensuada π^*. Es importante notar que si bien los clusterers tienen acceso a todos los datos, la función de consenso solo necesita acceder al ensamble, es decir que no requiere acceder a los objetos y sus características.

2.3.1. Generación del ensamble

Existen diferentes formas de generar un ensamble (Iam-On y otros, 2011; Kuncheva y Vetrov, 2006), donde se utilizan básicamente dos enfoques diferentes. El primer enfoque consiste en cambiar los datos, incluyendo estrategias como:

- **Muestreo de características.** En lugar de tomar todas las características para hacer clustering, se seleccionan distintos subconjuntos de ellas, generando diferentes versiones del conjunto de datos en las que se corren los algoritmos (Jing y otros, 2015; Topchy y otros, 2003; Wu y otros, 2015; Yu y otros, 2016, 2007). Por ejemplo, de un conjunto de datos con 30 atributos, se podrían generar varios subconjuntos tomando 3 atributos a la vez y realizando clustering sobre cada uno de ellos.

- **Muestreo de objetos.** En este caso se muestrean los objetos, generando subconjuntos de datos de un tamaño menor a n (Fischer y Buhmann, 2003; Minaei-Bidgoli y otros, 2004; Topchy y otros, 2004; Yang y Jiang, 2016). Es importante notar que, ya que en este caso las particiones generadas no poseerán información de agrupamiento sobre todos los objetos, la función de consenso necesitará que exista algún tipo de solapamiento de los objetos dentro del ensamble (Strehl y otros, 2002).

- **Proyecciones de los datos.** En lugar de seleccionar diferentes subconjuntos de objetos o características, esta opción aplica una transformación en los datos (por ejemplo, una transformación lineal), proyectándolos a un nuevo sistema de coordenadas. Las proyecciones podrían hacerse con técnicas conocidas como el análisis de componentes principales (Jolliffe, 2002), o utilizando proyecciones aleatorias (Fern y Brodley, 2003).

El otro enfoque consiste en cambiar los algoritmos y sus parámetros (Fred y Jain, 2002; Kuncheva y Vetrov, 2006; Yang y Chen, 2011). Algunas estrategias dentro de este conjunto son:

- **Utilizar el mismo algoritmo con diferentes inicializaciones.** Esta estrategia (Hadjitodorov y otros, 2006; Zhong y otros, 2015) se utiliza comúnmente con el algoritmo k-medias, donde se mantiene fijo el parámetro k (generalmente se utiliza como regla $k = \sqrt{n}$) y se varía aleatoriamente

la posición inicial de los centroides. El objetivo de esta estrategia consiste en generar una mayor cantidad de particiones diferentes con una misma cantidad de clusters.

- **Utilizar el mismo algoritmo variando sus parámetros.** En este caso se varían los parámetros de mayor importancia en el algoritmo, intentando generar una mayor diversidad (Fred y Jain, 2005; Huang y otros, 2016). En el caso de k-medias este parámetro es generalmente el número de clusters k. La idea es que el ensamble posea particiones con diferente cantidad de clusters, y así diferentes puntos de vista sobre los agrupamientos.

- **Variar el algoritmo.** Esta estrategia (Law y otros, 2004; Strehl y otros, 2002) intenta agregar aún más diversidad utilizando algoritmos heterogéneos que, como se ha explicado anteriormente, tienen diferentes suposiciones sobre la distribución de los datos y la forma de los clusters a encontrar.

Es importante que el ensamble posea algún tipo de diversidad entre sus particiones miembros (Fern y Lin, 2008; Hu y otros, 2016; Iam-On y otros, 2011). Como se ha mencionado en el Capítulo 1, si bien la diversidad es una cuestión compleja en sí misma y será tratada más adelante, es claro que la función de consenso no puede extraer agrupamientos novedosos de un ensamble completamente homogéneo.

2.3.2. Funciones de consenso

Una vez generado el ensamble, el paso siguiente consiste en combinarlo para generar una única partición de los datos (Topchy y otros, 2005). Básicamente, dado un ensamble con P particiones, una función de consenso es definida como $\Gamma(\Pi) \rightarrow \pi^*$, donde el ensamble Π se representa como una matriz de enteros de tamaño $P \times n$, y π^* es la partición consensuada de $1 \times n$. De esta forma, un conjunto de P soluciones de clustering sobre n objetos es mapeado a una única solución combinada.

Existen varios enfoques para consensuar las decisiones de agrupamiento contenidas en el ensamble, y a continuación se detallan dos de los enfoques más utilizados: por un lado los que emplean teoría de grafos y, por el otro, los que se basan en acumulación de evidencia.

2.3.2.1. Basadas en teoría de grafos

Las funciones de consenso basadas en teoría de grafos son la primera familia de métodos que surgieron para el problema de los ensambles de agrupamientos (Strehl y otros, 2002). Estas funciones suponen que todas las particiones del ensamble son igualmente importantes, y por lo tanto intentan buscar una solución de consenso que de alguna manera maximice la información compartida con el ensamble. Para lograr esto, se define una medida que compara un ensamble Π con una partición de consenso π^*, basándose en la información mutua normalizada (NMI) (definida en (2.23)). Básicamente, se calcula el promedio del NMI entre la partición π^* y cada miembro del ensamble Π. Esta medida recibe el nombre de ANMI (*Average Normalized Mutual Information*, en inglés) y queda definida como

$$\bar{\Upsilon}(\pi^*, \Pi) = \frac{1}{P} \sum_{\forall \pi_i \in \Pi} \Upsilon(\pi^*, \pi_i). \tag{2.30}$$

Por lo tanto, la función objetivo de este tipo de métodos de consenso se define como

$$\pi^* = \arg\max_{\check{\pi}} \sum_{i=1}^{P} \Upsilon(\check{\pi}, \pi_i). \tag{2.31}$$

Esta función objetivo representa un problema de optimización combinatorio muy difícil. Por ejemplo, existen más de 170 millones de formas diferentes de agrupar 16 objetos en 4 clusters (Jain y Dubes, 1988). Por esta razón, en (Strehl y otros, 2002) se proponen tres funciones de consenso basadas en heurísticas intuitivas más que métodos de optimización directos.

Estos métodos primero transforman el ensamble de entrada en una representación de hipergrafos. Un hipergrafo es una generalización de un grafo, y consiste en vértices e hiperaristas. Si bien una arista en un grafo regular conecta exactamente dos vértices, una hiperarista puede conectar cualquier *conjunto* de vértices. En la Tabla 2.3 se muestra un ensamble de ejemplo y su representación como hipergrafo. Para cada partición π_i del ensamble se construye una matriz de membresía binaria H_i, con una columna por cada cluster (que ahora se representa como una hiperarista). Puede suceder que una determinada partición sea parcial (como π_4), donde no todos los objetos fueron agrupados. Para este caso

	π_1	π_2	π_3	π_4			H_1			H_2			H_3			H_4	
							h_1	h_2	h_3	h_4	h_5	h_6	h_7	h_8	h_9	h_{10}	h_{11}
x_1	1	2	1	1		v_1	1	0	0	0	1	0	1	0	0	1	0
x_2	1	2	1	2		v_2	1	0	0	0	1	0	1	0	0	0	1
x_3	1	2	2	?	$\Leftrightarrow$	v_3	1	0	0	0	1	0	0	1	0	0	0
x_4	2	3	2	1		v_4	0	1	0	0	0	1	0	1	0	1	0
x_5	2	3	3	2		v_5	0	1	0	0	0	1	0	0	1	0	1
x_6	3	1	3	?		v_6	0	0	1	1	0	0	0	0	1	0	0
x_7	3	1	3	?		v_7	0	0	1	1	0	0	0	0	1	0	0

Tabla 2.3: Ejemplo ilustrativo de un ensamble de agrupamientos con $P = 4$ particiones, donde π_1, π_2 y π_3 poseen 3 clusters, y π_4 posee 2 clusters (tomado de (Strehl y otros, 2002)). A la izquierda se muestran las particiones originales, y a la derecha la representación en un hipergrafo con 11 hiperaristas.

particular, en el que para una determinada partición π_i no se conoce a qué cluster pertenece un objeto x_j, la entrada correspondiente a la fila v_j de la matriz H_i es igual a 0.

Llamamos a $H = (H_1, \ldots, H_P)$ como la matriz de adyacencia de un hipergrafo con n vértices y tantas hiperaristas como la suma de la cantidad de clusters de todas las particiones miembro del ensamble. Cada columna h_j hace referencia a una hiperarista, donde un valor de 1 indica que el vértice correspondiente a la fila es parte de dicha hiperarista, y un 0 indica que no lo es. De esta manera, cada cluster ha sido mapeado a una hiperarista, y por lo tanto el ensamble completo a un hipergrafo.

Algoritmo de particionamiento por similaridad de clusters

Este método, conocido como CSPA (*Cluster-based Similarity Partitioning Algorithm*, en inglés), es uno de los más sencillos. El mismo parte de la idea de que una partición de los datos indica una relación entre los objetos de un mismo cluster, y por lo tanto esta idea se puede utilizar para establecer una medida de similaridad par a par. Esta *medida inducida de similaridad* se utiliza luego para volver a hacer clustering sobre los mismos objetos, lo que produce una solución consensuada.

La nueva medida consiste en asignar un valor de 1 cuando dos objetos pertenecen al mismo cluster, y un valor de 0 en caso contrario. Por lo tanto, para cada partición π_i se crea una matriz de similaridad de $n \times n$. El promedio de las P matrices, las cuales representan cada una a las P particiones del ensamble,

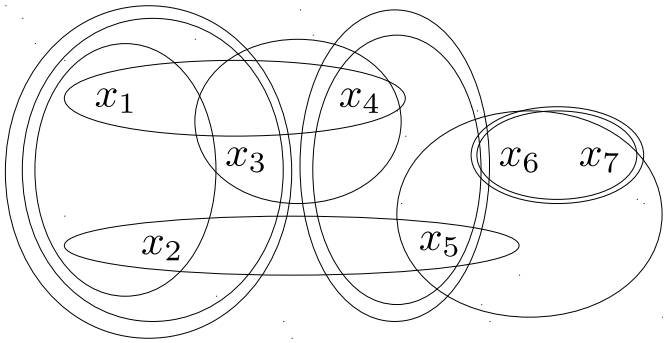


Figura 2.4. Ejemplo del algoritmo HGPA para el problema de ensambles mostrado en la Tabla 2.3 (tomado de (Strehl y otros, 2002)). Cada hiperarista se muestra como una curva cerrada que abarca los vértices que conecta.

producen una matriz de similaridad general Q. Cada entrada de la matriz Q, por lo tanto, tiene el promedio de veces en que dos objetos estuvieron en el mismo cluster, y esta puede ser calculada directamente como $Q = \frac{1}{P} H H^\top$.

Una vez calculada la matriz Q, se puede utilizar cualquier algoritmo de clustering basado en matrices de similaridad. En el caso de CSPA se transforma la matriz Q en un grafo donde cada vértice representa un objeto de los datos, y el peso de cada arista es la similaridad entre un par de vértices (objetos). Luego se aplica un algoritmo de particionamiento de grafos (Karypis y Kumar, 1998).

Si bien CSPA es el algoritmo más simple y representa la heurística más rápidamente deducible, tanto su costo computacional como de memoria son cuadráticos en n. Los siguientes dos algoritmos, en cambio, son aproximadamente lineales en n.

Algoritmo de particionamiento de hipergrafos

Para este algoritmo, llamado HGPA (*HyperGraph Partitioning Algorithm*, en inglés), se aplica un enfoque donde los datos son reparticionados considerando a los clusters como vínculos fuertes entre los objetos. De esta forma, el problema de los ensambles de agrupamientos se formula como el particionamiento de un hipergrafo donde se intenta cortar el mínimo número de hiperaristas. Todas las hiperaristas tienen el mismo peso, al igual que los vértices. A diferencia de CSPA, donde se considera solamente la relación de par a par, en este caso se tiene en cuenta la relación entre todos los objetos de cada cluster. Como puede verse en la Figura 2.4, aquí se intenta buscar un separador de hiperaristas que particione

el grafo en k componentes no conectados de aproximadamente el mismo tamaño.

El algoritmo utilizado para particionar el hipergrafo es el propuesto en (Karypis y otros, 1999), el cual genera particiones de alta calidad y es escalable. Sin embargo, este tipo de algoritmos en general no provee información sobre cortes parciales de los hipergrafos, y esto puede ser problemático al aplicarlo sobre ensambles de agrupamientos. Esto significa que, luego de realizar el particionamiento, no hay sensibilidad por la cantidad de vértices de la hiperarista que se dejan en el mismo grupo. En la práctica, esto puede hacer que el algoritmo elija particionamientos de menor calidad. Otra característica es que los grupos formados luego del particionamiento tienen tamaños similares (Strehl y Ghosh, 2003), por lo tanto si los datos poseen clusters altamente irregulares en tamaño, este enfoque puede no ser conveniente.

Algoritmo de meta-clustering

El tercer algoritmo es MCLA (*Meta-CLustering Algorithm*, en inglés), y se basa en hacer clustering sobre los clusters. Además de devolver una partición consensuada de los datos, también ofrece información sobre la confianza acerca de la pertenencia de un objeto a su cluster.

El primer paso de este algoritmo es transformar cada hiperarista h_i de H (es decir, las 11 hiperaristas que se ven en la Tabla 2.3) en vértices de otro grafo regular no dirigido, que aquí se llama meta-grafo. Las aristas de este meta-grafo tienen un peso que es proporcional a la similaridad de sus vértices. Una medida de similaridad apropiada en este caso es el índice de Jaccard binario, ya que representa la relación entre la intersección y la unión de dos conjuntos de objetos. Formalmente, el peso de una arista $w_{a,b}$ entre los vértices h_a y h_b queda definido como

$$w_{a,b} = \frac{h_a^\top h_b}{\|h_a\|_2^2 + \|h_b\|_2^2 - h_a^\top h_b}. \tag{2.32}$$

Ya que los clusters originales no se solapan (particionamiento exclusivo), no existen aristas entre los vértices de una misma matriz de membresía binaria H_q y, por lo tanto, el meta-grafo es P-partito.

Una vez formado el meta-grafo, el mismo es particionado en k meta-clusters. Para esto se utiliza el mismo algoritmo de particionamiento de grafos que en

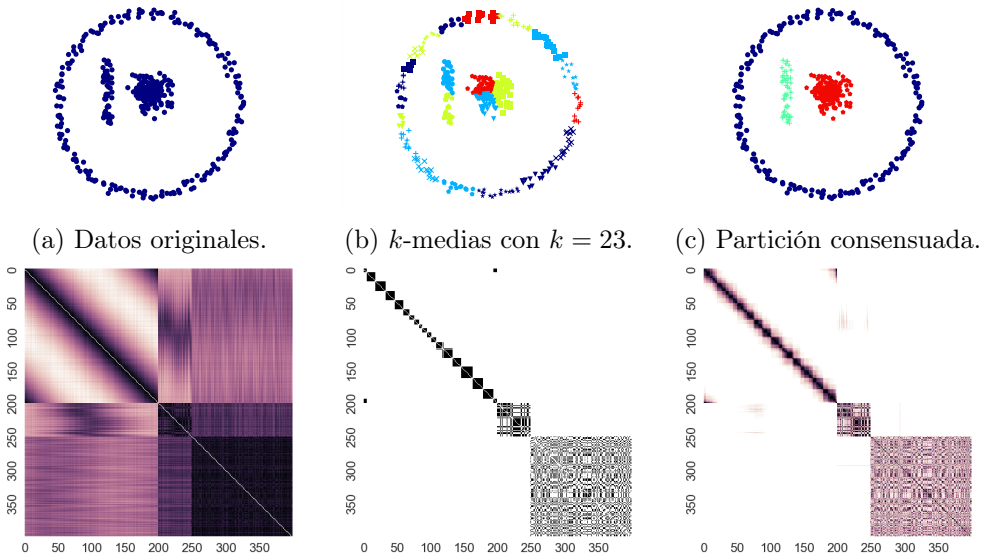
CSPA. El resultado es una solución de clustering donde se han agrupado los vectores h. Cada meta-cluster tiene aproximadamente P vértices, y ya que cada vértice en el meta-grafo corresponde a un cluster del ensamble de entrada, un meta-cluster es un conjunto de clusters. Para cada uno de los k meta-clusters, las hiperaristas h_i se "pliegan" en una meta-hiperarista. Cada meta-hiperarista posee un vector de asociación que contiene una entrada por cada objeto describiendo su nivel de asociación con el meta-cluster correspondiente. Este nivel de asociación se calcula haciendo el promedio de cada h_i del meta-cluster[1]. Finalmente, cada meta-cluster compite por los objetos, es decir, cada objeto es asignado al meta-cluster con el cual tiene una mayor asociación. La confianza de la asociación de un objeto a un meta-cluster queda reflejada por el nivel de asociación. Es importante notar que no hay garantía de que todos los meta-clusters ganen al menos un objeto. Por lo tanto, MCLA devuelve una partición consensuada π^* con k grupos como máximo.

2.3.2.2. Basadas en acumulación de evidencia

Esta clase de métodos de consenso se basan en la idea de *acumulación de evidencia* (EAC, por sus siglas en inglés) para combinar o consensuar un ensamble de agrupamientos (Fred y Jain, 2002, 2005). En esta sección se describirá primero cómo estos enfoques representan los agrupamientos del ensamble acumulando la información de cada partición miembro en una matriz de similaridad de $n \times n$. Luego, se explicarán los métodos de consenso basados en clustering jerárquico que se aplican sobre esta, los que producen una partición consensuada.

Dado que un ensamble puede poseer particiones con diferentes cantidades de clusters, este enfoque de consenso propone un mecanismo de votación para realizar la combinación del ensamble, el cual resulta en una nueva medida de similaridad entre los objetos. La suposición de fondo es que los objetos que pertenezcan a clusters "naturales" probablemente sean agrupados juntos en las diferentes particiones miembro del ensamble. Considerando las coocurrencias de pares de objetos en el mismo cluster como votos a favor de su asociación, las P particiones de los n objetos son mapeadas a una matriz de coasociación de tamaño $n \times n$

[1]Un valor de 0 o 1 indica la asociación más débil o más fuerte, respectivamente.



(a) Datos originales. (b) k-medias con $k = 23$. (c) Partición consensuada.

(d) Matriz de similaridad de los datos. (e) Matriz de coasociación de k-medias (b). (f) Matriz de coasociación final de EAC.

Figura 2.5. Ejemplo de la aplicación de los métodos de acumulación de evidencia para consensuar un ensamble de particiones hecho con k-medias sobre datos artificiales. El ejemplo ha sido reproducido de (Fred y Jain, 2005).

$$\mathcal{C}(i,j) = \frac{n_{ij}}{P}, \tag{2.33}$$

donde n_{ij} es el número de veces que el par de objetos $(\mathbf{x}_i, \mathbf{x}_j)$ es asignado al mismo cluster entre las P particiones del ensamble.

Un ejemplo de cómo funciona el proceso de acumulación de evidencia se ilustra en la Figura 2.5, donde se utiliza un conjunto de datos artificialmente generado con tres clusters y 400 objetos (Figura 2.5a). El primer grupo corresponde a un anillo exterior, el cual posee 200 objetos; el segundo, de 50 objetos, tiene forma rectangular; y finalmente el último grupo corresponde a puntos normalmente distribuidos, y posee 150 objetos. Con el propósito de comparar luego con la matriz de coasociación (2.33), se ha graficado en la Figura 2.5d la matriz de similaridad de los datos originales, donde $sim(\mathbf{x}_i, \mathbf{x}_j) = \max_{\ell,m}\{d_e(\mathbf{x}_\ell, \mathbf{x}_m)\} - d_e(\mathbf{x}_i, \mathbf{x}_j)$. Esta matriz posee en las coordenadas los índices de los datos: 1-200 corresponde al anillo exterior, 201-250 al rectángulo, y 251-400 al cluster gaussiano. Los colores van del blanco (similaridad nula) a negro (máxima similaridad).

Sobre estos datos, se ha generado un ensamble con 30 particiones ($P = 30$) utilizando el algoritmo k-medias con inicializaciones aleatorias de los centroides, y eligiendo un valor de k aleatorio dentro del rango $[10, 30]$. En la Figura 2.5b se

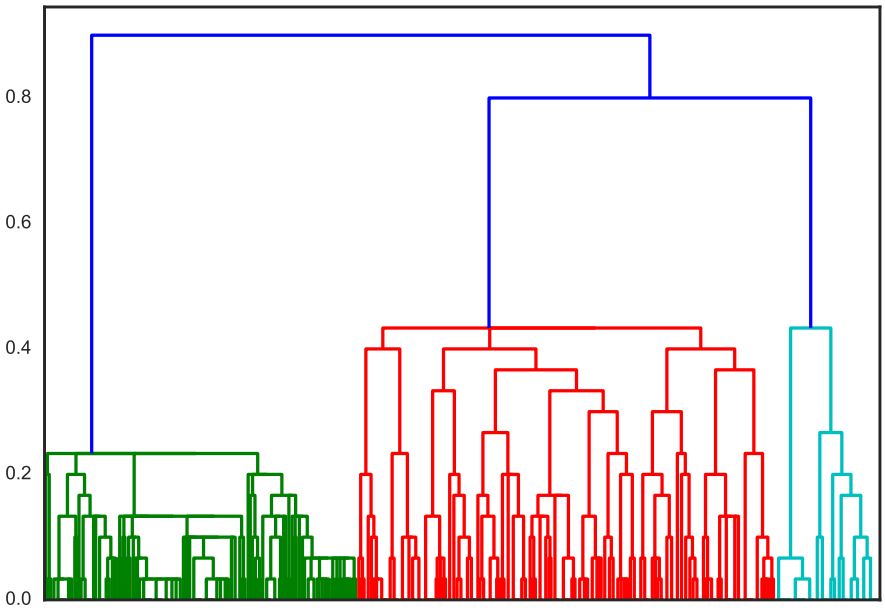


Figura 2.6. Dendrograma derivado de aplicar un criterio de enlace simple sobre la matriz de coasociación mostrada en la Figura 2.5f. Las distancias se muestran en el eje vertical.

puede ver una partición miembro del ensamble con $k = 23$, y en la Figura 2.5e se muestra su matriz de coasociación correspondiente (en este caso, para esa sola partición) donde el color negro indica que el par de objetos está en el mismo cluster. En esta matriz, por ejemplo, puede observarse en el sector inferior derecho los puntos blancos que indican cómo el cluster gaussiano ha sido dividido en clusters más pequeños. La matriz de coasociación final, que acumula los votos de todas las particiones miembro del ensamble, puede verse en la Figura 2.5f. A diferencia de la matriz de similaridad de los datos originales (Figura 2.5d), en la matriz final de coasociación la estructura original de los datos resulta mucho más evidente, donde los clusters se encuentran más claramente separados.

Una vez obtenida la matriz de coasociación se aplica un algoritmo de clustering jerárquico sobre ella, derivando así una partición de consenso final. Para el ejemplo anterior, se aplicó este algoritmo utilizando el criterio de enlace simple, lográndose así recuperar la estructura original de los datos, como puede verse en la Figura 2.5c. En la Figura 2.6 se muestra el dendrograma resultante para este ejemplo, donde se ve reflejada claramente la estructura con tres grupos en los datos originales.

El mecanismo de acumulación de evidencia, por lo tanto, mapea las par-

ticiones en el ensamble de entrada a una nueva medida de similaridad entre los objetos, la cual se resume en la matriz de coasociación. Este cambio corresponde intrínsecamente a una transformación no lineal del espacio original de los datos a una nueva representación, sobre la que cualquier algoritmo de clustering basado en medidas de similaridad puede ser aplicado.

2.3.2.3. Supraconsenso

Dado que existen varias funciones de consenso, y que un criterio común para elegir una partición consensuada es medir la consistencia de esta con el ensamble de entrada (Fred y Jain, 2005; Strehl y otros, 2002), es posible definir una función de *supraconsenso*. Esta función, que trabaja de forma no supervisada, escoge entre las particiones consensuadas a aquella que maximiza algún criterio de consistencia. El criterio más común es el promedio de la información mutua normalizada entre la partición de consenso y cada miembro del ensamble (ANMI), que ya ha sido definido en (2.30).

La función de supraconsenso se configura con un conjunto de métodos de consenso a aplicar, y se le da como parámetro un ensamble de agrupamientos. La función finalmente devuelve la partición consensuada que maximiza (2.30).

2.3.3. Medidas de diversidad en los ensambles de agrupamientos

Durante el desarrollo de la propuesta de los ensambles de agrupamientos, ha surgido en la comunidad la pregunta de qué características afectan el desempeño de la función de consenso. Si bien los trabajos iniciales, como se ha mencionado antes, han intuido la necesidad de que exista cierto tipo de heterogeneidad en el ensamble, los estudios sobre esta cuestión han arribado al concepto de diversidad, aunque sin una clara definición al respecto. En estos trabajos se ha intentado explicar la calidad de la partición de consenso en función de la diversidad del ensamble, e intentar a partir de allí encontrar un nivel de diversidad óptimo. Esta cuestión se analizará más profundamente en el siguiente capítulo, y aquí se describirán las diferentes medidas de diversidad que la comunidad ha propuesto para cuantificar el nivel de disimilaridad entre las particiones miembro del ensamble.

Durante el transcurso de este documento, nos referiremos a las medidas de

diversidad también como *medidas de ensambles*, ya que estas cuantifican ciertas características del ensamble, es decir, de la entrada a la función de consenso. De esta forma se diferencian de las medidas de clustering definidas anteriormente, que cuantifican determinadas propiedades de la salida del consenso, es decir, la partición consensuada.

2.3.3.1. Medidas internas de diversidad

Dado un ensamble Π de tamaño P, una primera medida de diversidad consiste en calcular la disimilaridad entre las particiones miembro del ensamble (Fern y Brodley, 2003), la cual se define como el promedio de las disimilaridades de las particiones en Π,

$$D_{\mathrm{I1}} = \frac{2}{P(P-1)} \sum_{i=1}^{P-1} \sum_{j=i+1}^{P} (1 - \Upsilon(\pi_i, \pi_j)). \tag{2.34}$$

La segunda medida, llamada D_{I2}, consiste en cuantificar la propagación de diversidad (Hadjitodorov y otros, 2006), la cual se obtiene calculando la desviación estándar de los valores de disimilaridad entre las particiones miembro del ensamble.

2.3.3.2. Medidas externas de diversidad

El segundo grupo de medidas de diversidad tienen en cuenta la diferencia entre las particiones miembro del ensamble y alguna partición consensuada (Hadjitodorov y otros, 2006; Kuncheva y Hadjitodorov, 2004). De esta forma se define la medida D_{E1} como el promedio de las diferencias entre cada partición del ensamble y una solución de consenso π^*. Esto es,

$$D_{\mathrm{E1}} = \frac{1}{P} \sum_{\forall \pi_i \in \Pi} 1 - \Upsilon(\pi^*, \pi_i). \tag{2.35}$$

Nuevamente, una medida alternativa se obtiene al calcular la desviación estándar de estas diferencias, la cual se denomina D_{E2}.

CAPÍTULO 3

Control de la diversidad en ensambles de agrupamientos

Un hombre solo se conoce a sí mismo en la acción, durante la acción, mientras está en acción.

— Luigi Giussani

En este capítulo se describirá un método para controlar la diversidad de los ensambles de agrupamientos. Como se ha explicado en el Sección 1.1, aunque el clustering de consenso ha representado un gran avance sobre las técnicas tradicionales de clustering, aún existe un problema importante relacionado con la diversidad de los ensambles, dado que no se conoce con precisión cómo esta afecta los resultados del consenso. Si bien muchos trabajos se han abocado a este estudio, los resultados presentados en ellos no han logrado responder al interrogante sobre qué relación existe entre la diversidad de los ensambles y el desempeño final de la partición de consenso.

La razón fundamental detrás de esta falta de respuesta es que ningún enfoque actual logra producir un comportamiento suave de la función de consenso, y por lo tanto es muy difícil estudiar cómo la diversidad impacta en las soluciones consensuadas. El método de control de la diversidad presentado en este capítulo logra cambiar de manera suave la diversidad de los ensambles, e inducir así un comportamiento suave en la función de consenso. Este enfoque propuesto, al generar ensambles con diversidad controlada, permite facilitar futuros estudios sobre este problema, representando así un importante aporte para el área.

3.1. Introducción

Se han utilizado varias estrategias para explorar cómo la diversidad afecta el desempeño del consenso (Fern y Brodley, 2003; Iam-On y otros, 2011), el cual suele medirse como la exactitud de la solución consensuada con respecto a una partición de referencia (por ejemplo, las clases de los datos). En estas estrategias actuales se genera un conjunto de ensambles con diferentes diversidades como paso previo a observar la exactitud de la función de consenso. Uno de los enfoques más comunes incluye la generación de los miembros del ensamble variando aleatoriamente un parámetro del proceso, como se ha explicado en la Sección 2.3.1. Este parámetro puede ser el algoritmo de clustering mismo (Law y otros, 2004; Strehl y otros, 2002), el número de clusters (Fred y Jain, 2002; Yang y Chen, 2011; Zhong y otros, 2015) o la inicialización (Kuncheva y Vetrov, 2006; Topchy y otros, 2003). En lugar de cambiar el algoritmo, otro método comúnmente utilizado consiste en cambiar los datos eligiendo aleatoriamente subconjuntos de los objetos (Fischer y Buhmann, 2003; Minaei-Bidgoli y otros, 2004; Topchy y otros, 2004; Yang y Jiang, 2016), utilizando diferentes características (Jing y otros, 2015; Topchy y otros, 2003; Wu y otros, 2015; Yu y otros, 2016, 2007), transformando los datos mediante proyecciones aleatorias (Fern y Brodley, 2003; Topchy y otros, 2003, 2005) o combinando varios de estos métodos (Yang y otros, 2014; Yu y otros, 2012a). Una alternativa al método puramente aleatorio consiste en generar un ensamble que maximice algún criterio dado. En este caso, primero se crea un grupo inicial de particiones utilizando alguno de los métodos enumerados antes y, luego, un subconjunto de este grupo es seleccionado, el cual maximiza la función objetivo. Esta función objetivo puede definirse para generar ensambles con diversidad baja, media y alta, como en (Iam-On y otros, 2011).

Los métodos mencionados han sido ampliamente utilizados para explorar la disimilaridad dentro de un ensamble, pero los resultados obtenidos indican que existe un problema importante con estos enfoques. Ciertamente, los estudios previos ofrecen opiniones opuestas acerca de este problema, donde algunos sugieren que los ensambles más diversos son mejores para obtener soluciones más exactas (Fern y Brodley, 2003; Iam-On y otros, 2011); otros han propuesto los ensambles de diversidad moderada como la opción preferida (Hadjitodorov y otros, 2006); incluso existen trabajos donde, si bien se afirma la importancia de la di-

versidad en los resultados, también se concluye que no existe una relación clara entre esta y la exactitud de las soluciones consensuadas (Hu y otros, 2016). Además de estos resultados contradictorios, se ha reportado una alta variabilidad no solo al utilizar diferentes conjuntos de datos, sino también al emplear distintas estrategias de generación de los ensambles o diferentes funciones de consenso. Asimismo, las gráficas de exactitud en función de la diversidad muestran que los ensambles con diversidades similares pueden diferir mucho en la exactitud que producen (Hadjitodorov y otros, 2006; Iam-On y otros, 2011). Estos resultados confusos muestran que los enfoques actuales ciertamente pueden generar diversidad, pero no pueden controlarla, y esta limitación puede llevar a la función de consenso a producir salidas impredecibles. Este es un problema importante y debe ser abordado antes de realizar cualquier análisis del impacto de la diversidad en el clustering de consenso. El comportamiento impredecible sucede porque mientras se observa una única medida de diversidad, otras propiedades del ensamble cambian, y de ese modo se produce un comportamiento errático de la función de consenso. Además, es difícil generar ensambles que estén uniformemente distribuidos en el rango de diversidad bajo evaluación, lo que puede provocar un análisis sesgado. Estas dos razones demuestran que es necesario controlar la diversidad de los ensambles para poder analizar efectivamente su impacto en los resultados de consenso.

Debido a la importancia de la diversidad en los ensambles de agrupamientos y los problemas antes indicados, en el marco de esta tesis doctoral se definió y exploró un nuevo problema en el área, proponiendo un novedoso método que permite tener un control fino de la diversidad (o nivel de desacuerdo/disimilaridad) del ensamble. Hasta el momento, no se han propuesto métodos para controlar el nivel de desacuerdo entre los miembros de un ensamble. El enfoque que se desarrolla en este capítulo extrae información desde la estructura del ensamble y luego la utiliza para realizar pequeños cambios, los cuales permiten disminuir o aumentar la diversidad del ensamble de una manera suave. Los resultados obtenidos usando seis conjuntos de datos conocidos en el área demuestran que este método es efectivo para controlar la diversidad del ensamble, el cual logra que la función de consenso se comporte de una manera suave. Esto provee un novedoso enfoque para realizar en el futuro mejores estudios sobre el impacto de la diversidad en el clustering de consenso.

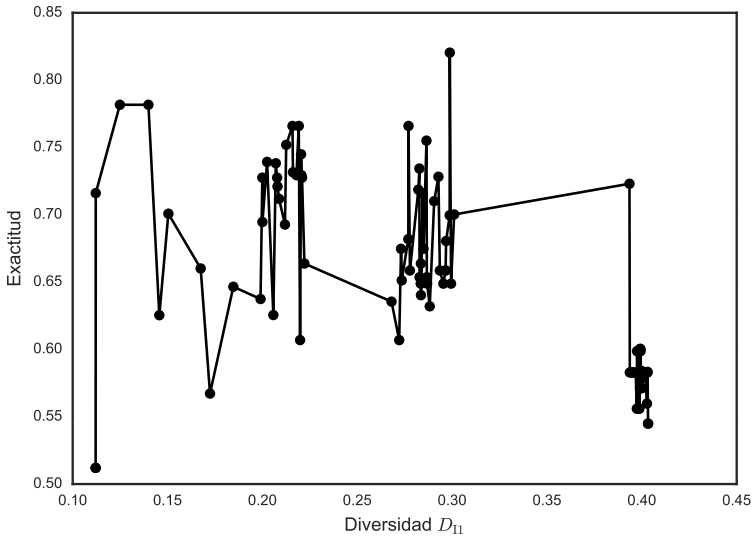


Figura 3.1. Resultados de un método actual para analizar el impacto de la diversidad (Iam-On y otros, 2011) sobre la calidad de la partición de consenso final.

3.2. Método para el control de la diversidad

Los métodos actuales suponen implícitamente que los ensambles con un mismo nivel de diversidad son similares; de esta forma, iguales valores de diversidad deberían representar ensambles parecidos, o al menos entradas similares a la función de consenso, de la cual se espera que produzca resultados semejantes. Sin embargo, esto puede no suceder en la práctica. En la Figura 3.1 se graficó la exactitud de la partición de consenso en función de la diversidad interna del ensamble (Iam-On y otros, 2011). En (Hadjitodorov y otros, 2006) se pueden encontrar resultados similares. Al observar la figura se hacen evidentes dos problemas. El primero es el comportamiento de la exactitud de la salida del consenso (eje y) cuando la diversidad interna (eje x) se encuentra alrededor de 0.21, donde si bien los valores de diversidad están cerca entre sí, no sucede lo mismo con la exactitud, la cual varía considerablemente. Un comportamiento similar puede observarse alrededor de la diversidad 0.28. De esta forma, se observa cómo ensambles con diversidades similares pueden representar entradas sumamente diferentes para la función de consenso. El segundo problema es que el rango de diversidad no siempre es muestreado uniformemente; por ejemplo, hay muchos menos ensambles con diversidades en $[0.10, 0.20]$ y prácticamente ninguno en $[0.31, 0.38]$. Como se ha dicho antes, una posible explicación al comportamiento errático de la función de consenso es que, mientras se observa una única medida de diver-

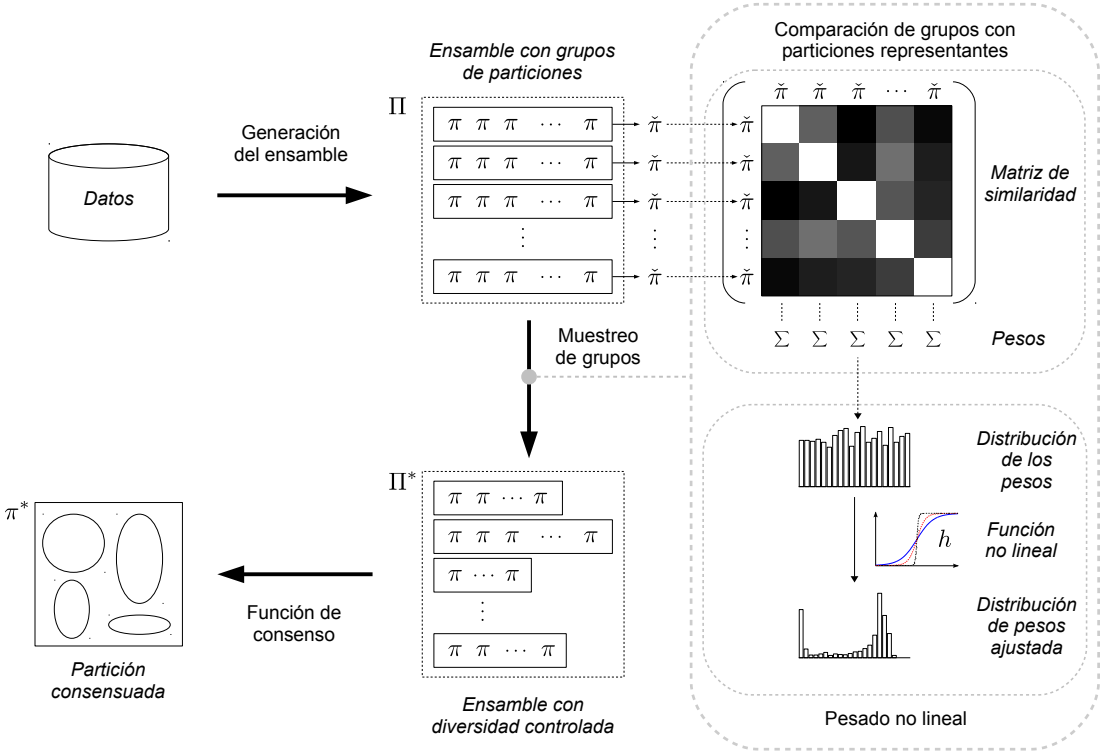


Figura 3.2. El método de control de diversidad: pasos y salidas intermedias.

sidad, en realidad pueden estar cambiando diferentes propiedades del ensamble. Esto hace muy complicado el análisis de cómo la diversidad afecta a la función de consenso, porque no es posible llegar a conclusiones fuertes cuando la variable objetivo (en este caso la exactitud) presenta un comportamiento errático. Además, dicho análisis se vuelve aún más difícil debido a la falta de uniformidad al muestrear los ensambles en el rango de diversidad bajo evaluación.

En esta sección se explica un nuevo método para el control de la diversidad. En la Figura 3.2 se muestra el proceso general del método, y en la Tabla 3.1 se puede ver un resumen de la notación utilizada. El primer paso consiste en generar un ensamble Π a partir de los datos. Éste es luego dividido en *grupos de particiones*, los cuales son creados haciendo clustering sobre los miembros del ensamble. El ensamble con diversidad controlada Π^*, el cual es producido muestreando desde los grupos originales, es finalmente combinado para obtener una partición de consenso π^*. El paso donde se realiza el muestreo se muestra en detalle a la derecha de la figura. Primero, se extrae información sobre la estructura del ensamble, la cual se obtiene comparando cada grupo con los demás. En lugar de una comparación directa entre grupos, se definen *particiones representantes*

Símbolo	Descripción
π	Partición de los datos
Π	Ensamble con grupos de particiones
q	Número de grupo de particiones en Π
Π_i	Grupo de particiones (miembro de Π)
p_i	Tamaño del grupo Π_i
$\check{\pi}_i$	Partición representante del grupo Π_i
Π^*	Ensamble con diversidad controlada
π^*	Partición de consenso final
ϕ_i	Clusterer
m	Número de clusterers usados
g	Número de corridas de un clusterer
Υ	Información mutua normalizada

Tabla 3.1: Notaciones utilizadas en el método propuesto para controlar la diversidad.

$\check{\pi}$ para cada uno. Se calcula un peso para cada grupo basado en la matriz de similaridad entre representantes. Luego, la distribución de estos pesos se ajusta usando una función no lineal, siendo estos utilizados para muestrear desde los grupos de particiones en Π. Basado en la distribución ajustada, el proceso de muestreo modifica el tamaño de los grupos para así cambiar la diversidad del ensamble de una manera suave. De esta forma, se crea un nuevo ensamble Π^* con su diversidad controlada por la distribución de los pesos. Finalmente, se utiliza la función de consenso para derivar una partición final π^* a partir de Π^*. Las siguientes subsecciones explican estos pasos del método de control en detalle.

3.2.1. Generación del ensamble basado en grupos de particiones

La generación del ensamble, que es el primer paso del método de control de diversidad, recibe como entrada un conjunto de datos y produce un ensamble organizado en grupos de particiones. Este paso se muestra en detalle en la Figura 3.3 y contiene dos fases: 1) *clustering sobre los datos*, donde se obtienen varias particiones y 2) *agrupamiento de las particiones*, la cual identifica diferentes grupos de soluciones.

Fase 1: Clustering sobre los datos. Se pueden utilizar varios métodos para crear un ensamble, como se ha explicado en detalle en la Sección 2.3.1. Uno de los más comunes consiste en producir diferentes particiones de los datos utilizando un único algoritmo de clustering y variando aleatoriamente alguno de

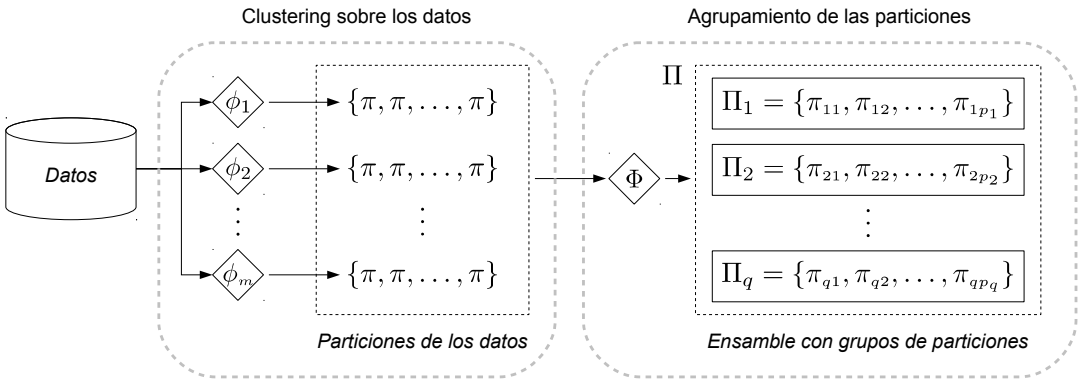


Figura 3.3. Generación de un ensamble basado en grupos de particiones. El proceso de generación contiene dos fases: 1) clustering sobre los datos y 2) agrupamiento de las particiones.

sus parámetros (Iam-On y otros, 2011; Kuncheva y Vetrov, 2006). En particular, k-medias es una opción frecuente y comúnmente utilizada, donde se emplea alguno de los siguientes esquemas para seleccionar el número de clusters: se fija el valor de k y los centroides se inicializan aleatoriamente (Hadjitodorov y otros, 2006; Zhong y otros, 2015); o se elige el valor de k aleatoriamente dentro de un intervalo de valores $[k_{min}, k_{max}]$ (Fred y Jain, 2005; Huang y otros, 2016; Kuncheva y Vetrov, 2006). El método presentado utiliza una combinación de ambos esquemas. Un diagrama que ilustra este proceso de clustering se puede ver en la primer parte de la Figura 3.3. Un clusterer ϕ_i se define para cada k en el intervalo seleccionado, donde ϕ_i se corre g veces con inicializaciones aleatorias, produciendo así un conjunto de $m \times g$ particiones. Este proceso intenta obtener tantas particiones diferentes como sea posible. Los ensambles generados de esta forma tienen una estructura conocida que contiene conjuntos de particiones, donde cada conjunto es generado por el mismo clusterer variando únicamente las inicializaciones.

Fase 2: Agrupamiento de las particiones. Luego de producir el ensamble inicial, la segunda fase de este paso consiste en obtener información acerca de su estructura. Esta fase básicamente aplica un algoritmo de clustering Φ sobre los miembros del ensamble generado previamente, el cual obtiene grupos de particiones denotados como Π_i, de tamaño p_i. Por lo tanto, esta fase no modifica el ensamble inicial sino que simplemente reagrupa sus miembros. Se desarrollaron tres algoritmos de agrupamiento para este propósito, cada uno de los cuales produce grupos de particiones con diferentes propiedades. El primer algoritmo es el más simple y agrupa los miembros del ensamble de acuerdo al número de clus-

ters. Así, cada grupo de particiones producido usando este enfoque tiene el mismo valor de k y este método es llamado agrupamiento de ensambles según k (Gk). El segundo método, llamado agrupamiento de ensamble completo (FG), consiste en aplicar un algoritmo de clustering jerárquico aglomerativo sobre los miembros del ensamble. Este método utiliza la distancia $1 - \Upsilon(\pi_i, \pi_j)$, donde $\Upsilon(\pi_i, \pi_j)$ es la información mutua normalizada (NMI) según se la ha definido en (2.23). En contraste con Gk, FG permite especificar el número de grupos q a ser producidos. El tercer algoritmo, llamado FGk, es una combinación de los dos anteriores, el cual toma los grupos formados por FG y luego los divide según Gk.

3.2.2. Comparación de grupos mediante particiones representantes

El paso previo del método produce un ensamble Π con grupos de particiones Π_i. El siguiente paso consiste en comparar dichos grupos para obtener información acerca de la estructura de Π, lo cual se puede lograr simplemente realizando una comparación exhaustiva entre cada par de grupos. Sin embargo, dada la estructura de Π, es posible definir un método más eficiente para comparar los grupos. Hay que notar que cada miembro de un grupo comparte ciertas propiedades con los otros, y así cada grupo de particiones puede ser considerado como un cluster. De esta forma, en lugar de comparar todos los miembros de los grupos, la similaridad entre dos grupos puede ser estimada comparando sus correspondientes particiones representantes. Una partición representante $\check{\pi}_i$ (ver Figura 3.2) es la partición que mejor representa a su grupo Π_i. Una definición conveniente del representante es aquella partición que maximiza la información compartida con los miembros del grupo:

$$\check{\pi}_i = \arg\max_{\check{\pi}} \sum_{\forall \pi \in \Pi_i} \Upsilon(\pi, \check{\pi}). \tag{3.1}$$

Se puede utilizar una función de consenso como método para obtener un representante $\check{\pi}_i$. Los miembros de un grupo pueden ciertamente tener una cantidad diferente de clusters (particularmente al utilizar FG), por lo tanto es necesario decidir cuántos clusters debería contener la partición representante. Esto puede lograrse obteniendo una partición de consenso por cada k en Π_i, y seleccionando como representante a aquel con el máximo nivel de acuerdo con los miembros del

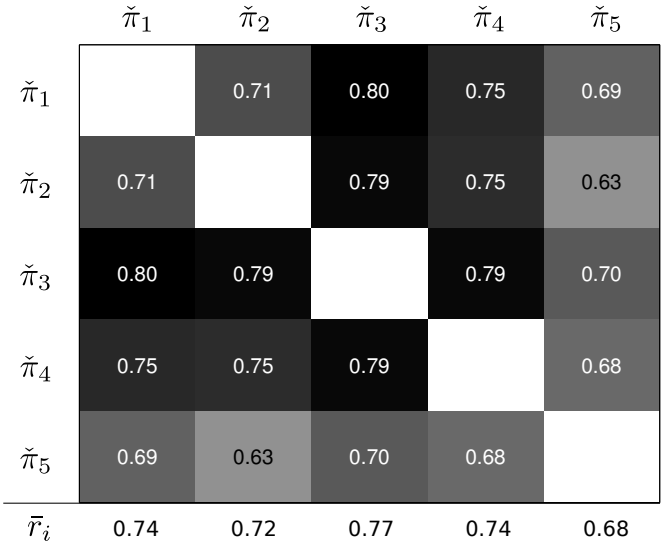


Figura 3.4. Matriz de similaridad de las particiones representantes generadas para el conjunto de datos Wine. La última fila es el promedio por columnas (descartando la diagonal principal).

grupo. Por ejemplo, si un grupo Π_i contiene varias particiones con $k = 3$, 5 y 7, entonces se derivan tres particiones de consenso de Π_i, con 3, 5 y 7 clusters. Así, la partición de consenso que maximiza (3.1) es seleccionada como representante para el grupo Π_i.

Luego de obtener todas las particiones representantes, estas se comparan entre ellas generando la matriz de similaridad

$$r_{ij} = \Upsilon(\check{\pi}_i, \check{\pi}_j). \tag{3.2}$$

Esta matriz contiene información acerca de la estructura del ensamble. La similaridad de un grupo con respecto a los otros es estimada calculando $\bar{r}_i = \frac{1}{q-1} \sum_{\forall j \neq i} r_{ij}$. En la Figura 3.4 se muestra un ejemplo de una matriz de similaridad en la que se comparan cinco representantes. De acuerdo a esta matriz, el grupo Π_3 es el más similar al resto del ensamble y Π_5 el más diferente. Esta información se puede utilizar para modificar Π y obtener un nuevo ensamble Π^* con diversidad controlada.

3.2.3. Pesado no lineal y muestreo de grupos

Luego de obtener información acerca de la estructura del ensamble, las etapas siguientes producen un nuevo ensamble Π^* (con diversidad controlada) ajus-

tando la distribución de $\bar{r}_i$ para obtener un peso w_i para cada grupo, y luego muestreando desde los grupos de Π. Si se quieren obtener pequeños incrementos en la diversidad, una idea intuitiva es aumentar levemente el tamaño de los grupos más diferentes, mientras que se reduce la proporción de los más similares. Por el contrario, la operación opuesta debería decrementar la diversidad. Ambos mecanismos proveen una manera directa de controlar la diversidad.

La similaridad de un grupo con el resto del ensamble, $\bar{r}_i$, se utiliza para calcular un peso w_i. El mecanismo empleado para obtener un control fino sobre la diversidad del ensamble consiste en aplicar una función no lineal a los valores de $\bar{r}_i$. Esta función se usa para ajustar la distribución de $\bar{r}_i$ de una manera suave, y así permitir pequeños aumentos o disminuciones en la diversidad. En el método presentado se emplea una función sigmoidea, y el cálculo de los pesos de los grupos queda dado por

$$w_i(h) = \frac{p_i}{1 + e^{-h(\bar{\bar{r}} - \bar{r}_i)}}, \tag{3.3}$$

donde p_i es el tamaño del grupo Π_i, $\bar{\bar{r}}$ es la media de $\bar{r}_i$, y h controla el cambio en la diversidad. Para cada valor de h se puede crear un nuevo ensamble Π^* (con $\Pi^* = \Pi$ para $h = 0$). Así, cuando $h > 0$, el método hace que Π^* sea más diverso que Π porque los grupos más diferentes en el ensamble reciben mayor peso. Por el contrario, cuando $h < 0$, Π^* tiene menor diversidad que Π porque los grupos más similares son favorecidos. De esta forma, es posible obtener incrementos (decrementos) en la diversidad del ensamble realizando incrementos (decrementos) graduales en h.

En la Figura 3.5 se muestra un ejemplo de la aplicación de este método donde se ven tres histogramas. La distribución original de $\bar{r}_i$ (en el medio) se modifica usando dos valores diferentes para h (arriba y abajo). Dado que para la mayoría de los grupos su similaridad $\bar{r}_i$ es mayor que el promedio de todos los valores de similaridad (histograma del medio), la mayoría de los grupos son favorecidos cuando el método intenta decrementar la diversidad (histograma de abajo con $h = -75$). Por otro lado, al intentar incrementar la diversidad del ensamble, se obtiene el efecto opuesto (histograma de arriba con $h = 75$).

Después de obtener los pesos w_i, los nuevos grupos del ensamble Π^* se producen al muestrear los grupos en Π con probabilidad $P_i = w_i / \sum_i w_i$. Por ejemplo, si suponemos que hay tres grupos, los cuales obtuvieron los pesos $w_1 =$

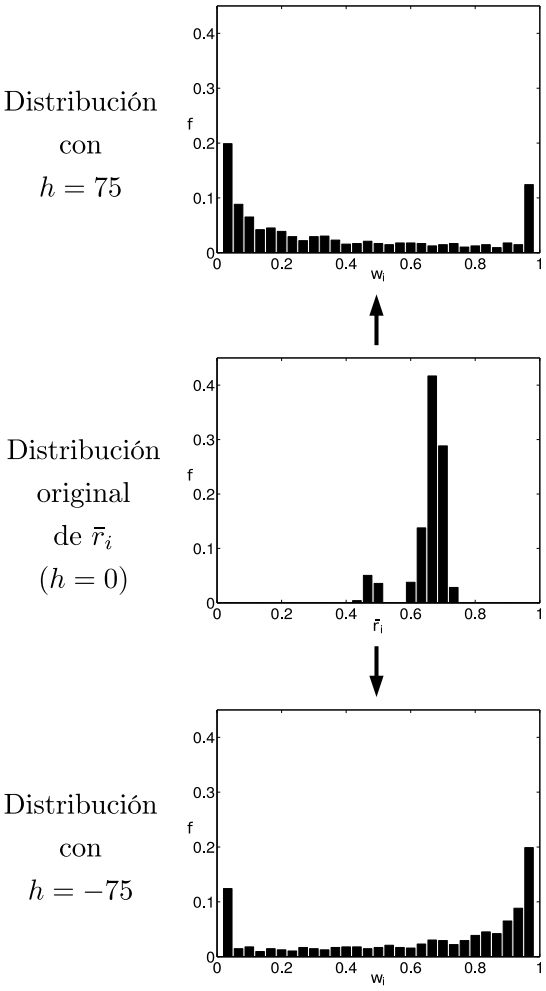


Figura 3.5. Ajuste de la distribución de $\bar{r}_i$ usando una función sigmoidea con dos valores diferentes para el parámetro h.

0.8, $w_2 = 0.3$ y $w_3 = 0.4$ para un valor determinado de h, y que el tamaño deseado del ensamble Π^* es 100, entonces según P_1, P_2 y P_3, aproximadamente 53 de sus miembros serán muestreados uniformemente del grupo Π_1, 20 de Π_2 y 27 de Π_3. Cuando se utiliza un valor ligeramente diferente para h al calcular w_i, la probabilidad de muestrear cada grupo será también modificada ligeramente, obteniendo así ensambles que sufren pequeños cambios.

El paso final en el método general consiste en derivar una partición de consenso a partir del ensamble Π^*. En la Sección 2.3.2 se han descripto algunos de los métodos de consenso más comúnmente utilizados: aquellos basados en grafos y los que utilizan acumulación de evidencia. Cualquier técnica de consenso puede ser utilizada durante este paso en el método aquí presentado.

Conjunto de datos	n	D	k
Difficult Doughnut	500	12	2
Four Gaussian	100	12	4
Iris	150	4	3
Ionosphere	351	34	2
Glass	214	9	6
Wine	178	13	3

Tabla 3.2: Descripción de los datos artificiales y reales utilizados en los experimentos. n es el número de objetos, D la cantidad de características o dimensiones, y k el número de clases.

3.3. Datos y medidas de desempeño

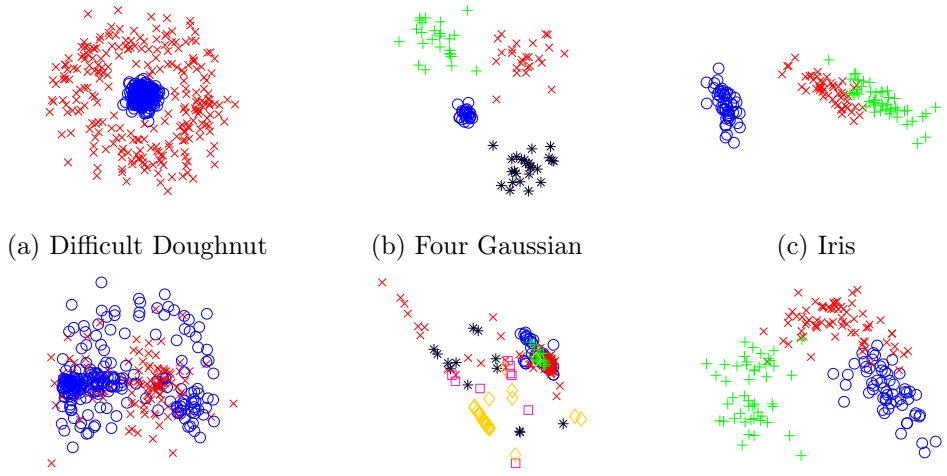
En esta sección se describirán los conjuntos de datos utilizados para evaluar el método presentado, así como también una medida propuesta para cuantificar la suavidad del control sobre la diversidad del ensamble. Las demás medidas de desempeño empleadas en este trabajo, considerando tanto los índices externos como internos, ya fueron descriptas en la Sección 2.2.

3.3.1. Conjuntos de datos

Se utilizaron seis conjuntos de datos artificiales y reales muy conocidos en la comunidad de aprendizaje de máquina (Tabla 3.2). Difficult Doughnut y Four Gaussian fueron creados artificialmente generando dos dimensiones de datos y luego agregando diez dimensiones de ruido uniformemente distribuido. Estos dos conjuntos de datos tienen clusters con formas muy diferentes (ver las Figuras 3.6a y 3.6b), representando así distintos niveles de dificultad para el algoritmo de clustering. Los conjuntos de datos reales fueron obtenidos del repositorio de UCI (Lichman, 2013). Como puede verse en la Tabla 3.2 y en la Figura 3.6, estos poseen diferentes cantidades de objetos, características y clases; y además sus clusters tienen distintas formas y presentan diferentes niveles de solapamiento.

3.3.2. Medida de suavidad

En este trabajo también se ha desarrollado y utilizado una medida de suavidad para evaluar la capacidad de hacer un control fino de la diversidad del ensamble, la cual ha permitido valorar cómo evolucionaban las demás medidas



(a) Difficult Doughnut (b) Four Gaussian (c) Iris

(d) Ionosphere (e) Glass (f) Wine

Figura 3.6. Los clusters de los conjuntos de datos empleados. Para los conjuntos artificiales (a y b) se utilizaron las dos dimensiones sin ruido para obtener las gráficas. Para los conjuntos reales se emplearon proyecciones en componentes principales.

de ensambles y las medidas de clustering sobre el consenso al cambiar la diversidad. Un nivel más alto de suavidad es indicativo de un mejor control sobre la diversidad del ensamble.

El control de la diversidad puede ser valorado estudiando la evolución de las medidas de ensambles y las medidas de clustering, es decir, observando el comportamiento de las entradas a la función de consenso (los ensambles Π^*) y sus correspondientes salidas (las particiones consensuadas π^*). Por un lado, si se realiza el control sobre una única medida de diversidad, se espera que las otras medidas de diversidad cambien de una manera suave. Por otro lado, los métodos actuales suponen que los ensambles con una diversidad similar representan entradas similares a la función de consenso, y así se espera que estas produzcan particiones consensuadas parecidas. Para evaluar si esta propiedad realmente se verifica, se pueden calcular las medidas de clustering sobre las particiones de consenso. Si estas medidas cambian de una forma suave, esto es indicativo de que los ensambles con diversidades similares representan realmente entradas parecidas a la función de consenso.

Para medir la suavidad se puede utilizar el coeficiente de autocorrelación con corrimiento 1. Se sabe que este coeficiente se puede emplear para detectar cuándo no hay aleatoriedad (Box y otros, 2013), y esto está en sintonía con el concepto

intuitivo de suavidad que se ha descripto antes. Desde este punto de vista, la aleatoriedad en una secuencia se refiere a la independencia o impredecibilidad de un valor relativo a otro cercano. Por ejemplo, para la exactitud, este coeficiente puede ser calculado como

$$\rho_1 = \frac{\sum_d (A_d - \bar{A})(A_{d+1} - \bar{A})}{\sum_d (A_d - \bar{A})^2}, \tag{3.4}$$

donde A es una secuencia de valores de exactitud ordenados por diversidad d, y $\bar{A}$ es su media. En lugar de la exactitud, se pueden utilizar cualquiera de las medidas de ensambles o de clustering definidas previamente. Los valores de ρ_1 cercanos a 1 indican un comportamiento suave de la secuencia de valores, y así el control sobre la diversidad del ensamble es mejor. Por el contrario, valores de ρ_1 cercanos a 0 indican cambios bruscos, demostrando así que no hay control sobre la diversidad del ensamble.

3.4. Resultados y discusión

En esta sección se describen la metodología y configuraciones experimentales empleadas para evaluar el método presentado. De acuerdo a la configuración experimental descripta a continuación, se han empleado todos los conjuntos de datos presentados en la Sección 3.3.1 para evaluar la capacidad del método de controlar la diversidad de los ensambles. Luego, se utilizaron las medidas de clustering definidas en la Sección 2.2 para observar los efectos del método en la evolución del desempeño de la función de consenso. Finalmente, para poder cuantificar el grado de control de la diversidad que logran los diferentes métodos evaluados, se calculó el coeficiente de autocorrelación con corrimiento 1 (ρ_1) tanto para las medidas de ensambles como para las medidas de clustering.

3.4.1. Configuración experimental

Para la fase 1 del paso de generación del ensamble (Sección 3.2.1), se ha utilizado k-medias como algoritmo de clustering base, con $k_{min} = 2$ y $k_{max} = 20$ ($m = 19$), y 20 inicializaciones aleatorias ($g = 20$) para cada k. Ya que con esta configuración Gk produce exactamente 19 grupos de particiones, se ha configurado FG para que obtenga el mismo número de grupos ($q = 19$). Se utilizó la

función de supraconsenso (Sección 2.3.2.3) tanto para derivar el representante $\check{\pi}_i$ de cada grupo como la partición de consenso π^* del paso final. Esta función de supraconsenso se configuró para utilizar 10 métodos de consenso diferentes: tres basados en grafos (CSPA, HGPA y MCLA), descriptos en la Sección 2.3.2.1; y los métodos basados en acumulación de evidencia (Sección 2.3.2.2). Ya que estos últimos permiten aplicar cualquier algoritmo de clustering sobre la matriz de coasociación, en los experimentos se han utilizado siete criterios de enlace diferentes: el enlace simple, completo, promedio no pesado, promedio pesado, centroide pesado, centroide no pesado y el criterio de Ward. El experimento completo se repitió 50 veces.

El método de control de diversidad ha sido evaluado al final de cada uno de sus múltiples pasos y etapas. Se ha observado su desempeño utilizando diferentes tipos de medidas que cuantifican distintos aspectos del proceso completo: las medidas de ensambles, que brindan información sobre la entrada de la función de consenso; y las medidas de clustering, que se aplican sobre la partición consensuada final. Las medidas de ensambles básicamente cuantifican la diversidad de los ensambles generados utilizando las cuatro medidas de diversidad anteriormente presentadas (Sección 2.3.3): las dos medidas internas D_{I1} y D_{I2}, y las externas D_{E1} y D_{E2}. Las medidas de clustering presentadas en la Sección 2.2 fueron utilizadas para observar el impacto de los ensambles generados sobre la partición de consenso, teniendo en cuenta diferentes perspectivas. Se ha utilizado como medida externa la exactitud (conocida en la bibliografía comúnmente como *accuracy*)

$$A = \Upsilon(\pi', \pi^*), \tag{3.5}$$

donde Υ se refiere al NMI definido en (2.23), π' es la partición de referencia basada en las clases de los datos, y π^* la partición consensuada derivada a partir de los ensambles con diversidad controlada. Como medidas internas se usaron la compactitud (2.26), la separación (2.27), el índice de Davies-Bouldin (2.28) y el índice de Dunn (2.29). Por último, se ha utilizado la medida de suavidad (3.4) para poder cuantificar el grado de control sobre la diversidad que los diferentes métodos son capaces de alcanzar.

El código fuente completo[1] está disponible públicamente para ser descargado, y tener así la posibilidad de probar con diferentes configuraciones expe-

[1] `https://sourceforge.net/projects/sourcesinc/files/divcontrol/0.10/`

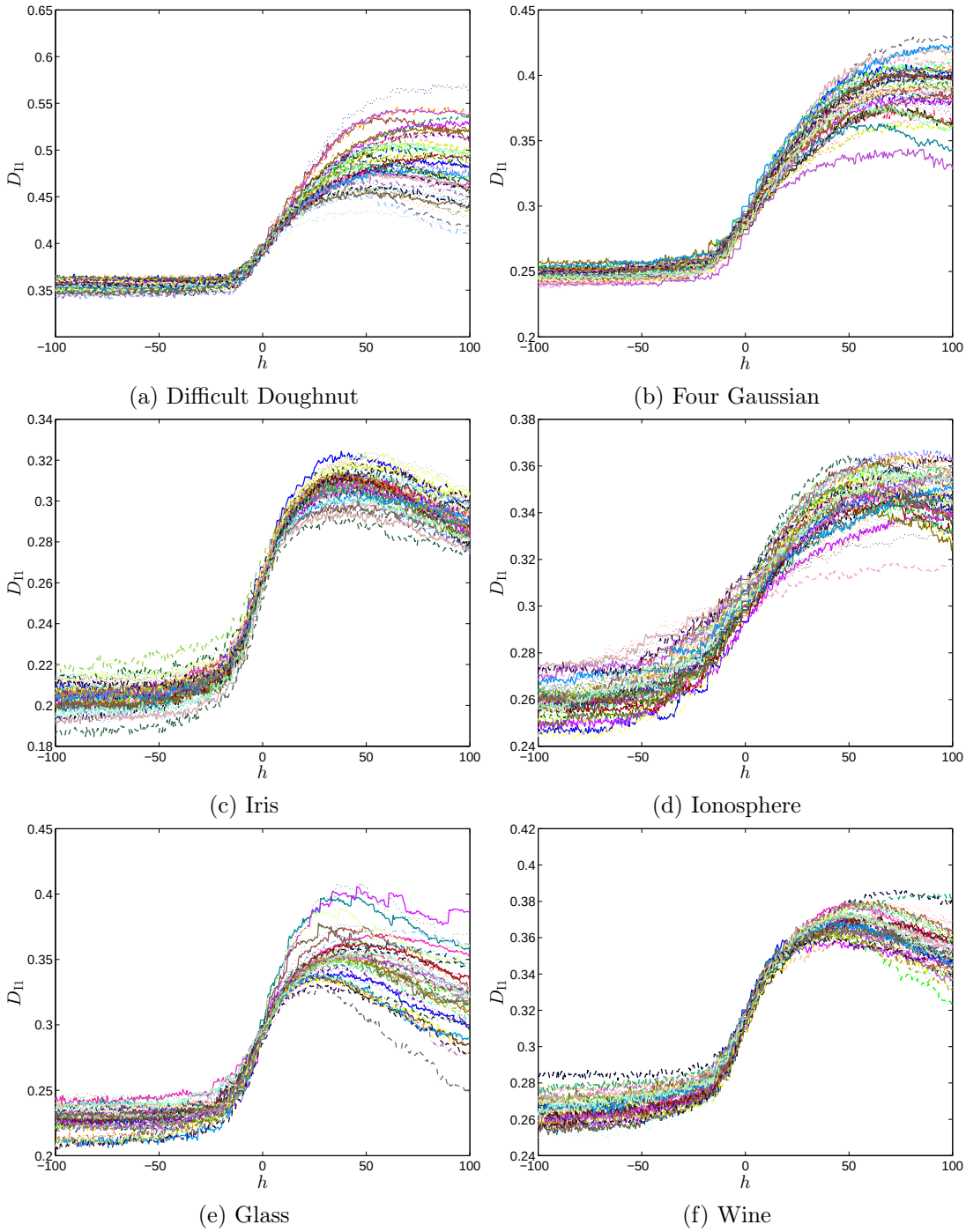
rimentales. Además, atendiendo a uno de los objetivos específicos de esta tesis, que consiste en garantizar la reproducibilidad de los resultados, el algoritmo de control de diversidad puede ser rápidamente probado usando un *web-demo*[2], que consiste en una interfaz web fácilmente accesible y que ha sido desarrollado con la herramienta *Generador de web-demos* (Stegmayer y otros, 2016).

3.4.2. Control de la diversidad de los ensambles

En la Figura 3.7 se muestran los resultados del control de diversidad para los seis conjuntos de datos de la Sección 3.3.1, donde se graficó la medida interna de diversidad D_{I1} como función del parámetro h de (3.3). Cada repetición es indicada por un estilo y color de línea diferente. Se generó un ensamble Π^* por cada valor de h considerando un intervalo suficientemente amplio de $[-100, 100]$, con el objetivo de observar cómo se comportaba el método en casos intermedios, pero también en los extremos. El tamaño de Π^* fue de 100 y el algoritmo de agrupamiento utilizado fue FG.

Los resultados muestran que, aunque se observan comportamientos diferentes para cada conjunto de datos, en todos los casos el método pudo controlar, aumentando y disminuyendo, la diversidad interna al cambiar el parámetro h. Además, el método produce efectivamente un cambio suave en D_{I1}, donde el rango completo de diversidad fue uniformemente muestreado. Al utilizar el método para incrementar o decrementar D_{I1}, cada curva finalmente convergió a un cierto nivel, aunque cada una llegó a diferentes límites inferiores y superiores. La saturación ocurrió en ambos lados de la curva porque al usarse valores absolutos altos para h la función sigmoidea sufre cambios cada vez menores, y entre los nuevos ensambles creados casi no hay diferencias en el tamaño de los grupos. En esta situación extrema, la única fuente de cambio es la selección aleatoria de los miembros de cada grupo de particiones. Por lo tanto, es esperable encontrar ensambles sin grandes cambios en sus diversidades al utilizar los valores más altos de h. Otro comportamiento interesante se observa cuando la diversidad es incrementada ($h > 0$), donde se alcanza un punto de máxima diversidad y luego, a partir de ahí, esta decrece hasta un determinado nivel. Este cambio ocurre porque los valores altos de h producen algunos pesos $w_i \approx 0$, generando así gru-

[2] `http://fich.unl.edu.ar/sinc/web-demo/divcontrol/`



(a) Difficult Doughnut

(b) Four Gaussian

(c) Iris

(d) Ionosphere

(e) Glass

(f) Wine

Figura 3.7. Evolución del control de la diversidad para seis conjuntos de datos.

pos vacíos, mientras que otros w_i fueron suficientemente grandes para introducir muchas particiones repetidas.

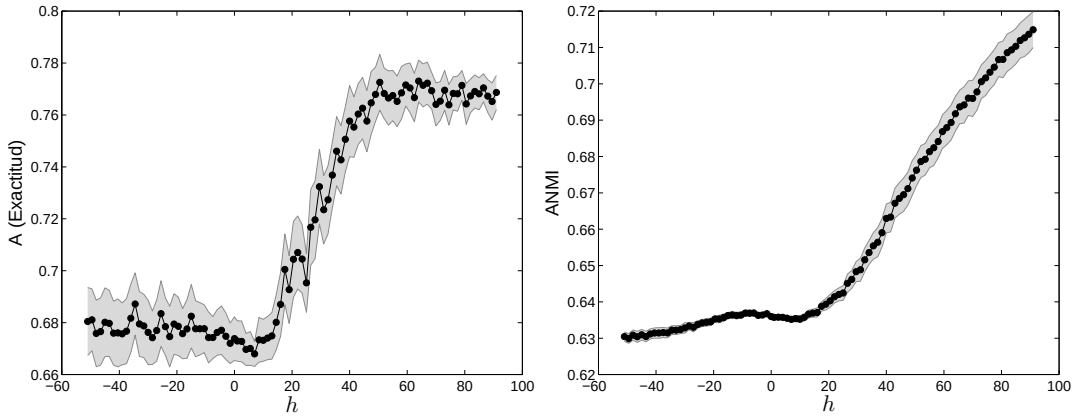
Los resultados muestran claramente que el método propuesto pudo disminuir e incrementar la diversidad de los ensambles de una manera efectiva. Si bien se empleó un rango amplio de valores de h, en la práctica sería necesario concen-

trarse en un rango más útil alrededor de $h = 0$, donde la diversidad exhibe los cambios más grandes. Ya que h puede tomar valores continuos, se puede aumentar la resolución de modo sencillo y explorar un determinado rango útil, que es precisamente lo que hemos realizado en los experimentos siguientes. La diversidad pudo ser controlada independientemente de las características propias de los datos, tanto de aquellos con clusters más compactos y bien separados (como Four Gaussian), como también de otros que poseen estructuras mucho más complejas (como Ionosphere o Glass). Estos resultados indican que el método es capaz de controlar con precisión el nivel de desacuerdo entre los miembros de un ensamble de una manera suave.

3.4.3. Evaluación del desempeño del consenso

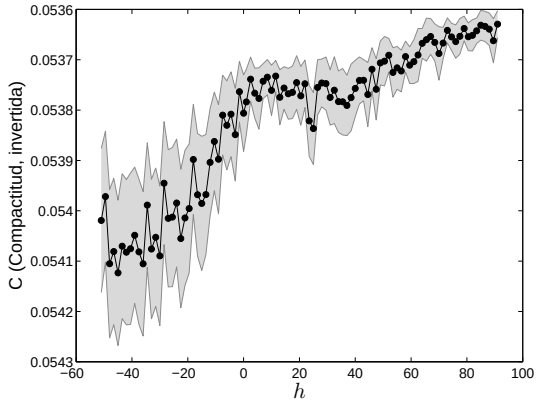
Como paso siguiente, se calcularon las medidas de clustering mencionadas anteriormente sobre la partición de consenso final, lo que permitió observar cómo estos valores evolucionaban para los ensambles de diversidad controlada. Los resultados para todos los conjuntos de datos descriptos anteriormente se muestran en las Figuras 3.8 a 3.13. En estas figuras se han graficado todas las medidas de clustering, donde se observa el promedio de todas las repeticiones para cada h. En gris se muestran también los correspondientes intervalos de confianza ($\alpha = 0.05$), que ayudan a identificar las zonas del rango de h donde la función de consenso ha sido más estable.

Para todos los conjuntos de datos puede verse que, además del control de la diversidad del ensamble como se demostró antes, el método propuesto pudo también inducir cambios suaves en las medidas de clustering sobre las particiones consensuadas. En los resultados para Wine (Figura 3.8) puede verse que la mayoría de las medidas tendieron a mejorar con valores más grandes de h (ensambles más diversos). Gran parte de las medidas no parecen haber sido afectadas significativamente, pero la exactitud y el ANMI exhibieron incrementos relativamente altos. De esta forma, si bien las particiones π^* parecieron tener clusters casi igualmente compactos y separados para todos los valores de h, definitivamente representan particiones diferentes de los datos. Es interesante observar lo que sucede en todas las medidas en conjunto para ciertos valores de h. En este caso, si bien la compactitud y Dunn tienen un comportamiento creciente sin so-

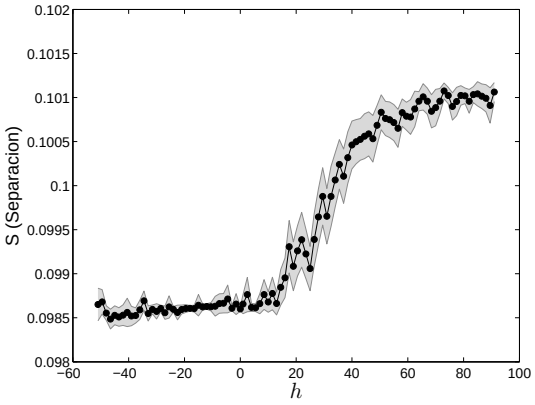


(a) Exactitud (NMI)

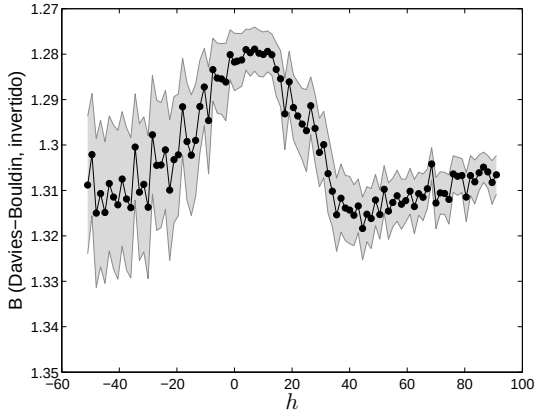
(b) ANMI



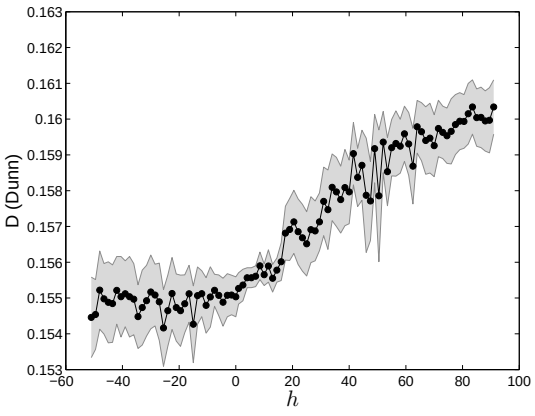
(c) Compactitud (invertido)



(d) Separación



(e) Davies-Bouldin (invertido)

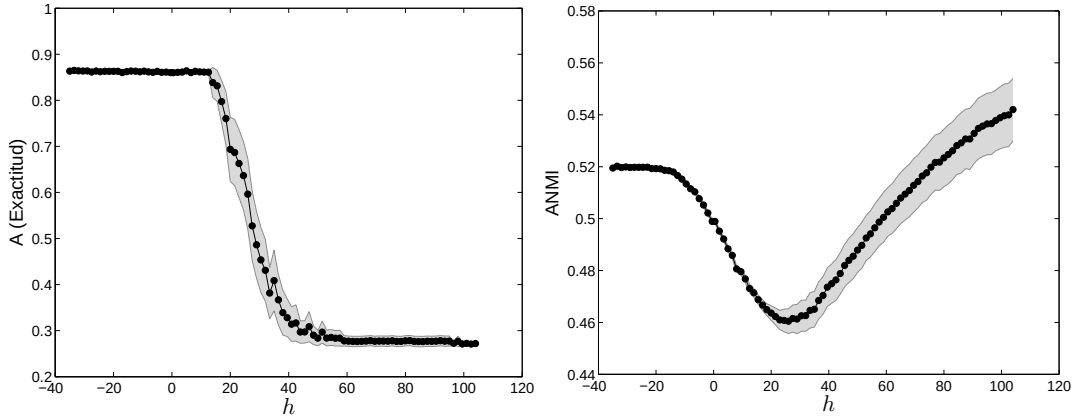


(f) Dunn

Figura 3.8. Seis medidas de clustering calculadas para las particiones de consenso obtenidas usando el conjunto de datos Wine. La compactitud y el índice de Davies-Bouldin están invertidos, así los valores de arriba son mejores para todas las medidas.

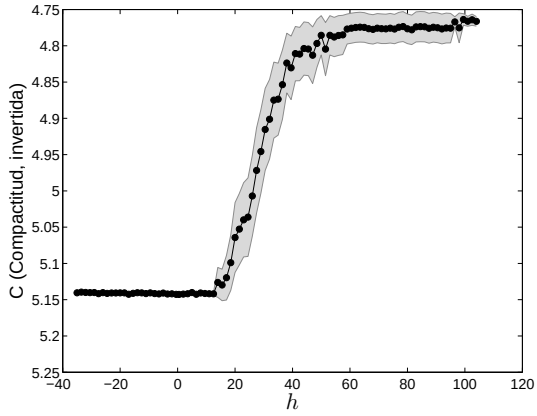
bresaltos, las demás medidas de clustering parecen sufrir algún tipo de cambio importante en $h \approx 10$. Por ejemplo, Davies-Bouldin alcanza su máximo valor, la separación de los clusters comienza a crecer y la partición de consenso comienza a compartir más información con el ensamble (ANMI). Es importante notar que este análisis se puede realizar sin observar la exactitud, que no puede calcularse en una aplicación real. Sin embargo, en este caso podemos ver cómo $h \approx 10$ también representa un punto de cambio para esta medida, donde las particiones consensuadas comienzan a ser más exactas.

Un análisis completo de cómo la diversidad afecta el desempeño del consenso está fuera del alcance de este trabajo. Sin embargo, una mirada general contemplando los resultados en todos los conjuntos de datos puede ayudar a direccionar los esfuerzos futuros para entender mejor el impacto que tiene la diversidad. Por ejemplo, al observar cómo los ensambles con diversidad controlada afectan las medidas de clustering para Difficult Doughnut (Figura 3.9), pueden verse similitudes y diferencias importantes con respecto a Wine (Figura 3.8). Lo similar es que el método de control de diversidad nuevamente logra inducir cambios suaves en la función de consenso. De hecho, como se mostrará en breve, Difficult Doughnut es el conjunto de datos donde se logra el mayor nivel de control sobre las medidas de clustering. Otra similitud con Wine es que la mayoría de las medidas mejoran al aumentar la diversidad de los ensambles. Sin embargo, una mirada más atenta sobre su evolución muestra un comportamiento interesante en todas ellas. Al igual que se observó con Wine, en donde cierto valor de h parecía marcar un cambio importante, en este caso los cambios suceden en $h \approx 15$. Las particiones consensuadas generadas con $h < 15$ son las que poseen clusters menos compactos y separados, sin embargo tienen los mayores valores de Dunn y se comportan de forma estable según todas las medidas. Este conjunto representa a las particiones más exactas. Para $h > 15$ la compactitud, la separación y Davies-Bouldin comienzan a crecer rápidamente, mientras que Dunn comienza a bajar y a comportarse de manera muy inestable (esto se ve por los intervalos de confianza más amplios). Esto es esperable si se consideran las características de este conjunto de datos y se observa también la evolución de la exactitud, donde a diferencia de Wine los ensambles más diversos producen las peores particiones consensuadas. En este conjunto de soluciones, el cluster representado por el anillo exterior de Difficult Doughnut es particionado de diferentes maneras, mezclándose con el cluster cen-

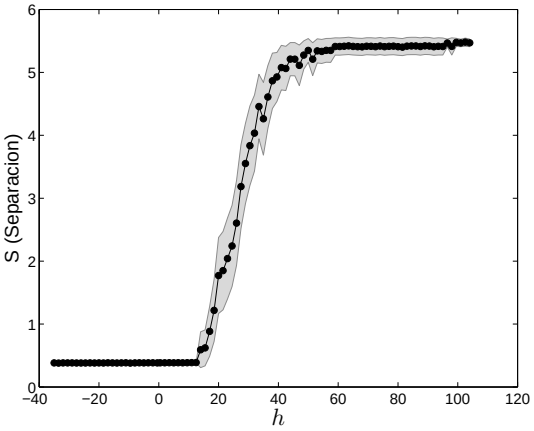


(a) Exactitud (NMI)

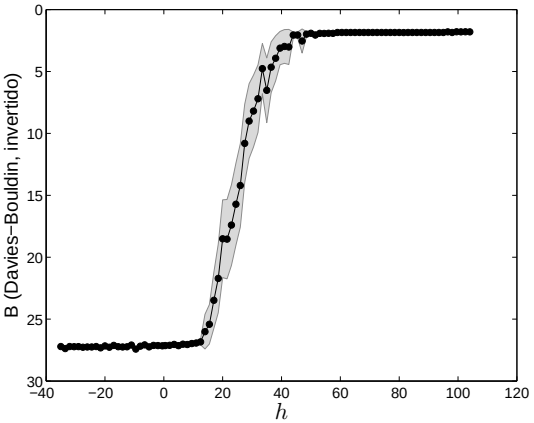
(b) ANMI



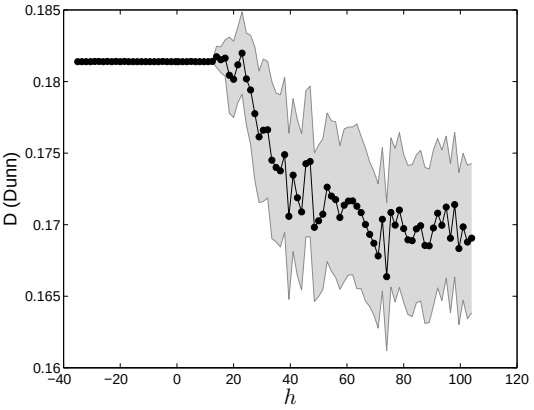
(c) Compactitud (invertido)



(d) Separación



(e) Davies-Bouldin (invertido)



(f) Dunn

Figura 3.9. Seis medidas de clustering calculadas para las particiones de consenso obtenidas usando el conjunto de datos Difficult Doughnut. La compactitud y el índice de Davies-Bouldin están invertidos, así los valores de arriba son mejores para todas las medidas.

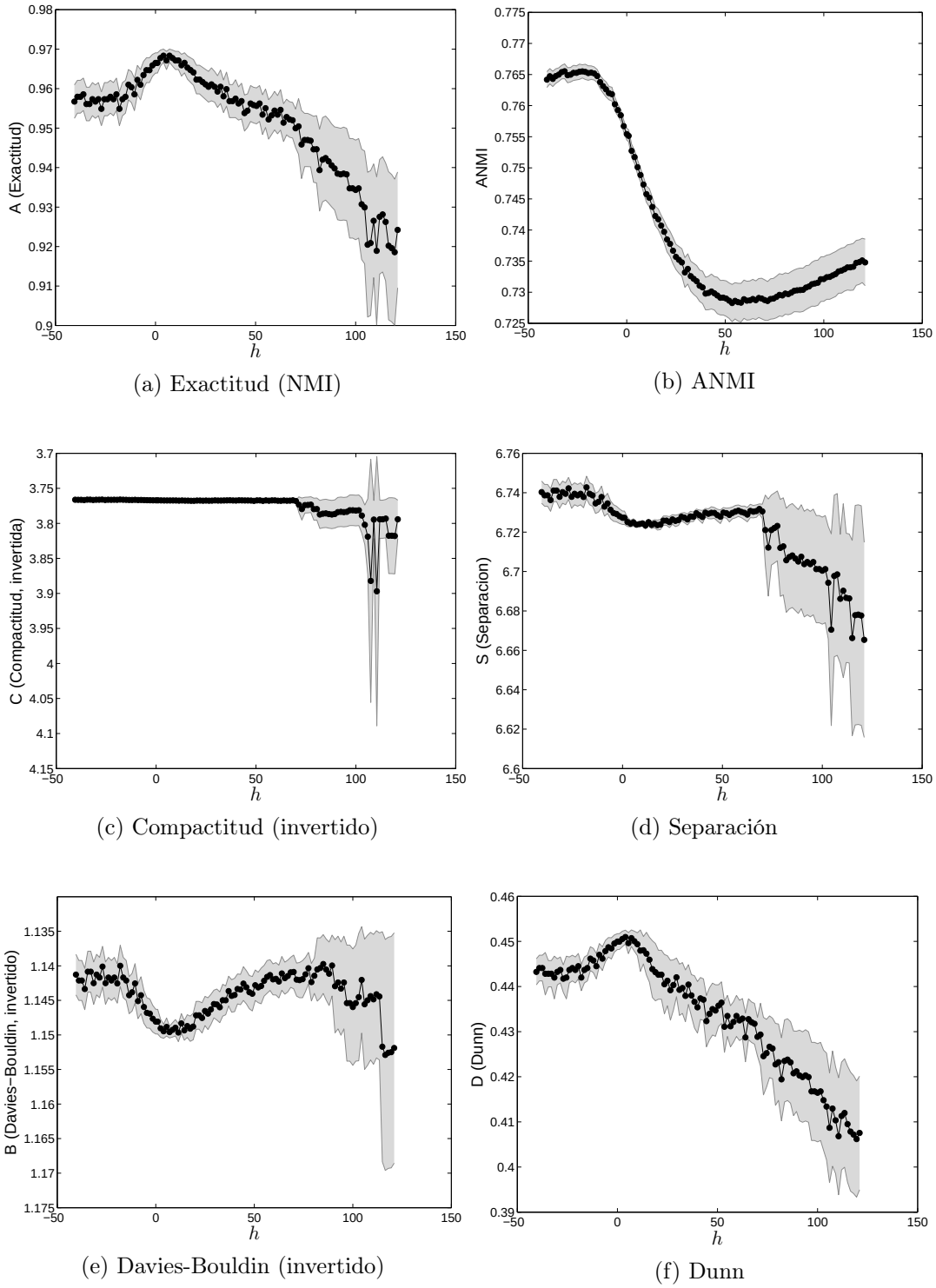
tral, y produciendo por ende valores bajos de exactitud y una alta varianza para el índice de Dunn.

Si se observan las figuras de los otros conjuntos de datos, también se podrían sacar conclusiones similares observando las diferentes medidas de clustering e identificando ciertos valores de h donde estás presentan algún tipo de cambio. Nuevamente, es importante notar que estos valores pueden identificarse prescindiendo de la exactitud, que no está disponible en una aplicación real. Para Four Gaussian en la Figura 3.10, dos momentos interesantes son $h \approx 5$ y $h \approx 70$. En $h \approx 5$, la separación y Davies-Bouldin alcanzan valores bajos, mientras que Dunn llega a su máximo. A partir de ahí, estas tres medidas crecen o decrecen de forma aproximadamente monótona hasta $h \approx 70$, donde todas comienzan a comportarse de forma más inestable. Si se observan los resultados con Iris (Figura 3.11), Ionosphere (Figura 3.12), y Glass (Figura 3.13), también allí se pueden encontrar distintos valores de h donde se producen cambios interesantes. Toda esta información se podría utilizar para entender mejor la estructura de los datos, identificando conjuntos de particiones muy distintas (como pasa con Difficult Doughnut), aquellas que pertenecen a un conjunto de comportamiento estable, que tienen clusters con grados de compactitud o separación muy diferentes, o las que obtienen los valores mínimos o máximos para ciertas medidas.

El análisis de los resultados producidos por el método de control de diversidad sugiere que no existe un nivel de diversidad universalmente más adecuado, como intentan encontrar los diferentes trabajos actuales. Esto es importante para direccionar correctamente los futuros estudios sobre la diversidad. Los ensambles de diversidad controlada logran producir cambios suaves en las particiones de consenso finales y permiten detectar momentos interesantes donde estas cambian, como se ha mostrado identificando ciertos valores de h.

3.4.4. Suavidad de las medidas de desempeño

Finalmente, se calculó la suavidad (ρ_1) para las medidas de ensambles y las de clustering usando diferentes enfoques para generar ensambles diversos. Se comparó el método presentado en este capítulo con dos métodos existentes para analizar el impacto de la diversidad en el consenso: un método de generación aleatorio (RN) descripto en (Fern y Brodley, 2003; Hadjitodorov y otros, 2006;

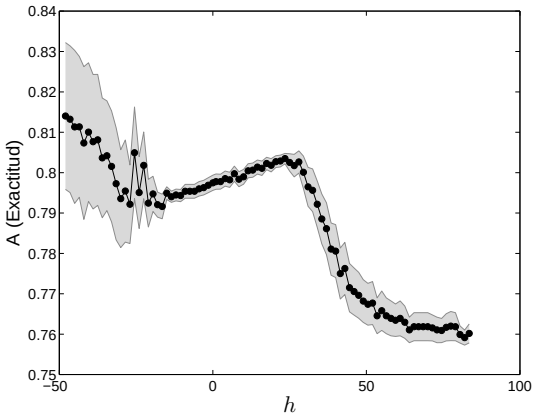


(a) Exactitud (NMI) (b) ANMI

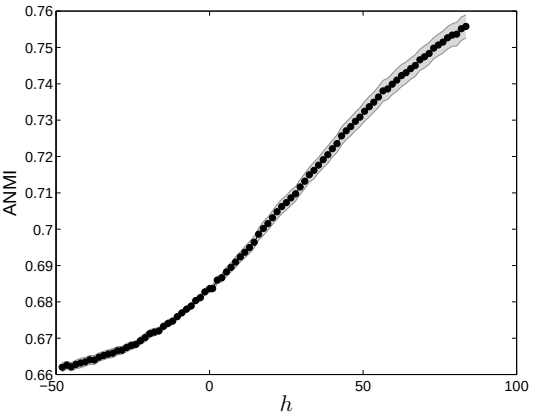
(c) Compactitud (invertido) (d) Separación

(e) Davies-Bouldin (invertido) (f) Dunn

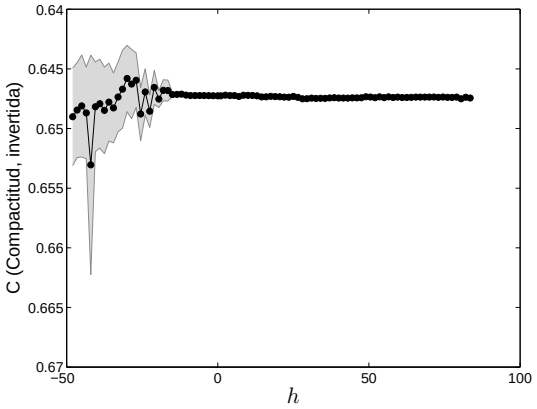
Figura 3.10. Seis medidas de clustering calculadas para las particiones de consenso obtenidas usando el conjunto de datos Four Gaussian. La compactitud y el índice de Davies-Bouldin están invertidos, así los valores de arriba son mejores para todas las medidas.



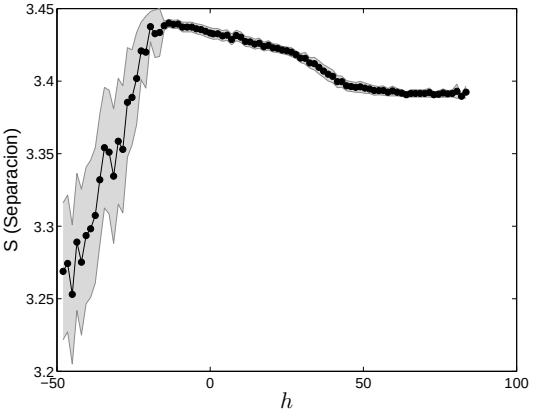
(a) Exactitud (NMI)



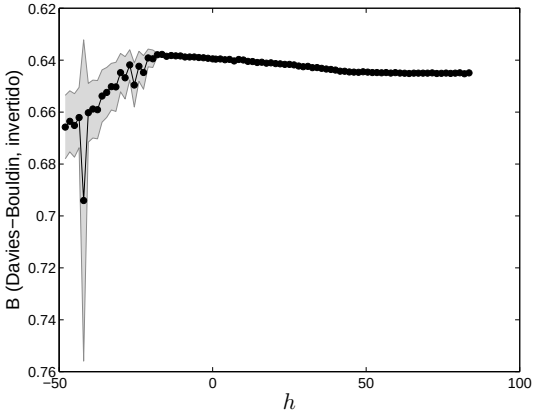
(b) ANMI



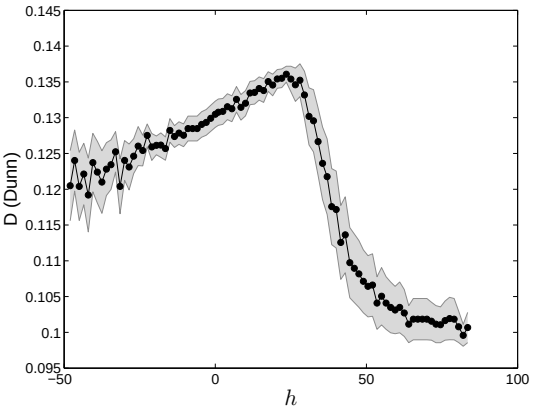
(c) Compactitud (invertido)



(d) Separación

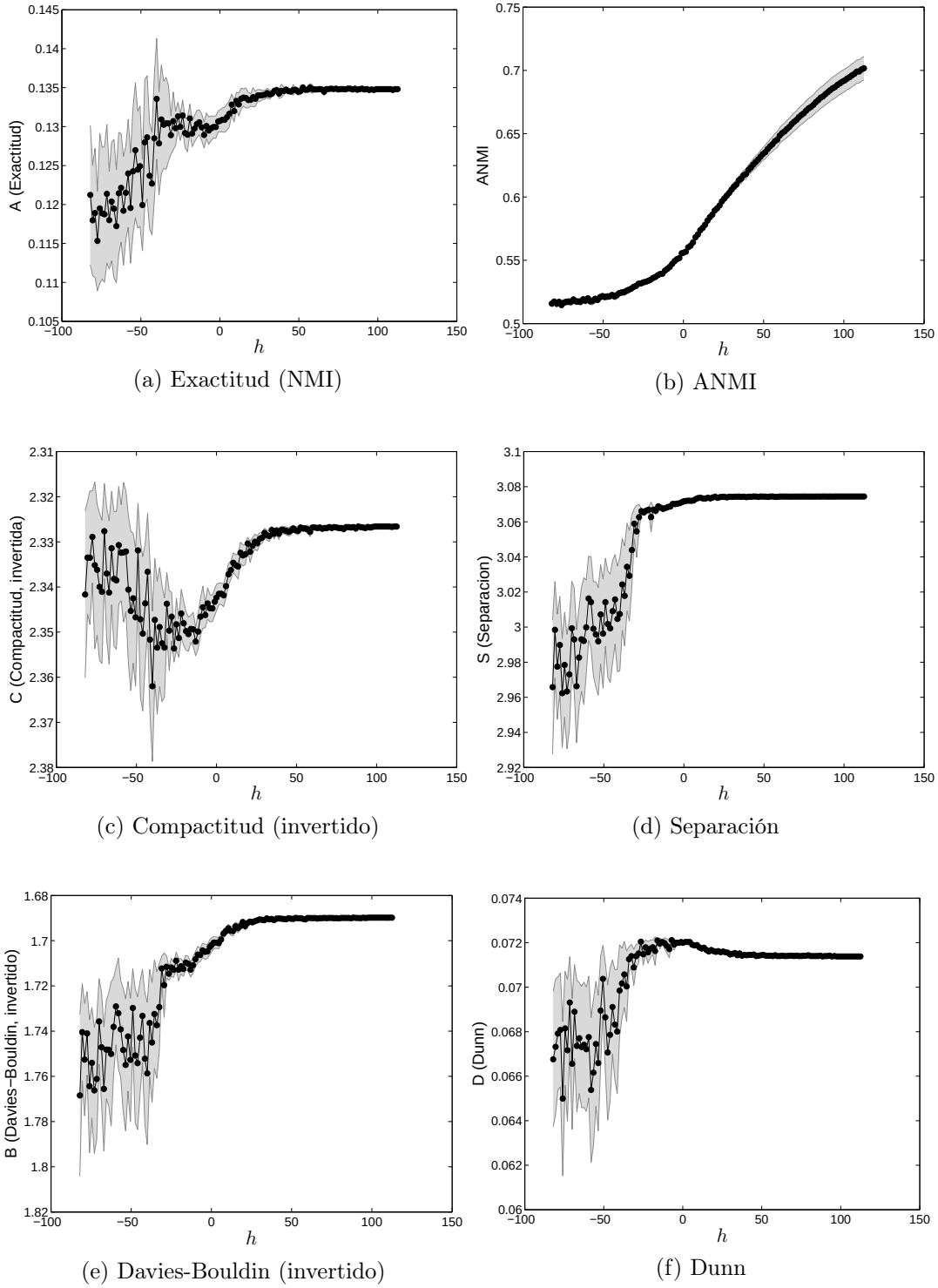


(e) Davies-Bouldin (invertido)



(f) Dunn

Figura 3.11. Seis medidas de clustering calculadas para las particiones de consenso obtenidas usando el conjunto de datos Iris. La compactitud y el índice de Davies-Bouldin están invertidos, así los valores de arriba son mejores para todas las medidas.



(a) Exactitud (NMI)

(b) ANMI

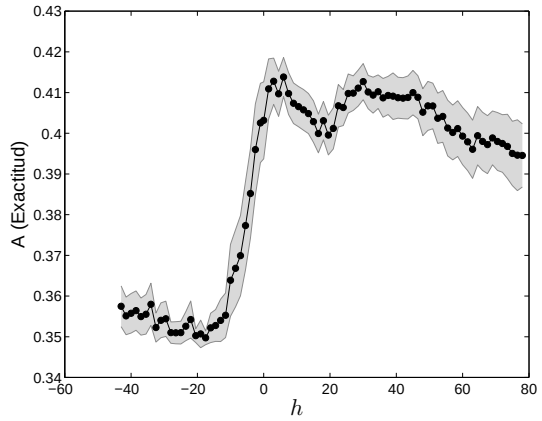
(c) Compactitud (invertido)

(d) Separación

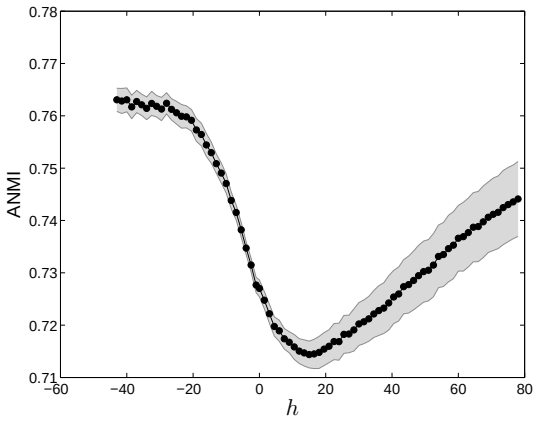
(e) Davies-Bouldin (invertido)

(f) Dunn

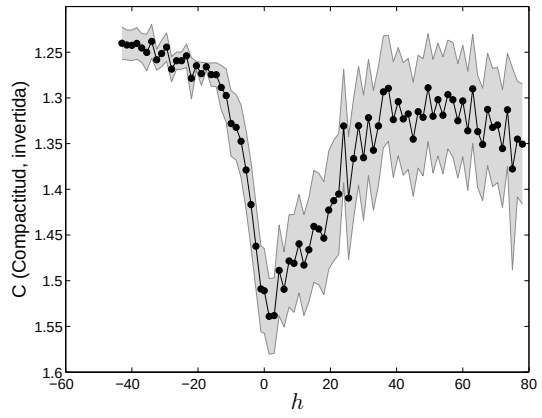
Figura 3.12. Seis medidas de clustering calculadas para las particiones de consenso obtenidas usando el conjunto de datos Ionosphere. La compactitud y el índice de Davies-Bouldin están invertidos, así los valores de arriba son mejores para todas las medidas.



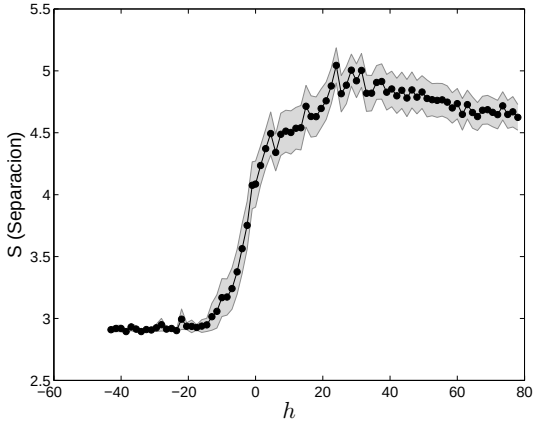
(a) Exactitud (NMI)



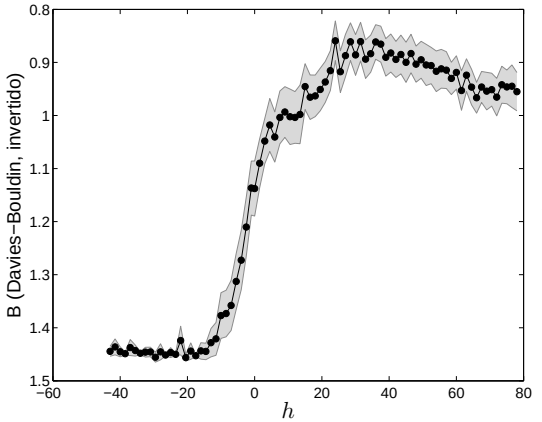
(b) ANMI



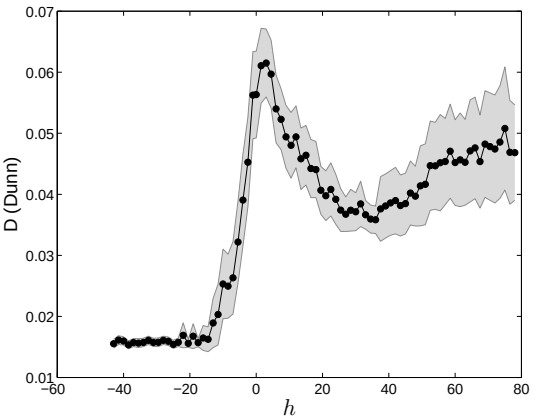
(c) Compactitud (invertido)



(d) Separación



(e) Davies-Bouldin (invertido)



(f) Dunn

Figura 3.13. Seis medidas de clustering calculadas para las particiones de consenso obtenidas usando el conjunto de datos Glass. La compactitud y el índice de Davies-Bouldin están invertidos, así los valores de arriba son mejores para todas las medidas.

ρ_1	Difficult Doughnut					Four Gaussian				
	RN	BC	Gk	FG	FGk	RN	BC	Gk	FG	FGk
D_{I2}	0.69	0.95	**0.98**	0.98	0.98	0.75	0.95	0.98	0.98	**0.98**
D_{E1}	0.66	0.71	0.96	0.97	**0.98**	0.15	0.74	0.95	0.97	**0.98**
D_{E2}	0.48	0.95	0.86	**0.97**	0.92	0.03	0.57	0.64	0.80	**0.89**

ρ_1	Iris					Ionosphere				
	RN	BC	Gk	FG	FGk	RN	BC	Gk	FG	FGk
D_{I2}	0.61	0.95	0.98	0.98	**0.98**	0.17	0.90	0.97	**0.98**	0.98
D_{E1}	0.07	0.70	0.97	0.97	**0.98**	0.65	0.62	0.98	**0.98**	0.98
D_{E2}	0.10	0.84	0.96	0.97	**0.97**	0.50	0.72	0.97	0.97	**0.98**

ρ_1	Glass					Wine				
	RN	BC	Gk	FG	FGk	RN	BC	Gk	FG	FGk
D_{I2}	0.51	0.95	0.97	0.97	**0.98**	0.56	0.93	0.98	0.98	**0.98**
D_{E1}	0.46	0.92	0.94	0.96	**0.97**	0.02	0.64	0.96	0.96	**0.96**
D_{E2}	0.02	0.86	0.92	0.88	**0.94**	0.17	0.73	0.93	**0.94**	0.93

Tabla 3.3: Suavidad de las medidas de ensambles.

Hu y otros, 2016; Kuncheva y Hadjitodorov, 2004), el cual crea aleatoriamente un conjunto de ensambles con diferentes diversidades; y el método propuesto en (Iam-On y otros, 2011), al cual nos referimos como basado en categorías (BC), y que genera conjuntos de ensambles que pertenecen a las categorías de diversidad "baja", "media" y "alta". El método de control de diversidad se utilizó con los tres algoritmos de agrupamiento: Gk, FG y FGk. Es importante notar que en estos experimentos se empleó solo un rango útil para h, es decir, un rango donde la diversidad exhibió cambios grandes. Utilizando estos cinco métodos, se generaron varias secuencias de ensambles ordenadas por diversidad ascendente. Se calculó el promedio de suavidad para las medidas de ensambles y de clustering para todas las repeticiones.

En la Tabla 3.3 se presentan los resultados de suavidad para las medidas de ensambles, donde se calculó el promedio de ρ_1 para las tres medidas de diversidad: D_{I2}, D_{E1} y D_{E2}. Los números en negrita indican el máximo valor promedio de ρ_1 obtenido para cada conjunto de datos y medida de ensamble. Las diferencias entre valores subrayados y no subrayados son estadísticamente significativas ($\alpha = 0.05$). Para todos los conjuntos de datos se encontró que RN siempre obtenía los valores más bajos para ρ_1. Esto significa que los valores de diversidad para D_{I2}, D_{E1} y D_{E2} cambiaron bruscamente comparados con D_{I1}. En contraposición, BC obtuvo mayores valores de ρ_1, lo que indica que pudo generar ensambles cuya diversidad cambia más suavemente que en los generados por el enfoque aleatorio. Esto podría ser explicado por la naturaleza propia de BC (avara, del inglés *greedy*)

al momento de crear ensambles en las diferentes categorías de diversidad, porque aunque este método contiene un componente aleatorio (la primer partición del ensamble es seleccionada aleatoriamente), el mismo elige la siguiente partición miembro incrementalmente de acuerdo a la categoría en cuestión. Cada categoría de diversidad generalmente produce un conjunto de ensambles similares, obteniendo así diferencias suaves entre ellos. Sin embargo, este comportamiento tiene un efecto secundario porque las categorías de diversidad frecuentemente difieren mucho entre ellas, por lo tanto BC no logra muestrear uniformemente el rango de diversidad.

El método de control de diversidad (con sus tres alternativas) obtuvo los mayores valores de ρ_1 para todos los conjuntos de datos, con diferencias estadísticamente significativas comparado con los otros métodos. Esto es así porque la capacidad de controlar la diversidad resultó en pequeños cambios entre los ensambles, y de esta forma *todos* los ensambles tuvieron diferencias suaves con los demás, mientras que BC creó categorías de ensambles cohesivas pero separadas. En general, todos los algoritmos de agrupamiento obtuvieron un desempeño similar, pero FGk produjo los mejores resultados, seguido de cerca por FG. El desempeño superior de estos algoritmos comparados con Gk fue debido a la cohesión de los grupos de particiones generados. Ciertamente, Gk solo agrupa por k, lo que podría dejar particiones muy diferentes dentro del mismo grupo.

La Tabla 3.4 presenta la suavidad en las salidas de la función de consenso, donde se muestran cinco medidas de clustering: la exactitud (A), compactitud (C), separación (W), el índice de Davies-Bouldin (B) y el índice de Dunn (D). Las tendencias en estos resultados son similares a las obtenidas para las medidas de ensambles. Los ensambles generados por RN ocasionaron que la función de consenso derive particiones impredecibles. Usando este método, la suavidad para las medidas de clustering fue siempre cercana a 0, lo que indica que los ensambles cambiaron de una forma altamente irregular de acuerdo a las medidas sobre las salidas de la función de consenso. Aunque BC fue muy superior a RN, el método de control de diversidad obtuvo el mejor desempeño. Además de producir ensambles que cambiaron de una forma suave desde el punto de vista de la diversidad (Tabla 3.3), el método de control de diversidad pudo también inducir cambios suaves en las particiones de consenso.

Para el método de control de diversidad, un análisis conjunto de la suavidad

ρ_1	Difficult Doughnut					Four Gaussian				
	RN	BC	Gk	FG	FGk	RN	BC	Gk	FG	FGk
A	0.03	0.91	0.77	**0.95**	0.82	0.03	0.64	0.60	0.76	**0.79**
C	0.02	0.88	0.68	**0.93**	0.77	0.03	0.47	0.55	0.66	**0.75**
W	0.02	0.92	0.77	**0.95**	0.82	0.04	0.47	0.60	0.72	**0.77**
B	0.02	0.89	0.80	**0.95**	0.82	0.03	0.17	0.60	0.71	**0.78**
D	0.28	0.35	0.51	**0.73**	0.57	0.02	0.57	0.65	0.74	**0.80**

ρ_1	Iris					Ionosphere				
	RN	BC	Gk	FG	FGk	RN	BC	Gk	FG	FGk
A	0.04	0.39	0.61	**0.81**	0.71	0.06	**0.58**	0.32	0.43	0.41
C	0.01	0.45	0.29	0.59	**0.70**	0.19	0.18	0.34	**0.50**	0.31
W	0.01	0.41	0.48	**0.78**	0.75	0.15	**0.61**	0.57	0.46	0.55
B	0.07	0.16	0.52	**0.76**	0.70	0.23	0.50	**0.56**	0.45	0.52
D	0.07	0.44	0.67	**0.85**	0.71	0.05	0.27	**0.45**	0.33	0.27

ρ_1	Glass					Wine				
	RN	BC	Gk	FG	FGk	RN	BC	Gk	FG	FGk
A	0.04	0.32	0.74	**0.83**	0.78	0.05	0.40	0.56	**0.71**	0.62
C	0.03	0.28	0.60	0.60	**0.70**	0.03	**0.46**	0.28	0.37	0.39
W	0.11	0.57	0.87	0.88	**0.90**	0.03	0.55	0.72	**0.81**	0.72
B	0.11	0.42	0.87	0.88	**0.89**	0.03	0.35	0.28	**0.44**	0.38
D	0.04	0.57	0.77	0.77	**0.82**	0.04	0.33	0.37	**0.50**	0.39

Tabla 3.4: Suavidad de las medidas de clustering.

para las medidas de ensambles y para las medidas de clustering muestra que dado un cambio leve en la entrada, la función de consenso pudo producir un cambio aceptablemente suave en la salida. Este fue el caso para los conjuntos de datos Four Gaussian, Iris y Glass, pero especialmente para Difficult Doughnut, donde el método propuesto pareció incluso controlar la calidad de la partición de consenso.

Estos resultados sugieren que los métodos actuales pueden generar ensambles con diferentes diversidades, pero sus estructuras internas pueden en realidad diferir de una forma impredecible. Así, estos métodos son menos útiles y fiables en casos donde es necesario analizar cómo la diversidad afecta el desempeño del consenso. Por el contrario, el método propuesto puede controlar una medida particular de diversidad del ensamble (como se mostró en la Sección 3.4.2), garantizando que todos los aspectos del mismo cambian de una manera suave. Esto fue demostrado por los altos valores de suavidad obtenidos desde las perspectivas de la diversidad y de la función de consenso. Por lo tanto, se puede afirmar que el método propuesto es más efectivo para generar ensambles que cambien gradualmente sus propiedades gracias a un control fino de sus diversidades.

3.5. Conclusiones

En este capítulo se ha presentado la contribución principal de esta tesis, donde se identificó y abordó un problema central en los métodos actuales para generar ensambles diversos. Cuando se utilizan estos métodos para generar ensambles de agrupamientos con diferentes niveles de diversidad, aquellos ensambles resultantes con diversidades similares pueden en realidad poseer propiedades muy diferentes, provocando que la función de consenso presente un comportamiento impredecible. Una consecuencia de esto es que se hace prácticamente imposible estudiar cómo la diversidad impacta en los resultados finales. Por el contrario, el método presentado puede incrementar y decrementar la disimilaridad entre los miembros del ensamble de una manera suave, proveyendo así un novedoso enfoque para estudiar la diversidad. El método analiza la estructura del ensamble inicial y produce en él cambios pequeños, lo que permite un control fino de la diversidad. El buen desempeño del mismo ha sido demostrado experimentalmente al evaluar sus diferentes pasos y etapas. El método produce ensambles que cambian de manera suave no solo desde el punto de vista de la diversidad, sino también de acuerdo a diferentes medidas de calidad calculadas en la salida de la función de consenso.

CAPÍTULO 4

Ensambles de agrupamientos para grandes datos en bioinformática

El objeto de tener una mente abierta, así como tener la boca abierta, es el de cerrarla otra vez sobre algo sólido.
— G. K. Chesterton

En este capítulo se presenta un conjunto de métodos y aplicaciones basados en ensambles de agrupamientos, que intentan abordar tres de las propiedades comúnmente reconocidas en los llamados "grandes datos" (Nature, 2008; Wu y otros, 2014): la heterogeneidad o *variedad* de fuentes de información, el gran *volumen* de datos disponible, y la *velocidad* de generación de los mismos. Estas características representan un gran desafío para la aplicación de un proceso de minería de datos. Primero se tratará el aspecto de la heterogeneidad o variedad sobre un conjunto de datos biológicos. Este trabajo ha resultado en una satisfactoria aplicación de un método de ensambles, junto con una propuesta de validación de resultados. Luego, se diseñó una arquitectura general para incluir el trabajo previo y abordar también los otros dos aspectos de los grandes datos: el volumen y la velocidad.

4.1. Método de ensambles sobre fuentes biológicas heterogéneas

Comúnmente, cuando se habla de grandes datos se sobrevalora el aspecto relacionado al volumen, es decir cuántos terabytes de información se deben procesar, pero la variedad de la información es quizá una cuestión aun más desafiante (Jagadish, 2015; Nature, 2008). Como se ha descripto en el capítulo anterior,

existe una importante dificultad a la hora de elegir un algoritmo y sus parámetros para un problema de clustering dado. Esto es aún más crítico cuando los datos presentan fuentes diversas, incluyendo quizá no solo datos numéricos sino también categóricos. La heterogeneidad de las fuentes de información es una característica habitual en varios conjuntos de datos de origen biológico, e influye significativamente en las tareas de minería sobre los mismos. Ciertamente, es común encontrar medidas heterogéneas sobre los mismos componentes biológicos. Estas medidas pueden incluir por ejemplo datos de expresión de genes, capacidad antioxidante de los materiales, o perfiles metabólicos; si bien estas brindan información sobre los mismos organismos o condiciones experimentales, presentan una significativa diferenciación en la distribución de las mediciones y escalas que comúnmente requieren un complejo paso previo de preprocesamiento (Basheer y Hajmeer, 2000; Mohabeer y otros, 2011).

Estas tareas previas a la aplicación de un algoritmo de aprendizaje maquinal transforman los datos para que estos posean ciertas características que, en teoría, produzcan un mejor desempeño del método a aplicar. Un ejemplo es cuando se aplica un método de estandarización como los explicados en el Capítulo 2. Sin embargo, la aplicación de dichos métodos no es trivial y presenta varios desafíos y dificultades para el usuario final. En esta sección se describirá la aplicación de un método de ensambles de agrupamientos sobre un conjunto de datos biológicos reales (Pividori y otros, 2013a). Como se mostrará, el mismo ha resultado ser más efectivo que las técnicas convencionales de clustering para estos escenarios, y representa un avance en cuanto al tratamiento de la primer característica de los grandes datos: la heterogeneidad o variedad de fuentes de información.

Durante el desarrollo de esta tesis doctoral, el doctorando ha codirigido un proyecto final de grado donde se ha implementado una herramienta web para utilizar estos métodos con otras fuentes de datos[1].

4.1.1. Análisis de clusters sobre accesiones de tomate

Uno de los problemas menos atendidos en la producción hortícola local es la pérdida de variabilidad en los materiales biológicos. Sin embargo, se presume que sí hay variabilidad en los materiales "criollos" que han sido mantenidos por

[1] `https://bitbucket.org/sinc-lab/webclusterensembles`

pequeños productores en determinadas zonas geográficas del país. En el marco del proyecto INTA AEBIO 243542, del cual nuestro grupo de investigación es parte, se planteó la recuperación y medición de diversas características de materiales criollos de tomate, para dar cuenta del enorme potencial que tiene el acervo genético mantenido por los productores, con subsistencia durante generaciones. A esos materiales se les midieron caracteres agronómicos asociados al rendimiento, y aquellos asociados a la calidad de los frutos (componentes del metabolismo primario y secundario). El análisis de 68 cultivares criollos permitió la identificación de distintos grupos definidos por los principales caracteres agronómicos asociados al rendimiento (Asprelli y otros, 2011). Un subgrupo de 35 de estos cultivares, caracterizados por sus complementos metabólicos en los frutos, también permitió distinguir grupos definidos. Estos resultados preliminares permitieron proveer conocimiento acerca de la variabilidad presente en los recursos genéticos de especies de interés comercial, para el uso racional de los mismos y su conservación. Además, mostraron la necesidad de diseñar métodos y herramientas que permitan el análisis de la variabilidad a partir de la integración de datos fenotípicos, genéticos, genómicos y metabólicos desde una perspectiva sistémica en biología.

En este marco, se utilizaron los datos de diferentes mediciones sobre 8 accesiones de tomate (*Solanum Lycopersicum*) cultivadas y recolectadas a lo largo de los valles andinos de la Argentina (Peralta y otros, 2008). Estos lugares son indicados con puntos rojos en la Figura 4.1, donde los demás puntos indican otros lugares donde se han colectado cultivos de pimiento y zapallo, cuyos datos no han sido incluidos en este trabajo. Cada accesión de tomate presenta una serie de características propias como el color y la forma. Los códigos de las accesiones utilizadas en esta tesis son: CMP, 557/C237, M82, GPEA, 3806/ALGR, 572/C526, 552/C169 y 569/C369.

El conjunto de datos utilizado cuenta con distintas fuentes de información, las cuales poseen a su vez diferentes cantidades de características (Tabla 4.1): I) los perfiles metabólicos (21 características), II) la capacidad antioxidante (3 características), III) pigmentos (2 características), IV) los paneles sensoriales (12 características), V) vitamina E (4 características) y VI) los volátiles (100 características). Ya que estas medidas se han tomado sobre 3 réplicas independientes de cada una de las 8 accesiones, se trabajó sobre un total de 24 objetos. Una réplica es una muestra independiente sobre el mismo material biológico. La principal

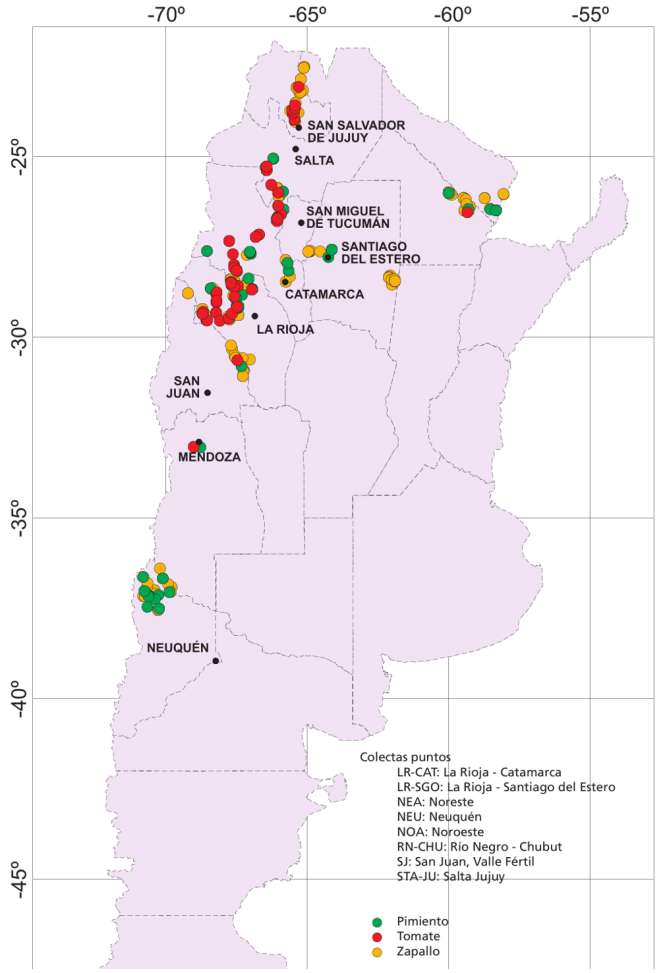
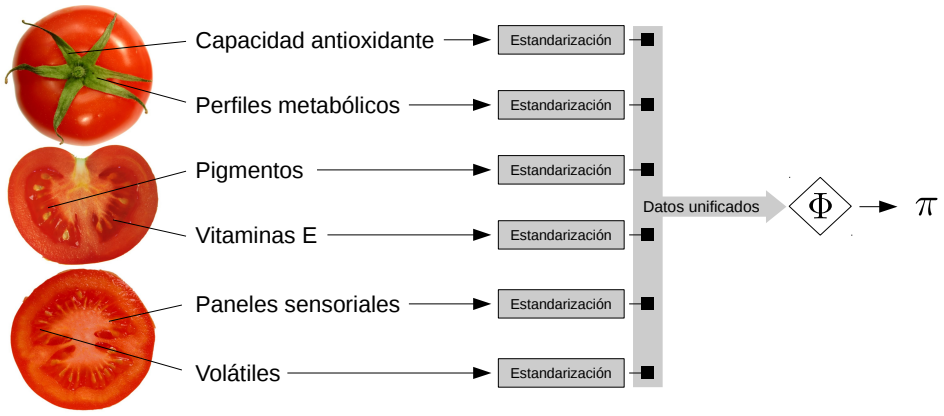


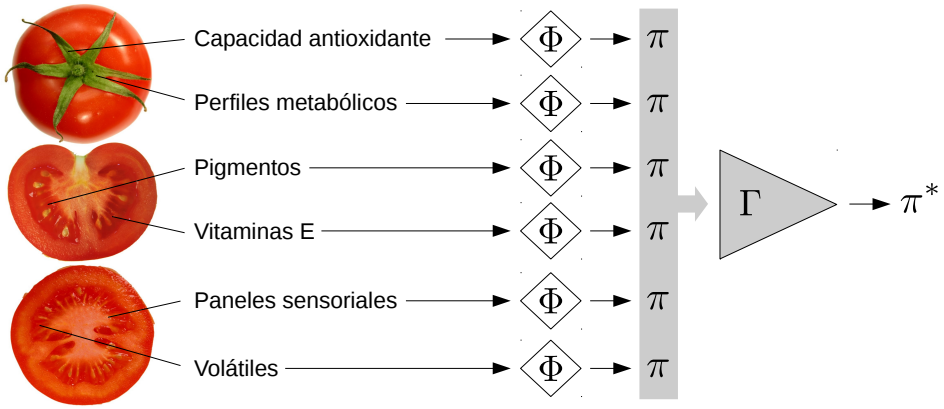
Figura 4.1. Valles andinos de Argentina donde se han colectado los tomates (indicados con puntos rojos) cuyos datos han sido utilizados en este trabajo. Figura tomada del trabajo (Peralta y otros, 2008).

Perfiles metabólicos	**Capacidad antioxidante**	**Pigmentos**	**Paneles sensoriales**	**Vitamina E**	**Volátiles**
citrate	Frap	licopenos	apariencia general	alphatocopherol	ethylacetate
fructoseDL	TEAC HS	carotenos	color piel	betatocopherol	3methylbutanal
glucose	TEAC LS		forma fruto	gammatocopherol	1penten3one
ethanol			color pulpa	total tocoferol	2ethylfuran
GABA			abundancia pulpa		3methyl1butanol
alanineDL			olor		2methyl1butanol
asparagineDL			sabor		UNK1volatiles
aspartateDL			dulzura		2pentenal
glutamateDL			acidez		toluene
glutamineDL			jugoso		hexenalm
isoleucineDL			textura pulpa		2hexenal
malateDL			textura piel		...
phenylalanineDL					alphaterpineol
threonineDL					decanal
tryptophanDL					methylsalicilate
valineDL					3methyl3hepten2one
methanol					eugenol
2oxoglutarate					arhimachalen2ol
succinate					propilsalicilate
sucrose					isoamylsalicilate
trigonelline					3hexenal
					hexanal

Tabla 4.1: Seis fuentes de datos y sus características. Por cuestiones de espacio, solo se muestra un subconjunto de las 100 características medidas de los volátiles.



(a) Enfoque de clustering tradicional sobre fuentes heterogéneas de datos. La integración es a nivel de fuentes.



(b) Enfoque propuesto de ensambles de agrupamientos sobre fuentes heterogéneas de datos. La integración es a nivel de ensamble o soluciones de clustering.

Figura 4.2. Comparación entre un proceso tradicional de clustering y el propuesto en este capítulo para el análisis de datos con fuentes heterogéneas. En gris se indican los componentes relacionados con el paso de integración.

complejidad de los datos aquí utilizados reside en la heterogeneidad de sus fuentes, que como se ha dicho también representa una característica de los grandes datos, además de ser una propiedad comúnmente encontrada en varios conjuntos de datos de origen biológico.

La aplicación de un algoritmo de clustering tradicional sobre estas fuentes heterogéneas requiere una serie de pasos manuales, como indica la Figura 4.2a. En ella pueden verse las diferentes medidas que se han tomado de los tomates, las cuales primero es necesario estandarizar por separado utilizando algún método de estandarización como los que se han explicado en la Sección 2.1.1. Finalmente, luego de estas tareas manuales, es posible aplicar un algoritmo tradicional de clustering Φ sobre el conjunto unificado de datos, el cual produce una partición

π de los mismos.

Si bien es posible analizar este tipo de datos con algoritmos convencionales de clustering, es importante tener en cuenta algunos de los problemas asociados al paso previo de estandarización: I) al cambiar los datos se pueden estar perdiendo propiedades importantes de los mismos que no solo afecten las soluciones de clustering, sino que también dificulten la posterior interpretación de los resultados; II) es necesario conocer previamente o poder estimar con certeza las propiedades de los datos (como su distribución) para lograr una correcta aplicación de los métodos de estandarización (Gan y otros, 2007; Kaufman y Rousseeuw, 2005); y además III) las diferentes estandarizaciones aplicadas a cada fuente deben producir un resultado tal que permita comparar adecuadamente los objetos, sin que una fuente domine a otras. Estas tres dificultades representan un gran desafío para el usuario final, que en la mayoría de los casos lleva a cabo estos pasos previos de estandarización posiblemente sin tener en cuenta sus implicaciones.

Con los ensambles de agrupamientos es posible plantear un esquema más simple y efectivo para este tipo de escenarios, el cual se muestra en la Figura 4.2b. A diferencia del esquema tradicional descripto antes, en este caso cada fuente es procesada de forma individual por un algoritmo de clustering Φ. Este tratamiento individual, al evitar la integración de datos, no requiere en principio ningún tipo de transformación previa de los mismos. Cada algoritmo Φ trabaja sobre los mismos componentes biológicos, pero utilizando diferentes fuentes de información sobre estos. Esto permite generar un ensamble de soluciones diversas, como se puede observar en el recuadro gris, que luego son integradas o combinadas por una función de consenso Γ. Este paso de integración produce finalmente una solución consensuada π^*. Como puede verse, la integración en este caso es a nivel de ensamble o soluciones de clustering, no a nivel de datos como en el enfoque tradicional. Esto permite no solo evitar los problemas de estandarización mencionados antes, sino que también ofrece todas las ventajas de los ensambles de agrupamientos, como proveer soluciones de mayor calidad y simplificar el cálculo distribuido. A continuación se darán más detalles de la aplicación del método propuesto en este dominio biológico.

4.1.2. Consenso de grupos de soluciones

El enfoque propuesto, llamado *consenso de grupos de soluciones*, posee dos pasos para facilitar la tarea de clustering sobre los datos. Primero, se aplica un método de ensambles de agrupamientos sobre las diferentes fuentes de datos. Luego, se utiliza un método de evaluación que cuantifica la información compartida entre la solución consensuada y las distintas fuentes.

El primer paso, que aplica un método de ensambles de agrupamientos considerando la heterogeneidad de las fuentes de datos (ver Figura 4.2b), consta de dos etapas. La primer etapa es la generación del ensamble, donde se utiliza un algoritmo simple y eficiente como k-medias para cada fuente individualmente, variando la cantidad de clusters k en un intervalo $[k_{min}, k_{max}]$. Para cada valor diferente de k, k-medias es ejecutado q veces inicializando aleatoriamente sus centroides. Este procedimiento es muy similar al utilizado en el Capítulo 3 para la generación de los ensambles. El objetivo de este procedimiento es obtener un *grupo de soluciones*, el cual se define como un conjunto con la mayor cantidad de estructuras posibles con un mismo número de cluster. Por lo tanto todos los grupos tienen q particiones con la misma cantidad de clusters, siendo cada una de ellas generada con inicializaciones aleatorias. Esto produce un ensamble final altamente diverso, con una gran variedad de *puntos de vista* sobre la estructura posible de los datos. La segunda etapa consiste en la combinación de este ensamble utilizando una función de supraconsenso, como se ha explicado en la Sección 2.3.2.3. De esta forma, la partición consensuada es aquella que maximiza la información compartida entre esta y cada solución miembro del ensamble. Es importante notar que en este paso ya no se necesita acceder más a las mediciones originales, sino solo a las particiones (el ensamble).

Una vez obtenidas las particiones finales de interés, se calcula la cantidad de información compartida entre estas y cada fuente de datos. Esto permite tener rápidamente una idea de la consistencia que hay con las diferentes fuentes. Para lograr esto, utilizando el mismo procedimiento empleado por el método de control de diversidad (Capítulo 3), se obtienen para cada fuente de datos un grupo de *particiones representantes* $\check{\Pi} = \{\check{\pi}_i\}$, donde cada representante $\check{\pi}_i$ tiene diferentes cantidades de clusters, según un intervalo predefinido $[\check{k}_{min}, \check{k}_{max}]$. Estos representantes son en realidad particiones consensuadas derivadas de los

ensambles individuales de cada fuente de datos. Por ejemplo, se podrían obtener nueve particiones representantes utilizando $\check{k}_{min} = 2$ y $\check{k}_{max} = 10$, sobre un ensamble generado sobre cierta fuente de datos. Este grupo de nueve representantes contienen de manera compacta la información estructural del ensamble.

Para cuantificar la información compartida entre la partición final y cada grupo de representantes se optó en este caso por el índice de información mutua ajustada (AMI, por sus siglas en inglés), definido en (2.25). Como se ha indicado en la Sección 2.2.1, a diferencia de otras medidas populares basadas en información mutua como el NMI (2.23), el AMI tiene la propiedad de estar ajustado con la probabilidad asociada para particiones aleatorias. Esta propiedad ofrece una ventaja cuando se comparan particiones con diferentes cantidades de clusters. Ciertamente, la información mutua tiende a conferir valores mayores a medida que se incrementa el número de clusters en relación a la cantidad de datos, por ejemplo, cuando $n/k < 100$ (Vinh y otros, 2010). Esta desventaja también afecta al NMI, al no ser una medida ajustada como el AMI. Dado que para nuestro caso se necesita comparar particiones con diferentes valores de k, y que la relación n/k no es mayor a 12 (siendo $n = 24$ y $k = 2$), resulta indispensable emplear una medida ajustada. Análogamente al ANMI, en este caso se ha usado el promedio del AMI (AAMI, por sus siglas en inglés) como

$$\bar{\Psi}(\pi^*, \check{\Pi}) = \frac{1}{P} \sum_{\forall \check{\pi}_i \in \check{\Pi}} \Psi(\pi^*, \check{\pi}_i), \tag{4.1}$$

donde π^* es una partición final y $\check{\Pi}$ un grupo de P particiones representantes de una fuente de datos.

4.1.3. Configuración experimental

Utilizando los datos ya descriptos, se ha comparado el método propuesto de ensambles y consenso de grupos con un enfoque tradicional usando k-medias. Para el método de consenso de grupos se generó el ensamble corriendo k-medias sobre cada fuente individualmente, con $k_{min} = 2$ y $k_{max} = 12$, obteniendo así 11 grupos de soluciones. El parámetro q fue establecido en 10, resultando en un total de 110 particiones por cada fuente de datos. En la función de supraconsenso se utilizaron los tres métodos de consenso basados en teoría de grafos, descriptos en la Sección 2.3.2.1: CSPA, HGPA y MCLA. Para el enfoque tradicional usando

k-medias se han unificado todas las fuentes de datos y aplicado luego el método de estandarización de unidad tipificada z-score (Sección 2.1.1) para cada una de las características de forma independiente.

Se han utilizado dos medidas de validación para cuantificar el desempeño de cada método. La primera de ellas se deriva de las propiedades de los datos en sí: la existencia de réplicas para cada una de las 8 accesiones de tomate puede considerarse como información *a priori* para definir una posible partición de referencia. Ciertamente, es esperable que las réplicas de una misma accesión (material) sean muy similares, y por lo tanto se agrupen juntas. Así, la partición de referencia tendrá 8 clusters, conteniendo cada uno a las 3 réplicas de cada accesión. La segunda medida de calidad cuantificará la información compartida entre la partición final con cada una de las fuentes, coincidiendo así con el método de evaluación descripto antes. Para obtener los representantes de cada fuente, se ha establecido $\check{k}_{min} = 2$ y $\check{k}_{max} = 12$. Todos los valores informados se obtuvieron tras correr los experimentos 100 veces.

4.1.4. Resultados y discusión

Para evaluar ambos métodos con respecto a la partición de referencia basada en las réplicas, cada uno fue ejecutado usando $k = 8$. El consenso de grupos siempre agrupó juntas las réplicas de una misma accesión, logrando una tasa de error del 0 %. El método tradicional usando k-medias, muy por el contrario, ha fallado en el 94 % de los casos, agrupando las réplicas de una misma accesión en clusters diferentes.

Los resultados con respecto a la segunda métrica de calidad pueden verse en la Figura 4.3, donde se muestra un gráfico de barras por cada fuente de datos. Los valores mostrados corresponden al promedio de las 100 repeticiones experimentales, indicando el intervalo de confianza de 95 % con una línea negra en la parte superior de cada barra. En el eje de las x se observa el número de clusters utilizado para la partición final. En el eje de las y se observa el valor de AAMI promedio alcanzado para cada una de las fuentes de datos. Por ejemplo, las soluciones finales con $k = 8$ obtenidas con el método de consenso de grupos obtuvo un AAMI promedio de 0.75 al compararse con la fuente de paneles sensoriales. El método tradicional con k-medias, en cambio, logró un AAMI promedio

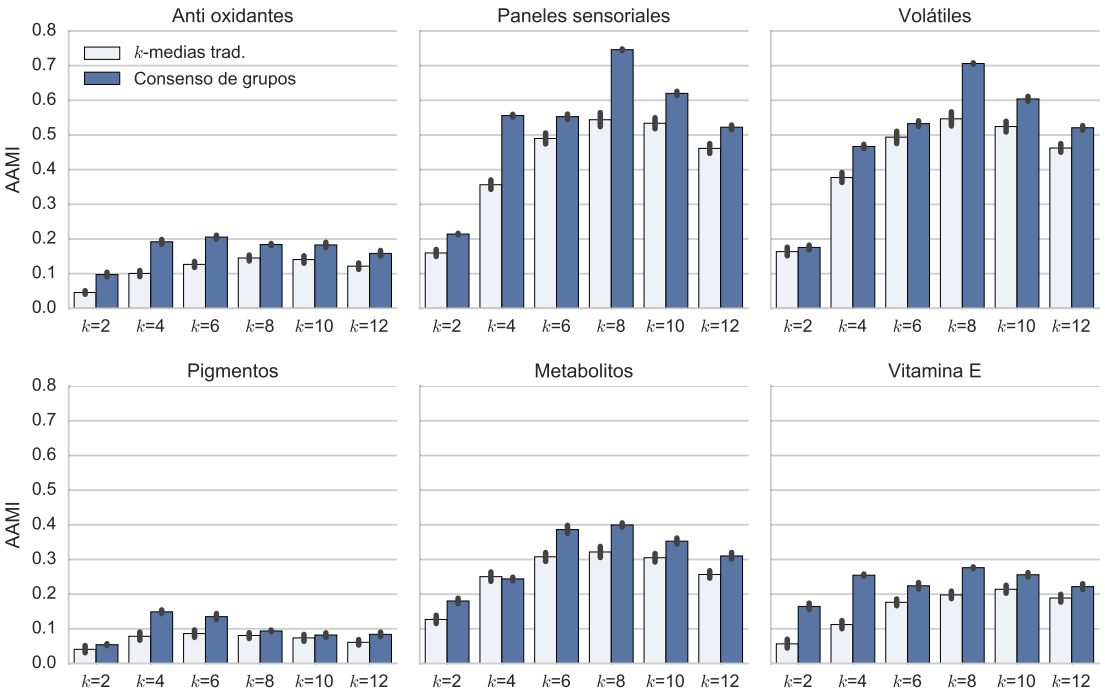


Figura 4.3. Información compartida entre cada partición final y cada una de las fuentes de datos para k igual a 2, 4, 6, 8, 10 y 12. Las lineas negras en cada barra indican los intervalos de confianza (95 %).

de 0.54. Al observar todas las fuentes, el método de consenso de grupos ha sido ampliamente superior al método tradicional, y en casi todos los casos la diferencia es estadísticamente significativa.

Es interesante analizar también la cantidad de información retenida por la misma partición final entre todas las fuentes de datos. Si una solución de clustering retiene más información de una fuente en particular, esto puede implicar una menor utilización de otras fuentes. Aunque el método tradicional con k-medias obtenga valores de AAMI promedio similares al consenso de grupos para ciertas configuraciones y fuentes, el método propuesto siempre logra una mayor utilización de información de las 6 fuentes de datos. Por ejemplo, para el caso de los perfiles metabólicos y $k = 4$, el método tradicional con k-medias logra un promedio de AAMI equivalente al consenso de grupos. Sin embargo, para esta partición con 4 clusters, el método propuesto retiene más información de todas las demás fuentes de datos. De esta forma, el consenso de grupos logra un mejor balance respecto a la consistencia con todas las fuentes en general, sin preferir una en particular. En cambio, k-medias puede estar siendo influenciado por la escala o distribución particular de las mediciones de metabolitos.

Un comportamiento interesante de la Figura 4.3 es que, para ambos métodos, algunas fuentes de datos tienen un peso mayor que otras en la solución final. Los paneles sensoriales y los volátiles claramente tienen una preponderancia muy superior a las otras cuatro fuentes. De hecho, la cantidad de información compartida entre cada una de las particiones finales (para todos los valores de k) con estas dos fuentes presenta un comportamiento muy similar. Esto se puede explicar por las propiedades que tiene una partición consensuada. La función de supraconsenso combina el ensamble completo (que incluye particiones de todas las fuentes) en una única solución que intenta maximizar el nivel de acuerdo con los miembros del ensamble. El hecho de que, en promedio, esta partición consensuada final comparta más información con determinadas fuentes significa que las estructuras que existen en ellas tienen más coincidencias con el resto. Concretamente, los 24 objetos descriptos por las 12 características de los paneles sensoriales o las 100 de los volátiles, poseen una estructura de clusters que tiene más coincidencias con los clusters de las fuentes restantes. Al contrario, los antioxidantes y los pigmentos poseen una estructura muy diferente a las otras fuentes. Estas dos fuentes comparten poca información con la estructura global resumida en la partición consensuada. Toda esta información brindada por el método propuesto puede ser muy interesante para el usuario del dominio, al cual se le brinda no solo un método para obtener mejores soluciones de clustering, sino también para entender mejor la relación existente entre las diferentes mediciones de los datos y entre los distintos materiales biológicos.

Los resultados del AAMI sobre cada fuente parecen también brindar información adicional sobre los clusters de las particiones finales. Al observar las dos fuentes de las que se retiene más información (paneles sensoriales y volátiles), puede verse que con $k = 8$ se obtiene el mayor AAMI. Esto resulta mucho más claro con el método de consenso de grupos que con el método tradicional. Este comportamiento representa un resultado interesante ya que los datos poseen 8 accesiones, y el método podría ser útil también para identificar el mejor número de clusters. Este hecho puede verificarse en los datos de tomate calculando qué tan estables son las particiones consensuadas (Kuncheva y Vetrov, 2006). Una medida de estabilidad muy utilizada es el *índice de consenso* (CI, por sus siglas en inglés) (Vinh y otros, 2010), el cual se basa en la similaridad interna de un grupo de soluciones para estimar el número de clusters en los datos. Para su cálculo,

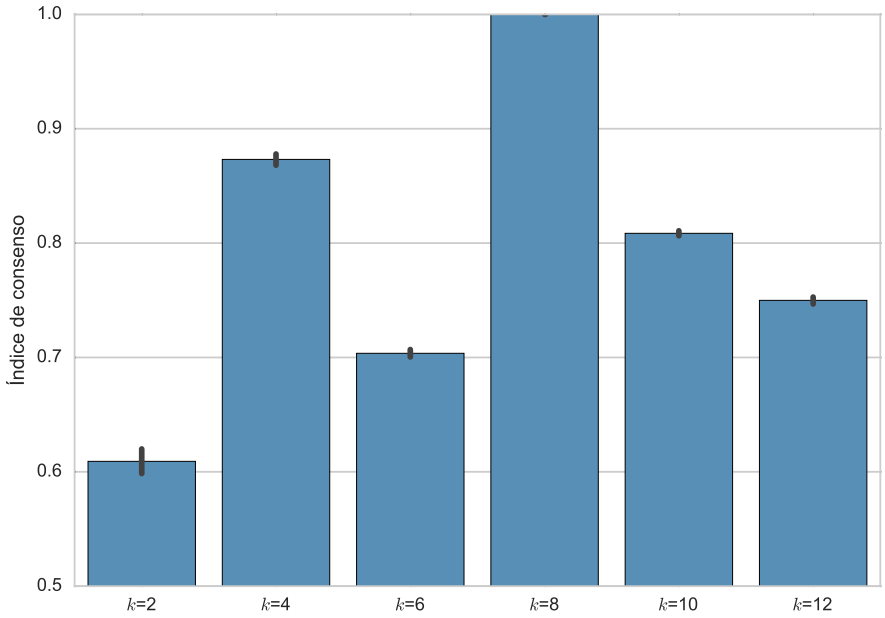


Figura 4.4. Índice de consenso para los datos de tomates. El índice fue calculado para las particiones consensuadas finales con $k = 2, 4, 6, 8, 10$ y 12, y los valores corresponden al promedio de 100 repeticiones.

primero se generan grupos Π_k con varias particiones usando el mismo número de clusters k. Luego se mide qué tan similares son los miembros de cada grupo y, si este valor es relativamente alto, se considera como una indicación de que los datos podrían tener esa estructura. Dado un determinado valor de k, el CI se calcula generando primero P particiones con k clusters, $\Pi_k = \{\pi_1, \pi_2, \ldots, \pi_P\}$ y luego

$$CI(\Pi_k) = \frac{2}{P(P-1)} \sum_{i<j} \Psi(\pi_i, \pi_j), \tag{4.2}$$

donde un valor cercano a 1 para $CI(\Pi_k)$ es indicativo de que los datos podrían tener k clusters. Los resultados de este índice de consenso pueden verse en la Figura 4.4, donde se han creado seis grupos de particiones Π_k con $k = 2, 4, 6, 8, 10$ y 12. Puede verse claramente que, según el índice de consenso, los datos poseen 8 clusters, ya que el $CI(\Pi_8)$ alcanzó el valor máximo de 1.0. Esto coincide con lo observado en la Figura 4.3, donde las fuentes de datos más representativas también indican una posible estructura con 8 clusters.

Por último, y tomando como sugerencia la segunda estructura más estable según el índice de consenso, en la Figura 4.5 puede verse una solución de ejemplo usando consenso de grupos con $k = 4$. Los puntos con la misma forma representan a las tres réplicas de cada accesión, mientras que el cluster es indicado con

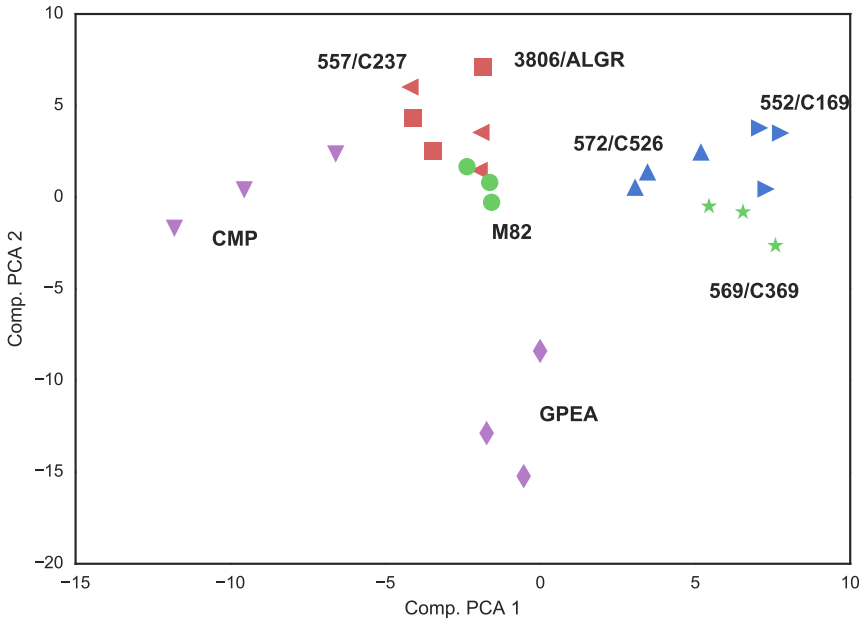


Figura 4.5. Solución de 4 clusters obtenida con el método de consenso de grupos. Los puntos con la misma forma indican las réplicas de cada accesión, y el color representa a los clusters.

un color diferente (rojo, azul, verde y púrpura). Este gráfico se realizó utilizando una proyección de los datos estandarizados con dos componentes principales (PCA). Según la interpretación biológica provista por el experto del dominio, esta partición agrupa las accesiones de tomate exclusivamente por sus complementos metabólicos, y existe un vínculo con los grupos genéticos a los cuales estos fueron asociados utilizando marcadores moleculares (Asprelli y otros, 2011). Por ejemplo, las accesiones 572/C526 y 552/C169 han sido clasificadas en dos grupos genéticos diferentes (G3 y G4), sin embargo ambas fueron colectadas en la misma región andina (Catamarca). El cluster indicado con color rojo agrupó dos accesiones (557/C237 y 3806/ALGR) las cuales también pertenecen a dos grupos genéticos diferentes, aunque ambas presentan frutas con formas similares. Estos resultados sugirieron al experto de dominio que, dado que las accesiones son indistinguibles en términos genéticos, será difícil cultivar tomates con mejores valores nutricionales si los programas de mejora biotecnológica no son dirigidos a estudiar los mecanismos detrás de la baja heredabilidad de estos tipos de caracteres biológicos.

4.2. Arquitectura para grandes volúmenes de datos

En la sección anterior se desarrolló un método para tratar efectivamente la heterogeneidad de las fuentes. La propuesta planteada permite en principio tratar cada fuente de datos con un método de clustering específico, y luego combinar las particiones generadas gracias a una función de consenso. Este enfoque permite que el componente de integración se desentienda de la complejidad asociada al preprocesamiento de los datos en formatos variados. Los grandes datos también pueden presentar enormes volúmenes de información que se producen a una gran velocidad (Labrinidis y Jagadish, 2012; Mervis, 2012). Ciertamente, en la actualidad 2.5 quintillones de bytes son generados cada día, y el 90 % de los datos en todo el mundo hoy fue producido en los dos años anteriores[2]. Los datos de origen biológico son un ejemplo contundente en este sentido, debido al rápido avance de nuevas tecnologías con una alta producción de datos novedosos y de gran valor (Li y Chen, 2014). Por ejemplo, gracias a las modernas tecnologías de secuenciación de alto rendimiento (NGS, del inglés *next-generation sequencing*), capaces de producir una enorme cantidad de información, se han logrado realizar descubrimientos de relevancia (Gou y otros, 2014; Howe y otros, 2008; Lupski y otros, 2010; Ng y otros, 2009). Esto significa que es de gran importancia el diseño de nuevos métodos que permitan explorar este tipo de datos, que ha probado ser una importante fuente de nuevo conocimiento.

El desafío para los métodos de minería de datos en estos escenarios no es solo ser capaz de procesar esta gran cantidad de información, sino también de poder hacerlo en un lapso de tiempo razonable (Wu y otros, 2014). En esta sección se describirá una arquitectura general que intenta guiar el diseño de un enfoque de clustering para escenarios de grandes datos (Pividori y otros, 2015). Incorporando el método descripto en la sección anterior y aprovechando las ventajas de los ensambles de agrupamientos y el control de la diversidad, se ha propuesto una arquitectura que contemple también los problemas de grandes volúmenes de información que se producen a gran velocidad. Además, guiado por esta arquitectura propuesta, se describirá un nuevo tipo de función de consenso capaz de

[2] `https://www-01.ibm.com/software/data/bigdata/what-is-big-data.html`

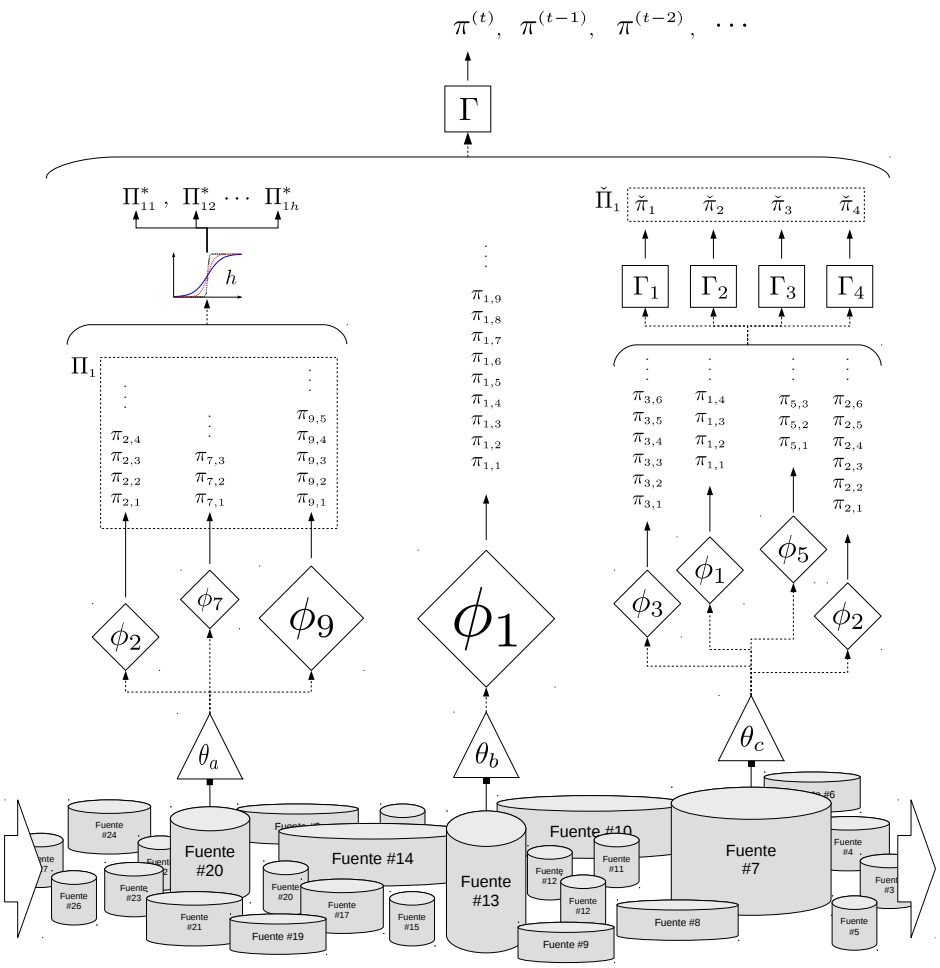


Figura 4.6. Arquitectura BigCE.

procesar una gran cantidad de datos de manera incremental.

4.2.1. Grandes ensambles de agrupamientos: arquitectura y componentes

Los ensambles de agrupamientos han probado ser útiles para escenarios de computación distribuida (Hore y otros, 2009; Strehl y otros, 2002), lo cual los hace interesantes para problemas de minería con grandes datos. Si bien se han propuesto unas pocas alternativas para tratar con enormes volúmenes de datos utilizando ensambles (Hore y otros, 2009; Traganitis y otros, 2014), estas se centran en la optimización individual de ciertos componentes, en lugar de un enfoque general de consenso para grandes datos.

En la Figura 4.6 se muestra una arquitectura propuesta para escenarios de grandes datos, a la cual llamamos grandes ensambles de agrupamientos (BigCE,

del inglés *big cluster ensembles*). En la parte inferior de la figura se puede ver una gran cantidad de fuentes de datos heterogéneas, las cuales contienen información en una variada cantidad de formatos. Además, estas fuentes pueden ser dinámicas, dado que constantemente hay información nueva que puede estar disponible y otra volverse rápidamente obsoleta. En este escenario el usuario final quiere obtener conocimiento actualizado de este tipo de datos cambiante, y por esta razón las particiones consensuadas finales $\pi^{(t)}$ (en la parte superior de la figura) también cambian con el tiempo.

El primer componente de esta arquitectura son los *muestreadores*, denotados con θ_i. Estos son responsables de acceder directamente a los datos, y tienen la tarea de obtener diferentes muestras de los objetos y sus características. El objetivo es dividir el problema de clustering, creando subtareas factibles de procesar por cualquier algoritmo convencional. El siguiente componente, llamado *clusterer* y denotado como ϕ_i, obtiene particiones π_{ij} de esta muestra de los datos. Como se ha explicado en la Sección 2.3.1, un clusterer es una instancia de un algoritmo de clustering tradicional con un conjunto fijo de parámetros. Éstos pueden correrse varias veces sobre la misma muestra variando algún parámetro o su inicialización, y generar así diversidad en los ensambles creados, de la misma forman que se hizo en el Capítulo 3 con el método de control de diversidad. La configuración de un clusterer puede ser ajustada teniendo en cuenta los recursos computacionales existentes en el nodo donde se ejecutará, lo que posibilita un modelo de escalado horizontal (Singh y Reddy, 2014). Esta característica se muestra en la figura utilizando diferentes tamaños para los clusterers ϕ_i.

Los componentes descriptos hasta ahora son los responsables de generar los ensambles. Como puede verse, el ensamble global consiste en varios subensambles, generados por diferentes clusterers que procesan posiblemente diferentes fuentes de los mismos datos. Dado que el trabajo de los clusterers es continuo debido a la naturaleza dinámica de los datos, los ensambles se generan continuamente. El tamaño de los diferentes subensambles puede fijarse de antemano según los recursos disponibles y la complejidad de los datos. Las particiones más viejas pueden ser reemplazadas por las más nuevas cada cierto tiempo. Sin embargo, los subensambles pueden ser de un tamaño significativo, con miles de particiones generadas sobre un volumen de datos considerable.

Un ensamble puede representarse como una matriz de $n \times p$, conteniendo

p particiones sobre n objetos. Debido a que esta matriz puede significar un problema inviable de resolver por una función de consenso, el ensamble puede ser, igual que los datos, reducido de diferentes maneras. Una forma sencilla es tomar muestras, análogamente a como se muestrean los datos. Estas muestras del ensamble pueden ser aleatorias o utilizando algún criterio de *selección de ensembles de agrupamientos* que, teniendo en cuenta la diversidad y diferentes métricas sobre las particiones miembro, eligen un subconjunto de ellas (Azimi y Fern, 2009; Gullo y otros, 2009; Naldi y otros, 2013; Yu y otros, 2014). Otra manera puede consistir en utilizar *particiones representantes* del ensamble, como se ha hecho en el Capítulo 3 con el método de control de diversidad. En lugar de utilizar todos los miembros del mismo, la información de la estructura de los datos puede resumirse en un conjunto reducido $\check{\Pi}$ de particiones consensuadas intermedias que en la figura se denotan como $\check{\pi}_i$. En este caso, se emplean funciones de consenso Γ_i para obtener representantes de alta calidad, utilizando diferentes métodos y cantidad de clusters para las soluciones. Finalmente, en lugar de simplemente reemplazar el ensamble por otro de menor tamaño, este puede reemplazarse por un conjunto de ensambles mediante el método de control de diversidad, extensamente detallado en el Capítulo 3 de la tesis. Ciertamente, partiendo de un ensamble inicial Π_1 de gran tamaño, es posible generar varios ensambles con diversidad controlada Π^*_{1j} que de cierta forma representen al ensamble inicial. Así como los componentes θ_i y ϕ_j generan subconjuntos de datos y de particiones diversas, el método de control de diversidad puede partir de un gran ensamble para generar otros con diferentes grados de diversidad e incluso menor tamaño. Esta tarea se realiza sin acceder a los datos originales, y permite explorar mejor las estructuras contenidas en el ensamble inicial.

Las distintas alternativas para lidiar con subensambles de gran tamaño pueden combinarse entre ellas. Donde haya un conjunto de particiones, es decir un ensamble, puede aplicarse una técnica de selección de agrupamientos, correr el método de control de diversidad, o utilizar una función de consenso para obtener representantes. Como estas tres técnicas a su vez generan otros ensambles, estas pueden combinarse de diferentes maneras según las necesidades y recursos disponibles. Un aspecto interesante aquí, y al igual que sucede con los muestreadores, es la posibilidad de reducir el tamaño de los ensambles, permitiendo así utilizar también los métodos de consenso tradicionales, los que poseen diferentes propie-

dades que los hacen aptos para gran variedad problemas. Esto es importante ya que la arquitectura BigCE, al contemplar el problema en su conjunto, no se centra en un único componente en particular como los enfoques actuales, lo que permite el uso de diferentes métodos y técnicas existentes.

El último componente en la arquitectura propuesta es una función de consenso que toma toda la información producida y contenida en los diferentes tipos de ensambles y genera una única partición consensuada final $\pi^{(t)}$. Como se observa en la parte superior de la figura, esta partición final puede ir cambiando a medida que se procesen más datos y se generen más soluciones intermedias. Esta función de consenso superior tiene ciertas características particulares que la diferencian de los métodos de consenso tradicionales que se utilizan, por ejemplo, para obtener particiones representantes. A diferencia de los clusterers ϕ_i que procesan cantidades de datos reducidas gracias a los muestreadores θ_j, la función de consenso superior debe ser capaz de procesar todos los objetos juntos. En este nivel ya no hay posibilidad de volver a reducir la cantidad de datos, como se hace en los niveles inferiores. Sin embargo, tal cantidad de información es imposible de procesar por los métodos tradicionales (tanto de clustering como de consenso), ya que en muchos casos ni siquiera es posible cargar toda la información en la memoria principal. Esta limitación, evidenciada en la arquitectura propuesta, motivó el desarrollo de una función de consenso que pueda trabajar con todos los datos pero procesándolos por lotes. A continuación se describirán los detalles de su funcionamiento y los resultados iniciales obtenidos.

4.2.2. Función de consenso incremental

Un ensamble, en su definición más básica, puede pensarse como una matriz de $n \times p$, con p particiones diferentes de los n objetos. Todos los métodos de consenso actuales, como se explicó en la Sección 2.3.2, básicamente transforman esta representación del ensamble en otra y luego aplican sobre ella un algoritmo de clustering tradicional. El hecho de que los métodos de consenso funcionen mejor reside en que esta nueva representación de los datos, basada en la similaridad entre ellos según pertenezcan o no a los mismos clusters, logra capturar mejor su estructura. Los métodos basados en teoría de grafos, descriptos en la Sección 2.3.2.1, representan el ensamble como grafos o hipergrafos, y luego aplican

	Caract. categóricas			
	π_1	π_2	π_3	π_4
$\mathbf{x}_1$	1	3	2	1
$\mathbf{x}_2$	1	3	2	1
$\mathbf{x}_3$	1	2	1	?
$\mathbf{x}_4$	2	2	1	1
$\mathbf{x}_5$	2	2	3	2
$\mathbf{x}_6$	2	1	3	2
$\mathbf{x}_7$	2	1	4	?

	Caract. numéricas										
	Ω_1^1	Ω_2^1	Ω_1^2	Ω_2^2	Ω_3^2	Ω_1^3	Ω_2^3	Ω_3^3	Ω_4^3	Ω_1^4	Ω_2^4
$\mathbf{x}_1$	1	0	0	0	1	0	1	0	0	1	0
$\mathbf{x}_2$	1	0	0	0	1	0	1	0	0	1	0
$\mathbf{x}_3$	1	0	0	1	0	1	0	0	0	0	0
$\mathbf{x}_4$	0	1	0	1	0	1	0	0	0	1	0
$\mathbf{x}_5$	0	1	0	1	0	0	0	1	0	0	1
$\mathbf{x}_6$	0	1	1	0	0	0	0	1	0	0	1
$\mathbf{x}_7$	0	1	1	0	0	0	0	0	1	0	0

Tabla 4.2: Dos representaciones de un ensamble *basadas en características*: como características categóricas (izquierda) y como características numéricas usando la matriz binaria de asociación de clusters (derecha).

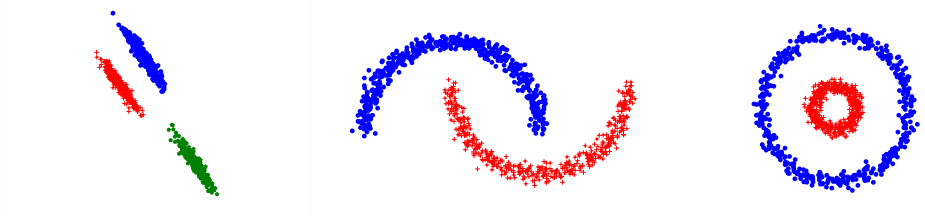
un algoritmo de particionamiento sobre ellos. Por otro lado, los métodos basados en acumulación de evidencia, descriptos en la Sección 2.3.2.2, utilizan una representación del ensamble llamada matriz de coasociación. Allí, en la Figura 2.5, se puede ver claramente cómo la matriz de similaridad según la coasociación supera a aquella basada en los datos originales, permitiendo así revelar mejor su estructura.

En la literatura se han propuesto también otras representaciones de los ensambles, acompañadas de nuevos métodos para derivar una partición de consenso (Fern y Brodley, 2004; Topchy y otros, 2003). Dos de estas representaciones, ambas *basadas en características*, se muestran en la Tabla 4.2. A la izquierda puede verse un ensamble con cuatro particiones π_j sobre siete objetos $\mathbf{x}_i$. La primer partición, π_1, posee 2 clusters ($k = 2$), mientras que π_2, π_3 y π_4 tienen 3, 4 y 2 clusters, respectivamente. La partición π_4 no tiene información de agrupamiento de todos los objetos, posiblemente por haberse obtenido a partir de un subconjunto de ellos. Esta capacidad de representar agrupamientos parciales es especialmente útil en la arquitectura BigCE. En esta primer representación, las particiones son tratadas como nuevas características categóricas de los objetos (Topchy y otros, 2005). Luego, cualquier algoritmo de clustering para datos categóricos sirve de función de consenso para esta representación. La representación de la derecha, conocida como matriz binaria de asociación de clusters (Fern y Brodley, 2004; Iam-On y otros, 2011), es otra forma de representar la información de agrupamiento del ensamble. En este caso, cada columna representa un cluster Ω_i^j de la partición j, y cada valor de la matriz puede ser 1 o 0, de acuerdo a si el objeto pertenece o no al cluster. Si bien la información contenida en esta matriz binaria ha sido utilizada para formular el problema de consenso utilizando grafos (Fern y

Brodley, 2004; Strehl y otros, 2002), como se explicó en la Sección 2.3.2.1, se la ha empleado también considerando a las columnas como nuevas características de los objetos (Iam-On y otros, 2011), aunque en este caso no son categóricas sino numéricas. Para la función de consenso propuesta se usará esta última representación, que utiliza la matriz binaria de asociación de clusters como nuevas características de los datos, y aplica sobre ella un algoritmo de clustering incremental.

En cada iteración del algoritmo clásico de k-medias (explicado en la Sección 2.1.2) se calcula la distancia de *todos* los objetos con los k centroides, asignándolos luego al más cercano. Este cómputo de distancias en k-medias es el paso más costoso computacionalmente (Capó y otros, 2016), y por esta razón han surgido varias alternativas que intentan reducir esta complejidad (Bradley y Fayyad, 1998; Davidson y Satyanarayana, 2003; Sculley, 2010). Entre estas alternativas para tratar con grandes volúmenes de datos, se encuentra una llamada *k-medias basado en mini-lotes* (Sculley, 2010) (*mini-batch k-means*, en inglés), y nos referiremos a ella como MBkM. Esta variante, en lugar de usar todos los objetos en cada iteración, emplea un conjunto más pequeño, reduciendo así la cantidad de distancias a computar. Este conjunto reducido de datos es una muestra aleatoria de tamaño fijo llamada *mini-lote*. Luego, en la etapa de inicialización, se escogen aleatoriamente k objetos como centroides, al igual que el k-medias clásico, y se elige de manera aleatoria los mini-lotes de b objetos. De esta manera, para cada objeto en un mini-lote, se obtiene el centroide más cercano y luego se utiliza un algoritmo basado en descenso por gradiente incremental (o estocástico) para actualizar la posición de los centroides (Bottou y Bengio, 1995). Si bien las soluciones halladas por MBkM pueden tener una calidad ligeramente inferior al método tradicional (Sculley, 2010), la reducción del tiempo para procesar grandes volúmenes de datos es significativa, y el hecho de poder hacerlo de manera incremental lo convierte en una opción interesante para la arquitectura propuesta en esta sección.

La función de consenso incremental propuesta utiliza esta versión de k-medias basada en mini-lotes para derivar del ensamble una partición de consenso (nos referimos a ella como C-MBkM). Además de utilizar la matriz binaria de asociación de clusters, también se propone una variante donde se aplica análisis de componentes principales (PCA, por sus siglas en inglés) a dicha matriz, obteniéndose una proyección de la misma utilizando dos componentes principales.



(a) Gaussians. (b) Moons. (c) Circles.

Figura 4.7. Tres conjuntos de datos con 10 millones de objetos generados artificialmente.

El objetivo de esta variante es permitir explorar el comportamiento de C-MBkM con datos binarios y continuos. Debido al tamaño de los datos, el cálculo de los componentes principales se realiza utilizando IPCA (Ross y otros, 2008), una versión incremental de PCA que, al igual que MBkM, trabaja utilizando mini-lotes de b objetos. Este algoritmo tiene una complejidad de memoria constante, en el orden de b. A diferencia de PCA, que ejecuta una única descomposición en valores singulares (SVD, por sus siglas en inglés) de $O(nm^2)$, IPCA realiza n/b cálculos de SVD de $O(bm^2)$, donde m es el número de características de los datos.

Por último, como tercer variante, se ha utilizado la función de supraconsenso (Sección 2.3.2.3) configurada para correr los dos métodos anteriores, C-MBkM y C-MBkM+PCA, la cual escoge la solución que mejor representa al ensamble (nos referiremos a ella como supra-C-MBkM). Debido al gran volumen de datos, y a que el índice AMI definido en (2.25) es más costoso de calcular, en este caso supra-C-MBkM utiliza ARI (2.22) para medir el nivel de acuerdo entre la partición consensuada final y cada miembro del ensamble. Análogamente a como se definió el ANMI en (2.30), definimos el promedio del ARI (AARI, por sus siglas en inglés) como

$$\bar{\Lambda}(\pi^*, \Pi) = \frac{1}{P} \sum_{\forall \pi_i \in \Pi} \Lambda(\pi^*, \pi_i), \tag{4.3}$$

donde π^* es la partición de consenso y Π un ensamble con P miembros.

4.2.3. Configuración experimental

Los tres métodos propuestos (C-MBkM, C-MBkM+PCA y supra-C-MBkM) han sido comparados con MBkM utilizando tanto los datos originales como una proyección de estos con PCA (con dos componentes

principales). Para la evaluación se han utilizado tres conjuntos de datos artificialmente generados[3], los cuales se muestran en la Figura 4.7. Todos tienen 10 millones de objetos y 10 características, de las cuales dos son reales y ocho contienen ruido uniformemente distribuido. El primer conjunto de datos, llamado Gaussians, posee 3 clusters gaussianos elongados, y como puede verse en la figura representa la estructura más sencilla de todas. El segundo conjunto de datos, Moons, contiene 2 clusters, aunque en este caso el nivel de dificultad es mayor que Gaussians. Por último, Circles posee 2 clusters que representan dos círculos concéntricos. Este tipo de estructura, utilizando los datos originales, es imposible de detectar por MBkM. Se ha utilizado la exactitud como medida de desempeño, definida como $\Lambda(\pi', \pi^*)$, donde π' es la partición de referencia de los datos y π^* la solución brindada por los métodos evaluados.

Los métodos MBkM y C-MBkM han sido entrenados con una única pasada simple por los 10 millones de objetos, con un tamaño de lote igual al 2 % ($b = 200.000$). Los ensambles han sido generados ejecutando MBkM sobre los datos originales, utilizando un rango de k entre 2 y 30, produciendo así ensambles con 29 soluciones. El método supra-C-MBkM se configuró para correr tanto C-MBkM como C-MBkM+PCA, cinco veces cada uno con inicializaciones aleatorias de los centroides, escogiendo aquella partición que maximiza el AARI definido en (4.3). Todos los métodos utilizaron un valor de k igual al número original de clusters en los datos, y se han realizado 10 corridas independientes.

4.2.4. Resultados y discusión

Los resultados del experimento pueden verse en la Tabla 4.3. Los datos presentados corresponden a los valores mínimos, la media y desviación estándar ($\mu(\sigma)$), y los máximos del ARI entre la partición final obtenida por cada método y la partición de referencia de cada conjunto de datos. En general, los métodos de consenso presentados han obtenido resultados superiores a las técnicas convencionales. Para el caso de Circles, cuya estructura es imposible de recuperar por MBkM, los métodos de consenso propuestos C-MBkM y C-MBkM+PCA han logrado mayores valores de ARI, aunque las desviaciones acusan cierta inestabili-

[3]Se han utilizado las funciones de generación de datos artificiales de scikit-learn, descriptas en: `http://scikit-learn.org/stable/datasets/index.html`

Λ	Circles			Moons			Gaussians		
	mín	$\mu(\sigma)$	máx	mín	$\mu(\sigma)$	máx	mín	$\mu(\sigma)$	máx
MBkM	0.00	0.00(0.00)	0.01	0.25	0.25(0.00)	0.26	0.44	0.84(0.23)	1.00
MBkM+PCA	0.00	0.00(0.00)	0.01	0.20	0.38(0.15)	0.54	0.44	0.94(0.17)	1.00
C-MBkM	0.02	0.24(0.19)	0.51	0.01	0.25(0.18)	0.41	0.44	0.74(0.24)	1.00
C-MBkM+PCA	0.18	0.36(0.32)	1.00	0.01	0.27(0.13)	0.37	0.45	0.73(0.27)	1.00
supra-C-MBkM	1.00	**1.00**(0.00)	1.00	0.26	**0.40**(0.18)	0.75	1.00	**1.00**(0.00)	1.00

Tabla 4.3: Resultados para los tres conjuntos datos Circles, Moons y Gaussians empleando los cinco métodos descriptos, donde los dos primeros son los algoritmos tradicionales sobre los datos originales y sus proyecciones con PCA; los tres últimos son los métodos de consenso incrementales propuestos.

dad, provocada posiblemente por las inicializaciones aleatorias de los centroides. Si bien la variante que usa PCA ha logrado desempeñarse mejor, solo el criterio de maximización del AARI de supra-C-MBkM ha logrado una estabilidad perfecta, recuperando siempre la verdadera estructura de los datos.

Al observar los resultados para Moons, puede verse que todos los métodos han obtenido resultados similares, aunque el que alcanza el mayor promedio es supra-C-MBkM. Los métodos MBkM, en sus dos versiones, se han comportado de forma similar a las dos variantes de C-MBkM. Si bien con estos datos la mayoría de los métodos se comportan de manera inestable, supra-C-MBkM ha obtenido consistentemente los máximos valores de exactitud.

Por último, los métodos tradicionales y las dos funciones de consenso propuestas han logrado desempeñarse de forma similar con el conjunto de datos Gaussians. En este caso, MBkM+PCA ha logrado en promedio una exactitud mayor, aunque con una alta inestabilidad. Nuevamente, solo el criterio de supra-C-MBkM ha logrado resultados perfectamente estables y exactos. La representación del ensamble utilizando una matriz binaria de asociación ha resultado efectiva para encontrar estructuras imposibles de hallar por los métodos tradicionales, como se mostró con el conjunto de datos Circles. Si bien C-MBkM y C-MBkM+PCA muestran cierta inestabilidad en su comportamiento, el criterio de selección de supra-C-MBkM ha mostrado ser eficaz para paliar estos problemas y lograr una estabilidad y exactitud muy elevadas.

Es importante notar que un k-medias tradicional necesitaría cargar en la memoria todos los datos al mismo tiempo para poder procesarlos. Para estos tres casos, considerando que cada valor de punto flotante ocupa 8 bytes en la memoria, una matriz de $1 \cdot 10^7 \times 10$ representa alrededor de 760 MB. Los métodos

incrementales, por el contrario, solo necesitan alrededor de 15 MB por mini-lote en la configuración experimental empleada, lo que supone una reducción muy significativa. Las funciones de consenso incrementales propuestas, al trabajar sobre una representación de los datos basada en el ensamble de entrada, requieren otro cálculo para conocer la memoria utilizada por cada mini-lote. Para estos tres conjuntos de datos, donde el ensamble creado posee 29 particiones con k en el intervalo $[2, 30]$, la representación basada en la matriz de asociación de clusters (Tabla 4.2) posee 464 columnas, que corresponden a la cantidad total de clusters del ensamble. Considerando que cada valor binario ocupa 1 byte de memoria en una implementación estándar de estos métodos, cada mini-lote de esta matriz necesita alrededor de 90 MB según la configuración experimental usada aquí. Sin embargo, existen otras implementaciones para matrices binarias que son más eficientes para almacenar estos valores, y para este caso el requerimiento por mini-lote podría reducirse a alrededor de 11 MB. Además de la memoria necesaria, el tiempo requerido para procesar los datos por ambos tipos de métodos (k-medias y MBkM) difieren en un orden de magnitud según las pruebas realizadas. De esta forma, los resultados mostrados en los tres conjuntos de datos y la capacidad de ajustar el tamaño de los mini-lotes de acuerdo a los recursos computacionales disponibles, convierten a los métodos incrementales en opciones interesantes para combinar un conjunto de particiones sobre grandes volúmenes de datos.

4.3. Conclusiones

El método de consenso de grupos de soluciones sobre datos biológicos con fuentes heterogéneas, presentado en la primer parte de este capítulo, representa un paso importante para simplificar el proceso de clustering y al mismo tiempo mejorar la calidad de las soluciones encontradas. Esto se ha mostrado en base a su utilidad para entender mejor la relación entre las diferentes accesiones de tomate y obtener conocimiento para direccionar futuros planes de mejora. El enfoque propuesto, en contraste con las técnicas tradicionales, no solamente obtiene soluciones que maximizan la información retenida de todas las fuentes de datos, sino que también evita realizar un complejo paso previo de preprocesamiento fuente por fuente. Al producir varias soluciones con diferentes configuraciones para luego combinarlas en una única partición consensuada, el método reduce significativa-

mente la necesidad de que el usuario conozca los algoritmos y parámetros más apropiados para analizar un conjunto de datos. Además, se ha desarrollado una metodología para evaluar la calidad de la solución final obtenida y entender mejor la relación entre las fuentes de datos. Esto se realiza midiendo la información compartida con cada fuente, asistiendo al usuario final en elegir mejores particiones. En este sentido, se ha mostrado también como el método podría sugerir la cantidad adecuada de clusters en los datos, lo cual representa un aspecto a explorar en trabajos futuros. El consenso de grupos provee una solución eficaz para mitigar los problemas asociados a la variedad o heterogeneidad presente en las diferentes fuentes de los grandes datos.

La arquitectura BigCE, propuesta en la segunda parte del presente capítulo, ha sido desarrollada incorporando el método de consenso de grupos y adaptándose a las otras dos características de los grandes datos: el gran volumen de información y la velocidad con que esta se produce. BigCE posee varias ventajas interesantes. La primera tiene que ver con una característica inherente de los ensambles: solo los clusterers necesitan acceder a los datos. Todos los demás componentes intermedios solo necesitan saber a qué clusters pertenecen los objetos. Esto permite, por un lado, simplificar el cómputo distribuido, y por otro paliar los problemas relacionados con la privacidad de los datos. Los problemas de heterogeneidad en las fuentes no necesitan ser abordados de la forma tradicional, con las inherentes dificultades relacionadas al preprocesamiento de los datos. La arquitectura propuesta permite utilizar algoritmos de clustering tradicionales bien conocidos, como k-medias o un método jerárquico, ya que tiene en cuenta la división del problema para que este sea factible de resolver. Las funciones de consenso intermedias para obtener los representantes también pueden ser las más comúnmente utilizadas, como las que se explicaron en la Sección 2.3.2. Sin embargo, la función de consenso superior, encargada de generar una solución combinada de todos los datos juntos, debe ser capaz de procesar la información de tal forma que no sea necesario cargar todo en la memoria, como requieren los métodos de consenso tradicionales. Este problema ha sido abordado con la propuesta de tres variantes basadas en una versión por lotes de k-medias, las cuales no solo son capaces de procesar todos los datos sino también de obtener soluciones de alta calidad.

CAPÍTULO 5

Generador de web-demos: investigaciones reproducibles y accesibles

Misterio eterno de nuestra vida. Natura humana, ¿cómo, si polvo y sombra eres, si eres frágil y vil, tan alto sientes?
— Giacomo Leopardi

Una cuestión transversal a todos los trabajos presentados en esta tesis ha sido la reproducibilidad de los resultados, es decir, la posibilidad de que el lector de un artículo pueda reproducir completamente los resultados que se presentan. Si bien una forma tradicional de lograr esto es publicar el código fuente de los métodos desarrollados junto con los datos utilizados, esta opción por sí misma presenta una serie de problemas que dificultan la rápida y efectiva reproducción de los resultados. Esto es especialmente cierto en la actualidad dentro de ciertas comunidades científicas, como la de bioinformática, donde los nuevos métodos y algoritmos poseen una complejidad cada vez mayor, y por ende no son sencillos de ejecutar sin complicados pasos previos de instalación y configuración de dependencias externas. En este capítulo se presentará una herramienta llamada *generador de web-demos*, desarrollada durante el transcurso de la tesis doctoral. Esta herramienta brinda una solución a estos problemas produciendo automáticamente una interfaz web al código fuente sin requerir conocimientos avanzados.

5.1. Reproducibilidad: estado del arte y propuesta

La reproducibilidad y corroboración de los resultados son de suma importancia en el proceso de generación del conocimiento científico (Vandewalle y otros, 2009). Esto es particularmente cierto en la comunidad de biología computacional

y bioinformática (Bissell, 2013; Editorial, 2014; Huang y Gottardo, 2013; Piwowar y otros, 2007), donde constantemente se publican nuevos métodos y algoritmos. Esta cuestión ha tomado una relevancia creciente en los últimos años. Por ejemplo, desde octubre de 2014, los artículos en las revistas de *Nature* deben hacer accesible el código fuente[1]. Sin embargo, el código fuente por si solo no garantiza que los resultados publicados sean realmente reproducibles (Hurley y otros, 2015). Esto es especialmente cierto hoy en día, donde las técnicas y algoritmos se vuelven cada vez más complejos y por lo tanto más difíciles de reproducir y validar (Hurley y otros, 2015; Smith y otros, 2014). Si bien la disponibilidad del código fuente es un paso importante, no es suficiente para lograr un adecuado grado de reproducibilidad. De hecho, en muchos casos ya desde el comienzo es necesario poseer conocimiento técnico avanzado para configurar el entorno correctamente y poder así ejecutar el código, una tarea que puede acarrear mucho tiempo y un esfuerzo considerable.

Frecuentemente, es necesario lidiar con una serie de inconvenientes antes de tener la posibilidad de efectivamente ejecutar el código fuente. Uno de estos pasos previos se relaciona a las dependencias que deben ser instaladas, desde el entorno de ejecución en sí (como Matlab o R), hasta restricciones relacionadas al sistema operativo, sin nombrar los posibles problemas relacionados a licencias o permisos. Es usual también que el código del método a validar dependa de versiones específicas de bibliotecas o herramientas externas, que muchas veces ya no son más compatibles con los nuevos sistemas operativos. Resolver este tipo de problemas puede en muchos casos impedir totalmente la posibilidad de ejecutar el software o, en el mejor de los casos, requerir una larga serie de tareas previas que muchas veces son de prueba y error, lo que provoca una importante pérdida de tiempo. Finalmente, los usuarios o lectores del trabajo publicado que quieran reproducir o utilizar los métodos deben volver a realizar la configuración del entorno, y esto en muchos casos puede tener una relación directa con la cantidad de citas recibidas por el artículo (Piwowar y otros, 2007).

Actualmente, existen varios enfoques propuestos para intentar paliar los problemas a la hora de reproducir los resultados de las investigaciones. Por ejemplo, una opción reciente sugiere aprovechar la creciente disponibilidad del ancho de banda para acceder a Internet, el espacio de disco cada vez más abundante y bara-

[1]Editorial de Nature. Code Share. Nature, 514(7524):536, 2014

to y el software de código abierto para lograr resultados realmente reproducibles utilizando tecnologías de virtualización (Hurley y otros, 2015). Concretamente, estos autores sugieren que cada publicación presente sus resultados computacionales acompañados de un *entorno virtual* (una máquina virtual) almacenado en la "nube". Este entorno necesita ser solo una implementación mínima y suficiente para que los métodos puedan ser reproducidos total o parcialmente. Otra opción propuesta consiste en integrar los datos y el código fuente más estrechamente con la publicación en sí (Jasny y otros, 2011; Peng, 2011; Sandve y otros, 2013). Por ejemplo, en el año 2011, miembros de la comunidad de ciencias de la computación propusieron la idea de los *artículos ejecutables*, donde el software y los datos utilizados están "dentro" del artículo. Esta original idea permite ejecutar el código mientras se lee el artículo, lo que provee una lectura sin interrupciones. Desafortunadamente, la propuesta no ha tenido gran adhesión fuera de la comunidad de informática, debido principalmente a que es necesario poseer conocimientos técnicos avanzados por parte del autor. Por ejemplo, es necesario configurar el lenguaje de programación y seleccionar varios parámetros complejos. Además, las opciones brindadas muchas veces no son suficientemente abarcativas para lograr satisfacer las necesidades de todos los autores. En este sentido, por ejemplo, no se ha considerado el acceso a herramientas externas como software para plegado de secuencias ARN (ácido ribonucleico), alineación y comparación de secuencias, o el acceso a bases de datos públicas con anotaciones genómicas. Estos inconvenientes representan una gran barrera para los autores no expertos en estas cuestiones técnicas.

Por las razones expuestas, es evidente la necesidad de una nueva herramienta que sea simple y fácil de usar y configurar, intuitiva para el autor, y al mismo tiempo rápida. Por ejemplo, una característica deseable sería que las entradas y salidas del algoritmo se detecten automáticamente. Luego, debería poder generarse una interfaz para realizar pruebas rápidas y sin las complicaciones del método a reproducir. Una interfaz web, por ejemplo, es fácilmente accesible por cualquier persona con acceso a Internet, en cualquier lugar del mundo y con cualquier dispositivo. Sin embargo, la generación de una interfaz web al código fuente no es para nada una tarea sencilla ni directa, y nuevamente caeríamos en una solución que requiera conocimientos avanzados por parte del autor.

En el resto de este capítulo se describirá una herramienta que soluciona los

problemas mencionados anteriormente. Ésta consiste en un generador de interfaces web para un determinado código fuente, con el objetivo de ser sencillo y rápido para el autor que quiere publicar su código, y fácil de utilizar y ejecutar para el lector del artículo que desea verificar o utilizar los métodos presentados.

5.2. Web-demos: resultados reproducibles y accesibles

Las páginas web son hoy en día accedidas por todos, en todo el mundo, de una manera muy sencilla. Sólo es necesario entrar a una determinada dirección web (URL) para poder visualizarlas. Éstas necesitan que solamente contemos con un navegador web, que en la actualidad puede ser instalado muy fácilmente, no solo en las computadoras de escritorio, sino también en los dispositivos móviles. Una interfaz web para el código fuente de los métodos presentados resulta en una solución ideal para abstraer a sus usuarios de todos los problemas inherentes a su configuración e instalación. En esta sección se describirá una herramienta llamada *generador de web-demos* que permite crear simples interfaces web al código fuente, que reciben el nombre de *web-demos* (Stegmayer y otros, 2016). Un web-demo hace de interfaz entre el código fuente (que puede contar con una alta complejidad) y el usuario que desea ejecutarlo, quien solo debe ingresar las entradas requeridas por el método y luego visualizar las salidas, que pueden ser numéricas, en formato de tablas, o complejos gráficos, y que pueden incluso brindar cierta capacidad de interacción.

El generador de web-demos simplemente necesita recibir el código fuente original en un archivo comprimido, luego detecta automáticamente las entradas y salidas del algoritmo, y finalmente genera una interfaz web sencilla para usar y probar el código. Este web-demo queda fijo con una URL determinada, lo que permite compartir muy fácilmente el acceso con otras personas. Esta sencilla interfaz web ofrece un acceso mucho más simplificado para utilizar el código fuente y ver sus resultados, lo que permite mejorar enormemente la accesibilidad a los métodos publicados, su diseminación y uso. El código fuente del generador de web-demos está disponible gratuitamente[2], lo que permite su instalación de

[2] `https://bitbucket.org/sinc-lab/webdemobuilder`

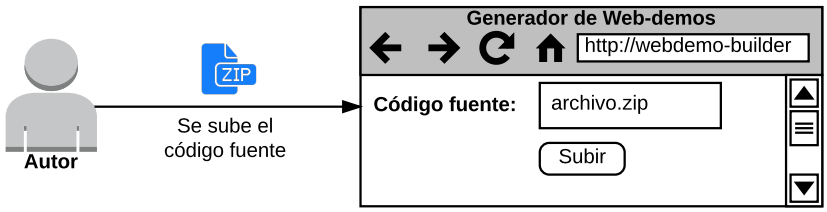
forma local o en un servidor, e incluso es posible utilizarlo desde la nube[3].

El generador de web-demos actualmente permite generar interfaces web a partir de código escrito en R, Python o Matlab, los cuales representan las plataformas de programación más comúnmente utilizadas en la comunidad de bioinformática. Con esta herramienta, el autor de un artículo puede hacer que su investigación sea no solo reproducible, sino también rápidamente accesible y fácil de probar con diferentes entradas posibles. Una vez que el autor genera su web-demo con el generador, los usuarios (como por ejemplo, un lector del artículo) pueden probar a través de él los métodos presentados, utilizando sus propias entradas (que pueden ser datos o parámetros del algoritmo), o utilizando archivos de ejemplo que el autor puede incluir en la interfaz web. Al momento de crear el web-demo, el autor puede utilizar cualquiera de los entornos que provee el generador, que cuentan con la configuración estándar de R, Python y Matlab.

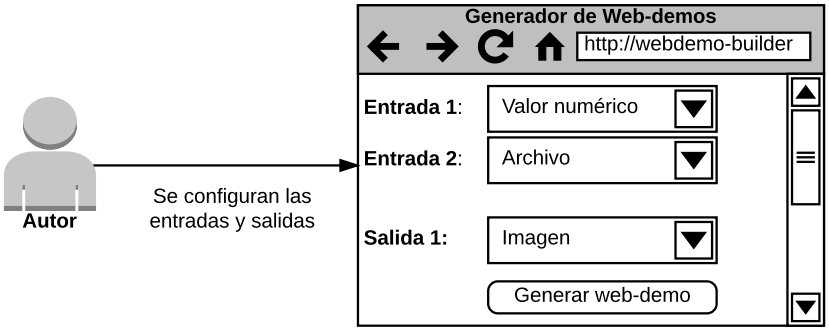
El generador toma el código fuente del algoritmo y produce un web-demo que actúa como una interfaz web al código fuente, al cual le envía las entradas y luego toma las salidas para presentarlas en el navegador del usuario. Este proceso se realiza en dos simples pasos, como puede verse en la Figura 5.1. El autor solo debe escribir una función principal en el lenguaje de programación elegido y empaquetarla junto con el resto del código en un archivo comprimido. Esta función simplemente debe recibir las entradas como parámetros, realizar el procesamiento necesario, y devolver luego los resultados como salidas. En el primer paso (Figura 5.1a), este archivo empaquetado es subido al generador de web-demos, el cual detecta automáticamente las entradas y salidas del método analizando la función principal que escribió el autor. El segundo paso (Figura 5.1b) consiste en especificar de qué tipo son las entradas y las salidas: un parámetro de entrada podría ser una simple cadena de texto, algún valor numérico o un archivo; el tipo de las salidas podría ser un número, un tabla o una imagen con, por ejemplo, un histograma o una gráfica mostrando la evolución del error de una función.

Estos simples pasos generan un web-demo, como el que puede verse en la Figura 5.2, al que el lector puede acceder por una URL fija. El web-demo resultante es una interfaz web con la que se puede interactuar: se completan los campos de entrada, se presiona el botón para ejecutar el código, y finalmente se obtienen los resultados en la misma página web. Cada usuario trabaja con

[3] `http://sinc.unl.edu.ar/papers/bib-2015-1-amazonaws`



(a) Primer paso: el autor sube su código fuente empaquetado en un archivo.



(b) Segundo paso: el generador detecta las entradas y salidas del método, y el autor configura luego el tipo de dato de cada una.

Figura 5.1. Pasos en la generación de un web-demo.

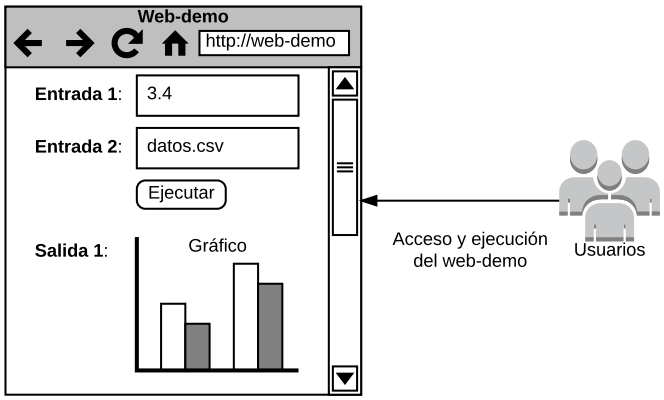


Figura 5.2. Acceso al web-demo creado a partir de los pasos mostrados en la Figura 5.1.

su propia instancia del web-demo, lo que permite aislar sus acciones del resto de los usuarios, sin conflicto alguno. Es importante mencionar que cada web-demo queda totalmente fijo no solo a una cierta versión del código fuente, sino también al entorno de ejecución seleccionado. Esta es una ventaja importante porque significa que el web-demo seguirá funcionando en el tiempo, produciendo los mismos resultados exhibidos en el artículo original, más allá de que haya cambios en el servidor donde se ejecuta (como por ejemplo, una actualización de algún paquete o incluso del sistema operativo).

Todos los pasos para crear un web-demo se realizan sin requerir ningún tipo de conocimiento técnico específico por parte del autor. Desde el punto de vista del lector, quien accede al web-demo, solo es necesario especificar las entradas del algoritmo (si es necesario, el autor puede proveer en el web-demo valores preestablecidos y archivos de ejemplo para una prueba rápida). Luego de eso, el algoritmo desarrollado se ejecuta en el servidor dentro de una moderna tecnología de virtualización llamada *contenedor de software* (Abdelbaky y otros, 2015; Preeth y otros, 2015). Una vez que el algoritmo finaliza, los resultados se presentan en la interfaz web. Es importante notar que el usuario del web-demo no necesita bajar ni saber instalar ningún tipo de biblioteca o lenguaje de programación, ni configurar ninguna otra herramienta usada por el autor del artículo; el usuario solo interactúa con la interfaz web. Esto representa una significativa reducción del esfuerzo necesario para tener un primer vistazo de cómo funcionan los métodos, y si uno desea todavía es posible analizar más detalladamente la implementación bajando el código fuente del algoritmo detrás del web-demo. Además, si el autor incluye datos confidenciales dentro del web-demo, un lector puede correr los algoritmos que los utilizan cambiando, por ejemplo, alguno de los parámetros y obteniendo luego los resultados, sin la necesidad de acceder a los datos directamente. De esta forma es posible verificar hallazgos realizados con datos propios y privados que no pueden ser compartidos, preservando la confidencialidad de los mismos.

5.3. Implementación

En esta sección se darán más detalles sobre la implementación del generador de web-demos y cómo funcionan los web-demos producidos, así como las propiedades que surgen de su diseño. La arquitectura completa puede observarse en la Figura 5.3, donde en color verde y con fuente de texto regular se indican los componentes tanto del generador como de los web-demos producidos. Ambos se muestran aparte porque efectivamente son dos aplicaciones distintas, donde una genera a la otra. Ambas son aplicaciones web que dependen de tecnologías externas (marcadas en color amarillo y con fuente en cursivas) como PHP, HTML y Javascript, y corren dentro de una plataforma de virtualización marcada con bordes punteados y fondo blanco.

El generador y las interfaces web de cada web-demo corren dentro de un

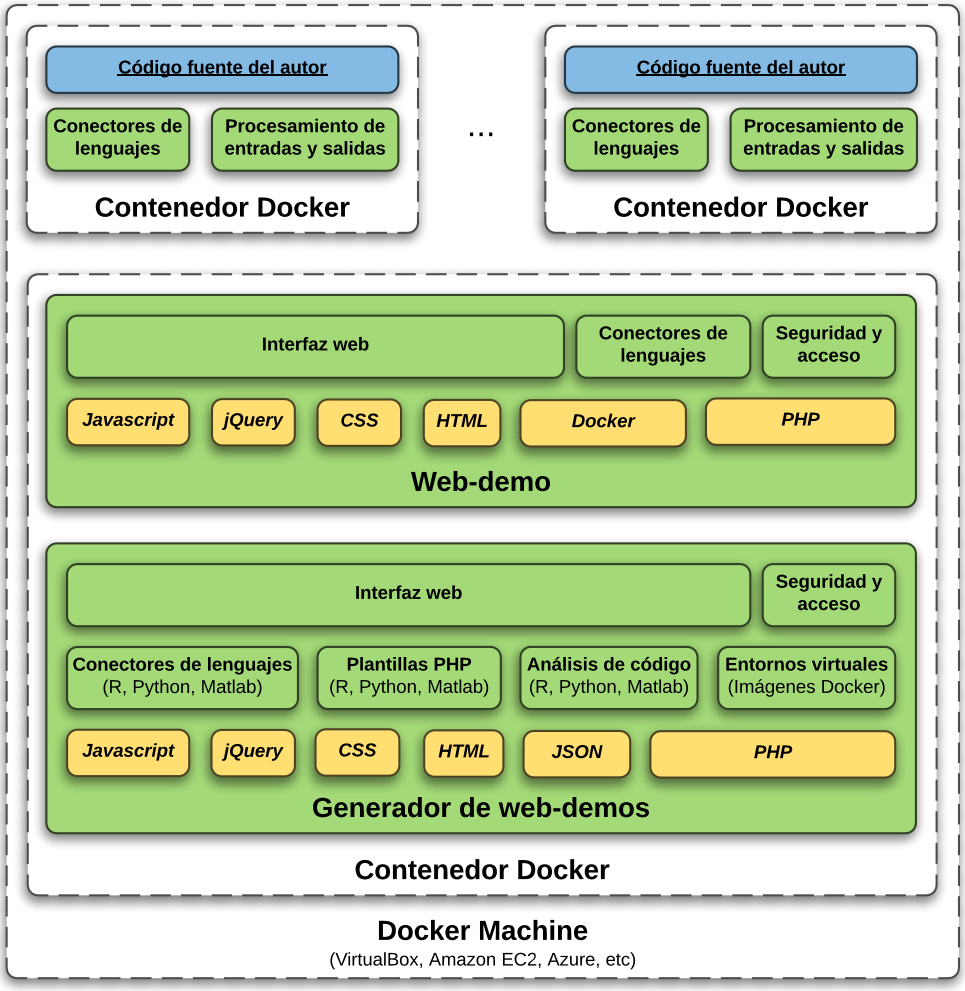


Figura 5.3. Arquitectura completa del generador y de los web-demos producidos.

mismo *contenedor Docker* o, más genéricamente, *contenedor de software*. En la Figura 5.3 se observa que ambas aplicaciones comparten el mismo contenedor ya que poseen las mismas dependencias web; sin embargo, esto no es un requisito y ambas podrían ejecutarse en contenedores distintos. Docker[4] es una moderna tecnología de virtualización, mucho más liviana que las tradicionales máquinas virtuales, y ampliamente utilizada en la actualidad para distribuir software. Esta popularidad se debe a que Docker permite empaquetar una aplicación junto con todas sus dependencias (incluso a nivel de sistema operativo) en una única unidad estandarizada y totalmente aislada, llamada justamente *contenedor*. Un contenedor es similar a una máquina virtual en el sentido de que el software que corre dentro se encuentra totalmente aislado. Sin embargo, en lugar de virtua-

[4] `http://www.docker.com/`

lizar el hardware como hace una máquina virtual tradicional, en realidad corre como un proceso más del sistema operativo exterior que lo alberga (el sistema anfitrión). Esto se traduce en una gran reducción de los recursos computacionales consumidos por una tecnología basada en contenedores de software. Docker utiliza funcionalidades internas del kernel del sistema operativo para aislar el proceso, haciéndole ver un determinado sistema de archivos, bibliotecas instaladas, programas disponibles, y hasta una capacidad de CPU y memoria que se especifican de antemano, y que pueden diferir del sistema anfitrión.

El generador y todas las interfaces web de los web-demos creados por este corren dentro de un contenedor basado en Linux, que tiene instalado PHP y todas las bibliotecas necesarias. Esta descripción de lo que tiene instalado dentro un contenedor se define por medio de una especificación en formato de texto que comúnmente se conoce como *Dockerfile*. Un Dockerfile contiene instrucciones de cómo crear una *imagen Docker*, y a partir de esta se instancian los contenedores. Este proceso de construcción de una imagen Docker y la ejecución de un contenedor basado en ella se muestra en la Figura 5.4. Allí puede verse como el Dockerfile especifica el sistema operativo en el que se basará la imagen, y luego los pasos de instalación necesarios para crear, en este caso, el entorno donde se ejecutará el generador y los web-demos. Una imagen Docker es el resultado de ejecutar todos esos pasos. Un contenedor es una instancia de dicha imagen, y al encontrarse completamente encapsulado en una unidad estandarizada, puede correrse en cualquier sistema operativo que soporte Docker, en la PC local o en un servidor remoto, con completa independencia del sistema anfitrión que lo alberga.

Todo el proceso de despliegue de un contenedor Docker para poner en funcionamiento el generador de web-demos puede realizarse en pocos pasos con una herramienta llamada *Docker Machine*[5]. Esto se muestra en la Figura 5.3 con el recuadro exterior en blanco y lineas punteadas. Docker Machine automatiza completamente el despliegue del generador de web-demos, tanto en una máquina virtual tradicional (por ejemplo, en una PC local) como en cualquier servicio de computación en la nube, como Amazon EC2, Azure o Digital Ocean. Esto hace que la instalación del generador sea relativamente sencilla y rápida.

Un aspecto importante de la arquitectura es el lugar donde se ubica el código del autor, que se muestra en color celeste y subrayado en la parte superior de la

[5] `https://docs.docker.com/machine/`

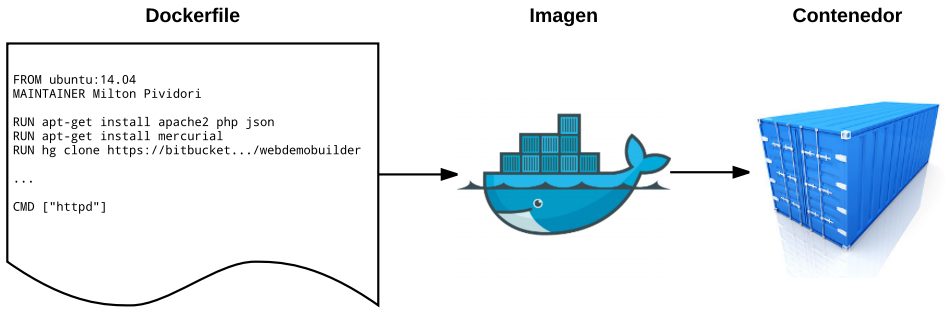


Figura 5.4. Ejemplo de creación de una imagen Docker y ejecución de un contenedor basado en ella. Una imagen se especifica por medio de un Dockerfile. Figura adaptada de la documentación de Docker[7].

Figura 5.3. Cuando el usuario del web-demo (por ejemplo un lector del artículo) carga las entradas y presiona el botón de ejecución, el web-demo ejecuta el código del autor en su propio contenedor. Este contenedor, que como puede verse es diferente al del generador, posee todas las dependencias necesarias para que el algoritmo se ejecute correctamente, y este es elegido o especificado por el autor. El web-demo posee las funcionalidades necesarias para ejecutar código Matlab, R o Python utilizando imágenes Docker ya ofrecidas por el generador (como se muestra en la figura de la arquitectura con el nombre de "entornos virtuales"), u otras imágenes más especializadas que pueden proveer los mismos autores. De esta forma, un autor, que por ejemplo ha escrito su algoritmo en Python utilizando varias dependencias externas, tiene la posibilidad de especificar su propio entorno virtual con gran flexibilidad y nivel de detalle, escribiendo su propio Dockerfile personalizado. Esta especificación puede subirse luego a cualquier servicio público de imágenes Docker, como por ejemplo Docker Hub[6], donde automáticamente se crea la imagen a partir del Dockerfile, y esta queda disponible para utilizarse con un identificador único. Al momento de crear el web-demo, el autor puede proporcionar este identificador único, que será utilizado luego por el web-demo para bajar la imagen Docker y crear el contenedor necesario para el código fuente. Cada usuario de un web-demo en particular ejecuta sus propias pruebas en un contenedor nuevo, totalmente aislado del resto de los usuarios del mismo web-demo, como muestra la Figura 5.3 en la parte superior.

Esta posibilidad ofrece muchas ventajas al autor. Siendo este el que mejor

[7] `https://docs.docker.com/engine/understanding-docker/`
[6] `https://hub.docker.com/`

conoce la forma de configurar un entorno para ejecutar su método, es el único que realiza en realidad este trabajo, y lo hace una sola vez al momento de elegir la imagen Docker o crear la suya propia. Luego, el web-demo utiliza esta imagen como "caja negra" para ofrecerle al lector la experiencia totalmente directa de probar los algoritmos, donde fácilmente puede cambiar las entradas y observar diferentes salidas, sin nunca tener que entrar en los detalles de configuración de las dependencias del código fuente.

En el desarrollo del generador de web-demos se tuvieron en cuenta dos aspectos importantes: la posibilidad de extender fácilmente la lista de lenguajes de programación soportados, y la seguridad y acceso tanto al generador como a cada web-demo. Con respecto al primero, se desarrollaron módulos especiales llamados *conectores*, que son los responsables de definir el formato de las entradas y salidas de una función para cada lenguaje, además de los comandos necesarios para ejecutar un script. Como puede verse en la Figura 5.3, los conectores se encuentran presentes tanto en el generador como en cada web-demo. Esto es así porque el generador, al momento de crear un web-demo, necesita analizar sintácticamente la función principal escrita por el autor, y determinar así las entradas y salidas posibles; y el web-demo, responsable de la ejecución del código, debe conocer el comando correcto de cada lenguaje para ejecutar un script. Con respecto a los aspectos relacionados a la seguridad, el acceso al generador puede ser protegido mediante la especificación de un usuario y contraseña. Si se utiliza esta opción, solo las personas autorizadas podrán accederlo para generar web-demos. Incluso cada web-demo en particular puede tener un usuario y contraseña, lo que en este caso permite restringir el uso de un web-demo determinado si es necesario.

El generador sigue siendo activamente desarrollado en la actualidad, al cual se están incorporando un conjunto de nuevas características. Una de las más interesantes extiende el concepto de accesibilidad aún más, ofreciendo la posibilidad de utilizar las funciones expuestas por el código desde cualquier otro software. Esto significa que, además de una interfaz web fácilmente utilizable por un usuario no experto, también se podrá contar con una interfaz REST (*Representational State Transfer*, en inglés) (Li y otros, 2016) que permite que las características ofrecidas por el código fuente del autor también estén disponibles a cualquier otro software mediante, por ejemplo, métodos HTTP estándar. Esto representa

un paso que va más allá de la reproducibilidad, y extiende la idea de accesibilidad ya presentada, transformando los métodos del autor en piezas de software totalmente reutilizables por cualquier otro investigador a través de otros métodos y lenguajes. Por ejemplo, una vez verificado el funcionamiento del web-demo, un lector podría escribir las líneas de código necesarias para que sus métodos utilicen la funcionalidad exhibida en el web-demo. En este caso, el código del lector actúa de cliente, y solo debe enviar las entradas y recibir las salidas. Si, por ejemplo, el web-demo ofrece un novedoso método de clustering, un investigador externo podría conectar sus propios métodos, enviando las entradas requeridas y recibiendo las salidas producidas por el método de clustering. Todo esto sin tener que configurar absolutamente nada.

5.4. Conclusiones

En este capítulo se ha abordado y cumplido el último objetivo específico de la tesis, donde se propuso una nueva forma de mejorar la reproducibilidad de los resultados. Se ha descripto la motivación, propuesta e implementación del generador de web-demos. El mismo representa una herramienta sumamente útil para mejorar la reproducibilidad de los resultados de las investigaciones así como también su accesibilidad. Las funcionalidades de este sistema permiten crear una interfaz web al código fuente del autor de un artículo, lo que permite una mejor experiencia para los usuarios del método en cuestión, evitando complejos pasos previos de configuración y la consiguiente pérdida de tiempo.

Este sistema actualmente es muy utilizado en el Instituto de Investigación en Señales, Sistemas e Inteligencia Computacional (sinc(i)), como puede verse en su sitio web institucional[8], donde se puede acceder a varios web-demos ya generados por los investigadores y becarios del instituto.

[8] `http://fich.unl.edu.ar/sinc/web-demo/`

CAPÍTULO 6

Conclusiones finales

Yo deseo para mí y les deseo a ustedes que nunca estemos tranquilos, que no nos quedemos tranquilos jamás.

— Luigi Giussani

En esta tesis se han estudiado, extendido y aplicado las ideas sobre ensambles de agrupamientos, un conjunto de métodos recientes en el área de aprendizaje de máquina que han adquirido gran relevancia debido a las soluciones consensuadas de mayor calidad que ofrecen. En las investigaciones actuales no se ha logrado identificar cómo impacta la diversidad de los ensambles en la partición de consenso e incluso se ha llegado a conclusiones opuestas. La principal contribución de esta tesis ha sido estudiar en profundidad esta cuestión central de los ensambles de agrupamientos y proponer además un método novedoso y efectivo para controlar la diversidad. El método de control de diversidad analiza primero la estructura del ensamble y, a partir de esta información, realiza pequeños cambios en él, y produce así ensambles con diversidad controlada que inducen cambios suaves en la función de consenso. Mediante una medida de suavidad basada en la autocorrelación, se ha demostrado que nuestro enfoque supera ampliamente a los métodos actuales. Finalmente, se ha mostrado que nuestra propuesta representa un paso importante para direccionar adecuadamente futuros estudios sobre el impacto de la diversidad en el desempeño del consenso.

En el desarrollo de la tesis también se han aplicado exitosamente los ensambles de agrupamientos junto con las ideas del método de control de la diversidad a un problema de bioinformática con datos reales de cultivares de tomates. Estos datos biológicos presentan una característica común en los hoy llamados grandes datos: la heterogeneidad de las fuentes de información, que representan un verdadero desafío para los algoritmos de aprendizaje de máquina. La aplicación

satisfactoria de los ensambles de agrupamientos en este caso real ha motivado el desarrollo de una arquitectura inicial para problemas de clustering en los grandes datos, que contempla además los enormes volúmenes de información que estos presentan.

Finalmente, se ha abordado una cuestión de especial importancia en la investigación científica, particularmente en la comunidad de bioinformática: la reproducibilidad de los resultados. Observando la gran necesidad actual de mejores metodologías y herramientas que faciliten la reproducción de los hallazgos científicos en el área, se ha propuesto una novedosa herramienta para generar interfaces web al código fuente desarrollado por los investigadores. Estas interfaces, llamadas web-demos, facilitan significativamente la tarea de revisión y prueba de los nuevos métodos presentados en publicaciones científicas.

A continuación se detallan las conclusiones específicas de cada aporte en esta tesis:

1. **Control de la diversidad en los ensambles de agrupamientos**

   - Se ha determinado un problema central en los enfoques actuales para analizar cómo la diversidad de los ensambles afecta el desempeño.
   - Se han identificado las razones que causan este problema, es decir, por qué los métodos actuales para estudiar la diversidad han llegado muchas veces a conclusiones opuestas.
   - Las razones identificadas previamente han permitido diseñar un novedoso método de control de la diversidad, que recaba información estructural del ensamble y logra producir pequeños cambios en él, generando así un conjunto de ensambles con diversidad controlada.
   - Debido a que el enfoque de control de la diversidad representa una idea novedosa en el área, se ha diseñado una medida para evaluar el grado de control que un método logra sobre la diversidad. Esta medida, basada en el coeficiente de autocorrelación, ha permitido evaluar objetivamente el grado de suavidad que el método de control es capaz de producir no solo en la evolución de las medidas sobre ensambles, sino también en las medidas de clustering sobre las particiones de consenso derivadas.

- Las pruebas realizadas en seis conjuntos de datos ampliamente conocidos han demostrado que el método no solo representa un enfoque efectivo para el control de la diversidad sino que además, al inducir cambios suaves sobre el consenso, ha sentado las bases para futuros estudios de análisis de la diversidad.

2. **Ensambles para grandes datos en bioinformática**

   - Se han aplicado satisfactoriamente los métodos de ensambles y las ideas del control de diversidad sobre un conjunto de datos biológicos reales de cultivares de tomates provenientes de fuentes de información heterogéneas.
   - En los resultados presentados, los ensambles de agrupamientos han demostrado ser sumamente útiles y efectivos en este escenario con fuentes de datos de gran variedad, donde los enfoques tradicionales requieren complejos pasos previos de preprocesamiento, lo que dificulta significativamente el trabajo del usuario final y hace que los resultados dependan fuertemente de su pericia.
   - Mediante la utilización de las ideas propuestas en el método de control de diversidad, particularmente de las particiones representantes, se ha diseñado un método para medir la información compartida entre la partición de consenso final y cada fuente de datos, lo que ha permitido arribar a importantes conclusiones. En primer lugar, la aplicación de ensambles sobre estos datos ha superado ampliamente a los métodos tradicionales. En segundo lugar, los gráficos de información compartida con cada fuente han mostrado ser sumamente útiles para conocer mejor la estructura de los datos y sugieren incluso la cantidad de clusters presentes en ellos.
   - Los métodos empleados en esta aplicación han sido implementados en una aplicación web que permite utilizarlos fácilmente con cualquier tipo de datos que procedan de diferentes fuentes, donde también es posible configurar los parámetros de los métodos de clustering y de consenso usados.
   - Luego de esta satisfactoria aplicación a datos reales de fuentes de infor-

mación heterogéneas y considerando que los grandes datos comparten esta característica, se ha propuesto una arquitectura inicial que contemple también sus otras dos propiedades: el gran volumen de información y la alta velocidad con la que esta se genera. Esta arquitectura propuesta, basada en los ensambles de agrupamientos y el control de diversidad, describe varios componentes de procesamiento necesarios en estos escenarios y sirve como marco general de futuros métodos para analizar grandes datos, lo que ayuda a vislumbrar los principales desafíos.

- Utilizando esta arquitectura general, se ha propuesto en esta tesis una función de consenso que permita combinar, de una manera escalable e incremental, la gran cantidad de información generada por los demás componentes de la arquitectura. Los resultados iniciales obtenidos sugieren que la propuesta puede ser de gran utilidad y representa así un método interesante para seguir estudiando, extendiendo y mejorando en el futuro.

3. **Generador de web-demos: investigaciones reproducibles y accesibles**

   - Considerando la gran importancia que tiene la reproducibilidad de los resultados en la comunidad científica, particularmente en la de bioinformática, se han identificado los principales problemas presentes a la hora de revisar y verificar los resultados publicados en un artículo. Esto ha servido como base para diseñar e implementar una solución que permite, de manera muy sencilla, generar interfaces web (web-demos) al código fuente de los métodos desarrollados por los investigadores. Esto ha permitido mejorar significativamente no solo la reproducibilidad, sino también la accesibilidad, diseminación y uso de las nuevas técnicas desarrolladas por sus autores.
   - Los web-demos representan una valiosa herramienta no solo para los autores de los métodos computacionales propuestos, sino también para los revisores de los artículos, y sus potenciales lectores y usuarios. El valor reside no solo en la gran reducción del esfuerzo necesario para la configuración de todas las dependencias que pueda tener el código

fuente, sino también de la importante reducción del tiempo requerido y los posibles errores involucrados en este proceso.

- El generador de web-demos implementado, basado en contenedores de software (una moderna tecnología de virtualización), provee una arquitectura especialmente diseñada para ajustarse a las necesidades más exigentes de los investigadores, que necesitan tener gran flexibilidad para incluir todas las dependencias y configuraciones necesarias para lograr una completa y eficaz reproducibilidad de sus resultados.

APÉNDICE A

Generador de web-demos: un ejemplo guiado

La página de documentación del generador[1] contiene código fuente y datos de ejemplos sencillos para verificar cómo funciona todo en la práctica. Estos ejemplos consisten en tres piezas de código diferentes, que realizan distintas tareas aunque relacionadas. En esta sección se mostrará el paso a paso de cómo generar el web-demo para uno de esos ejemplos, el cual se encuentra implementado en Matlab y realiza el plegado de secuencias de ARN. En la página indicada se puede bajar el código en formato zip, así como un archivo de ejemplo que es la entrada del algoritmo.

Para crear este primer web-demo es necesario ingresar a la instancia de prueba del generador de web-demos[2], y en la sección llamada *"General description"*, ingresar un nombre para el web-demo (por ejemplo, "Matlab miRNA folding"), una descripción y una versión, como puede verse en la Figura A.1. Luego, en la sección *"Source code"* es necesario elegir primero el lenguaje de programación (en este caso Matlab). Lo siguiente, que es opcional, consiste en elegir un entorno virtual. Si esto es necesario (por ejemplo, si el autor creó su propio entorno y lo ha publicado en Docker Hub), se puede ingresar el identificador de la imagen Docker en el campo *"Virtual environment"*. Si no, se puede dejar la opción estándar ofrecida por el generador. En este caso, el identificador de la imagen Docker para Matlab es `sinclab/webdemo-base-matlab`. El siguiente paso es elegir el archivo de código fuente, donde se incluya también la función principal que el autor debe escribir. En el Listado A.1 se puede ver la función escrita para este ejemplo, llamada `main`, la cual simplemente posee una entrada (`filename`) y devuelve una

[1] `https://bitbucket.org/sinc-lab/webdemobuilder/wiki/Home`

[2] `http://sinc.unl.edu.ar/papers/bib-2015-1-amazonaws/`

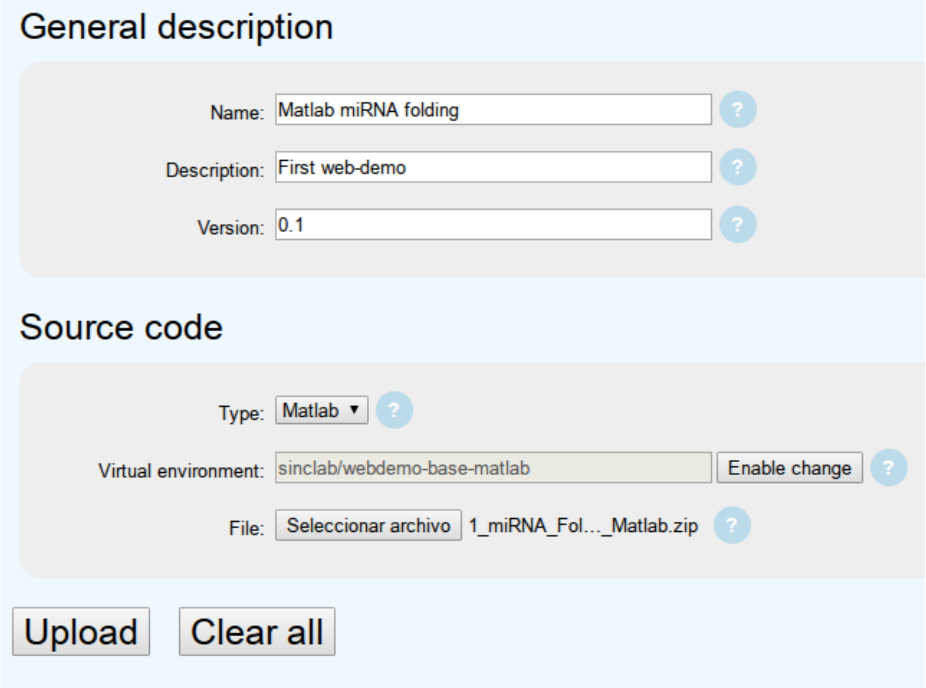


Figura A.1. Pantalla principal del generador de web-demos, con un ejemplo de código en Matlab.

```
function [output] = main(filename)
  [heads, seqs] = fastaread(filename);
  n = length(seqs);
  foldedseqs = {};
  for i = 1:n
    foldedseqs{i} = rnafold(seqs{i});
  end;

  output = ['RNAfolding_results.fa'];
  fid = fopen(output,'wt');
  for j = 1:n
    fprintf(fid, '> %s\n', cell2mat(heads(j)));
    fprintf(fid, '%s\n', cell2mat(seqs(j)));
    fprintf(fid, '%s\n', cell2mat(foldedseqs(j)));
  end;
  fclose(fid);
end
```

Listado A.1. Archivo `main.m` con el código en Matlab de la función principal escrita por el autor.

salida (`output`). Finalmente, el autor debe presionar el botón *"Upload"* y esperar que el generador procese el código.

El siguiente paso consiste en configurar las entradas y salidas, que es la próxima pantalla que el generador muestra una vez que ha procesado el código

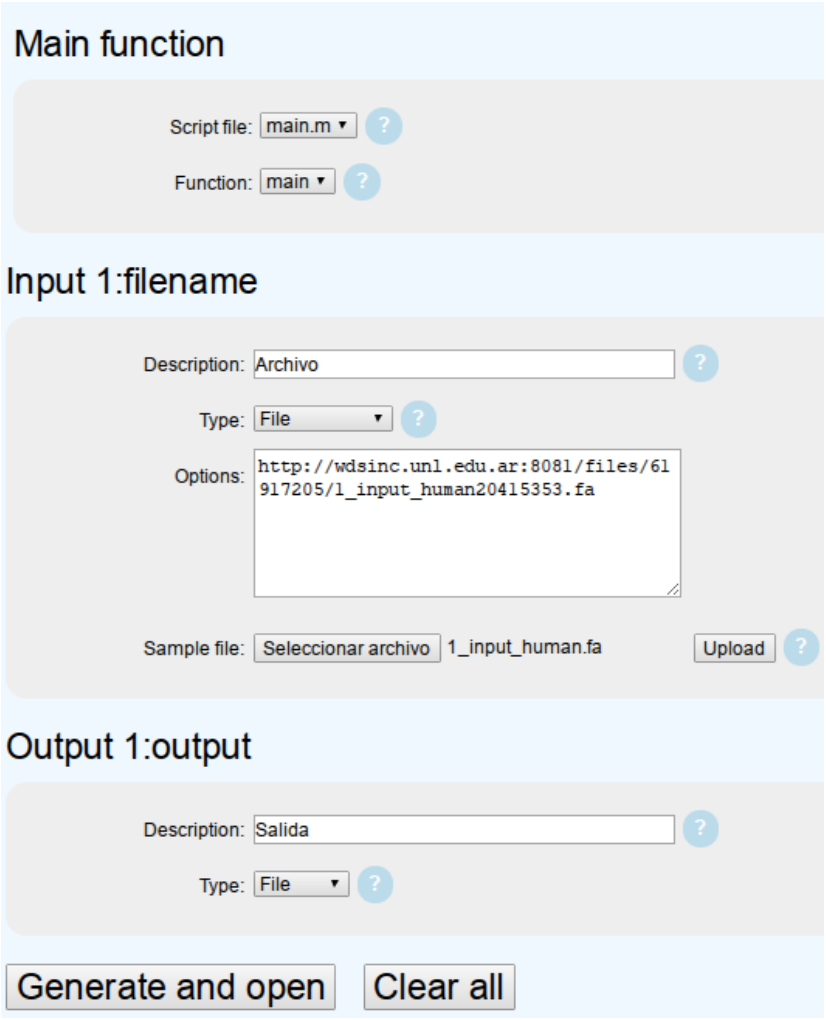


Figura A.2. Ejemplo de configuración de entradas y salidas de un web-demo.

fuente y que puede verse en la Figura A.2. Lo primero que debe seleccionarse es el archivo y la función principal (*"Main function"*), que en el ejemplo es `main.m` y `main`, respectivamente. Esta será la función de la cual el generador leerá las entradas y salidas, y la que el web-demo resultante utilizará para ejecutar el código fuente. Como se mostró antes en el Listado A.1, esta función principal tiene una sola entrada y una sola salida, como ha detectado el generador y se muestra en la Figura A.2. Luego se pasa a configurar el tipo de dato para cada entrada y salida; la entrada número 1 (*"Input 1"*) ha sido configurada para recibir un archivo (*"Type: File"*), y también se ha proporcionado uno de ejemplo para el usuario del web-demo (*"Sample file: 1_input_human.fa"*). La única salida de este ejemplo es otro archivo, y en la figura se puede ver que se ha colocado una descripción y se ha escogido el tipo *"File"*.

Figura A.3. Ejemplo de un web-demo funcionando, con un archivo como entrada y otro como salida.

Lo siguiente es presionar el botón *"Generate and open"*, lo que abrirá una nueva ventana con el web-demo resultante. Una vez que se carga el archivo de entrada de ejemplo (`1_input_human.fa`, en este caso un archivo fasta con secuencias de ARN), se presiona el botón *"Upload"* y luego *"Submit"*, el web-demo ejecuta el código fuente dentro de un contenedor Docker con Matlab. La salida, una vez terminado el procesamiento, puede verse en la Figura A.3, donde es posible bajar el archivo generado `RNAFolding_results.fa`.

Bibliografía

Abdelbaky, M.; Diaz-Montes, J.; Parashar, M.; Unuvar, M. y Steinder, M. (2015). «Docker Containers across Multiple Clouds and Data Centers». En: *2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC)*pp. 368–371.

Arthur, David y Vassilvitskii, Sergei (2007). «K-means++: The Advantages of Careful Seeding». En: *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*SODA '07, pp. 1027–1035. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

Asprelli, P.D.; Occhiuto, P.N.; Makuch, M.A.; Lorello, I.M.; Togno, L.S.; García Lampasona, S.C. y Peralta, I. (2011). «Recolección de germoplasma criollo de especies cultivadas y su distribución en regiones andinas de Argentina». *Horticultura Argentina*, **30(71)**, pp. 30–45.

Azimi, Javad y Fern, Xiaoli (2009). «Adaptive Cluster Ensemble Selection». En: *Proceedings of the 21st International Jont Conference on Artifical Intelligence*IJCAI'09, pp. 992–997. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Basheer, I.A y Hajmeer, M (2000). «Artificial neural networks: fundamentals, computing, design, and application». *Journal of Microbiological Methods*, **43(1)**, pp. 3–31. Neural Computting in Micrbiology.

Bishop, Christopher M. (2006). *Pattern Recognition and Machine Learning*. Springer, New York.

Bissell, Mina (2013). «Reproducibility: The risks of the replication drive». *Nature*, **503(7476)**, pp. 333–334.

Bottou, Léon y Bengio, Yoshua (1995). «Convergence Properties of the K-

Means Algorithms». En: *Advances in Neural Information Processing Systems 7*pp. 585–592.

Box, G.E.P.; Jenkins, G.M. y Reinsel, G.C. (2013). *Time Series Analysis: Forecasting and Control*. Wiley Series in Probability and Statistics. Wiley.

Bradley, Paul S. y Fayyad, Usama M. (1998). «Refining Initial Points for K-Means Clustering». En: *Proceedings of the Fifteenth International Conference on Machine Learning*ICML '98, pp. 91–99.

Capó, Marco; Pérez, Aritz y Lozano, Jose A. (2016). «An efficient approximation to the K-means clustering for massive data». *Knowledge-Based Systems*. En prensa.

Cover, Thomas M. y Thomas, Joy A. (2006). *Elements of Information Theory, 2nd Edition*.

Davidson, I. y Satyanarayana, A. (2003). «Speeding up k-means clustering by bootstrap averaging». En: *IEEE data mining workshop on clustering large data sets*pp. 16–25.

Davies, David L. y Bouldin, Donald W. (1979). «A Cluster Separation Measure». *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **PAMI-1(2)**, pp. 224–227.

Dunn, J. C. (1974). «Well separated clusters and optimal fuzzy-partitions». *Journal of Cybernetics*, **4(1)**, pp. 95–104.

Editorial, Nature (2014). «Journals unite for reproducibility». *Nature*, **515(7525)**, p. 7.

Eisen, Michael B.; Spellman, Paul T.; Brown, Patrick O. y Botstein, David (1998). «Cluster analysis and display of genome-wide expression patterns». *Proceedings of the National Academy of Sciences*, **95(25)**, pp. 14863–14868.

Everitt, Briand; Landau, Sabine; Leese, Morven y Stahl, Daniel (2011). *Cluster Analysis, 5th Edition*. Wiley.

Fern, Xiaoli Z. y Lin, Wei (2008). «Cluster Ensemble Selection». *Stat. Anal. Data Min.*, **1(3)**, pp. 128–141.

Fern, Xiaoli Zhang y Brodley, Carla E. (2003). «Random projection for high dimensional data clustering: A cluster ensemble approach». En: *ICML-2003*pp. 186–193.

Fern, Xiaoli Zhang y Brodley, Carla E. (2004). «Solving Cluster Ensemble Problems by Bipartite Graph Partitioning». En: *Proceedings of the Twenty-first International Conference on Machine Learning*ICML '04, pp. 36–. ACM, New York, NY, USA.

Fiori, Alessandro; Mignone, Andrea y Rospo, Giuseppe (2016). «DeCoClu: Density consensus clustering approach for public transport data». *Information Sciences*, **328**, pp. 378–388.

Fischer, B. y Buhmann, J.M. (2003). «Bagging for path-based clustering». *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **25(11)**, pp. 1411–1415.

Fred, A. L N y Jain, A.K. (2002). «Data clustering using evidence accumulation». En: *Pattern Recognition, 2002. Proceedings. 16th International Conference on*volumen 4, pp. 276–2804.

Fred, AL.N. y Jain, AK. (2005). «Combining multiple clusterings using evidence accumulation». *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **27(6)**, pp. 835–850.

Gan, Guojun; Ma, Chaoqun y Wu, Jianhong (2007). *Data Clustering: Theory, Algorithms, and Applications*. Philadelphia.

Gordon, A. D. (1998). *Data Science, Classification, and Related Methods*. capítulo Cluster Validation, pp. 22–39. Springer Japan, Tokyo.

Gordon, A. D. (1999). *Classification*. Monographs on statistics and applied probability. Chapman Hall, Boca Raton (Fla.), London, New York.

Gou, X.; Wang, Z.; Li, N.; Qiu, F.; Xu, Z.; Yan, D.; Yang, S.; Jia; Konga, X.; Wei, Z.; Lu, S.; Lian, L.; Wu, C.; Wang, X.; Li, G.; Teng, M.; Jiang,

Q.; Zhao, X.; Yang, J.; Liu, B.; Wei, D.; Li, H.; Yang, J.; Yan, Y.; Zhao, G.; Dong, X.; Li, M.; Deng, W.; Leng, J.; Wei, C.; Wang, C.; Mao, H.; Zhang, H.; Ding, G. y Li, Y. (2014). «Whole-genome sequencing of six dog breeds from continuous altitudes reveals adaptation to high-altitude hypoxia». *Genome Research*, **24(8)**, pp. 1308–1315.

Gower, J. C. (1967). «A comparison of some methods of cluster analysis». *Biometrics*, **23(4)**, pp. 623–637.

Gullo, Francesco; Tagarelli, Andrea y Greco, Sergio (2009). «Diversity-based Weighting Schemes for Clustering Ensembles». En: *Proceedings of the 2009 SIAM International Conference on Data Mining*pp. 437–448.

Hadjitodorov, Stefan T.; Kuncheva, Ludmila I. y Todorova, Ludmila P. (2006). «Moderate diversity for better cluster ensembles». *Information Fusion*, **7(3)**, pp. 264–275.

Halkidi, Maria; Batistakis, Yannis y Vazirgiannis, Michalis (2001). «On Clustering Validation Techniques». *Journal of Intelligent Information Systems*, **17(2-3)**, pp. 107–145.

Han, Jiawei; Kamber, Micheline y Professor, Jian Pei (2011). *Data Mining: Concepts and Techniques*.

Hore, Prodip; Hall, Lawrence O. y Goldgof, Dmitry B. (2009). «A scalable framework for cluster ensembles». *Pattern Recognition*, **42(5)**, pp. 676–688.

Howe, D.; Costanzo, M.; Fey, P.; Gojobori, T.; Hannick, L.; Hide, W.; Hill, D.P.; Kania, R.; Schaeffer, M.; St Pierre, S.; Twigger, S.; White, O. y Yon Rhee, S. (2008). «Big data: The future of biocuration». *Nature*, **455(7209)**, pp. 47–50.

Hu, Jie; Li, Tianrui; Wang, Hongjun y Fujita, Hamido (2016). «Hierarchical cluster ensemble model based on knowledge granulation». *Knowledge-Based Systems*, **91**, pp. 179–188.

Huang, Dong; Lai, Jianhuang y Wang, Chang-Dong (2016). «Ensemble clustering using factor graph». *Pattern Recognition*, **50**, pp. 131–142.

Huang, Xiaodi; Zheng, Xiaodong; Yuan, Wei; Wang, Fei y Zhu, Shanfeng (2011). «Enhanced clustering of biomedical documents using ensemble non-negative matrix factorization». *Information Sciences*, **181(11)**, pp. 2293–2302.

Huang, Yunda y Gottardo, Raphael (2013). «Comparability and reproducibility of biomedical data». *Briefings in Bioinformatics*, **14(4)**, pp. 391–401.

Hubert, Lawrence y Arabie, Phipps (1985). «Comparing partitions». *Journal of Classification*, **2(1)**, pp. 193–218.

Hurley, Daniel G.; Budden, David M. y Crampin, Edmund J. (2015). «Virtual Reference Environments: a simple way to make research reproducible». *Briefings in Bioinformatics*, **16(5)**, pp. 901–903.

Iam-On, N.; Boongeon, T.; Garrett, S. y Price, C. (2012). «A Link-Based Cluster Ensemble Approach for Categorical Data Clustering». *Knowledge and Data Engineering, IEEE Transactions on*, **24(3)**, pp. 413–425.

Iam-On, N.; Boongoen, T.; Garrett, S. y Price, C. (2011). «A Link-Based Approach to the Cluster Ensemble Problem». *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **33(12)**, pp. 2396–2409.

Jagadish, H.V. (2015). «Big Data and Science: Myths and Reality». *Big Data Research*, **2(2)**, pp. 49–52.

Jain, Anil K. (2010). «Data clustering: 50 years beyond K-means». *Pattern Recogn. Lett.*, **31(8)**, pp. 651–666.

Jain, Anil K. y Dubes, Richard C. (1988). *Algorithms for Clustering Data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Jasny, Barbara R.; Chin, Gilbert; Chong, Lisa y Vignieri, Sacha (2011). «Again, and Again, and Again ...». *Science*, **334(6060)**, pp. 1225–1225.

Jing, Liping; Tian, Kuang y Huang, Joshua Z. (2015). «Stratified feature sampling method for ensemble clustering of high dimensional data». *Pattern Recognition*, **48(11)**, pp. 3688–3702.

Johnson, Stephen C. (1967). «Hierarchical clustering schemes». *Psychometrika*, **32(3)**, pp. 241–254.

Jolliffe, I.T. (2002). *Principal Component Analysis*. Springer-Verlag New York.

Karypis, G.; Aggarwal, R.; Kumar, V. y Shekhar, S. (1999). «Multilevel hypergraph partitioning: applications in VLSI domain». *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, **7(1)**, pp. 69–79.

Karypis, George y Kumar, Vipin (1998). «A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs». *SIAM Journal on Scientific Computing*, **20(1)**, pp. 359–392.

Kaufman, Leonard y Rousseeuw, Peter (2005). *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley.

Kleinberg, Jon (2002). «An Impossibility Theorem for Clustering». En: *Neural Information Processing Systems*pp. 446–453. MIT Press.

Kogan, Jacob; Nicholas, Charles y Teboulle, Marc (2006). *Grouping Multidimensional Data: Recent Advances in Clustering*. Springer-Verlag Berlin Heidelberg.

Kohonen, Teuvo (1982). «Self-organized formation of topologically correct feature maps». *Biological Cybernetics*, **43(1)**, pp. 59–69.

Kuncheva, L. I. y Vetrov, D. P. (2006). «Evaluation of Stability of k-Means Cluster Ensembles with Respect to Random Initialization». *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **28(11)**, pp. 1798–1808.

Kuncheva, L.I. y Hadjitodorov, S.T. (2004). «Using diversity in cluster ensembles». En: *Systems, Man and Cybernetics, 2004 IEEE International Conference on*volumen 2, pp. 1214–12192.

Labrinidis, Alexandros y Jagadish, H. V. (2012). «Challenges and Opportunities with Big Data». *Proc. VLDB Endow.*, **5(12)**, pp. 2032–2033.

Lance, G. N. y Williams, W. T. (1967). «A General Theory of Classificatory Sorting Strategies: 1. Hierarchical Systems». *The Computer Journal*, **9(4)**, pp. 373–380.

Law, M.H.C.; Topchy, A.P. y Jain, A.K. (2004). «Multiobjective data clustering». En: *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*volumen 2, pp. II–424–II–430 Vol.2.

Li, L.; Chou, W.; Zhou, W. y Luo, M. (2016). «Design Patterns and Extensibility of REST API for Networking Applications». *IEEE Transactions on Network and Service Management*, **13(1)**, pp. 154–167.

Li, Yixue y Chen, Luonan (2014). «Big Biological Data: Challenges and Opportunities». *Genomics, Proteomics & Bioinformatics*, **12(5)**, pp. 187–189.

Lichman, M. (2013). «UCI Machine Learning Repository». `http://archive.ics.uci.edu/ml`

Lupski, J.R.; Reid, J.G.; Gonzaga-Jauregui, C.; Deiros, D.R.; Chen, D.C.Y.; Nazareth, L.; Bainbridge, M.; Dinh, H.; Jing, C.; Wheeler, D.A.; McGuire, A.L.; Zhang, F.; Stankiewicz, P.; Halperin, J.J.; Yang, C.; Gehman, C.; Guo, D.; Irikat, R.K.; Tom, W.; Fantin, N.J.; Muzny, D.M. y Gibbs, R.A. (2010). «Whole-genome sequencing in a patient with Charcot-Marie-Tooth neuropathy». *New England Journal of Medicine*, **362(13)**, pp. 1181–1191.

McQuitty, Louis L. (1966). «Similarity Analysis by Reciprocal Pairs for Discrete and Continuous Data». *Educational and Psychological Measurement*, **26(4)**, pp. 825–831.

Meilă, Marina (2005). «Comparing clusterings: an axiomatic view». En: *In ICML '05: Proceedings of the 22nd international conference on machine learning*pp. 577–584.

Mervis, Jeffrey (2012). «Agencies Rally to Tackle Big Data». *Science*, **336(6077)**, pp. 22–22.

Minaei-Bidgoli, Behrouz; Topchy, Alexander P. y Punch, William F. (2004). «A Comparison of Resampling Methods for Clustering Ensembles». En: *IC-AI*pp. 939–945.

Mohabeer, Hemant; Soyjaudah, K.M. Sunjiv y Pavaday, Narainsamy (2011). «Enhancing The Performance Of Neural Network Classifiers Using Selected Biometric Features». En: *SENSORCOMM 2011, The Fifth International Conference on Sensor Technologies and Applications*pp. 140–144.

Murphy, Kevin P. (2012). *Machine Learning: A Probabilistic Perspective*. The MIT Press.

Naldi, M.C.; Carvalho, A.C.P.L.F. y Campello, R.J.G.B. (2013). «Cluster ensemble selection based on relative validity indexes». *Data Mining and Knowledge Discovery*, **27(2)**, pp. 259–289.

Nature (2008). «Community cleverness required». *Nature*, **455(7209)**, pp. 1–1.

Ng, S.B.; Turner, E.H.; Robertson, P.D.; Flygare, S.D.; Bigham, A.W.; Lee, C.; Shaffer, T.; Wong, M.; Bhattacharjee, A.; Eichler, E.E.; Bamshad, M.; Nickerson, D.A. y Shendure, J. (2009). «Targeted capture and massively parallel sequencing of 12 human exomes». *Nature*, **461(7261)**, pp. 272–276.

Peng, Roger D. (2011). «Reproducible Research in Computational Science». *Science*, **334(6060)**, pp. 1226–1227.

Peralta, I. E.; Makuch, M.; Lampasona, S. García; Occhiuto, P. N.; Asprelli, P. D.; Lorello, I. M. y Togno, L. (2008). *Catálogo de poblaciones criollas de pimiento, tomate y zapallo colectadas en valles andinos de la Argentina*.

Pividori, M.; Stegmayer, G.; Carrari, F. y Milone, D. H. (2013a). «Consensus clustering from heterogeneous measures of S. Lycopersicum». En: *4to. Congreso Argentino de Bioinformática y Biología Computacional (4CAB2C)*.

Pividori, M.; Stegmayer, G. y Milone, D. H. (2013b). «A Novel Method to Control the Diversity in Cluster Ensembles». En: *Argentine Symposium on Artificial Intelligence (ASAI 2013) - 42º JAIIO*pp. 121–132.

Pividori, M.; Stegmayer, G. y Milone, D. H. (2014). «A Method to Improve the Analysis of Cluster Ensembles». *Revista Iberoamericana de Inteligencia Artificial*, **17(53)**, pp. 46–56.

Pividori, M.; Stegmayer, G. y Milone, D. H. (2015). «Cluster Ensembles for Big Data Mining Problems». En: *Argentine Symposium on Big Data, 44 JAIIO (Argentine Conference on Informatics)*.

Pividori, M.; Stegmayer, G. y Milone, D. H. (2016a). «Diversity control for improving the analysis of consensus clustering». *Information Sciences*, **361**, pp. 120–134.

Pividori, M.; Stegmayer, G. y Milone, D. H. (2016b). «Multi-hierarchical clustering for exploring hierarchies networks on gene expression data». Manuscrito en preparación.

Pividori, M.; Stegmayer, G. y Milone, D. H. (2016c). «Scalable and flexible consensus clustering framework for big data mining problems». Manuscrito en preparación.

Piwowar, Heather A.; Day, Roger S. y Fridsma, Douglas B. (2007). «Sharing Detailed Research Data Is Associated with Increased Citation Rate». *PLoS ONE*, **2(3)**, pp. 1–5.

Preeth, E. N.; Mulerickal, F. J. P.; Paul, B. y Sastri, Y. (2015). «Evaluation of Docker containers based on hardware utilization». En: *2015 International Conference on Control Communication Computing India (ICCC)*pp. 697–700.

Rand, William M. (1971). «Objective Criteria for the Evaluation of Clustering Methods». *Journal of the American Statistical Association*, **66(336)**, pp. 846–850.

Ross, David A.; Lim, Jongwoo; Lin, Ruei-Sung y Yang, Ming-Hsuan (2008). «Incremental Learning for Robust Visual Tracking». *International Journal of Computer Vision*, **77(1)**, pp. 125–141.

Sandve, Geir Kjetil; Nekrutenko, Anton; Taylor, James y Hovig, Eivind (2013). «Ten Simple Rules for Reproducible Computational Research». *PLoS Comput Biol*, **9(10)**, pp. 1–4.

Sculley, D. (2010). «Web-scale K-means Clustering». En: *Proceedings of the 19th International Conference on World Wide Web*pp. 1177–1178.

Sibson, R. (1973). «SLINK: An optimally efficient algorithm for the single-link cluster method». *The Computer Journal*, **16(1)**, pp. 30–34.

Singh, Dilpreet y Reddy, Chandan K. (2014). «A survey on platforms for big data analytics». *Journal of Big Data*, **2(1)**, pp. 1–20.

Smith, Rob; Ventura, Dan y Prince, John T. (2014). «Controlling for confounding variables in MS-omics protocol: why modularity matters». *Briefings in Bioinformatics*, **15(5)**, pp. 768–770.

Sokal, R. R. y Michener, C. D. (1958). «A statistical method for evaluating systematic relationships». *University of Kansas Scientific Bulletin*, **38**, pp. 1409–1438.

Stegmayer, G.; Pividori, M. y Milone, D. H. (2016). «A very simple and fast way to access and validate algorithms in reproducible research». *Briefings in Bioinformatics*, **17(1)**, pp. 180–183.

Strehl, Alexander y Ghosh, Joydeep (2003). «Relationship-Based Clustering and Visualization for High-Dimensional Data Mining». *INFORMS Journal on Computing*, **15(2)**, pp. 208–230.

Strehl, Alexander; Ghosh, Joydeep y Cardie, Claire (2002). «Cluster Ensembles - A Knowledge Reuse Framework for Combining Multiple Partitions». *Journal of Machine Learning Research*, **3**, pp. 583–617.

Theodoridis, Sergios y Koutroumbas, Konstantinos (2006). *Pattern Recognition (Third Edition)*. San Diego, third edition[a] edición.

Topchy, A.; Jain, A.K. y Punch, W. (2003). «Combining multiple weak clusterings». En: *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*pp. 331–338.

Topchy, A.; Jain, A.K. y Punch, W. (2005). «Clustering ensembles: models of consensus and weak partitions». *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **27(12)**, pp. 1866–1881.

Topchy, A.; Minaei-Bidgoli, B.; Jain, A.K. y Punch, W.F. (2004). «Adaptive clustering ensembles». En: *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*volumen 1, pp. 272–2751.

Traganitis, P. A.; Slavakis, K. y Giannakis, G. B. (2014). «Clustering high-dimensional data via random sampling and consensus». En: *Signal and Information Processing (GlobalSIP), 2014 IEEE Global Conference on*pp. 307–311.

Vandewalle, P.; Kovacevic, J. y Vetterli, M. (2009). «Reproducible research in signal processing». *IEEE Signal Processing Magazine*, **26(3)**, pp. 37–47.

Vinh, Nguyen Xuan; Epps, Julien y Bailey, James (2009). «Information Theoretic Measures for Clusterings Comparison: Is a Correction for Chance Necessary?» En: *Proceedings of the 26th Annual International Conference on Machine Learning*ICML '09, pp. 1073–1080.

Vinh, Nguyen Xuan; Epps, Julien y Bailey, James (2010). «Information Theoretic Measures for Clusterings Comparison: Variants, Properties, Normalization and Correction for Chance». *Journal of Machine Learning Research*, **11**, pp. 2837–2854.

Wang, H.; Li, Z. y Cheng, Y. (2008). «Distributed and Parallelled EM Algorithm for Distributed Cluster Ensemble». En: *Computational Intelligence and Industrial Application, 2008. PACIIA '08. Pacific-Asia Workshop on*volumen 2, pp. 3–8.

Ward, J. (1963). «Hierarchical Grouping to Optimize an Objective Function». *Journal of the American Statistical Association*, **58(301)**, pp. 236–244.

Witten, Ian H.; Frank, Eibe y Hall, Mark A. (2011). *Data Mining: Practical Machine Learning Tools and Techniques (Third Edition)*. Morgan Kaufmann Publishers Inc., Boston.

Wu, Junjie; Liu, Hongfu; Xiong, Hui; Cao, Jie y Chen, Jian (2015). «K-Means-Based Consensus Clustering: A Unified View». *Knowledge and Data Engineering, IEEE Transactions on*, **27(1)**, pp. 155–169.

Wu, Junjie; Xiong, Hui y Chen, Jian (2009). «Adapting the Right Measures for K-means Clustering». En: *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*KDD '09, pp. 877–886.

Wu, X.; Zhu, X.; Wu, G. Q. y Ding, W. (2014). «Data mining with big data». *IEEE Transactions on Knowledge and Data Engineering*, **26(1)**, pp. 97–107.

Xu, Rui y Wunsch, Don (2009). *Clustering*. Wiley-IEEE Press.

Yang, Fan; Li, Xuan; Li, Qianmu y Li, Tao (2014). «Exploring the diversity in cluster ensemble generation: Random sampling and random projection». *Expert Systems with Applications*, **41(10)**, pp. 4844–4866.

Yang, Y. y Jiang, J. (2016). «Hybrid Sampling-Based Clustering Ensemble With Global and Local Constitutions». *IEEE Transactions on Neural Networks and Learning Systems*, **27(5)**, pp. 952–965.

Yang, Yun y Chen, Ke (2011). «Temporal Data Clustering via Weighted Clustering Ensemble with Different Representations». *Knowledge and Data Engineering, IEEE Transactions on*, **23(2)**, pp. 307–320.

Yi, Jinfeng; Yang, Tianbao; Jin, Rong; Jain, A.K. y Mahdavi, M. (2012). «Robust Ensemble Clustering by Matrix Completion». En: *Data Mining (ICDM), 2012 IEEE 12th International Conference on*pp. 1176–1181.

Yu, Z.; Luo, P.; You, J.; Wong, H. S.; Leung, H.; Wu, S.; Zhang, J. y Han, G. (2016). «Incremental Semi-Supervised Clustering Ensemble for High Dimensional Data Clustering». *IEEE Transactions on Knowledge and Data Engineering*, **28(3)**, pp. 701–714.

Yu, Zhiwen; Li, Le; Gao, Yunjun; You, Jane; Liu, Jiming; Wong, Hau-San y Han, Guoqiang (2014). «Hybrid clustering solution selection strategy». *Pattern Recognition*, **47(10)**, pp. 3362–3375.

Yu, Zhiwen; Wong, Hau-San y Wang, Hongqiang (2007). «Graph-based consensus clustering for class discovery from gene expression data». *Bioinformatics*, **23(21)**, pp. 2888–2896.

Yu, Zhiwen; Wong, Hau-San; You, Jane; Yu, Guoxian y Han, Guoqiang (2012a). «Hybrid cluster ensemble framework based on the random combination of data transformation operators». *Pattern Recognition*, **45(5)**, pp. 1826–1837.

Yu, Zhiwen; You, Jane; Wong, Hau-San y Han, Guoqiang (2012b). «From cluster ensemble to structure ensemble». *Information Sciences*, **198**, pp. 81–99.

Zhong, Caiming; Yue, Xiaodong; Zhang, Zehua y Lei, Jingsheng (2015). «A clustering ensemble: Two-level-refined co-association matrix with path-based transformation». *Pattern Recognition*, **48(8)**, pp. 2699–2709.