



**UNIVERSIDAD
DE GRANADA**

**TRABAJO DE FIN DE GRADO
INGENIERÍA INFORMÁTICA**

Clustering Con Restricciones

Un Marco Unificado

Autor

Germán González Almagro

Directores

Salvador García López
Julián Luengo Martín



**ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS INFORMÁTICA Y DE
TELECOMUNICACIÓN**

—
Granada, mes de Junio

Clustering Con Restricciones

Un Marco Unificado.

Autor

Germán González Almagro

Directores

Salvador García López
Julián Luengo Martín

Clustering Con Restricciones: Un Marco Unificado

Germán González Almagro

Palabras clave: palabra_clave1, palabra_clave2, palabra_clave3,

Resumen

Poner aquí el resumen.

Constrained Clustering: A Unified Framework

Germán González Almagro

Keywords: Keyword1, Keyword2, Keyword3,

Abstract

Write here the abstract in English.

Yo, **Germán González Almagro**, alumno de la titulación en ingeniería informática de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 76593910T, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Germán González Almagro

Granada a 17 de mes de Junio.

D. Salvador García López, Profesor del Departamento Ciencias de la Computación en Inteligencia Artificial de la Universidad de Granada.

D. Julián Luengo Martín, Profesor del Departamento Ciencias de la Computación en Inteligencia Artificial de la Universidad de Granada.

Informan:

Que el presente trabajo, titulado *Clustering Con Restricciones, Un Marco Unificado*, ha sido realizado bajo su supervisión por **Germán González Almagro**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a 17 de mes de Junio.

Los directores:

Salvador García López Julián Luengo Martín

AGRADECIMIENTOS

Poner aquí agradecimientos...

ÍNDICE GENERAL

1 INTRODUCCIÓN	1
1.1 El problema del Aprendizaje Automático	2
1.1.1 ¿Qué es el aprendizaje?	2
1.1.2 Tipos de Aprendizaje Automático	2
1.2 Objetivos	3
1.3 Estructura	4
2 BREVE INTRODUCCIÓN AL CLUSTERING	5
2.1 Motivación para la clasificación	5
2.2 Métodos numéricos para el Clustering	5
2.2.1 ¿Qué es un cluster?	7
2.3 Aplicaciones del clustering	9
2.3.1 Aplicaciones en marketing	9
2.3.2 Aplicaciones en astronomía	10
2.3.3 Aplicaciones en psiquiatría	10
2.3.4 Aplicaciones en meteorología y climatología .	11
2.3.5 Aplicaciones en arqueología	11
2.3.6 Aplicaciones en bioinformática y genética .	12
2.4 Resumen	13
3 CLUSTERING CON RESTRICCIONES	15
3.1 Motivación para el clustering con restricciones	15
3.2 Definición de las restricciones	16
3.3 Uso de las restricciones	16
3.3.1 Métodos basados en restricciones	17
3.3.2 Métodos basados en distancia	18
3.4 Aplicaciones del clustering con restricciones	19
3.4.1 Aplicaciones en análisis de imágenes	19
3.4.2 Aplicaciones en análisis de vídeos	21
3.4.3 Aplicaciones en genética	21
3.4.4 Aplicaciones en análisis de textos	22
3.4.5 Aplicaciones en datos web	22
3.4.6 Aplicaciones en datos de audio	23
3.4.7 Aplicaciones en datos de GPS	23
3.5 Beneficios del uso de restricciones	24
3.6 Problemas del uso de restricciones	25
3.6.1 El problema de la factibilidad	25
3.6.2 El problema de la utilidad de conjuntos de res- tricciones	26
3.6.3 Soluciones al problema de la factibilidad	27
3.6.4 Soluciones al problema de la utilidad de con- juntos de restricciones	28
3.7 Resumen	29
4 ALGORITMOS DE CLUSTERING CON RESTRICCIONES	31

4.1	Formalización del problema	31
4.2	COP-K-means (Constrained K-means)	31
4.3	CEKM (Constrained Evidential K-means)	33
4.3.1	Funciones de creencia	33
4.3.2	FKM (fuzzy K-means) y sus variantes	34
4.3.3	El algoritmo EKM (Evidential K-means)	35
4.3.4	Incorporación de restricciones a EKM	37
4.3.5	Función objetivo de CEKM	38
4.3.6	Proceso de optimización de CEKM	38
4.4	Linear Constrained Vector Quantization Error (LCVQE)	40
4.4.1	El algoritmo CVQE	40
4.4.2	El algoritmo LCVQE	43
4.5	Relational Dirichlet Process - Means (RDP - Means)	45
4.5.1	El modelo Bayesiano no paramétrico	47
4.5.2	Reparametrización de la familia exponencial para RDP-means	51
4.5.3	Escalando las distribuciones para RDP-means . .	52
4.5.4	Asintóticos de TVClust	52
4.5.5	Muestreo de los parámetros de los clusters	53
4.5.6	Función objetivo de RDP-means	53
4.5.7	Efectos de los cambios sobre ξ_1 y ξ_2	54
5	IMPLEMENTACIÓN	57
5.1	Entorno de desarrollo	57
5.2	Bibliotecas empleadas en el desarrollo	57
5.3	Funcionalidades desarrolladas	58
5.3.1	Función para generar restricciones	58
5.3.2	Función para aplicar COP-K-Medias	59
5.3.3	Función para aplicar CEKM	60
5.3.4	Función para aplicar LCVQE	60
5.3.5	Función para aplicar RDPM	61
5.3.6	Función para aplicar TVClust	62
6	EXPERIMENTACIÓN	63
6.1	Conjuntos de datos considerados	63
6.1.1	Conjuntos de datos reales	63
6.1.2	Conjuntos de datos artificiales	65
6.2	Medida del error	65
6.3	Generación de las restricciones	66
6.4	Resultados obtenidos con COP-K-Medias	66
6.5	Resultados obtenidos con CEKM	69
6.6	Resultados obtenidos con LCVQE	70
6.7	Resultados obtenidos con RDP-Means	71
6.8	Resultados obtenidos con TVClust	73
6.9	Comparativa general de resultados	74
7	CONCLUSIONES	75
7.1	Trabajo Futuro	75
A	EL ALGORITMO K-MEDIAS	77

ÍNDICE DE FIGURAS

Figura 2.1	Clusters con cohesión interna y/o aislamiento externo	8
Figura 2.2	Distribución uniforme de puntos	8
Figura 2.3	Distribución uniforme de puntos clasificados . .	8
Figura 3.1	Restricciones de tipo delta y epsilon.	17
Figura 3.2	Restricciones sobre un conjunto de datos.	18
Figura 3.3	Clustering que satisface todas las restricciones. .	18
Figura 3.4	Restricciones sobre un conjunto de datos.	19
Figura 3.5	Clustering basado en métrica aprendida en base a las restricciones.	19
Figura 3.6	Caras de la base de datos de CMU.	20
Figura 3.7	Restricciones de tipo Cannot-Link (CL) entre caras de la misma persona.	20
Figura 3.8	Método de clustering empleado en el sistema de navegación del robot Aibo.	20
Figura 3.9	Diferentes tipos de restricciones en datos de video.	21
Figura 3.10	Clustering de genes basado en microarrays. . . .	22
Figura 3.11	Uso de información GPS.	23
Figura 3.12	Clusters encontrados en datos GPS sin uso de restricciones.	24
Figura 3.13	Ejemplo de conjunto de restricciones informativo.	28
Figura 3.14	Ejemplo de conjunto de restricciones contradictorio.	29
Figura 3.15	Representación de la medida de coherencia. . . .	29
Figura 4.1	Ejemplo de CVQE.	42
Figura 4.2	Ejemplo de CVQE.	43
Figura 6.1	Distribución de las dos primeras características de <i>Iris Dataset</i>	64
Figura 6.2	Conjuntos de datos artificiales.	65
Figura 6.3	Visualización de las restricciones.	66
Figura 6.4	Visualización de los resultados de COP-K-Medias	68
Figura 6.5	Visualización de los resultados de CEKM	69
Figura 6.6	Visualización de los resultados de LCVQE . . .	70
Figura 6.7	Visualización de los resultados de RDPM con diferentes valores de λ	72
Figura 6.8	Visualización de los resultados de TVClust . . .	73

ÍNDICE DE TABLAS

Tabla 3.1	Complejidad del problema del clustering en función del tipo de restricciones	26
Tabla 4.1	Ejemplo de partición de creencia	36
Tabla 4.2	Valores de CVQE para el ejemplo	43
Tabla 4.3	Valores de LCVQE para el ejemplo	45
Tabla 6.1	Características de los conjuntos de datos reales considerados	64
Tabla 6.2	Resultados obtenidos con COP-K-Medias empleado restricciones CL y ML	67
Tabla 6.3	Resultados obtenidos con COP-K-Medias empleado solo restricciones ML	67
Tabla 6.4	Resultados obtenidos con CEKM	69
Tabla 6.5	Resultados obtenidos con LCVQE	70
Tabla 6.6	Resultados obtenidos con RDPM	71
Tabla 6.7	Resultados obtenidos con RDPM especificando un mayor para λ	71
Tabla 6.8	Resultados obtenidos con TVClust	73
Tabla 6.9	Comparativa general de resultados (RandIndex)	74
Tabla 6.10	Comparativa general de resultados (Tiempo) .	74

ÍNDICE DE ALGORITMOS

Algoritmo 1	COP-K-medias	32
Algoritmo 2	Constrained Evidential K-means (CEKM) . . .	39
Algoritmo 3	Linear Constrained Vector Quantization Error (LCVQE)	46
Algoritmo 4	Relational Dirichlet Process - Means (RDPM) .	55
Algoritmo 5	K-medias	77

ACRONYMS

AA	Aprendizaje Automático
AS	Aprendizaje Supervisado

ASS	Aprendizaje Semi-Supervisado
ANS	Aprendizaje no Supervisado
AR	Aprendizaje por Refuerzo
ML	Must-Link
CL	Cannot-Link
KM	K-medias
CEKM	Constrained Evidential K-means
FKM	Fuzzy K-means
EKM	Evidential K-means
NC	Noise Clustering
VQE	Vector Quantization Error
CVQE	Constrained Vector Quantization Error
LCVQE	Linear Constrained Vector Quantization Error
RDPM	Relational Dirichlet Process - Means
DPM	Dirichlet Process - Means
TVClust	Two Views Clustering
URL	Uniform Resource Locator
GPS	Global Positioning System

INTRODUCCIÓN

An intelligent being cannot treat every object it sees as unique entity unlike anything else in the universe. It has to put objects in categories so that it may apply its hard-won knowledge about similar objects encountered in the past, to the object at hand.

Steven Pinker, How the Mind Works, 1997

Una de las habilidades más básicas y primitivas de la que están dotados los seres humanos es la de agrupar objetos similares para producir una clasificación que les resulte útil. Habilidad que ya nuestros más antiguos ancestros debieron poseer, por ejemplo, debieron ser capaces de darse cuenta de qué objetos eran comestibles, cuales eran venenosos y cuales intentarían matarles.

La capacidad de clasificación, en su sentido más amplio, es necesaria para el desarrollo del lenguaje, que está formado por palabras que nos ayudan a reconocer diferentes tipos de eventos, acciones y entidades. En esencia, cada sustantivo es una etiqueta que empleamos para agrupar un colectivo de seres u objetos con características similares, de manera que podemos hacer referencia a todos ellos empleando la palabra que los une.

De igual forma que la clasificación es una habilidad básica para las personas en su vida cotidiana, es también esencial en la mayoría de las ramas de la ciencia. En biología, por ejemplo, la clasificación de los diferentes tipos de organismos ha sido objeto de estudio desde el comienzo de su existencia. Aristóteles construyó un elaborado sistema de clasificación animal que dividía a todas las criaturas en dos grupos, aquellos con sangre roja y aquellos que carecían de ella. Más tarde propuso una subdivisión que los clasificaba según la forma en la que nuevos individuos venían al mundo, ya sea vivos, mediante huevos, crisálidas, etc.

Siguiendo a Aristóteles, Teofrasto escribió el primer documento que recopilaba las directrices para la clasificación de las plantas. Los trabajos resultantes fueron tan amplios y detallados que sentaron las bases para la investigación en biología durante los siguientes siglos. Este trabajo no fue sustituido hasta 1737, cuando Carlos Linneo publicó su obra *Genera Plantarum*, de la que extraemos el siguiente fragmento:

All the real knowledge which we possess, depends on methods by which we distinguish the similar from the dissimilar. The greater the number of natural distinctions this method comprehends the clearer becomes our idea of things. The more nume-

rous the objects which employ our attention the more difficult it becomes to form such a method and the more necessary. For we must not join in the same genus the horse and the swine, though both species had been one hoof'd nor separate in different genera the goat, the reindeer and the elk, tho' they differ in the form of their horns. We ought therefore by attentive and diligent observation to determine the limits of the genera, since they cannot be determined a priori. This is the great work, the important labour, for should the genera be confused, all would be confusion.

Carlos Linneo, Genera Plantarum, 1737

La clasificación de los animales y las plantas ha sido de gran importancia en campos como la biología y la zoología. Particularmente, esta clasificación sentó las bases para el desarrollo de la teoría de la evolución de Darwin. Pero también ha sido de gran relevancia en áreas de conocimiento como la química y la física, con la clasificación de los elementos en la tabla periódica, propuesta por Mendeleev en la década de 1860; o en astronomía, con la clasificación de estrellas en enanas o gigantes empleando las directrices de Hertzsprung–Russell.

1.1 EL PROBLEMA DEL APRENDIZAJE AUTOMÁTICO

El Aprendizaje Automático (AA), del inglés *Machine Learning*, es una campo derivado de las ciencias de la computación, y una rama de la inteligencia artificial, que tiene como último objetivo desarrollar métodos que hagan posible que las máquinas aprendan. En otras palabras, las técnicas que se enmarcan dentro del AA tienen como objetivo desarrollar programas que guíen a las computadoras para que aprendan a partir de un conjunto de ejemplos.

1.1.1 ¿Qué es el aprendizaje?

La Real Academia Española define el aprendizaje de la siguiente manera: “Adquisición por la práctica de una conducta duradera”. Si bien esta definición es plenamente aplicable al objetivo del AA, es demasiado general como para abarcar lo que este realmente pretende. De ahora en adelante, en lo que a este trabajo respecta, entenderemos el aprendizaje como: “Cambios en un sistema adaptativo que hacen posible que el mismo realice una tarea conocida de manera más efectiva y eficiente” [1].

1.1.2 Tipos de Aprendizaje Automático

Existen varios enfoques dentro del campo del AA, estos se clasifican en función de la información que se le proporciona a la máquina para que aprenda. Así, encontramos cuatro tipos de aprendizaje distintos:

- **Aprendizaje Supervisado (AS):** del inglés *Supervised Learning*. En él, la máquina dispone del conjunto de datos así como de la clase a la que pertenecen los mismos. El objetivo es aprender una función que permita predecir la clase de ejemplos que no se encontraban en el conjunto que se ha empleado para el aprendizaje. Podemos entender este tipo de aprendizaje comparando el conjunto de entrenamiento con un profesor y la máquina como un alumno.
- **Aprendizaje no Supervisado (ANS):** del inglés *Unsupervised Learning*. A diferencia del caso anterior, la máquina sólo dispone del conjunto de datos, y no de la clase a la que estos pertenecen. El objetivo es obtener la clase a la que pertenecen los ejemplos que se encuentran en el conjunto de entrenamiento, y en algunas ocasiones extraer también una función que prediga las clases de nuevos ejemplos. Retomando la analogía del alumno y el profesor, podemos entender que en ANS es el alumno el que aprende por sí mismo y enseña al profesor, dado que este último no conoce nada sobre los datos.
- **Aprendizaje Semi-Supervisado (ASS):** del inglés *Semi Supervised Learning*. Este tipo de aprendizaje se encuentra entre los dos anteriores, la máquina dispone del conjunto de datos y de información parcial sobre los mismos. El objetivo es el igual que en ANS, pero las técnicas empleadas para lograrlo son diferentes. Ahora, tanto profesor como alumno aprenden, ya que ninguno de ellos dispone de toda la información. El alumno aprenderá primero del profesor y desarrollará ese nuevo conocimiento para más tarde mostrárselo a éste.
- **Aprendizaje por Refuerzo (AR):** del inglés *Reinforcement Learning*. En este caso la máquina dispone sólo del conjunto de datos, pero el esquema de aprendizaje es distinto a los anteriores. Éste se basa en el método ensayo-error, esto es, la máquina (el alumno) lleva a cabo acciones sobre los datos y éstas se ven recompensadas o castigadas por el entorno (el profesor).

1.2 OBJETIVOS

Este trabajo tiene como objetivo abordar el aprendizaje semi-supervisado desde el marco de las restricciones a nivel de instancia sobre el problema. Aunque hay multitud de estudios y bibliografía sobre esta área, no existen publicaciones recientes que lo aborden de manera práctica y unificada.

En este trabajo se presenta una revisión sobre las técnicas comunes para aplicar clustering con restricciones, se profundiza sobre el mismo y se exponen sus aplicaciones. Además, se estudian y ponen a disposición de la comunidad cinco algoritmos de clustering con

restricciones a nivel de instancia. El lenguaje escogido para la implementación de dichos métodos es Python, dada su popularidad en el área del AA y su facilidad de uso.

1.3 ESTRUCTURA

El objetivo de la Sección 2 es realizar una introducción al problema de clustering. Para ser más concretos, en ella se definen los conceptos básicos sobre el clustering y se exponen algunas de sus aplicaciones.

Por su parte, la Sección 3 define los conceptos relacionados con el clustering con restricciones, poniendo de manifiesto las diferencias que presenta el mismo respecto a otras técnicas. Además, aporta ejemplos de aplicación del mismo y realiza un estudio sobre las ventajas y desventajas que éste presenta. Será en la Sección 4 en la que analizaremos cinco de los métodos de clustering con restricciones existentes, profundizando en sus bases y unificando los conceptos comunes a los mismos.

En la Sección 5 encontramos una guía y documentación para el uso del software desarrollado, que corresponde a la implementación de los cinco métodos expuestos en la Sección 4. La Sección 6 está íntimamente relacionada con la 5, sirviendo como prueba de concepto para los métodos documentados en la misma.

Por último, en la Sección 7 presentamos las conclusiones sobre el trabajo realizado.

2

BREVE INTRODUCCIÓN AL CLUSTERING

En primer lugar, será necesario realizar una introducción, siquiera sea breve, al clustering y sus aplicaciones, que facilite la comprensión de las siguientes secciones. A tal fin se seguirá el estudio realizado por Brian S. Everitt et al. (2011) [2].

2.1 MOTIVACIÓN PARA LA CLASIFICACIÓN

La clasificación puede ser entendida como una forma de simplificar la información contenida en grandes conjuntos de datos, de un modo que sea fácilmente comprensible por las personas. De esta manera, los procesos de extracción de información útil y aplicación de la misma se simplifican. Así, si somos capaces de dividir de forma válida un gran conjunto de datos en subconjuntos o grupos, podremos extraer información común a todos los elementos del subconjunto y proporcionar una descripción precisa que los englobe.

La necesidad de analizar la información de esta manera crece con la aparición y disponibilidad de grandes conjuntos de datos en el ámbito de la ciencia. El análisis de este tipo de información mediante clasificación, o clustering, hoy en día es conocido como *Ciencia de datos*. En el siglo XXI surge un particular interés en la ciencia de datos desde la aparición de la *World Wide Web*, conocida como Internet, donde el objetivo se ha convertido en extraer información relevante de las páginas Web que forman esta vasta red.

Es importante destacar que en la mayoría de las ocasiones no hay un sólo criterio de clasificación para un mismo conjunto de datos, sino que existe una amplia variedad de los mismos. En el caso de las personas, podrían ser clasificadas, por ejemplo, en base a sus ingresos económicos, o según la cantidad de calorías que consumen a lo largo de un periodo de tiempo definido. Así, distintos criterios de clasificación no tienen por qué dar como resultado la misma división en grupos del conjunto a clasificar; de esta manera, diferentes criterios servirán a diferentes propósitos.

2.2 MÉTODOS NUMÉRICOS PARA EL CLUSTERING

Los métodos numéricos para el clustering surgen en ramas de las ciencias naturales, como la biología o la zoología, en un intento de eliminar la subjetividad implícita en el proceso de clasificación que se desencadena al descubrir una nueva especie. El objetivo es pro-

porcionar un método no subjetivo y estable para clasificar y agrupar elementos.

Estos métodos adoptan diversos nombres que varían según el campo en el que se apliquen: taxonomía numérica (*numerical taxonomy*) en biología, *Q* análisis (*Q analysis*) en psicología, o reconocimiento de patrones no supervisado (*unsupervised pattern recognition*) en el campo de la inteligencia artificial. No obstante, hoy en día, análisis de clusters (*Clusters analysis*) o simplemente clustering son los términos más ampliamente aceptados y extendidos para referirse a tareas que involucran el descubrimiento de subgrupos dentro de un conjunto de elementos.

En la mayoría de las aplicaciones del clustering, el objetivo es obtener una partición de los datos, en la que cada instancia u objeto pertenezca a un único cluster, y la unión de todos ellos contenga a todos los objetos individuales. Dicho esto, debe destacarse que en algunas circunstancias son aceptables soluciones en las que existe solapamiento entre clusters, así como el hecho de que puede no existir una partición aceptable de los datos.

La manera más ampliamente extendida de representar la información sobre la que se debe aplicar clustering es una matriz X de dimensión $n \times p$, en la que cada fila corresponde a una instancia u objeto a procesar, y cada columna corresponde a una de las variables que caracterizan dichas instancias u objetos. El término comúnmente aceptado para referirse a cada fila es el de *vector de características*.

$$X = \begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,p} \\ x_{2,1} & \cdots & \cdots & \cdots \\ \vdots & \vdots & \vdots & \vdots \\ x_{n,1} & \cdots & \cdots & x_{n,p} \end{pmatrix}$$

La entrada $x_{i,j}$ en X se corresponde con el valor de la variable j -esima en la instancia i .

Las variables en X pueden ser una mezcla de atributos en un dominio continuo, discreto o categórico. Además, es posible que, en problemas reales, algunas entradas no estén disponibles. Esta mezcla de tipos de variables y los valores perdidos pueden complicar la tarea del clustering. Sin embargo, existen métodos para tratar estos casos, como la inferencia de valores perdidos o las transformaciones de dominio.

En algunas aplicaciones, las filas de la matriz pueden contener medidas repetidas de la misma variable, aunque bajo diferentes condiciones, o en diferentes momentos, incluso en diferentes localizaciones espaciales. Un ejemplo de ello pueden ser las medidas de altura de un grupo de niños en un mismo mes a lo largo de diferentes años. Este tipo de datos posee una estructura que, de nuevo, puede complicar la tarea del clustering.

Algunos métodos de clustering conllevan realizar transformaciones sobre la matriz X para transformarla en una matriz de $n \times n$ en la que se almacenan medidas extraídas de la matriz X que relacionen una instancia con todas las demás, como pueden ser la similitud, distancia o disimilitud.

El clustering es, dicho de manera simple, descubrimiento de grupos en datos, y no debe ser en ningún caso confundido con los métodos de discriminación o asignación, conocidos en el ámbito de la inteligencia artificial como aprendizaje supervisado. En ellos los grupos son conocidos a priori y el objetivo del análisis es obtener una regla de clasificación o clasificador que permita asignar nuevas instancias o individuos a uno de los grupos ya conocidos.

Una vez definida la estructura general de los métodos de clustering, estaría justificado preguntar, ¿qué es un cluster? La Sección 2.2.1 intentará dar respuesta a esta pregunta.

2.2.1 *¿Qué es un cluster?*

Hasta este momento, los términos cluster, grupo y clase han sido empleados de una manera completamente intuitiva, sin necesidad alguna de definición formal, una prueba más de lo innato de estos conceptos en el ser humano. De hecho, dar una definición formal de cluster resulta una tarea no, sólo complicada, sino en muchas ocasiones poco útil. Bonner, por ejemplo, propuso en 1964 una definición de cluster completamente dependiente de la interpretación del usuario: en lo que a él respecta, un cluster es aquello que el usuario entiende como cluster sin haberle propuesto una definición formal del mismo.

Aunque la definición de Bonner es acertada en una amplio rango de situaciones, autores como Cormack, en 1971, o Gordon en 1999, proponen una definición algo más analítica desde el punto de vista matemático, definiendo un cluster en términos de cohesión interna (homogeneidad), y aislamiento externo (separación). La Figura 2.1 muestra de manera informal las propiedades descritas anteriormente de forma que, a cualquier observador le resultarán aparentes los clusters presentes en ella sin necesidad de una definición formal de cluster. Este hecho puede explicar por qué alcanzar una definición matemáticamente precisa de homogeneidad y separación puede llegar a ser innecesario.

No queda completamente clara la manera en que las personas reconocen diferentes clusters cuando éstos son representados en un plano, pero una de las variables que con certeza influye es la distribución de distancias relativas entre los objetos o puntos.

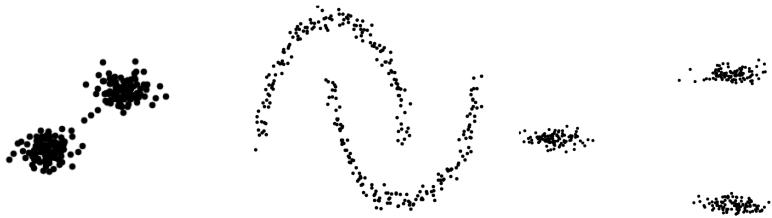


Figura 2.1: Clusters con cohesión interna y/o aislamiento externo

Por otra parte, como ya mencionamos anteriormente en esta sección, puede darse el caso de que en un conjunto de datos no exista una partición justificada. En la Figura 2.2 se muestra un conjunto de datos para el que la mayoría de observadores llegaría a la conclusión de que no existen grupos diferenciados, simplemente una nube de puntos uniformemente distribuidos. Idealmente, es de esperar que un método de clustering aplicado a este mismo conjunto de datos llegue a la misma conclusión.

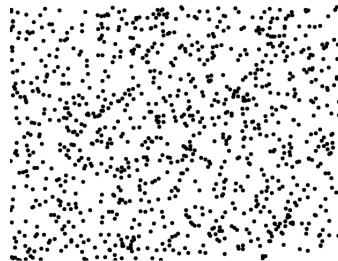


Figura 2.2: Distribución uniforme de puntos

Sin embargo, la mayoría de métodos de aprendizaje no supervisado darán como resultado un particionamiento uniforme como el que se muestra en la Figura 2.3. El número de particiones encontradas dependerá del método aplicado, si bien en cualquier caso obtendremos un particionamiento uniforme.

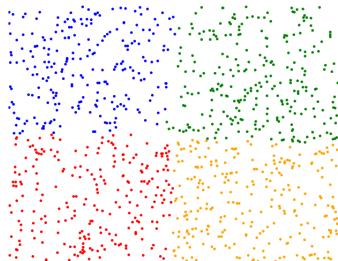


Figura 2.3: Distribución uniforme de puntos clasificados

El proceso de dividir una distribución homogénea de datos en diferentes grupos se conoce como disección, y tal proceso puede ser útil en ciertas circunstancias. Sin embargo, dado que en la mayoría de las ocasiones de aplicación real de métodos de clusters no se conoce a priori la estructura de los datos, existe el riesgo de interpretar todas

las soluciones en términos de existencia de subgrupos, lo que conllevaría la imposición de una estructura ficticia en datos en los que no hay estructura presente.

2.3 APLICACIONES DEL CLUSTERING

Como ya se ha indicado, el problema general al que intenta dar solución el clustering está presente en muchas ramas de la ciencia: biología, botánica, medicina, psicología, geografía, marketing, procesamiento de imágenes, psiquiatría, arqueología, etc. En esta sección se presentan algunas de las aplicaciones del clustering relacionadas con las citadas disciplinas.

2.3.1 *Aplicaciones en marketing*

Dividir los clientes en grupos homogéneos es una de las tareas más frecuentes en marketing. Un director de marketing podría preguntarse cómo agrupar los posibles clientes según los beneficios potenciales del producto que intenta introducir en el mercado. Por otra parte, un analista de marketing podría estar interesado en agrupar las empresas según sus características financieras, para poder analizarlas y predecir sus estrategias de mercado.

Un ejemplo de aplicación del clustering en este ámbito fue publicado por Green et al. (1967). Así, con un gran número de ciudades disponibles para el análisis, debieron restringir los lugares en los que llevar a cabo sus estudios, debido a motivos económicos. Para ello hicieron uso del análisis de clusters, clasificando las ciudades en pequeños grupos basándose en 14 características de las mismas, entre ellas el tamaño e ingresos medios *per capita*. Dado que se esperaba que las ciudades incluidas en un mismo grupo fueran muy similares, escogieron una ciudad de cada uno de ellos para realizar sus estudios.

Otra aplicación del análisis de clusters en el marketing fue descrita por Chakrapani (2004). En este caso, un fabricante de coches cree que comprar un coche deportivo no es una decisión basada sólo en capacidades económicas o edad, sino que es una decisión relacionada con el estilo de vida que llevan aquellos que deciden comprar un coche de estas características, frente a aquellos que no lo hacen. En consecuencia, el fabricante decide realizar un estudio, empleando análisis de clusters, que le permita identificar todas las características relacionadas con las personas que comprarían un coche deportivo, para así enfocar sus campañas de marketing a este sector específicamente.

2.3.2 *Aplicaciones en astronomía*

Dado un conjunto de datos astronómicos, los investigadores quieren saber, por ejemplo, cuantas clases de estrellas hay presentes en ellos, basándose en algun criterio estadístico. Las preguntas más frecuentes dentro de este ámbito son: ¿Cuantos objetos estadísticamente diferentes están presentes en los datos y a que clase debe ser asignado cada objeto? ¿Aparecen clases de objetos previamente desconocidas?. El análisis de clusters puede ser aplicado para dar respuesta a estas cuestiones, ayudando a detectar objetos estadísticamente anómalos, así como a guiar el proceso de clasificación de los mismos. Algunos ejemplos incluyen el descubrimiento de quasars con alto corrimiento al rojo, quasars de tipo 2 (altamente luminosos, núcleos galácticos activos a menudo oscurecidos por polvo y gas), y enanas marrones.

Un ejemplo específico viene dado por el estudio de Faúndez-Abans et al. (1996), que aplicó técnicas de clustering a datos sobre la composición química de 192 nebulosas planetarias. Se identificaron 6 grupos diferentes que eran similares en muchos aspectos a una clasificación previa de dichos objetos, pero que también mostraban diferencias interesantes que hasta ese momento los investigadores habían pasado por alto.

Un segundo ejemplo lo encontramos en el estudio de Celeux y Govaert (1992), quienes aplicaron clustering basado en distribuciones normales a un conjunto de 2370 estrellas, descritas por su velocidad relativa al núcleo galáctico y a la rotación galáctica. Usando un modelo de tres clusters, encontraron un cluster de gran tamaño y pequeño volumen, y dos de pequeño tamaño y gran volumen.

2.3.3 *Aplicaciones en psiquiatría*

Las enfermedades de la mente son a menudo más difíciles de diagnosticar que las enfermedades del cuerpo; es por ello que en el campo de la psiquiatría ha crecido el interés por las técnicas de análisis de clusters que permitan refinar, o incluso redefinir, las técnicas de diagnosis para este tipo de enfermedades. Gran parte de este trabajo involucra pacientes deprimidos, casos en los que el interés reside en distinguir entre dos tipos de depresión, la endógena (congénita), y la neurótica.

Pilowsky et al. (1968), por ejemplo, usando métodos desarrollados por otros autores, aplicaron técnicas de clustering a 200 pacientes en base a sus respuestas a un cuestionario sobre la depresión, junto a información sobre su sexo, edad, estado mental y enfermedad padecida. Éste es un claro ejemplo de variables de diferentes tipos incluidas en el mismo conjunto de datos. Uno de los grupos obtenidos como resultado de este estudio fue identificado como marcador de la depresión endógena.

El análisis de clusters también ha sido empleado para encontrar una clasificación de individuos que intentaron cometer suicidio, que podría sentar las bases para estudios posteriores sobre las causas y tratamientos del problema. Paykey y Rassaby (1978), por ejemplo, estudiaron 236 casos de suicidas fallidos registrados por el servicio de emergencias de una ciudad de los Estados Unidos de América. Del conjunto de posibles variables, 14 fueron seleccionadas como particularmente relevantes para la clasificación y, por tanto, fueron usadas en el análisis. Entre ellas se encontraban: edad, número de intentos de suicidio, gravedad de la depresión, grado de hostilidad, además de una serie de características demográficas. Al conjunto de datos resultante se le aplicaron métodos de clustering; el resultado más significativo obtenido corresponde a una división en tres clusters bien definidos.

2.3.4 *Aplicaciones en meteorología y climatología*

Diariamente se recogen enormes cantidades de datos sobre la meteorología mundial. Explorar estos datos mediante técnicas de clustering puede aportar nuevos enfoques para la climatología y el medio ambiente.

Littmann (2000), por ejemplo, aplicó clustering a los datos recogidos sobre los cambios diarios en la presión superficial en la cuenca Mediterránea, y encontró 20 grupos que explicaban la varianza de las lluvias en las regiones centrales del Mediterráneo. Otro ejemplo viene de la mano de Liu y George (2005), quienes usaron el algoritmo K-medias difuso (*Fuzzy K-means*, FKM) con datos espaciotemporales de la climatología de las regiones del sur central de EEUU.

2.3.5 *Aplicaciones en arqueología*

La arqueología es otra de las disciplinas en la que resulta útil la aplicación del clustering. La clasificación de los diferentes objetos encontrados en los yacimientos puede ayudar a descubrir su uso, los períodos a los que pertenecen, así como la población que los utilizó. De forma similar, el estudio de materiales fosilizados puede ayudar a revelar cómo vivieron las sociedades prehistóricas.

Un ejemplo temprano de la aplicación de clustering a objetos arqueológicos viene dado por Hodson et al. (1966), que aplicó técnicas de clustering a un grupo de broches que datan de la Edad de Hierro, encontrando una clasificación para los mismos de demostrada relevancia arqueológica. Otro ejemplo de la mano de Hodson (1971) es la aplicación del algoritmo K-medias (KM, Apéndice A) para construir una taxonomía de hachas de mano encontradas en las Islas Británicas. Las variables tenidas en cuenta para describir cada hacha incluyen longitud, anchura y valores en una escala que describen cómo de

puntiaguda era la herramienta. El clustering dio como resultado dos grupos de hachas, uno formado por las pequeñas y delgadas, y otro formado por las grandes y gruesas.

Respecto a materiales fosilizados, Sutton y Reinhard (1995) realizaron un estudio sobre 155 coprolitos encontrados en *Antelope House*, un yacimiento prehistórico en el Cañón de Chelly en Arizona. El estudio arrojó como resultado una interpretación de las diferencias entre coprolitos basada en la alimentación.

2.3.6 Aplicaciones en bioinformática y genética

Tiempos recientes están siendo testigo de un tremendo auge en el interés por la Bioinformática, acompañada por la biología molecular, ciencias de la computación, matemáticas y estadística. Tal aumento ha sido acelerado por la siempre creciente base de datos genómica y proteica, que son por sí mismas resultado de un grandísimo avance en las técnicas de secuenciación del ADN, medidas de expresión de los genes y compresión de las estructuras macromoleculares. La estadística ha resultado relevante en el estudio de la expresión de los genes. Los genes contenidos en el ADN de cada célula proporcionan las plantillas necesarias para la generación de las proteínas implicadas en la mayoría de los procesos estructurales y biomecánicos que tienen lugar en cada uno de nosotros. Sin embargo, aunque la mayoría de las células en los seres humanos contiene todos los complementos genéticos que componen el genoma humano, los genes se expresan de manera selectiva en cada célula dependiendo del tipo de la misma, del tejido y de las condiciones generales tanto dentro como fuera de la célula. La biología molecular ha puesto de manifiesto que la mayoría de los procesos en la vida de una célula están regulados por factores que afectan a la expresión de sus genes.

Como hemos visto, uno de los campos de investigación más activos hoy en día es el que estudia los procesos que regulan la expresión de los genes. Con el fin de almacenar la información relativa a esta área de estudio surgen los microarrays, (Cortesse, 2000). Desde el punto de vista del análisis de datos, una de las características relevantes en este tipo de información es que el número de características de cada instancia (p), supera con creces al número de instancias disponibles (n); conjuntos de datos como este son calificados como *datos de alta dimensionalidad*.

La mayoría de métodos estadísticos clásicos no pueden ser aplicados a este tipo de conjuntos de datos sin ser modificados de forma sustancial. Sin embargo, el análisis de clusters acepta bien tales conjuntos de datos y puede ser empleado para identificar grupos de genes con patrones de expresión similares, y dar respuesta a preguntas como por qué un gen se ve afectado por cierta enfermedad, o qué genes son responsables de enfermedades genéticas hereditarias.

Un ejemplo de aplicación lo encontramos en el trabajo de Selinski e Ickstadt (2008), quienes usaron clustering sobre polimorfismos de nucleótidos simples para detectar diferencias entre enfermedades a nivel genético.

2.4 RESUMEN

El clustering consiste en la exploración de conjuntos de datos. Su objetivo es discernir si pueden o no ser resumidos de manera significativa, en términos de un número relativamente pequeño de grupos o clusters de objetos o individuos, que se parecen unos a otros y que se diferencian de los que se encuentran en otros clusters.

Muchas ramas de la ciencia han hecho uso de las técnicas de clustering, de manera exitosa, para avanzar en sus respectivos campos, y procesar grandes cantidades de datos, cuyo análisis sería impensable afrontar con técnicas tradicionales.

3

CLUSTERING CON RESTRICCIONES

Realizada la introducción sobre los conceptos básicos del clustering, el objetivo de esta sección es definir los conceptos relacionados con el clustering con restricciones, así como dar ejemplos de su aplicación y destacar los beneficios y problemas que presenta su uso. A tal fin tomamos como referencia principal el trabajo de Ian Davidson y Sugato Basu (2007) [3] entre otros.

3.1 MOTIVACIÓN PARA EL CLUSTERING CON RESTRICCIONES

Tal y como hemos estudiado en la Sección 2, los métodos de clustering no supervisado son útiles para dotar de estructura a datos referentes a un área concreta. Un ejemplo de ello lo encontramos en la clasificación de textos; Cohn et al. (2003) [4] afrontan un problema propuesto por Yahoo!, que consiste en, dada una gran cantidad de documentos de texto, agruparlos según una taxonomía en la que los documentos con temáticas similares se encuentren cercanos. Para ello, los métodos de clustering no supervisado resultan de utilidad, ya que la información sobre el problema de la que se dispone inicialmente es limitada. Sin embargo, Wagstaff et al. (2001) mostraron que aplicando clustering no supervisado a ciertos problemas, como el de agrupar datos de GPS de forma que los clusters definan los carriles de una vía, no se obtienen resultados significativos, pues los clusters obtenidos distan mucho de la forma alargada que se esperaría como resultado. Para atajar el problema, introdujeron en el clustering un nuevo elemento, las restricciones a nivel de instancia, que permitían incluir conocimiento sobre los clusters que guiarían los métodos de clustering para obtener los resultados esperados. Bastaba con indicar que los carriles de la vía por la que circulan los vehículos miden cuatro metros de ancho, y por tanto cualquier vehículo que se encuentre a una distancia mayor de 4 metros de otro, en dirección perpendicular a la del desplazamiento, debe ser ubicado en un cluster diferente.

Nos situamos entonces en un nuevo escenario: es posible incorporar información adicional al proceso de clustering, además de la contenida en el propio conjunto de datos, para guiarlo en la formación de la partición y obtener resultados más precisos. Esto sitúa al clustering con restricciones en el marco del aprendizaje semisupervisado, a diferencia de los métodos de clustering tradicionales que se enmarcan en el área del clustering no supervisado.

3.2 DEFINICIÓN DE LAS RESTRICCIONES

El nuevo tipo de información que incorporamos al clustering viene dado en forma de restricciones a nivel de instancia, esto es, especificar si dos instancias del conjunto de datos deben estar en el mismo cluster o, por el contrario, deben estar en clusters separados.

A las restricciones que indican que dos puntos deben ser situados en el mismo cluster se las denomina *Must-link*, y se notan por $ML(x, y)$, donde x e y son dos instancias del conjunto de datos. De manera similar, a las restricciones que especifican lo contrario se las denomina *Cannot-link*, y se notan por $CL(x, y)$. [5]

Aunque pueden parecer simples, las restricciones definidas de la anterior forma poseen propiedades interesantes. Las Restricciones de tipo Must-link son un ejemplo de relación de equivalencia, y por tanto son simétricas, reflexivas y transitivas, formalizando:

Observación 3.1 *Las restricciones de tipo ML son transitivas.* Sean CC_i y CC_j componentes conexas, conectadas mediante restricciones ML, y sean x e y dos instancias en CC_i y CC_j respectivamente. Entonces $ML(x, y) : x \in CC_i, y \in CC_j \rightarrow ML(a, b) \forall a, b : a \in CC_i, b \in CC_j$. [3]

Observación 3.2 *Las restricciones de tipo CL pueden ser (encadenadas).* Sean CC_i y CC_j componentes conexas, conectadas mediante restricciones ML, y sean x e y dos instancias en CC_i y CC_j respectivamente. Entonces $CL(x, y) : x \in CC_i, y \in CC_j \rightarrow CL(a, b) \forall a, b : a \in CC_i, b \in CC_j$. [3]

Un claro ejemplo de uso de las restricciones lo encontramos en casos de aplicación de clustering en los que existen limitaciones en cuanto a las medidas de distancia, como sucedía en el supuesto de los datos GPS. Así, si queremos que las instancias que forman dos clusters estén separadas por una distancia mayor o igual a δ , basta con establecer restricciones de tipo ML entre todas aquellas instancias cuya distancia sea menor que δ . De manera similar, si queremos que el diámetro de los clusters sea como mucho ϵ , debemos establecer un conjunto de restricciones de tipo CL entre todas aquellas instancias que se encuentren a una distancia mayor que ϵ . La Figura 3.1 muestra una representación gráfica de estos dos tipos de restricciones.

3.3 USO DE LAS RESTRICCIONES

Mientras que el aprendizaje completamente supervisado implica conocer la etiqueta asociada a cada instancia, en el aprendizaje semi-supervisado sólo se dispone de un subconjunto de instancias etiquetadas. Por otra parte, en gran cantidad de dominios la información disponible se refiere a relaciones entre instancias, y no a la clase concreta a la que pertenecen las mismas. Es más, en montajes de clustering interactivo, un usuario no experto en el dominio del problema

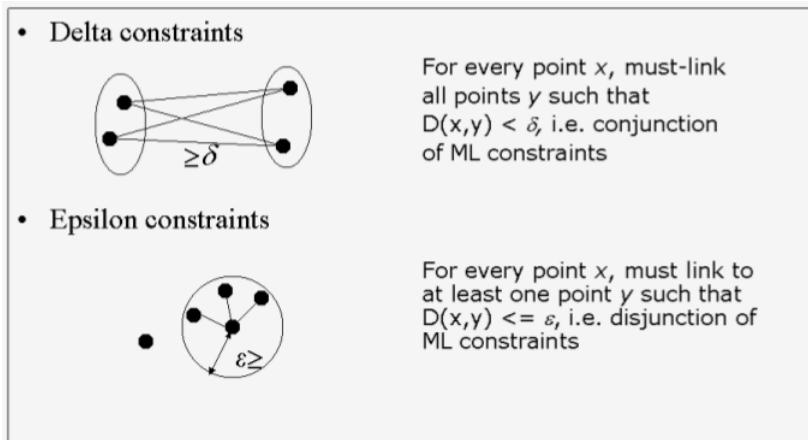


Figura 3.1: Restricciones de tipo *delta* y *epsilon*. [3]

podrá, probablemente, aportar información en forma de restricciones de tipo Must-Link (ML) y Cannot-Link (CL) [4][6], antes que aportar información sobre a qué clase concreta pertenecen ciertas instancias.

Habitualmente, las restricciones se incluyen en los problemas de clustering de dos maneras. Pueden ser empleadas para modificar la regla de asignación de instancias a clusters del método en cuestión, de forma que la solución satisfaga el máximo número de restricciones posible. Alternativamente, cabe la posibilidad de entrenar la función de distancia empleada por el método en base a las restricciones, ya sea antes o durante la aplicación del mismo. En cualquier caso, la fase de inicialización puede tomar en consideración las restricciones, de forma que las instancias asociadas con restricciones Must-Link (ML) serán situadas en el mismo cluster, y aquellas entre las que exista una restricción Cannot-Link (CL), quedarán en clusters diferentes. Basándonos en esta distinción, identificamos dos maneras de aproximar el problema, las basadas en restricciones (*constraint-based*), y las basadas en distancias (*distance-based*).

3.3.1 Métodos basados en restricciones

En los métodos basados en restricciones, el propio método de clustering es modificado de manera que la información disponible se emplea para sesgar la búsqueda y obtener una partición de los datos apropiada.

Existen dos modelos de métodos basados en restricciones: (1) aquellos que fuerzan el cumplimiento de las restricciones, e intentan encontrar la mejor asignación posible que no infrinja ninguna de ellas [7][8], y (2) los que hacen una interpretación relajada de las restricciones [9][10][11][12], permitiéndose incumplir un número mínimo de ellas para optimizar la función objetivo; de esta manera surge un compromiso entre el número de restricciones incumplidas y el valor

de la función objetivo. Existen diversas técnicas para obtener una partición atendiendo a las restricciones:

- Modificar la función objetivo de manera que incluya una penalización por incumplir restricciones. [13] [11]
- Agrupar instancias con información adicional obtenida de una distribución condicional en un espacio auxiliar. [14]
- Forzar el cumplimiento de todas las restricciones modificando la regla de asignación del método. [7]
- Inicializando los clusters en base a restricciones inferidas del conjunto de instancias etiquetadas.[15]

La Figura 3.2 muestra un conjunto de datos junto a sus restricciones asociadas. La Figura 3.3 propone un posible agrupamiento que satisface todas las restricciones.

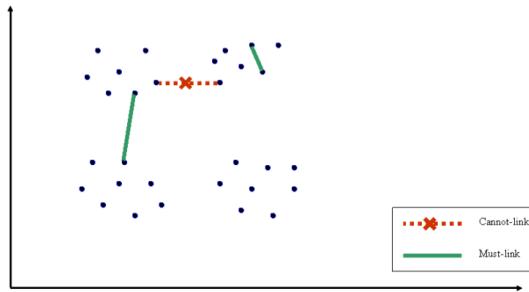


Figura 3.2: Restricciones sobre un conjunto de datos. [3]

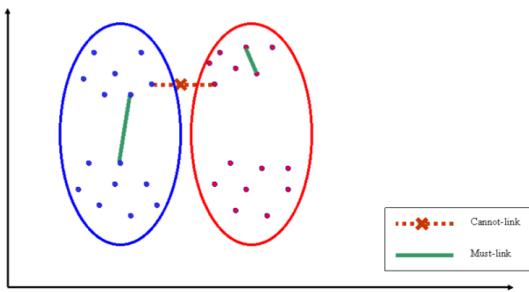


Figura 3.3: Clustering que satisface todas las restricciones. [3]

3.3.2 Métodos basados en distancia

En las aproximaciones basadas en distancias, se emplean métodos de clustering clásicos que hagan uso de una medida de distancia, de forma que dicha medida se modifica para que tenga en consideración las restricciones. En este contexto, satisfacer las restricciones significa que las instancias relacionadas con restricciones Must-Link (ML) se sitúan juntas en el espacio, y las relacionadas mediante Cannot-Link (CL) se encuentran separadas.

La Figura 3.5 muestra un posible agrupamiento basado en una métrica aprendida a partir de las restricciones especificadas en la Figura 3.4. Cabe destacar que en la Figura 3.5 el espacio en el que se encuentran los datos ha sido comprimido en el eje vertical y ensanchado en el eje horizontal para ajustarlo a la métrica de distancia aprendida.

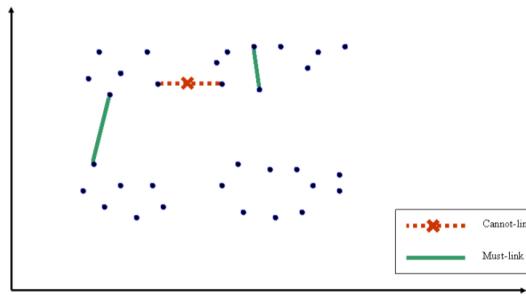


Figura 3.4: Restricciones sobre un conjunto de datos. [3]

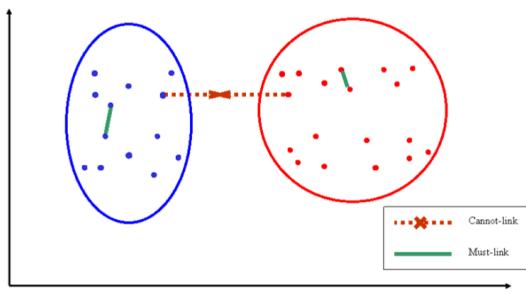


Figura 3.5: Clustering basado en métrica aprendida en base a las restricciones. [3]

3.4 APLICACIONES DEL CLUSTERING CON RESTRICCIONES

Esta sección muestra algunos casos de aplicación en los que el clustering con restricciones ha resultado ser una herramienta más útil que el clustering no supervisado. Para cada caso analizaremos cómo se obtuvieron las restricciones y cómo éstas mejoran los resultados en el clustering resultante.

3.4.1 Aplicaciones en análisis de imágenes

La Figura 3.6 muestra un extracto del conjunto de datos de caras de CMU (Carnegie Mellon University), en el que la tarea es agrupar caras en base a diferentes criterios. En este caso, el objetivo es agrupar las caras según su orientación.



Figura 3.6: Caras de la base de datos de CMU. [3]

El método empleado para extraer las restricciones es uno de los más populares en la literatura: establecer el número de clusters de la partición resultado igual al número de clases en la base de datos, y generar las restricciones a partir de un subconjunto de instancias etiquetadas; esto es, si dos instancias tiene diferentes etiquetas, establecer una restricción Cannot-Link (CL) entre ellas, en caso contrario una de tipo Must-Link (ML). De esta forma, entre las imágenes mostradas en la Figura 3.7 se establecen restricciones Cannot-Link (CL), ya que, aunque pertenecen a la misma persona, no presentan la misma orientación.



Figura 3.7: Restricciones de tipo CL entre caras de la misma persona. [3]

En la Figura 3.8 se muestra otro conjunto de datos de imágenes sobre el que se aplican técnicas de clustering con restricciones. En este caso, la tarea es realizar reconocimiento de objetos para incorporar el método al sistema de navegación del robot Aibo [11]. Para ello se emplean restricciones de distancia de tipo δ y ϵ como las descritas en la Figura 3.1. De esta manera se consiguen clusters bien diferenciados y por tanto útiles para las tareas de búsqueda de caminos que el robot realiza durante la navegación.



Figura 3.8: Método de clustering empleado en el sistema de navegación del robot Aibo. [3][11]

3.4.2 Aplicaciones en análisis de vídeos

Las bases de datos de vídeo son uno de los ejemplos en los que las restricciones pueden ser generadas directamente desde el dominio de datos, especialmente disponiendo de datos espacio-temporales sobre el vídeo [16]. En datos temporalmente sucesivos es posible establecer restricciones de tipo Must-Link (ML) entre grupos de píxeles de fotogramas (*frames*) cercanos en el tiempo. Esto es especialmente útil cuando la tarea es implementar reconocimiento de objetos basado en clustering y segmentación. También es posible añadir restricciones Cannot-Link (CL) a clusters localizados en el mismo fotograma, ya que existe una baja probabilidad de que estén asociados al mismo objeto. De hecho, en el dominio asociado a problemas de análisis de vídeo existen gran variedad de métodos de extracción de restricciones [16], la Figura 3.9 muestra algunos ejemplos.

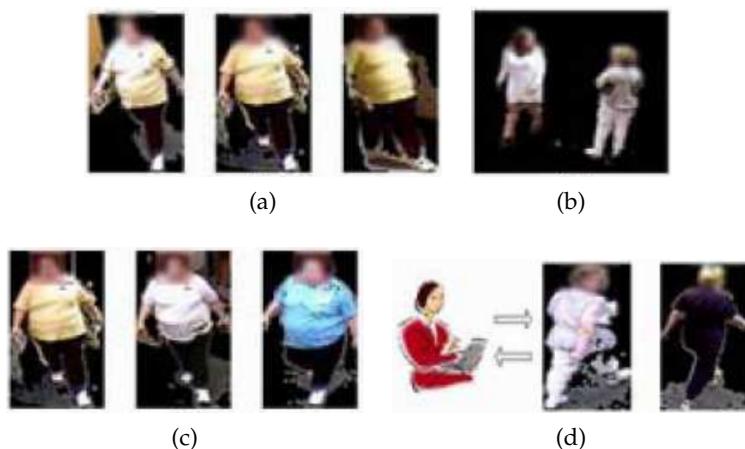


Figura 3.9: Diferentes tipos de restricciones en datos de video. [16] [3]

En la Figura 3.9, la imagen (a) corresponde a restricciones extraídas del seguimiento de una persona durante un periodo de tiempo, la (b) corresponde a restricciones espaciales que asocian dos objetos localizados en el mismo fotograma, la imagen (c) corresponde a restricciones obtenidas mediante reconocimiento facial y la (d) a las proporcionadas por el usuario.

Disponiendo de tantos métodos para extraer restricciones, cabe plantearse: ¿qué sucede si se establecen demasiadas restricciones? ¿Hace esto que el problema esté sobrestringido? En la Sección 3.6 abordaremos estas cuestiones.

3.4.3 Aplicaciones en genética

En clustering de genes basado en microarrays, los genes vienen representados por su perfil de expresión en diferentes experimentos y agrupados empleando diferentes métodos, en este caso métodos de

clustering con restricciones. La Figura 3.10 muestra un ejemplo: se trata de restricciones de tipo Must-Link (ML) que se establecen entre genes en base a los datos de co-ocurrencia almacenados en la base de datos de interacciones de proteínas, que contiene información sobre qué genes (y sus proteínas asociadas) están presentes en los mismos procesos celulares [17]. Esta información puede ser empleada para mejorar los resultados que proporcionan los métodos de clustering. [10]

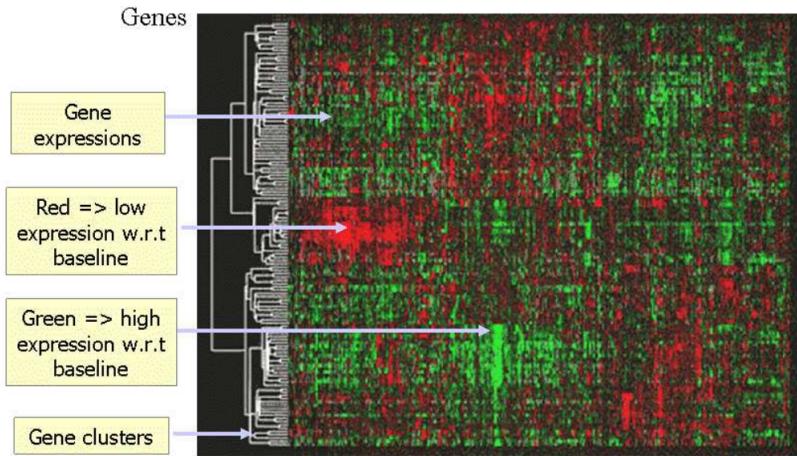


Figura 3.10: Clustering de genes basado en microarrays. [3]

3.4.4 Aplicaciones en análisis de textos

En tareas de clasificación de contenido, el objetivo es dividir de manera automática grandes cantidades de documentos en grupos o clusters. En este caso es posible extraer las restricciones de múltiples recursos auxiliares. Por ejemplo, si dos documentos se encuentran en el mismo directorio se podría inferir una restricción de tipo Must-Link (ML) entre ellos. De esta manera es posible modificar el clustering resultante para que se adapte a un criterio concreto, como por ejemplo crear una jerarquía de documentos semejante a la forma en la que se encuentran organizados en la estructura de directorios de entrada.

3.4.5 Aplicaciones en datos web

El clustering con restricciones resulta de gran utilidad en el procesamiento de datos de búsqueda en páginas web. Aquí, el objetivo es agrupar, de manera automática, los resultados de una consulta ambigua en el motor de búsqueda en clusters de URLs, que se refieran al concepto introducido como consulta en diferentes contextos. En este ámbito es posible extraer las restricciones a partir de búsquedas

realizadas anteriormente por los usuarios, de manera que se establece una restricción de tipo Must-Link (ML) entre URLs visitadas en la misma sesión de usuario. Aplicar clustering utilizando estas restricciones puede ayudar a sesgar el resultado de las búsquedas hacia las preferencias del usuario.

3.4.6 Aplicaciones en datos de audio

En ciertas tareas de análisis de audio, es posible que no se conozca el número de clases de objetos presentes en los datos, si bien las restricciones pueden extraerse directamente del dominio de datos. Esto sucede, por ejemplo, al aplicar clustering para reconocimiento de hablantes en una conversación [18]. En este caso el número de participantes no se conoce a priori, pero es fácil detectar si dos hablantes son diferentes o similares y establecer las restricciones en base a ello.

3.4.7 Aplicaciones en datos de GPS

Tal y como se indicó en el inicio de este capítulo, el clustering con restricciones sobre datos GPS se utiliza para identificar el carril por el que circula cada vehículo, como se muestra en la Figura 3.11. Cada instancia viene representada por la posición que ocupa en la vía en coordenadas cartesianas bidimensionales (x, y), obtenidas en base a los datos GPS. La Figura 3.12 muestra de manera gráfica esta representación de los datos (cabe destacar que múltiples instancias pueden referirse al mismo vehículo en distintos momentos en el tiempo).

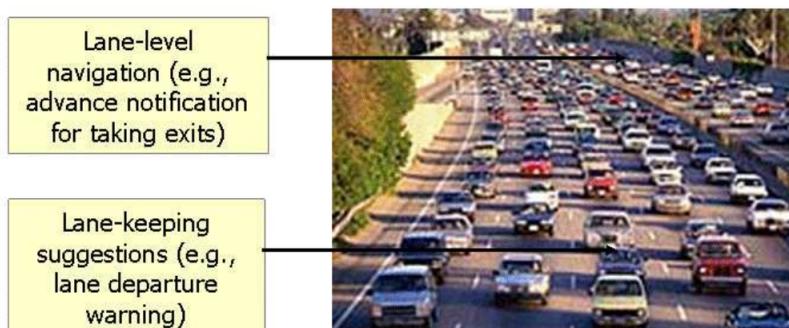


Figura 3.11: Uso de información GPS. [3] [7]

En este dominio, los clusters reales tienen forma alargada en el eje horizontal y se encuentran alineados perpendicularmente a la dirección de desplazamiento. Para lograr que los clusters resultantes tengan esta forma, podemos hacer uso de las restricciones. Estableceremos una restricción de tipo Cannot-Link (CL) entre aquellas instancias separadas más de 4 metros en dirección perpendicular a la del desplazamiento (ya que los carriles tienen una anchura máxima de 4 metros), y Must-Link (ML) entre aquellas instancias que presenten continuidad

en el eje horizontal, puesto que es probable que los vehículos que representan se encuentren en el mismo carril. Este modelo de clustering ha probado ser muy útil en navegación a tiempo real [7], permitiendo notificar al usuario cuando debe cambiar de carril, o cuando no debe abandonarlo.

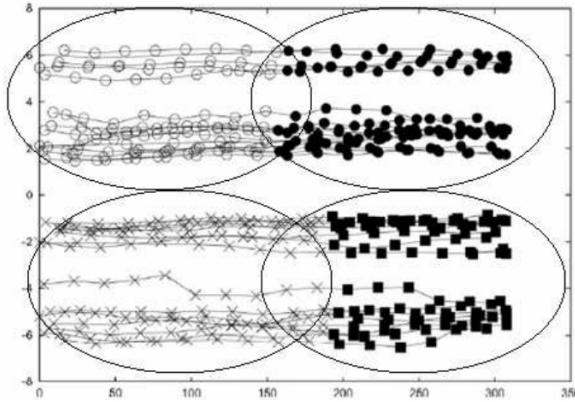


Figura 3.12: Clusters encontrados en datos GPS sin uso de restricciones. [3] [7]

3.5 BENEFICIOS DEL USO DE RESTRICCIONES

Encontramos dos beneficios principales en el uso de restricciones:

- Incremento de la exactitud en las predicciones de las etiquetas al generar restricciones en base a un subconjunto de instancias etiquetadas.
- Obtención de clusters con geometría adaptable a cada problema.

A continuación se analizan estos dos beneficios:

Dado $X = \{x_1 \dots x_u\}$, un gran conjunto de instancias no etiquetadas, y $L = \{(x_{u+1}, y_{u+1}) \dots (x_{u+l}, y_{u+l})\}$, un pequeño conjunto de instancias etiquetadas, es común escoger dos elementos de L (con reemplazamiento) y establecer una restricción ML entre ellos si pertenecen a la misma clase o, en caso contrario, una de tipo CL. Un método apropiado para evaluar los resultados ofrecidos por un método de clustering es medir el nivel de exactitud de éste a la hora de predecir las etiquetas del conjunto X . Esto normalmente requiere que se especifique el número de clusters deseados igual al número de clases conocidas en X ($K = K^*$). Para medir la exactitud se emplean métodos como *Rand index* [19].

El trabajo de Wagstaff y Cardie [5], en el que generaban las restricciones de la manera descrita anteriormente, demostraba que, cuando se realiza un promedio de la exactitud de las predicciones obtenidas con algoritmos de clustering con restricciones, variando estas últimas

entre experimentos, se obtienen resultados hasta un 20 % mejores que con las técnicas clásicas.

Observación 3.3 *El uso de restricciones, en promedio, incrementa la precisión. El rendimiento de un método al predecir etiquetas aumenta cuando se promedia empleando numerosos conjuntos de restricciones diferentes.* [3]

Esta regla, sin embargo, no es cierta en todos los casos, pues en conjuntos de datos como *Tic-Tac-Toe Endgame*, no se consigue ningún incremento en las predicciones sea cual sea el número de restricciones empleadas. La explicación dada por los autores citados para estas excepciones se basa en que establecer $K = K^*$ no es apropiado en este caso.

El otro beneficio que reporta el uso de restricciones es la posibilidad de obtener clusters con la geometría deseada, como el ejemplo de aplicar clustering a datos GPS, analizado en la Sección 3.4.7 de este trabajo.

3.6 PROBLEMAS DEL USO DE RESTRICCIONES

Aunque, tal y como hemos comprobado, la incorporación de restricciones a los métodos de clustering reporta beneficios en algunas aplicaciones, existen dos inconvenientes principales que se exponen a continuación, así como posibles soluciones a los mismos.

3.6.1 *El problema de la factibilidad*

La introducción de restricciones en el clustering cambia el problema al que éste da solución, que pasa a ser: *Encontrar la mejor partición que satisfaga todas las restricciones*. De esta manera, si aquellas no están bien especificadas o si los métodos de extracción son inadecuados, podemos encontrar que las restricciones se contradicen, lo que deriva en que no existe una asignación de instancias a clusters que las satisfaga todas. Por ejemplo, no existe asignación que satisfaga las restricciones $ML(x_1, x_2)$ y $CL(x_1, x_2)$, independientemente del valor de K . Lo mismo sucede para $K = 2$, y las restricciones $CL(x_1, x_2)$, $CL(x_2, x_3)$ y $CL(x_1, x_3)$. Formalizando, el problema de la factibilidad para problemas de clustering (no jerárquico) con restricciones viene definido por:

Definición 3.1 *Problema de la factibilidad para clustering con restricciones:* Dado un conjunto de datos X , un conjunto de restricciones R , un umbral superior K_l y un umbral superior K_u para el número de clusters resultantes, ¿Existe una partición de X en bloques tal que $K_l \leq K \leq K_u$ y todas las restricciones en R se satisfacen? [11] [3]

La complejidad teórica del problema dependerá del tipo de restricciones que se combinen en él. La Tabla 3.1 presenta, de manera resumida, la complejidad esperada en cada caso.

Complejidad del clustering con restricciones	
Restricciones	Complejidad
Restricciones δ	P
Restricciones ϵ	P
ML y δ	P
ML y ϵ	NP-completo
ϵ y δ	P
CL y otra	NP-completo

Tabla 3.1: Complejidad del problema del clustering en función del tipo de restricciones [3]

Tal y como queda reflejado en la Tabla 3.1, la utilización de restricciones Cannot-Link (CL) eleva el nivel de complejidad del clustering a NP-completo y, por tanto, el problema del clustering con restricciones es intratable. De manera intuitiva puede entenderse fácilmente que, si encontrar una sola partición que satisfaga las restricciones es un problema complejo, más complejo es aún encontrar la mejor.

Observación 3.4 *Saber que existe una solución factible no nos ayuda a encontrarla. Las consecuencias de este resultado sobre la complejidad del clustering con restricciones implican que, aun en caso de que exista una partición factible, no será fácil de encontrar, hablando en términos de complejidad algorítmica. [3]*

Los autores Wagstaff [20] y Davidson y Ravi [6] muestran que aun especificando el número de clusters de salida igual al de clases verdaderas ($K = K^*$), cosa que garantiza que existe una solución factible, algoritmos simples como la adaptación de K-medias (KM, Apéndice A) al clustering con restricciones (COP-K-means [7], Sección 4.2), pueden no converger debido al problema de la factibilidad.

3.6.2 El problema de la utilidad de conjuntos de restricciones

En el clustering con restricciones se asume que éstas son indicaciones que guían al algoritmo para encontrar la partición de los datos deseada. Entonces, está justificado pensar que, de cuanta más información adicional (restricciones) dispongamos, más cercano estará el resultado que obtengamos al que buscamos, tal y como la Observación 3.3 afirmaba. Sin embargo, y a pesar de lo dispuesto en dicha

observación, existen casos en los que, aun generando las restricciones sin ruido y en base a las etiquetas verdaderas, existen conjuntos de restricciones que, lejos de mejorar los resultados, los empeoran considerablemente [21]. Esto parece estar en desacuerdo con la Observación 3.3, sin embargo, recordemos que en ella se hace referencia al caso medio, y no a casos particulares.

Observación 3.5 *Conjuntos de restricciones particulares pueden causar efectos adversos. Algunos conjuntos de restricciones generados en base a las etiquetas verdaderas y libres de ruido pueden resultar en una pérdida de precisión a la hora de predecir esas mismas etiquetas. [3]*

3.6.3 Soluciones al problema de la factibilidad

El problema de la factibilidad puede ser abordado de varias maneras. La más inmediata quizá sea mantener el número de restricciones bajo, en proporción al número de instancias totales, para minimizar la probabilidad de que surjan inconsistencias. Sin embargo, no poder aumentar el número de restricciones si el problema lo requiere no es el escenario ideal. Por ello se debe poner interés en analizar cuando un problema pasa a estar sobrerestringido, ya que, como hemos estudiado en la Sección 3.6.2, incluso generando las restricciones en base a las etiquetas verdaderas, algoritmos como COP-K-medias (Sección 4.2) dejan de ser efectivos conforme aumenta el número de restricciones a satisfacer, incluso reiniciando de manera aleatoria el algoritmo varias veces.

El fenómeno de la sobrerestricción de problemas mediante el uso de restricciones Cannot-Link (CL) está íntimamente relacionado con el problema del coloreado de grafos; de hecho, ha sido demostrado que éste es equivalente al problema del clustering con restricciones CL [22]. Así, encontramos que resolver un problema con restricciones CL mediante algoritmos como COP-K-medias es, a efectos prácticos, resolver el problema del coloreado de grafos.

Observación 3.6 *El problema del clustering con restricciones CL es análogo al problema del coloreado de grafos. [3]*

Este resultado permite trasladar muchas de las propiedades del problema del coloreado de grafos al problema de clustering con restricciones. Por ejemplo, el teorema de Brook establece que el coloreado de grafos es sencillo cuando el número de colores disponibles (K en nuestro caso), es mayor que el máximo grado del grafo.

Observación 3.7 *El teorema de Brook es aplicable al problema del clustering con restricciones. Si $K >$ (Mayor número de restricciones CL sobre una instancia), entonces siempre existirá una partición factible. [3]*

Con esto, y aunque la Observación 3.4 indique lo contrario, cuando el problema del clustering cumple la condición expuesta en la Observación 3.7, podemos garantizar que siempre se encontrará una solución al problema en tiempo polinómico. Para asegurar la condición de Brook, es posible construir el conjunto de restricciones de manera que ninguna instancia tome partido en más de K restricciones Cannot-Link (CL). [22]

3.6.4 Soluciones al problema de la utilidad de conjuntos de restricciones

La solución al problema es simple: identificar aquellos conjuntos de restricciones verdaderamente útiles. Sin embargo, esto involucra aplicar algún tipo de métrica que permita evaluar cuándo un conjunto de restricciones dado cumple esta condición. A tal fin, Davidson, Wagstaff y Basu propusieron dos medidas: **informatividad** y **coherencia**.

La **informatividad** es una medida referida a la cantidad de información presente en el conjunto de restricciones que el algoritmo no puede determinar por sí mismo. Por ejemplo, en la Figura 3.13, un algoritmo como COP-K-medias (Sección 4.2) se vería inclinado a agrupar instancias cercanas en el espacio y colocar en clusters separados aquellas que se encuentren lejanas; sin embargo, las restricciones sesgan el espacio de soluciones evitando que esto suceda.

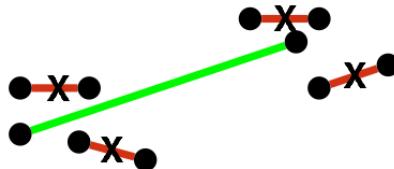


Figura 3.13: Ejemplo de conjunto de restricciones informativo. [3]

La informatividad se estima utilizando el conjunto de restricciones como un conjunto de test, de manera que se mide la habilidad del algoritmo para predecir las restricciones presentes en él. Formalizando, dado un conjunto de restricciones R y un algoritmo A , obtenemos la partición P_A aplicando el algoritmo al conjunto de datos de entrada especificando el conjunto de restricciones vacío. Calculamos entonces la fracción de las restricciones incumplidas por P_A [3]:

$$I_A(R) = \frac{1}{|R|} \left[\sum_{r \in R} \text{unsat}(r, P_A) \right] \quad (3.1)$$

Por otra parte, la **coherencia** mide el grado de concordancia dentro del propio conjunto de restricciones respecto a una métrica dada (D). Por ejemplo, la Figura 3.14 muestra dos restricciones paralelas y muy cercanas, pero de distinto tipo. Es en casos como este en los que se da una contradicción, ya que las restricciones Must-Link (ML) indican que la distancia entre las instancias involucradas en ellas es pequeña, mientras que las de tipo Cannot-Link (CL) deben indicar lo contrario.



Figura 3.14: Ejemplo de conjunto de restricciones contradictorio. [3]

Entonces, la medida de coherencia viene dada por el grado de solapamiento que presentan las restricciones al interpretarlas como vectores en el espacio y proyectarlas sobre uno de los ejes, tal y como se muestra en la Figura 3.15.

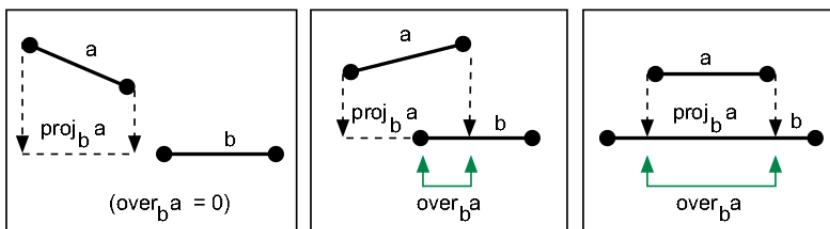


Figura 3.15: Representación de la medida de coherencia. [3]

3.7 RESUMEN

El clustering con restricciones incorpora nueva información al problema del clustering original, las restricciones. Dicha información viene dada en forma de restricciones, ya sean Must-Link (ML), Cannot-Link (CL), o restricciones de distancia, y es utilizada para guiar al método que apliquemos, al conjunto de datos en cuestión, en la búsqueda de la partición resultado.

Los métodos de clustering derivados de este concepto han demostrado ser de gran utilidad en múltiples ámbitos, así como también presenta problemas que pueden ser subsanados estudiando en profundidad las restricciones a emplear para resolver cada problema.

4

ALGORITMOS DE CLUSTERING CON RESTRICCIONES

Una vez introducido el problema del clustering con restricciones, pasamos a profundizar en los métodos para su aplicación. La siguiente sección presenta 5 algoritmos de clustering con restricciones, cuyos resultados serán expuestos más tarde, en la Sección 6.

4.1 FORMALIZACIÓN DEL PROBLEMA

Definido ya el problema del clustering con restricciones en la Sección 3, especificamos la manera de notar sus elementos, de forma que sea sencillo referirse a ellos.

- Notaremos con X a la matriz de $n \times p$ que contiene el conjunto de datos de entrada.
- Notaremos con x_i t.q. $i \in \{1, \dots, n\}$ a cada instancia de X , por lo que x_i es un vector en el espacio \mathbb{R}^p ($x_i \in \mathbb{R}^p$).
- Notaremos con R al conjunto de restricciones, tanto las de tipo ML como las CL, es decir $R = ML \cup CL$.
- Notaremos con K el número de clusters de la partición resultante.
- Notaremos con C el conjunto de clusters, y con c_i t.q. $i \in \{1, \dots, K\}$ a cada uno de ellos, por tanto $C = \{c_1, \dots, c_K\}$.
- Notaremos con V la matriz de $K \times p$ que almacena el conjunto de centroides asociados a los clusters, de manera que $v_i \in \mathbb{R}^p$ corresponde al cluster c_i .

Cabe destacar que los elementos y parámetros particulares de cada algoritmo serán definidos en la sección correspondiente al mismo, así como que no todos los elementos expuestos anteriormente son comunes a todos los algoritmos; si bien sí que lo son a la mayoría de ellos.

4.2 COP-K-MEANS (CONSTRAINED K-MEANS)

El algoritmo K-medias (KM, Apéndice A) es uno de los más básicos para aplicar clustering. Así, el algoritmo COP-K-medias (COP-K-means) es la adaptación inmediata de KM al clustering con restricciones. Para realizar un estudio detallado sobre el mismo tomaremos como base el trabajo de Wagstaff et al. (2001) [7].

El cambio más notable que supone COP-K-medias respecto al tradicional K-medias, consiste en modificar la regla de asignación de instancias a clusters de este último, para comprobar que dicha asignación no viola ninguna restricción. De esta manera, en cada iteración se intenta asignar cada instancia x_i al cluster más cercano c_j . Esta asignación sólo se llevará a cabo si, como hemos dicho, no se viola ninguna restricción. Si existe una instancia x_{ML} que debe ser asignada al mismo cluster que x_i , pero ya ha sido incluida en otro cluster, o existe una instancia x_{CL} en c_j que no puede ser agrupada junto a x_i , entonces x_i no puede ser asignado a c_j . El proceso continúa hasta encontrar una asignación legal para x_i , en caso de que no se encuentre se devuelve la partición vacía como resultado. Así el algoritmo proporciona una partición de X que cumple necesariamente todas las restricciones especificadas en R . El Algoritmo 1 corresponde al pseudocódigo asociado a COP-K-medias [7]:

Algoritmo 1: COP-K-medias

Entrada: Conjunto de datos X , conjunto de restricciones R , número de clusters K .

Salida: Partición P del conjunto de datos X .

función COP-K-medias(X, R, K) **begin**

1. Sean $V = \{v_1, \dots, v_K\}$ los centroides iniciales
2. Asignar cada instancia $x_i \in X$, al cluster c_j asociado al centroide más cercano v_j tal que $\text{ViolaRestriccion}(x_i, c_j, R) = \text{falso}$. Si no existe $c \in C$ t.q. $\text{ViolaRestriccion}(x_i, c, R) = \text{falso}$, **return** \emptyset .
3. Para cada cluster c_i , actualizar su centroide v_i realizando un promedio de todas las instancias x_i asignadas a él.
4. Iterar entre (1.) y (2.) hasta converger.
5. **return** C

end

Entrada: Instancia x , cluster c , conjunto de restricciones R

función ViolaRestriccion(x, c, R) **begin**

1. Para cada $(x, x_{ML}) \in ML$, si $x_{ML} \notin R$ **return true**.
2. Para cada $(x, x_{CL}) \in CL$, si $x_{CL} \notin R$ **return true**.
3. En otro caso, **return false**.

end

Existen multitud de criterios de convergencia estandarizados, aunque es común emplear uno adaptado al problema particular que queramos solucionar. El más extendido consiste en calcular la diferencia de la posición de los centroides entre dos iteraciones sucesivas, de forma que cuando ésta sea menor que un umbral dado detenemos el proceso de iteración.

4.3 CEKM (CONSTRAINED EVIDENTIAL K-MEANS)

Tal y como indican Violaine et al. (2012) [23], cuyo trabajo es el fundamento de la siguiente sección, para comprender el algoritmo Constrained Evidential K-means (CEKM), primero es necesario realizar una introducción al algoritmo K-medias difuso (*Fuzzy K-means*, FKM). En él, cada instancia puede pertenecer a uno o más clusters, con diferentes grados de pertenencia. La matriz que almacena esta información, es decir, la partición difusa, se nota con U , y se calcula minimizando la siguiente función:

$$\sum_{j=1}^K u_{ij} \quad t.q. \quad u_{ij} \in [0, 1] \forall i, j \quad (4.1)$$

Donde u_{ij} representa el grado de pertenencia de la instancia i al cluster j , y K es el número de clusters. Sin embargo, este método puede producir resultados contraintuitivos cuando los datos a los que se aplica son ruidosos o presentan instancias aisladas (*outliers*).

El clustering evidencial (*evidential clustering*) da solución a los problemas que presenta el algoritmo FKM, introduciendo el concepto de partición de creencia (*credal partition*), que extiende los conceptos existentes de particiones fuertes, difusas y probabilísticas. De esta forma, en una partición de creencia, se asigna a cada instancia una masa de creencia (*mass of belief*), no sólo para un único cluster, sino para cualquier conjunto de los mismos. El método CEKM combina las ventajas del uso de las restricciones con las del uso de funciones de creencia.

4.3.1 Funciones de creencia

La teoría de la evidencia de Dempster-Shafer ofrece un marco teórico para trabajar con información parcial y no completamente fiable; tomamos de ella los conceptos relativos a las funciones de creencia.

Consideremos la variable c que toma valores en el conjunto finito $C = \{c_1 \dots c_K\}$. El conocimiento parcial sujeto al valor real que adopta c puede ser representado mediante una función de masa m , que es una aplicación de C al intervalo $[0, 1]$:

$$\sum_{A \subseteq C} m(A) = 1 \quad (4.2)$$

A los subconjuntos A de C que cumplen que $m(A) > 0$ se los denomina conjuntos focales (*focal sets*) de m . El valor del conjunto focal $m(A)$ se interpreta como la fracción de una unidad de masa de creencia que está asignada a A y que no puede ser asignada a ningún otro subconjunto de A . Si el único conjunto focal es C , nos encontramos en el caso de completa ignorancia sobre los datos. Por el contrario, si la masa de creencia se asigna a un único elemento de C , estaríamos en el caso de certeza absoluta.

Se dice que una función de masa m está normalizada si $m(\emptyset) = 0$. Sin embargo, bajo la hipótesis de mundo abierto, una función de masa en la que $m(\emptyset) > 0$ se interpreta como la cantidad de creencia que se le asigna a la hipótesis de que el verdadero valor de c puede no encontrarse en C .

Dada una función de masa m , podemos definir una función de plausibilidad $pl : 2^C \rightarrow [0, 1]$ y una función de creencia $bel : 2^C \rightarrow [0, 1]$ de la siguiente manera:

$$pl(A) = \sum_{B \cap A \neq \emptyset} m(B) \quad \forall A \subseteq C \quad (4.3)$$

y

$$bel(A) = \sum_{B \subseteq A, B \neq \emptyset} m(B) \quad \forall A \subseteq C \quad (4.4)$$

De esta forma, las funciones pl y bel están relacionadas como sigue:

$$pl(A) = 1 - m(\emptyset) - bel(\bar{A}) \quad (4.5)$$

Donde \bar{A} representa el complemento de A . La cantidad $bel(A)$ se interpreta como el grado de creencia en A , tomando en consideración la masa de creencia asignada a A y a los subconjuntos no vacíos de A . Por el contrario, $pl(A)$ mide hasta qué punto es erróneo no creer en \bar{A} .

Con el objetivo de tomar decisiones en base al valor de c , es posible transformar la función de masa en una distribución de probabilidad pignística, definida, para una función de masa normalizada, como:

$$BetP(c) = \sum_{c \in A} \frac{m(A)}{|A|} \quad \forall c \in C \quad (4.6)$$

4.3.2 FKM (fuzzy K-means) y sus variantes

Cada cluster $c_j \in C$ con $j \in \{1, \dots, K\}$ está representado por un vector $v_j \in \mathbb{R}^p$, esto es, un centroide. Además, definimos V como la matriz compuesta por todos los centroides, y $U = (u_{ij})$ como la partición difusa que contiene los grados de pertenencia de cada instancia de X a cada cluster. El algoritmo Fuzzy K-means (FKM) calcula las matrices U y V de manera que minimiza (sujeto a las Ecuaciones 4.1 y 4.2) la siguiente función:

$$J_{FKM}(U, V) = \sum_{i=1}^n \sum_{j=1}^K u_{ij}^\beta d_{ij}^2 \quad (4.7)$$

Donde d_{ij} representa la distancia Euclídea entre el objeto x_i y el centroide v_j , y donde $\beta > 1$ es el exponente que controla el grado de difusión de la partición. La función objetivo se minimiza mediante un

algoritmo iterativo que optimiza los centroides y los grados de pertenencia de manera alterna. El algoritmo empieza con una asignación inicial sobre la que realiza modificaciones hasta que converge.

Para detectar datos ruidosos u outliers empleamos el algoritmo NC *Noise-Clustering*. Este método consiste en añadir a los K clusters iniciales uno adicional llamado “cluster ruidoso”, asociado a una distancia fija ρ respecto a todos los objetos. El parámetro ρ controla la cantidad de datos que serán considerados como outliers. La pertenencia u_{i*} de un objeto i al cluster ruidoso se calcula como:

$$u_{i*} = 1 - \sum_{j=1}^K u_{ij} \quad i = 1, \dots, n \quad (4.8)$$

Por tanto, la función objetivo que minimiza el algoritmo NC no es más que una combinación del cálculo de pertenencia de objetos al cluster ruidoso y de la que minimizaba el algoritmo FKM:

$$J_{NC}(U, V) = \sum_{i=1}^n \sum_{j=i}^K u_{ij}^\beta d_{ij}^2 + \sum_{i=1}^K \rho^2 u_{i*}^\beta \quad (4.9)$$

4.3.3 El algoritmo EKM (Evidential K-means)

Es posible obtener una versión credibilística del algoritmo NC reemplazando la matriz asociada a la partición difusa U con una partición de creencia, que notaremos con M . En este contexto, el conocimiento parcial asociado a la pertenencia de un objeto a una clase viene representado por una función de masa aplicada al conjunto C de posibles clases. Por tanto, la masa de creencia puede ser asignada a cualquier subconjunto A de C , y no sólo a elementos únicos de C . Este esquema hace posible modelar una amplia variedad de circunstancias, que van de la completa ignorancia sobre el conjunto de datos hasta la completa certeza sobre el mismo.

Para ilustrar estas ideas se propone el siguiente ejemplo: consideramos un conjunto de cuatro objetos que deben ser clasificados en dos clases. La Tabla 4.1 contiene la partición de creencia asociada a estos datos. La clase del primer objeto es conocida con certeza, ya que su masa de creencia está asignada a un solo elemento de C . Por el contrario, la clase del segundo objeto es completamente desconocida. La masa de creencia del tercer objeto está repartida entre dos elementos de C . Por tanto, tenemos conocimiento probabilístico sobre la clase a la que pertenece. El último objeto representa un outlier, ya que su masa de creencia está asignada al conjunto vacío.

Evidential K-means (EKM) es uno de los algoritmos que opera con una partición de creencia obtenida en base a los datos. Si tomamos m_{ij} como el grado de creencia de que el objeto x_i pertenece al subconjunto $A_j \subseteq C$, obtener una partición de creencia implica determinar, para cada x_i , las cantidades $m_{ij} = m_i(A_j) \quad \forall A_j \neq \emptyset, A_j \subseteq C$. De esta

Ejemplo de partición de creencia				
A	$m_1(A)$	$m_2(A)$	$m_3(A)$	$m_4(A)$
\emptyset	0	0	0	1
$\{c_1\}$	1	0	0,3	0
$\{c_2\}$	0	0	0,7	0
C	0	1	0	0

Tabla 4.1: Ejemplo de partición de creencia [23]

manera, cuando la distancia d_{ij} entre x_i y A_j es alta (baja), m_{ij} será un valor bajo (alto).

Tal y como sucedía en Fuzzy K-means (FKM), cada clase c_l está representada por un centroide $v_l \in \mathbb{R}^p$. Entonces, para cada subconjunto A_j de C distinto de \emptyset se calcula su centroide \bar{v}_j como el baricentro de los centros asociados a las clases presentes en A_j :

$$\bar{v}_j = \frac{1}{|A_j|} \sum_{l=1}^K S_{lj} v_l \quad (4.10)$$

donde el valor S_{lj} viene definido por:

$$S_{lj} = \begin{cases} 1 & \text{si } c_l \in A_j \\ 0 & \text{otro caso} \end{cases} \quad (4.11)$$

La distancia d_{ij} entre la instancia x_i y el conjunto focal A_j se calcula como:

$$d_{ij} = ||x_i - \bar{v}_j|| \quad (4.12)$$

Entonces, el algoritmo EKM calcula las matrices M y V tratando de minimizar un criterio similar al del algoritmo NC:

$$J_{EKM}(M, V) = \frac{1}{2^c n} \sum_{i=1}^n \sum_{A_j \neq \emptyset} |A_j|^\alpha m_{ij}^\beta d_{ij}^2 + \sum_{i=1}^n \rho^2 m_{i\emptyset}^\beta \quad (4.13)$$

sujeto a las restricciones $m_{ij} \geq 0 \ \forall i, j$, y $m_{i\emptyset} \geq 0 \ \forall i$, y:

$$\sum_{j/A_j \subseteq C, A_j \neq \emptyset} m_{ij} + m_{i\emptyset} = 1 \quad \forall i = 1, n \quad (4.14)$$

Donde $m_{i\emptyset}$ denota la cantidad de masa de creencia de la instancia x_i asignada al conjunto vacío. El parámetro ρ representa la distancia de cualquier objeto al conjunto vacío, y el parámetro α se introduce para controlar la penalización por asignar objetos a conjuntos con alta cardinalidad.

Gracias a la restricción expuesta en la Ecuación 4.14, podemos obtener equivalencia entre los algoritmos NC y EKM. El algoritmo EKM

asigna una gran masa de creencia al conjunto vacío para un objeto dado cuando éste se encuentra lejos de todos los subconjuntos A_j .

Como en FKM y NC, la partición de creencia se calcula aplicando un proceso de optimización iterativo, que actualiza las masas y los centroides de forma alterna. La regla de optimización de M es muy similar a su homóloga en NC, excepto por el número de valores m_{ij} a calcular, que en este caso es 2^c , en lugar de los $K + 1$ grados de pertenencia que eran necesarios en NC. De esta manera, la función de masa queda definida como:

$$m_{ij} = \frac{|A_j|^{-\alpha/(\beta-1)} d_{ij}^{-2/(\beta-1)}}{\sum_{A_l \neq \emptyset} |A_l|^{-\alpha/(\beta-1)} d_{ij}^{-2/(\beta-1)} + \rho^{-2/(\beta-1)}} \quad (4.15)$$

y

$$m_{i\emptyset} = 1 - \sum_{A_j \neq \emptyset} m_{ij} \quad i = 1, n \quad (4.16)$$

Por otra parte, la regla de actualización de los centroides resulta un poco más compleja. Requiere resolver un sistema de ecuaciones lineal en cada paso del proceso, en el que cada columna de V es la solución para un sistema lineal de K ecuaciones y K incógnitas. Tomamos B como la matriz de tamaño $(K \times p)$ definida por:

$$B_{lq} = \sum_{i=1}^n X_{iq} \sum_{A_j \ni c_l} |A_j|^{\alpha-1} m_{ij}^\beta \quad l = 1, K \quad q = 1, p \quad (4.17)$$

y la matriz H de tamaño $(K \times K)$ como:

$$H_{lt} = \sum_i \sum_{A_j \ni \{c_t, c_l\}} |A_j|^{\alpha-2} m_{ij}^\beta \quad t, l = 1, K \quad (4.18)$$

de forma que V es la solución del sistema de ecuaciones lineal $H \times V = B$, que puede ser resuelto mediante técnicas estándar.

4.3.4 Incorporación de restricciones a EKM

Una vez definidos los elementos que conforman el algoritmo EKM, debemos incorporar las restricciones a nivel de instancia al marco de las funciones de creencia, para integrarlas en el cálculo de la partición de creencia.

Siendo x_i y x_j dos instancias, podemos calcular la función de masa conjunta en el producto cartesiano $C \times C = C^2$ conociendo sus funciones de masa particulares m_i y m_j :

$$\begin{aligned} m_{i \times j}(A \times B) &= m_i(A)m_j(B) \quad A, B \subseteq C, A \neq \emptyset, B \neq \emptyset \\ m_{i \times j}(\emptyset) &= m_i(\emptyset) + m_j(\emptyset) - m_i(\emptyset)m_j(\emptyset) \end{aligned} \quad (4.19)$$

Conociendo $m_{i \times j}$ es posible calcular la plausibilidad asociada a que los objetos x_i y x_j pertenezcan a la misma clase que, en el espacio C^2 , corresponde al subconjunto $\theta = \{(c_1, c_1), (c_2, c_2), \dots, (c_K, c_K)\}$. El caso contrario corresponde al complemento de θ , es decir, $\bar{\theta}$:

$$\begin{aligned} pl_{i \times j}(\theta) &= \sum_{A \cap B \neq \emptyset} m_i(A)m_j(B) \\ pl_{i \times j}(\bar{\theta}) &= 1 - m_{i \times j}(\emptyset) - \sum_{l=1}^K m_i(\{c_l\})m_j(\{c_l\}) \end{aligned} \quad (4.20)$$

4.3.5 Función objetivo de CEKM

Asumimos ahora que la partición de creencia es desconocida, y que disponemos del conjunto de restricciones. En tal caso será necesario buscar una partición de creencia que considere las similitudes y diferencias obtenidas en base a los datos y a las restricciones. Para ello debemos buscar que $pl_{i \times j}(\theta)$ sea tan bajo como sea posible si $(x_i, x_j) \in CL$, así como que $pl_{i \times j}(\bar{\theta})$ también lo sea si $(x_i, x_j) \in ML$. A tal fin, integramos una penalización en el criterio de optimización de EKM de la siguiente manera:

$$J_{CONST} = \frac{1}{|R|} \left[\sum_{(x_i, x_j) \in ML} pl_{i \times j}(\bar{\theta}) + \sum_{(x_i, x_j) \in CL} pl_{i \times j}(\theta) \right] \quad (4.21)$$

De esta forma, la función objetivo a minimizar pasa a ser:

$$J_{CEKM}(M, V) = (1 - \xi)J_{EKM}(M, V) + \xi J_{CONST} \quad (4.22)$$

donde el parámetro $\xi \in [0, 1]$ se utiliza para controlar el compromiso entre las restricciones y el modelo geométrico asociado a la métrica de distancia.

4.3.6 Proceso de optimización de CEKM

De igual forma que en FKM, NC y EKM, el modelo de optimización de CEKM consiste en actualizar M y V de forma alterna. Cabe destacar que el término de penalización añadido a CEKM no depende de los centroides de los clusters, y por tanto se puede aplicar el mismo modelo de actualización que en EKM (Ecuaciones 4.17 y 4.18). Generalmente el problema es mucho más complejo para las masas de creencia. Sin embargo, fijando $\beta = 2$, la Ecuación 4.22, que describe la función objetivo, pasa a ser cuadrática respecto a m_{ij} . Con esto, y como las restricciones son lineales, se pueden emplear algoritmos de programación cuadrática para resolver el problema de la actualización de las masas de creencia. El proceso de cálculo asociado a CEKM queda resumido en el Algoritmo 2.

Algoritmo 2: Constrained Evidential K-means (CEKM)

Entrada: Conjunto de datos X , conjunto de restricciones R , número de clusters resultantes K

Salida: Partición de creencia M , centroides V

función CEKM(X, R, K) **begin**

1. Sean $V = \{v_1, \dots, v_K\}$ los centroides iniciales.
2. Actualizar las masas (M) resolviendo el problema de programación cuadrática definido por 4.22 sujeto a 4.14
3. Actualizar los centroides (V) resolviendo el sistema de ecuaciones lineal definido por 4.17 y 4.18 con $\beta = 2$
4. Iterar entre (2.) y (3.) hasta que no haya cambios significativos en V .
5. **return** M, V

end

En la mayoría de las ocasiones se requiere como resultado de un algoritmo de clustering una partición fuerte del conjunto de datos, y no una partición de creencia. Podemos obtener una partición difusa calculando la probabilidad pignística $BetP_i(\{c_j\})$ en base a cada función de masa m_i aplicando la Ecuación 4.6, e interpretar este valor como el grado de pertenencia del objeto i al cluster j . Una vez obtenida la partición difusa, podemos procesarla como precise el problema para obtener la partición fuerte. La manera más común de hacerlo es asignar cada instancia al cluster para el que presente un mayor grado de pertenencia.

Otra manera de extraer información de la partición de creencia es asignar cada objeto al subconjunto de clases que contengan un mayor porcentaje de su masa de creencia. De esta forma obtenemos una partición de creencia fuerte de, como mucho, 2^K clusters. A partir de esta partición es posible discernir qué objetos deben ser asignados a un único cluster sin ambigüedad, además de detectar aquellos que se encuentran en la frontera de dos o más clusters, que a menudo suelen ser relevantes.

4.4 LINEAR CONSTRAINED VECTOR QUANTIZATION ERROR (LCVQE)

Tomando como base el trabajo de Pelleg y Dorit (2007) [24], es necesario realizar una introducción al algoritmo Constrained Vector Quantization Error (CVQE) antes de detallar el algoritmo LCVQE, puesto que este último consiste en una modificación sobre el primero para mejorar, principalmente, su orden de complejidad.

4.4.1 El algoritmo CVQE

El algoritmo CVQE consiste en una generalización del algoritmo K-medias (KM, Apéndice A) para incluir las restricciones. En el algoritmo KM los centroides v_i se actualizan en cada iteración siguiendo la regla:

$$v_i = \frac{1}{|c_i|} \sum_{x_i \in c_i} x_i \quad (4.23)$$

que no es más que un promedio de todas las instancias asignadas al cluster asociado al centroide v_i . Tras la actualización, se recalculan las asignaciones de forma que cada instancia esté asociada al cluster más cercano. Esto deriva en una regla que minimiza la función Vector Quantization Error (VQE):

$$VQE = \frac{1}{2} \sum_{j=1}^K \sum_{x_i \in c_j} (v_j - x_i)^2 \quad (4.24)$$

CVQE modifica la función VQE, añadiendo a la misma un término de penalización que considera las restricciones incumplidas. Por simplicidad, notaremos el cardinal de ML con ml , así como el de CL con cl , y definiremos el conjunto de restricciones R como una lista de parejas de instancias:

$$\{(x_1(i), x_2(i))\}_{i=1}^{ml+cl} \quad (4.25)$$

Además definimos M como la función que, dada una instancia, devuelve el índice del cluster que la contiene: $M = \{j | x \in c_j\}$, y las funciones $g(i) = M(x_1(i))$ y $g'(i) = M(x_2(i))$. Por otra parte, definimos $h(i)$ como la función que devuelve el centroide más cercano al centroide v_i . Finalmente definimos $vl(i)$ de manera que indica si la restricción i -ésima ha sido violada, por tanto, para $i = 1, \dots, ml$ $vl(i) = 1 \Leftrightarrow g(i) \neq g'(i)$, de forma similar $i = ml + 1, \dots, ml + cl$ $vl(i) =$

$1 \leftrightarrow g(i) = g'(i)$. Definidas estas funciones, la regla de actualización de CVQE es:

$$v_j = \frac{1}{N_j} \left[\sum_{x_i \in c_i} x_i + \sum_{l=1, g(l)=j}^{ml} v_l(l) v_{g'(l)} + \sum_{l=ml+1, g(l)=j}^{ml+cl} v_l(l) v_{h(g'(l))} \right] \quad (4.26)$$

donde N_j viene definido por:

$$N_j = |c_j| + \sum_{l=1, g(l)=j}^{ml+cl} v_l(l) \quad (4.27)$$

De manera intuitiva, para las restricciones *ML* incumplidas, uno de los dos centroides afectados se mueve hacia el otro, mientras que para las *CL* incumplidas se mueve una de las instancias hacia el siguiente centroide más cercano a su cluster. De forma similar a KM, cada instancia se reasigna después de cada iteración para minimizar la función de error $CVQE = \sum_{j=1}^K CVQE_j$, donde $CVQE_j$ es:

$$CVQE_j = \frac{1}{2} \sum_{x_i \in c_j} T_{j,1} + \frac{1}{2} \sum_{l=1, g(l)=j}^{ml} T_{j,2} + \frac{1}{2} \sum_{l=ml+1, g(l)=j}^{ml+cl} T_{j,3} \quad (4.28)$$

y los términos $T_{j,1}$, $T_{j,2}$ y $T_{j,3}$ vienen definidos como:

$$\begin{aligned} T_{j,1} &= (v_j - x_i)^2 \\ T_{j,2} &= \left[(v_j - v_{g'(l)})^2 \cdot v_l(l) \right] \\ T_{j,3} &= \left[(v_j - v_{h(g'(l))})^2 \cdot v_l(l) \right] \end{aligned} \quad (4.29)$$

Cabe destacar que, al contrario de lo que sucede en K-medias (KM), en CVQE, un cluster c_i puede contener instancias para las que el centroide v_i no es el más cercano a ella.

En cada paso, el algoritmo CVQE asigna cada par de instancias implicadas en una restricción de manera que se minimiza la función $CVQE$.

A continuación se exponen algunas de las características del algoritmo CVQE que resultan relevantes para la compresión de LCVQE.

En primer lugar, el orden en el que se especifican las instancias implicadas en las restricciones es relevante. Consideremos una restricción formada por las instancias $(x_1(l), x_2(l))$, de forma que $x_1(l) \in c_{g(l)}$, y $x_2(l) \in c_{g'(l)}$. Sólo $c_{g(l)}$ se ve afectado por violar la restricción, mientras que sobre $c_{g'(l)}$ no se aplica ningún cambio. Esta regla se cumple tanto para las restricciones Must-Link (ML) como para las Cannot-Link (CL).

Observación 4.1 En CVQE el orden en el que se especifican las instancias implicadas en una restricción es relevante.

En segundo lugar, determinar la asignación que minimiza la función de error requiere $\mathcal{O}(K^2)$ cálculos para cada restricción. Por tanto, el método puede resultar computacionalmente pesado para grandes conjuntos de restricciones o clusters. Además, no es posible descartar ninguna opción de los cálculos aparte de las triviales.

Observación 4.2 *El método CVQE es computacionalmente pesado para grandes conjuntos de datos o restricciones.*

Para exemplificar esto consideramos la Figura 4.1. En ella, el par (x, y) es una restricción ML, y los centroides existentes son $\{v_1, \dots, v_6\}$. Dependiendo de los valores R , δ y ϵ las instancias pueden ser asignadas a los centroides (v_1, v_2) , (v_3, v_4) , (v_5, v_6) . Por tanto, las K^2 opciones deben ser consideradas para cada restricción al decidir una asignación.

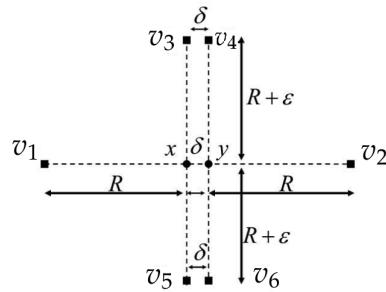


Figura 4.1: Ejemplo de CVQE. [24]

La última observación que debemos hacer está relacionada con el hecho de que la penalización por violar restricciones depende de la distancia entre los centroides implicados en ella, pero no de la distancia entre las instancias.

Observación 4.3 *El término de penalización por violar restricciones de CVQE sólo depende de la distancia entre los centroides implicados en la restricción.*

Para mostrar el problema que esto supone tomamos la Figura 4.2, en la que se presentan dos problemas de clustering. Para ambos, los centroides existentes son v_1 y v_2 , mientras que un problema incluye la restricción $ML(x_1, y)$ y el otro $ML(x_2, y)$.

La Tabla 4.2 recoge las asignaciones posibles en cada caso, así como el valor de la función CVQE. Vemos que, independientemente del valor de d y f , ambos problemas tiene la misma solución, mientras que intuitivamente diríamos que violar la restricción $ML(x_1, y)$ debería acarrear una penalización mayor que violar $ML(x_2, y)$. Es más, en ambos casos la acción que aplica CVQE es la misma, mover v_1 hacia v_2 en la recta que los une.

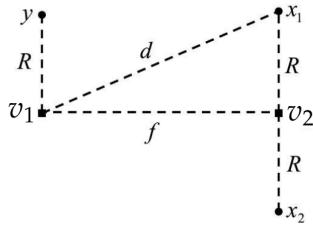


Figura 4.2: Ejemplo de CVQE. [24]

Valores de CVQE			
Restricción	$y \in c_1, x_i \in c_2$	$x_i, y \in c_1$	$x_i, y \in c_2$
$ML(x_1, y)$	$R^2 + R^2 + f^2$	$R^2 + f^2$	$R^2 + d^2$
$ML(x_2, y)$	$R^2 + R^2 + f^2$	$R^2 + f^2$	$R^2 + d^2$

Tabla 4.2: Valores de CVQE para el ejemplo [23]

4.4.2 El algoritmo LCVQE

El algoritmo LCVQE minimiza una función similar a la que minimiza CVQE. Con la diferencia de que, para cada asignación, LCVQE considera, como mucho, los dos clusters más naturalmente apropiados para la asignación. Por ello, la complejidad del algoritmo es independiente de K , al contrario de lo que sucedía en CVQE.

Intuitivamente, las restricciones ML violadas modifican la actualización de los centroides para desplazarlos hacia la instancia opuesta. Para las restricciones CL violadas, se determina cuál es la instancia más lejana al centroide común, sirviendo ésta como guía para mover el segundo centroide más cercano hacia ella. Con esto tenemos que, en LCVQE, las restricciones son simétricas, es decir, no es relevante el orden en el que se especifiquen las instancias implicadas en ellas.

Observación 4.4 En LCVQE las restricciones son simétricas, esto es, el orden en el que se especifiquen las instancias en ellas no es relevante.

Para formalizar la regla de actualización es necesario definir nuevas funciones. $L_j(i)$ devuelve la instancia de entre $x_1(i)$ y $x_2(i)$ que más lejos se encuentre del centroide v_j . $MM(x)$ devuelve el centroide más

cercano a x distinto de $v_{M(x)}$. Con esto la regla de actualización queda definida como:

$$\begin{aligned}
 v_j &= \frac{1}{N_j} \left[\sum_{x_i \in c_j} x_i + \frac{1}{2} S_1 + \frac{1}{2} S_2 + S_3 \right] \\
 S_1 &= \sum_{l=1, g(l)=j}^{ml} vl(l) \cdot x_2(l) \\
 S_2 &= \sum_{l=1, g'(l)=j}^{ml} vl(l) \cdot x_1(l) \\
 S_3 &= \sum_{l=ml+1, j=MM(L_{M(x_1(l))}(l))}^{ml+cl} vl(l) \cdot L_{M(x_1(l))}(l)
 \end{aligned} \tag{4.30}$$

donde N_j en este caso se calcula como:

$$N_j = |c_j| + \frac{1}{2} \sum_{l=1, g(l)=j}^{ml} vl(l) + \frac{1}{2} \sum_{l=1, g'(l)=j}^{ml} vl(l) + \sum_{l=ml+1, j=MM(L_{M(x_1(l))}(l))}^{ml+cl} vl(l) \tag{4.31}$$

Esta regla de actualización minimiza la función de error:

$$\begin{aligned}
 E_j &= \frac{1}{2} \sum_{x_i \in c_j} T_{j,1} + \frac{1}{2} \sum_{l=1, g(l)=j}^{ml} T_{j,2} + \frac{1}{2} \sum_{l=ml+1, g'(l)=j}^{ml} T_{j,3} + \\
 &\quad \frac{1}{2} \sum_{l=ml+1, j=MM(L_{M(x_1(l))}(l))}^{ml+cl} T_{j,4}
 \end{aligned} \tag{4.32}$$

quedando definidos los términos T como:

$$\begin{aligned}
 T_{j,1} &= (v_j - x_i)^2 \\
 T_{j,2} &= \left[\frac{1}{2} (v_j - x_2(l))^2 \cdot vl(l) \right] \\
 T_{j,3} &= \left[\frac{1}{2} (v_j - x_1(l))^2 \cdot vl(l) \right] \\
 T_{j,4} &= \left[(v_j - L_{M(x_1(l))}(l))^2 \cdot vl(l) \right]
 \end{aligned} \tag{4.33}$$

El Algoritmo 3 recoge de manera resumida el proceso que sigue LCVQE para obtener una partición de los datos.

El criterio de convergencia puede estar basado en la diferencia de los centroides entre iteraciones sucesivas, así como en criterios de asignación de instancias a clusters o evaluaciones totales de la función de error.

El algoritmo LCVQE requiere $\mathcal{O}(p)$ operaciones en cada paso, siendo p el número de dimensiones, ya que sólo se consideran tres posibles asignaciones, independientemente del valor de K . Por tanto LCVQE supera en eficiencia a CVQE, que era altamente sensible al número de clusters y de restricciones.

Observación 4.5 *El algoritmo LCVQE es computacionalmente más eficiente que CVQE, pues en cada paso realiza $\mathcal{O}(p)$ operaciones.*

Finalmente, podemos estudiar como se comporta el algoritmo frente al ejemplo de la Figura 4.2. La Tabla 4.3 muestra las asignaciones posibles en cada caso, así como los valores LCVQE. En ambos problemas asumimos que la asignación es $x_i \in c_2$ e $y \in c_1$. En el caso de la restricción (x_1, y) el centroide se actualiza según la regla $c_1 = (y + \frac{1}{2}x_1)/1,5$, y en el caso de (x_1, y) tenemos que $c_1 = (y + \frac{1}{2}x_2)/1,5$, resultados intuitivamente mejores que los que obteníamos con CVQE, ya que el centroide se mueve hacia la distancia media de las instancias en lugar de hacia el otro centroide c_2 .

Observación 4.6 *El término de penalización por violar restricciones de CVQE involucra las características espaciales de las instancias implicadas en ella.*

Valores de LCVQE			
Restricción	$y \in c_1, x_i \in c_2$	$x_i, y \in c_1$	$x_i, y \in c_2$
$ML(x_1, y)$	$(d^2 + d^2)/2$	d^2	d^2
$ML(x_2, y)$	$(d^2 + d^2)/2$	d^2	d^2

Tabla 4.3: Valores de LCVQE para el ejemplo [23]

4.5 RELATIONAL DIRICHLET PROCESS - MEANS (RDP - MEANS)

Tomamos como referencia principal el trabajo de Daniel Khashabi et al. (2015) [25]. En él, los autores incorporan las restricciones desde una perspectiva distinta a las anteriores. Así, modelan el conjunto de instancias y el conjunto de restricciones de manera diferente; es por ello que llaman Two Views Clustering (TVClust) al método base para Relational Dirichlet Process - Means (RDPM).

Para ser más específicos, TVClust combina una mezcla de Procesos de Dirichlet, aplicados sobre el conjunto de instancias, y una interpretación del conjunto de restricciones que las considera como un grafo aleatorio. Además, los autores derivan este modelo, basándose en el algoritmo Dirichlet Process - Means (DPM) [26] combinado con el modelo de las restricciones, para obtener un algoritmo determinista que incorpora las mismas a DPM, esto es, el algoritmo Relational Dirichlet Process - Means (RDPM).

Algoritmo 3: Linear Constrained Vector Quantization Error (LCVQE)

Entrada: Conjunto de datos X , conjunto de restricciones R , centroides iniciales V , número de clusters K

Salida: Partición P del conjunto de datos X , centroides V

función LCVQE(X, R, K) **begin**

 1. Inicialización de $GMLV_j = GCLV_j = \emptyset \quad \forall j \in \{1, \dots, K\}$

 2. Inicialización de C por la regla del centroide más cercano.

 3. Para cada $\{(x_1(i), x_2(i))\} \in ML$ siendo v_j el centroide más cercano a $x_1(i)$ y v_n el centroide más cercano a $x_2(i)$ calcular:

$$(a) \frac{1}{2} [(x_1(l) - v_j)^2 + (x_2(l) - v_n)^2] + \frac{1}{4} [(x_1(l) - v_n)^2 + (x_2(l) - v_j)^2]$$

$$(b) \frac{1}{2}(x_1(l) - v_j)^2 + \frac{1}{2}(x_2(l) - v_j)^2$$

$$(c) \frac{1}{2}(x_1(l) - v_n)^2 + \frac{1}{2}(x_2(l) - v_n)^2$$

Si (a) es minimal $\rightarrow GMLV_j = GMLV_j \cup x_2(l)$ y $GMLV_n = GMLV_n \cup x_1(l)$. Asignar x_1 a c_j y x_2 a c_n .

Si (b) es minimal \rightarrow asignar x_1 y x_2 a c_j .

Si (c) es minimal \rightarrow asignar x_1 y x_2 a c_n .

4. Para cada $\{(x_1(i), x_2(i))\} \in CL$ con v_j el centroide más cercano a $x_1(i)$ y v_n el centroide más cercano a $x_2(i)$, y siendo $Max_n(l) = argmax_{x_i(l)}(x_i(l) - v_n)^2$. Tomamos j como el centroide más cercano a $Max_n(l)$ que no sea n y calculamos:

$$(a) \frac{1}{2}(x_1(l) - v_j)^2 + \frac{1}{2}(x_2(l) - v_j)^2 + \frac{1}{2}(Max_n(l) - v_{MM(Max_j(l))})^2$$

$$(b) \frac{1}{2} [(x_1(l) - v_j)^2 + (x_2(l) - v_n)^2]$$

Si (a) es minimal \rightarrow

$GCLV_{MM(Max_j(l))} = GCLV_{MM(Max_j(l))} \cup Max_j(l)$ y asignar x_1 y x_2 a c_j .

Si (b) es minimal \rightarrow Asignar x_1 a c_j y x_2 a c_n .

5. Actualizar cada centroide v_j en V como sigue:

$$v_j = \frac{1}{N_j} [Sum(c_j) + \frac{1}{2}Sum(GMLV_l) + Sum(GCLV_j)]$$

6. Iterar entre (2.) y (5.) hasta converger

7. **return** C, V

end

4.5.1 El modelo Bayesiano no paramétrico

Introducimos las particularidades relacionadas con el modelo de datos sobre el que aplicaremos clustering. El conjunto de instancias X viene definido tal y como se especifica al inicio de la Sección 4, y el conjunto de restricciones R como una matriz simétrica de $n \times n$. En R se encuentran almacenadas todas las posibles restricciones que involucran a parejas de instancias. Así, tomando x_i y x_j como dos instancias, basta con obtener el valor $R_{i,j}$ en la matriz para saber la relación que existe entre ellas. Si $R_{i,j} = 1$, entonces existe una restricción de tipo Must-Link (ML) entre x_i y x_j , si $R_{i,j} = 0$ la restricción será de tipo Cannot-Link (CL), y no existirá relación entre las instancias en cualquier otro caso.

Consideramos los dos conjuntos de datos de los que disponemos, X y R , como dos formas distintas de ver la estructura de clusters subyacente. Cabe destacar que cualquiera de estas dos estructuras es suficiente para llevar a cabo procesos de clustering clásicos, ya sean métodos como K-medias (KM) (Sección A), en el caso de X , o *normalized graph-cut* en el caso de R .

La aproximación mediante TVClust consiste en agregar información de los dos modelos mediante un enfoque Bayesiano, de manera que sea posible alcanzar un consenso entre ambos, que tenga como resultado la partición de X . Dada la estructura de clustering latente, que se supone común, los datos de ambas interpretaciones se modelan mediante dos procesos generativos distintos: X se modela mediante Mezcla de Procesos de Dirichlet, y R mediante un grafo aleatorio.

Considerar modelos distintos para entender los datos y las restricciones es de especial utilidad cuando ninguno de los dos puede ser tomado como completamente fiable. Mientras que otros métodos de clustering como COP-K-medias (Sección 4.2) confían en la exactitud de las restricciones, TVClust hace una interpretación relajada de las mismas, haciéndolo más robusto ante errores. Podría decirse que, en TVClust, $R_{i,j} = 1$ equivale una una restricción *May-Link*, mientras que $R_{i,j} = 0$ se asociaría a una *May-Not-Link*, en contraste con las ya conocidas Must-Link (ML) y Cannot-Link (CL).

4.5.1.1 Modelo para las instancias

Como ya hemos visto, utilizamos una Mezcla de Procesos de Dirichlet como modelo de clustering subyacente para el conjunto de datos X . Tomamos θ_i como el parámetro del modelo asociado a la instancia x_i , que es modelado como una muestra independiente e idénticamente

te distribuida ¹ de una distribución aleatoria G , que se obtiene en base a un Proceso de Dirichlet $\text{DP}(\alpha, G_0)$:

$$\{\theta_0, \dots, \theta_n\} \text{ t.q. } G \stackrel{iid}{\sim} G, \quad G \sim \text{DP}(\alpha, G_0) \quad (4.34)$$

Con esto, conociendo la distribución de $\theta_{\setminus i}$ definida como $\theta_{\setminus i} = \{\theta_0, \dots, \theta_{i-1}, \theta_{i+1}, \dots, \theta_n\}$ podemos conocer la de θ_i aplicando el esquema de Balckwell-MacQueen [27]:

$$p(\theta_i | \theta_{\setminus i}) \propto \sum_{k=1}^K n_{-i,k} \delta_{\theta_k^*}(\theta_i) + \alpha G_0(\theta_i) \quad (4.35)$$

donde asumimos que existen K ² valores únicos entre $\theta_{\setminus i}$ notados como $\{\theta_1^*, \dots, \theta_K^*\}$, $\delta_{\theta_k^*}(\cdot)$ es la función delta de Kronecker, y $n_{-i,k}$ es el número de instancias agrupadas en el cluster c_k excluyendo la instancia i . Así, la Ecuación 4.35 puede entenderse dentro de los métodos de clustering tradicionales en el sentido de que con una probabilidad positiva, θ_i tomará uno de los valores del conjunto $\{\theta_1^*, \dots, \theta_K^*\}$, esto es, la instancia asociada se incluirá en uno de los clusters presentes. Para simplificar este concepto podemos emplear la Metáfora del Restaurante Chino, propuesta por Aldous (1983), en la que asignar θ_i a un cluster es análogo a sentar un cliente en una mesa del restaurante. El cliente puede sentarse en una mesa ya ocupada o en una que estaba vacía.

Dado θ_i , utilizamos una familia de funciones paramétricas $p(x_i | \theta_i)$ para modelar la instancia x_i , en concreto tomaremos las familia de las exponenciales:

$$p(x | \theta) = \exp(\langle T(x), \theta \rangle - \psi(\theta) - h(x)) \quad (4.36)$$

donde $\psi(\theta)$ es la función *log-partition* para la familia exponencial $\psi(\theta) = \log \int \exp(\langle x, \theta \rangle - h(x)) dx$, y $T(x)$ es el vector de estimadores suficientes de la distribución de probabilidad. En el caso de la familia exponencial la dimensión del vector de estimadores suficientes es igual al número de parámetros estimables. Por simplicidad asumimos que x es el vector de estimadores suficientes aumentado y obtenemos:

$$p(x | \theta) = \exp(\langle x, \theta \rangle - \psi(\theta) - h(x)) \quad (4.37)$$

Con esta formulación definimos:

$$\begin{aligned} \mathbb{E}_p[x] &= \nabla_\theta \psi(\theta) \\ \text{Cov}_p[x] &= \nabla_\theta^2 \psi(\theta) \end{aligned} \quad (4.38)$$

¹ iid es la abreviatura en inglés para “independiente e idénticamente distribuida”.

² La interpretación de K en esta sección es distinta a la propuesta al inicio de la Sección 4, en la introducción a la notación.

Y por conveniencia escogeremos la medida base para G_0 en $\text{DP}(\alpha, G_0)$ de entre la familia de conjugados, que toma la siguiente forma:

$$dG_0(\theta|\tau, \eta) = \exp(\langle\theta, \tau\rangle - \eta\psi(\theta) - m(\tau, \eta)) \quad (4.39)$$

donde τ y η son parámetros de la distribución base. Dadas estas definiciones de probabilidad y conjugado, la posterior distribución sobre θ será la familia exponencial de la misma forma que la inicial, pero escalando los parámetros a $\tau + x$ y $\eta + 1$ (para más detalles consultar [25]).

La familia exponencial contiene multitud de distribuciones empleadas en la práctica. Por ejemplo, las distribuciones Gaussianas se utilizan para modelar valores reales en un espacio \mathbb{R}^p .

4.5.1.2 *Modelo para las restricciones*

Dado $\theta_{1:n} = \{\theta_1, \dots, \theta_n\}$ podemos resumir las estructura del clustering mediante una matriz H con dimensiones $n \times n$, donde $H_{i,j} = \delta_{\theta_i}(\theta_j)$. Cabe destacar que la matriz H es distinta a la matriz R , ya que esta última representa las restricciones y puede ser entendida como una muestra aleatoria de la estructura de clustering contenida en H . Con esto, el objetivo es inferir H en base a R .

Modelamos R mediante el siguiente proceso generativo: con probabilidad p , una arista existente en el grafo descrito por H se conserva en R , y con probabilidad q , una falsa arista de H se añade a R . Por ejemplo, tomando $(i, j) \in \{1, \dots, n\}$:

$$\begin{cases} p(R_{i,j} = 0 | H_{i,j} = 1) = p \\ p(R_{i,j} = 1 | H_{i,j} = 0) = 1 - p \\ p(R_{i,j} = 0 | H_{i,j} = 0) = q \\ p(R_{i,j} = 1 | H_{i,j} = 0) = 1 - q \end{cases} \quad (4.40)$$

Dicho de otra forma, los parámetros p y q representan la credibilidad de los valores presentes en la matriz R , mientras que $1 - p$ y $1 - q$ representan la probabilidad de error. El valor concreto que se debe asignar a p y q varía para cada problema; pueden ser obtenidos mediante conocimiento experto o con técnicas de aprendizaje desde datos.

4.5.1.3 *Inferencia mediante muestreo de Gibbs*

Es posible derivar un esquema de muestro de Gibbs para el modelo de TVClust, que en esencia es similar al que emplea Dirichlet Process

- Means (DPM). La distribución de muestreo que emplearemos viene definida por:

$$p(\theta_i | \theta_{\setminus i}, x_i, R, p, q) \propto \sum_{k=1}^K n_{-i,k} p(x_i | \theta_k^*) \delta_{\theta_k^*}(\theta_i) \left(\frac{p}{1-q} \right)^{f_k^i} \left(\frac{1-p}{q} \right)^{s_k^i} + \alpha p_{G_0}(x_i) p_{G_0}(\theta_i | x_i) \quad (4.41)$$

donde

$$\begin{cases} f_k^i = \#\{j : \theta_j = \theta_k^*, R_{i,j} = 1\} \\ s_k^i = \#\{j : \theta_j = \theta_k^*, R_{i,j} = 0\} \end{cases} \quad (4.42)$$

El proceso de cálculo y simplificación que deriva en estas ecuaciones se encuentra detallado en el trabajo de Daniel Khashabi et al. (2015) [25].

Retomando la analogía del restaurante chino, podemos interpretar la distribución de muestreo de θ_i de la siguiente manera: si la instancia i es *amiga* de la instancia j , entonces $R_{i,j} = 1$, por otra parte, si dos instancias son *desconocidas* entonces $R_{i,j} = 0$. Con esto, podemos interpretar el valor f_i^k como el número de amigos de la instancia i en la mesa k , y s_i^k como el número de desconocidos.

Tal y como hemos visto, los valores p y q representan la credibilidad de las restricciones, esto es, el nivel de confianza en que son correctas. Para representar un grado de confianza alto generalmente estableceremos que $p > 1 - q$. Con esto, y tomando en consideración la Ecuación 4.41, la probabilidad de que una persona sea asignada a una mesa no sólo aumenta con la popularidad de la misma, sino que también lo hace con el número de amigos de la instancia en la mesa (f_i^k) y disminuye con el número de desconocidos (s_i^k).

Una mejora aplicable al modelo consiste en modificar la regla de actualización del conjunto de parámetros $\{\theta_1, \dots, \theta_n\}$, de manera que, en lugar de actualizar los parámetros particulares de cada instancia de manera secuencial, se actualiza un conjunto de parámetros asociados a los clusters $\{\theta_1^*, \dots, \theta_K^*\}$, así como los indicadores de asignación de instancias a clusters (las etiquetas) $\{z_1, \dots, z_n\}$ donde $z_i \in \{1, \dots, K\}$ indica el cluster al que esta asignado la instancia i . Entonces la nueva regla de actualización es:

$$\begin{cases} p(z_i = k) \propto n_{-i,k} p(x_i, \theta_k^*) \left(\frac{p}{1-q} \right)^{f_k^i} \left(\frac{1-p}{q} \right)^{s_k^i} \\ p(z_i = k_{nuevo}) \propto \alpha \int p(x_i, \theta_k^*) dG_0 \end{cases} \quad (4.43)$$

4.5.2 Reparametrización de la familia exponencial para RDP-means

Para comprender Relational Dirichlet Process - Means (RDPM) es necesario introducir el concepto de divergencia de Bregman, así como su relación con la familia de funciones exponenciales. Comenzamos con la definición formal de la divergencia de Bregman:

Definición 4.1 Divergencia de Bregman (1967): Definida una función estrictamente convexa $\phi : \mathcal{S} \rightarrow \mathbb{R}$, de forma que el dominio $\mathcal{S} \subseteq \mathbb{R}^p$ es un conjunto convexo, y ϕ es diferenciable en $ri(\mathcal{S})$, siendo este el interior relativo de \mathcal{S} donde su gradiente $\nabla\phi$ existe. Dados dos puntos $x, y \in \mathbb{R}^p$, la divergencia de Bregman $D_\phi(x, y) : \mathcal{S} \times ri(\mathcal{S}) \rightarrow [0, +\infty)$ viene definida como: [25]

$$D_\phi(x, y) = \phi(x) - \phi(y) - \langle x - y, \nabla\phi(y) \rangle$$

La divergencia de Bregman es una clase general de medidas de distancia, por ejemplo, tomando ϕ como una función cuadrática, la divergencia de Bregman es equivalente a la distancia Euclídea [28].

Existe una aplicación biyectiva entre la familia exponencial y las divergencias de Bregman, tal y como demostraron Foster y Warmuth (2002) [29]. Dada esta conexión, Banerjee et al. (2005) [28] construyeron un algoritmo en base a K-medias (KM) que utiliza la divergencia de Bregman como medida de distancia, en lugar de la distancia Euclídea.

Definición 4.2 El Conjugado de Legendre: Para una función $\psi(\cdot)$ definida sobre el dominio \mathbb{R}^p , se define como su conjugado convexo $\psi^*(\cdot)$ como $\psi^*(\mu) = \sum_{\theta \in \text{dom}(\psi)} \{\langle \mu, \theta \rangle - \psi(\theta)\}$. Además, si la función $\psi(\theta)$ es cerrada y convexa, entonces $(\psi^*)^* = \psi$. [25]

Se puede demostrar que la función *log-partition* de la familia exponencial es una función cerrada y convexa. Por tanto, existe una biyección entre el parámetro μ del conjugado de Legendre $\psi^*(\cdot)$, y el parámetro de la familia exponencial, θ , en la función *log-partition* definida para la familia exponencial en la Ecuación 4.36. Con esto, podemos reescribir la probabilidad definida por la Ecuación 4.37 usando la divergencia de Bregman y el conjugado de Legendre:

$$p(x|\theta) = p(x|\mu) = \exp(-D_{\psi^*}(x, \mu))f_{\psi^*}(x) \quad (4.44)$$

donde $f_{\psi^*}(x) = \exp(\psi^*(x) - h(x))$. Una consecuencia de esta nueva parametrización es que la probabilidad asociada a cualquier instancia x está relacionada con cómo de lejos está la misma del cluster parametrizado con μ , midiendo la distancia según la divergencia de Bregman $D_{\psi^*}(x, \mu)$.

De manera similar podemos reescribir la Ecuación 4.39 en términos de la divergencia de Bregman y el conjugado de Legendre:

$$p(\theta|\tau, \eta) = p(\mu|\tau, \eta) = \exp\left(-\eta D_{\psi^*}\left(\frac{\tau}{\eta}, \mu\right)\right) g_{\psi^*}(\tau, \eta) \quad (4.45)$$

donde $g_{\psi^*}(\tau, \eta) = \exp(\eta\psi(\theta) - m(\tau, \eta))$

4.5.3 Escalando las distribuciones para RDP-means

Lema 4.1 Jiang et al (2012) [26]: Dada la familia exponencial definida en la Ecuación 4.36, definimos otra distribución de probabilidad con parámetro $\tilde{\theta}$ y la función log-partition $\tilde{\psi}(\cdot)$, donde $\tilde{\theta} = \gamma\theta$ y $\tilde{\psi}(\tilde{\theta}) = \gamma\psi(\tilde{\theta}/\gamma)$, entonces:

1. La distribución de probabilidad escalada $\tilde{p}(\cdot)$ definida con parámetro $\tilde{\theta}$ y función log-partition $\tilde{\psi}(\cdot)$, es una distribución de probabilidad propia que pertenece a la familia exponencial.
2. La media y la varianza de la distribución de probabilidad $\tilde{p}(\cdot)$ son:

$$\mathbb{E}_{\tilde{p}}(x) = \mathbb{E}_p(x), \quad Cov_{\tilde{p}}(x) = \frac{1}{\gamma} Cov_p(X)$$

3. El conjugado de Legendre de $\tilde{\psi}(\cdot)$ es $\tilde{\psi}^*(\tilde{\theta}) = \gamma\psi^*(\tilde{\theta})$

El resultado más importante derivado del Lema 4.1 es que la covarianza $Cov_p(x)$ escala con $1/\gamma$, que tiende a 0 para valores grandes de γ , mientras que la media $\mathbb{E}_p(x)$ se mantiene invariante. Por tanto podemos obtener un algoritmo determinista cuando γ tiende a infinito.

Con esto, las Ecuaciones 4.44 y 4.45 pueden ser escritas como:

$$\begin{aligned} \tilde{p}(x|\theta, \gamma) &= \tilde{p}(x|\mu, \gamma) = \exp(-\gamma D_{\psi^*}(x, \mu)) f_{\gamma\psi^*}(x) \\ \tilde{p}(\theta|\tau, \eta, \gamma) &= \tilde{p}(\tilde{\mu}|\tau, \eta, \gamma) = \exp\left(-\eta D_{\psi^*}\left(\frac{\tau}{\eta}, \mu\right)\right) g_{\gamma\psi^*}(\tau/\gamma, \eta/\gamma) \end{aligned} \quad (4.46)$$

4.5.4 Asintóticos de TVClust

Usando la distribución escalada definida en la Ecuación 4.46 podemos reescribir la regla de actualización de Gibbs descrita en la Ecuación 4.43 como sigue:

$$\begin{cases} p(z_i = k) \propto n_{-i,k} \exp(-\gamma D_{\psi^*}(x_i, \mu_k)) \left(\frac{p}{1-q}\right)^{f_k^i} \left(\frac{1-p}{q}\right)^{s_k^i} \\ p(z_i = k_{nuevo}) \propto \alpha \int \tilde{p}(x_i|\theta) \tilde{p}(\theta|\tau, \eta) d\theta \end{cases} \quad (4.47)$$

Siguiendo el proceso de cálculo y simplificación detallado en el trabajo de Daniel Khashabi et al. (2015) [25], podemos transformar la Ecuación 4.47 en:

$$\begin{cases} p(z_i = k) \propto n_{-i,k} \exp(-\gamma D_{\psi^*}(x_i, \mu_k)) - \xi_1 f_k^i + \xi_2 s_k^i \\ p(z_i = k_{nuevo}) \propto \nu(x_i; \tau, \eta, \gamma) \exp(-\gamma \lambda) \end{cases} \quad (4.48)$$

Donde $\xi_1 = \ln\left(\frac{p}{1-p}\right)$ y $\xi_2 = \ln\left(\frac{q}{1-p}\right)$, esto es, ξ_1 y ξ_2 representan, respectivamente, la probabilidad de que exista o no exista una restricción.

Cuando γ tiende a infinito, el muestreador de Gibbs deriva en un algoritmo determinista, en el que, en cada iteración, la asignación de x_i a un cluster se determina comparando los $K + 1$ valores descritos por:

$$\{D_{\psi^*}(x_i, \mu_1) - \xi_1 f_1^i + \xi_2 s_1^i, \dots, D_{\psi^*}(x_i, \mu_K) - \xi_1 f_K^i + \xi_2 s_K^i, \lambda\} \quad (4.49)$$

Si el valor k -ésimo (con $k = 1, \dots, K$) es el más pequeño, entonces x_i se asigna al cluster k -ésimo, por el contrario, si λ es el menor valor, se crea un cluster nuevo con x_i como única instancia asignada.

4.5.5 Muestreo de los parámetros de los clusters

Daniel Khashabi et al. (2015) [25] demuestran que la función que muestrea los parámetros para los clusters es equivalente a una media de las instancias que éstos contienen bajo las condiciones adecuadas, esto es, tomando $\gamma \rightarrow \infty$ y argumentos iguales para la divergencia de Bregman. De esta manera, la ecuación de actualización de los parámetros para los clusters es:

$$\mu_k = \frac{1}{n_k} \sum_{i:z_i=k} x_i \quad (4.50)$$

Esto completa el algoritmo Relational Dirichlet Process - Means (RDPM); podemos resumir el proceso de cálculo que éste lleva a cabo en el Algoritmo 4:

4.5.6 Función objetivo de RDP-means

Teorema 4.1 El algoritmo de clustering con restricciones RDP-means minimiza de manera iterativa la siguiente función:

$$\min_{\{\mathcal{L}_k\}_{k=1}^K} \sum_{k=1}^K \sum_{i \in \mathcal{L}_k} \left[D_\phi(x_i, \mu_k) - \xi_1 f_k^i + \xi_2 s_k^i \right] + \lambda K \quad (4.51)$$

donde $\{\mathcal{L}_1, \dots, \mathcal{L}_k\}$ representan una partición de las n instancias.

4.5.7 Efectos de los cambios sobre ξ_1 y ξ_2

Tomando $\xi_1, \xi_2 \rightarrow 0$, el algoritmo RDPM se comporta de manera idéntica a DPM, esto es, no se tiene en consideración la información proporcionada por las restricciones. Por otra parte, si $\xi_1, \xi_2 \rightarrow \infty$ se pone todo el peso en las restricciones y no se consideran las instancias para el proceso de clustering; en otras palabras, el conjunto de cluster se genera de acuerdo a la matriz R únicamente. De manera similar, podemos escoger poner más peso en las restricciones *may-link* sobre las *may-not-link* estableciendo $\xi_1 > \xi_2$ y viceversa.

A pesar del control que ξ_1 y ξ_2 permiten sobre el proceso de clustering, la función objetivo de RDPM (Ecuación 4.51) posee un gran número de mínimos locales y, dado que el algoritmo la minimiza con un proceso *greedy*, corremos el riesgo de caer en un mínimo local. Los autores Daniel Khashabi et al. (2015) [25] muestran experimentalmente que establecer $\xi_1 = \xi_2 = \xi$, con un valor pequeño para ξ_0 , que se incrementará con el paso de las iteraciones en función de un ratio dado, proporciona mejores resultados que un esquema de libre configuración para ξ_1 y ξ_2 .

Algoritmo 4: Relational Dirichlet Process - Means (RDPM)

Entrada: Conjunto de datos X , matriz de restricciones R , parámetro de la divergencia de Bregman ψ^* , los parámetros λ , ξ_0 , y el ratio de incremento ξ_{rate}

Salida: La asignación $z = [z_1, z_2, \dots, z_n]$

función RDPM($X, R, \psi^*, \lambda, \xi_0, \xi_{rate}$) **begin**

1. Inicializar $\xi \leftarrow \xi_0$ y asignar todas las instancias a un mismo cluster
2. Iterar hasta converger como sigue:

while no convergencia **do**

for $x_i \in X$ **do**

for $\mu_k \in C$ **do**

 Calcular los valores f_k^i y s_k^i a partir de R .
 $dist(x_i, \mu_k) \leftarrow D_{\psi^*}(x_i, \mu_k) - f_k^i \xi_1 + \xi_2 s_k^i$

end

 // d_{min} es la distancia mínima e i_{min} es el índice de la misma

$[d_{min}, i_{min}] \leftarrow \{dist(x_i, \mu_1), \dots, dist(x_i, \mu_k)\}$

if $d_{min} < \lambda$ **then**

$z_i \leftarrow i_{min}$

else

$C \leftarrow \{C \cup \{x_i\}\}$ //Crear un nuevo cluster

$K \leftarrow K + 1$

end

end

for $\mu_k \in C$ **do**

if $|c_k| > 0$ **then**

$\mu_k \leftarrow \frac{\sum_{x_i \in c_j} x_i}{|c_j|}$

else

 Eliminar el cluster c_j y propagar los cambios

end

end

$\xi \leftarrow \xi \times \xi_{rate}$

end

return C

end

IMPLEMENTACIÓN

El objetivo de esta sección es dar una visión general de los métodos y herramientas empleadas para la implementación de los algoritmos expuestos en la Sección 4, así como servir de guía para su utilización. Para ello tomamos como referencia la implementación de estos métodos en Matlab, resultado de un trabajo colaborativo en el que participan varios de los autores citados en este trabajo (citar github).

5.1 ENTORNO DE DESARROLLO

Para la implementación es esencial disponer de un entorno de desarrollo que permita manejar de manera eficiente los múltiples archivos que contendrán el código. De igual forma debemos disponer de herramientas de depuración de fácil uso. Por ello el entorno recomendado es PyCharm, desarrollado por la empresa JetBrains.

PyCharm ofrece multitud de herramientas que hacen el desarrollo de grandes cantidades de código una tarea asequible, ayudándonos a seguir el estándar de programación de Python y poniendo a nuestra disposición un depurador de código completo.

5.2 BIBLIOTECAS EMPLEADAS EN EL DESARROLLO

Para la implementación de las funcionalidades será necesario hacer uso de algunas bibliotecas de Python que nos permitan realizar cálculos y procedimientos básicos de forma sencilla.

Como no podía ser de otra manera, haremos uso de NumPy, una de las bibliotecas más completas y extendidas de Python. NumPy nos permite trabajar con matrices de manera eficiente, lo que es esencial para trabajar con grandes cantidades de datos que, a menudo, viene especificados en forma de matriz. NumPy integra operaciones con matrices, como la suma, la resta o la multiplicación y el cálculo de la inversa o del determinante de manera sencilla.

Emplearemos también la biblioteca `math`, que pone a nuestra disposición multitud de operaciones que no viene definidas en el lenguaje, como pueden ser el cálculo del factorial o el logaritmo.

La biblioteca SciPy es una de las bibliotecas para Python que integra métodos numéricos. SciPy ofrece una API sencilla de usar para el usuario, con funcionalidades como el cálculo de distintas medidas de distancia en un espacio dado, o el de funciones complejas. Un ejemplo de estas últimas pueden ser: el valor absoluto del logaritmo de la función Gamma para entradas reales (`gammaln`), o el de la derivada

logarítmica de la función Gamma (digamma), cálculos necesarios para la implementación de algunos métodos.

Otra de las bibliotecas que podemos encontrar de utilidad para el desarrollo puede ser `scikit-learn`, una de las más ampliamente usadas por la comunidad para el desarrollo de aplicaciones de aprendizaje automático de cualquier tipo. Scikit-learn pone a nuestra disposición métodos clásicos, como K-medias (KM), o Fuzzy K-means (FKM), que a menudo se emplean como métodos de inicialización en funcionalidades más complejas.

Por último, y aunque no sea estrictamente necesaria para la implementación, emplearemos la biblioteca `Matplotlib` para obtener representaciones de los resultados obtenidos con los métodos implementados. `Matplotlib` permite un alto nivel de personalización en representaciones, haciendo posible incluir toda la información que sea necesaria para su comprensión.

5.3 FUNCIONALIDADES DESARROLLADAS

Uno de los objetivos de este trabajo es poner a disposición de la comunidad algunos de los métodos de clustering con restricciones que existen en la actualidad. A continuación se ofrece la documentación de los métodos implementados.

5.3.1 Función para generar restricciones

La función `gen_rand_const` crea un conjunto de restricciones en base a un conjunto de datos. La función que sigue para ello es simple: basta con seleccionar de forma aleatoria dos instancias y establecer una restricción de tipo ML o CL en función de si las instancias pertenecen o no a la misma clase. Dado que el método necesita como argumento las etiquetas (un oráculo), puede no ser de utilidad en un caso real en el que no conoczamos la clase a la que pertenece cada objeto; sin embargo es válido para realizar experimentos. A continuación se muestra el prototipo del método:

```
def gen_rand_const(X, y, mat_const, nb_const, noise = 0,
prop = False)
```

Los parámetros de la función se detallan a continuación:

- **X** → matriz de $n \times p$ que contiene el conjunto de datos
- **y** → lista de de etiquetas de longitud n etiquetas asociadas al conjunto de datos.
- **mat_const** → matriz de $n \times n$ de restricciones sobre las que se añadirán las nuevas.

- **nb_const** → número de restricciones a añadir.
- **noise** → porcentaje de ruido a introducir en las restricciones.
- **prop** → booleano que indica si las restricciones deben propagarse, es decir, si la matriz será simétrica respecto a la diagonal principal

La función devuelve una matriz de $n \times n$ que contiene **nb_const** nuevas restricciones. Cada elemento de la matriz indica si existe una restricción entre dos instancias y de qué tipo es, esto es, 0 indica que no existe restricción, 1 indica que existe una de tipo ML, y -1 una de tipo CL.

5.3.2 Función para aplicar COP-K-Medias

La función **COPKM** es la encargada de aplicar el algoritmo COP-k-medias (Sección 4.2) a un conjunto de datos dado. La implementación de esta función está basada en el trabajo de Behrouz Babaki [30]. A continuación se muestra el prototipo de la función y se detallan sus parámetros:

```
def COPKM(X, K, constraints, max_iter = 300, tol = 1e-4,
           init = 'rand')
```

- **X** → matriz de $n \times p$ que contiene el conjunto de datos
- **K** → número de clusters de la partición resultado
- **constraints** → matriz de $n \times n$ que contiene el conjunto de restricciones. Siendo i y j los índices que indican filas y columnas respectivamente, un 1 en la matriz indica una restricción ML entre i y j , un -1 indica una de tipo CL, y un 0 indica que no existe restricción.
- **max_iter** → límite de iteraciones para el algoritmo
- **tol** → factor multiplicativo para el cálculo del umbral del desplazamiento de los centroides por debajo del cual se detiene el proceso de iteración.
- **init** → modelo de inicialización de los centroides: 'rand' para inicialización aleatoria y 'kmpp' para inicialización con K-medias++

La función devuelve la lista de longitud n que indica la pertenencia de cada instancia a un cluster dado, es decir, la partición del conjunto de datos X , y los centroides V asociados a los clusters.

5.3.3 Función para aplicar CEKM

La función `CEKM` aplica el algoritmo CEKM (Sección 4.3) a un conjunto de datos dado. A continuación se muestra el prototipo de la función y se detallan sus parámetros:

```
def CEKM(X, K, constraints, max_iter = 300, alpha = 1,
          rho = 100, bal = 0.5, stop_thr = 1e-3, init = 'rand')
```

- **X** → matriz de $n \times p$ que contiene el conjunto de datos
- **K** → número de clusters de la partición resultado
- **constraints** → matriz de $n \times n$ que contiene el conjunto de restricciones. Siendo i y j los índices que indican filas y columnas respectivamente, un 1 en la matriz indica una restricción ML entre i y j , un -1 indica una de tipo CL, y un 0 indica que no existe restricción.
- **max_iter** → límite de iteraciones para el algoritmo
- **alpha** → exponente que controla la penalización por asignar instancias a conjuntos con alta cardinalidad.
- **rho** → distancia de todos los objetos al conjunto vacío.
- **bal** → compromiso entre la función objetivo J_{CEKM} y las restricciones: $J_{CEKM} = (1 - bal) \times J_{CEKM} + bal$
- **stop_thr** → umbral por debajo del cual la diferencia entre las posiciones de los centroides no es significativa. Por tanto si dicha diferencia es menor se detiene el proceso de iteración.
- **init** → modelo de inicialización de los centroides: 'rand' para inicialización aleatoria y 'fkm' para inicialización con Fuzzy K-means (FKM)

La función devuelve la partición difusa resultado de aplicar la función $BetP(c)$ (Ecuación 4.6) a la partición de creencia, los centroides V , la matriz que contiene los valores de la función de masa M , y el valor de la función J_{CEKM} .

5.3.4 Función para aplicar LCVQE

La función `LCVQE` es la encargada de aplicar el algoritmo LCVQE (Sección 4.4) a un conjunto de datos dado. A continuación se muestra el prototipo de la función y se detallan sus parámetros:

```
def LCVQE(X, K, constraints, max_iter = 300,  
          centroids = None)
```

- **X** → matriz de $n \times p$ que contiene el conjunto de datos
- **K** → número de clusters de la partición resultado
- **constraints** → matriz de restricciones de $|R| \times 3$, en la que cada fila responde al formato $\{x_1, x_2, r\}$, donde x_1 y x_2 son las instancias implicadas en la restricción, y r indica el tipo de la misma (1 para ML y -1 para CL).
- **centroids** → centroides iniciales
- **max_iter** → límite de iteraciones para el algoritmo

La función devuelve la lista de longitud n que indica la pertenencia de cada instancia a un cluster dado, es decir, la partición del conjunto de datos X , los centroides V asociados a los clusters, el número de iteraciones realizadas y el valor de la función $LCVQE$.

5.3.5 Función para aplicar RDPM

La función RDPM es la encargada de aplicar el algoritmo RDPM (Sección 4.5) a un conjunto de datos dado. A continuación se muestra el prototipo de la función y se detallan sus parámetros:

```
def RDPM(X, lamb, constraints, max_iter = 300, xi_0 = 0.1,  
          xi_rate = 1)
```

- **X** → matriz de $n \times p$ que contiene el conjunto de datos
- **lamb** → distancia mínima a la que se considera que una instancia no pertenece a un cluster, medida desde el centroide del mismo.
- **constraints** → matriz de $n \times n$ que contiene el conjunto de restricciones. Siendo i y j los índices que indican filas y columnas respectivamente, un 1 en la matriz indica una restricción ML entre i y j , un -1 indica una de tipo CL, y un 0 indica que no existe restricción.
- **max_iter** → límite de iteraciones para el algoritmo
- **xi_0** → parámetro ξ_0 del Algoritmo 4, controla el compromiso entre el peso del conjunto de datos y el del conjunto de restricciones.

- **xi_rate** → parámetro ξ_{rate} del Algoritmo 4, controla el ratio de incremento de ξ_0 .

La función devuelve la lista de longitud n que indica la pertenencia de cada instancia a un cluster dado, es decir, la partición del conjunto de datos X , y el número de clusters presentes en ella.

5.3.6 Función para aplicar TVClust

```
def TVClust(X, K, constraints, max_iter = 300,
              is_keep_l = 1, alpha0 = 1.2, stop_thr = 0.0005)
```

- **X** → matriz de $n \times p$ que contiene el conjunto de datos
- **K** → número de clusters de la partición resultado
- **constraints** → matriz de $n \times n$ que contiene el conjunto de restricciones. Siendo i y j los índices que indican filas y columnas respectivamente, un 1 en la matriz indica una restricción ML entre i y j , un 0 indica una de tipo CL, y un -1 indica que no existe restricción.
- **max_iter** → límite de iteraciones para el algoritmo
- **is_keep_l** → booleano que indica si se debe o no monitorizar la métrica de convergencia.
- **alpha** → parámetro α involucrado en el Proceso de Dirichlet (para más detalles consultar Sección 4.5).
- **stop_thr** → umbral por debajo del cual la mejora en la función objetivo de TVClust no se considera significativa y se detiene el proceso de iteración.

La función devuelve la lista de longitud n que indica la pertenencia de cada instancia a un cluster dado, es decir, la partición del conjunto de datos X , y el número de clusters presentes en ella.

6

EXPERIMENTACIÓN

Esta sección tiene como objetivo mostrar el correcto funcionamiento las funciones documentadas en la Sección 5. Para ello las aplicaremos sobre conjuntos de datos correspondientes a casos de aplicación reales, así como a conjuntos artificiales.

Cabe destacar que los parámetros dados como argumento a las funciones son los especificados por defecto en la Sección 5 a no ser que se especifique lo contrario, es decir, no se han optimizado para los datos concretos a los que se aplican en cada ocasión. La optimización de parámetros queda a cargo del usuario, que deberá realizar un estudio sobre los mismos para adaptarlos al problema concreto que intenta resolver.

6.1 CONJUNTOS DE DATOS CONSIDERADOS

Tal y como ya se ha mencionado, tomaremos conjuntos de datos reales y artificiales para poner a prueba los 5 métodos implementados. Teniendo en cuenta dicha distinción, a continuación se detallan los pormenores de cada uno de ellos.

6.1.1 *Conjuntos de datos reales*

Consideraremos 4 conjuntos de datos correspondientes a casos reales de aplicación de técnicas de Aprendizaje Automático (AA):

- **Conjunto de datos Iris:** el conjunto de datos Iris (*Iris dataset*) es uno de los más empleados en AA. Famoso por ser objeto del primer intento de aplicación de métodos de clustering por el biólogo Ronald Fisher en 1936, quien intentaba obtener un método para clasificar flores de la especie Iris en sus tres subespecies: *iris setosa*, *iris virginica* e *iris versicolor*. Compuesto por un total de 150 muestras de 3 clases distintas, caracterizadas cada una por 4 atributos en el dominio de los números reales positivos, a saber: altura del sépalo, anchura del sépalo, altura del pétalo y anchura del pétalo. Por facilidad de representación solo consideraremos las dos primeras características, la figura 6.1 muestra la distribución de las mismas.
- **Conjunto de datos Wine:** el conjunto de datos Wine (*Wine dataset*) recoge 178 muestras de 3 clases de vinos distintas, caracterizadas cada una por 13 atributos en el dominio de los números

reales positivos. Algunos de estos atributos son: contenido de alcohol, contenido de ácido málico, contenido de magnesio, etc.

- **Conjunto de datos Breast Cancer:** el conjunto de datos Breast Cancer (*Breast Cancer dataset*) recoge 569 asociadas cada una a una paciente que padecía o no de cáncer de mama (2 clases), cada muestra viene caracterizada por 30 atributos en el dominio de los números reales positivos. Algunos de los atributos son: edad de la paciente, tamaño del tumor, mama afectada, etc.
- **Conjunto de datos Glass:** el conjunto de datos Glass (*Glass dataset*) recoge 214 muestras de 6 clases de cristal distintas, caracterizadas cada una por 10 atributos en multitud de dominios. Algunos de estos atributos son: indice reactivo, contenido en magnesio, contenido en aluminio, finalidad de uso, etc.
- **Conjunto de datos Digits:** el conjunto de datos Digits (*Digits dataset*) recoge 1797 muestras de 10 clases de distintas, correspondientes a los dígitos manuscritos del 0 al 9. Cada muestra viene caracterizada por 64 atributos en el dominio de los enteros del 1 al 16.

La tabla 6.1 muestra de manera resumida las características de los estos 5 conjuntos de datos.

Conjuntos de datos reales				
Nombre	Muestras	Atributos	Clases	Dominio
Iris	150	4	3	\mathbb{R}^+
Wine	178	13	3	\mathbb{R}^+
Breast Cancer	569	30	2	\mathbb{R}^+
Glass	214	10	6	Variados
Digits	1797	64	10	$\{0 - 16\}$

Tabla 6.1: Características de los conjuntos de datos reales considerados

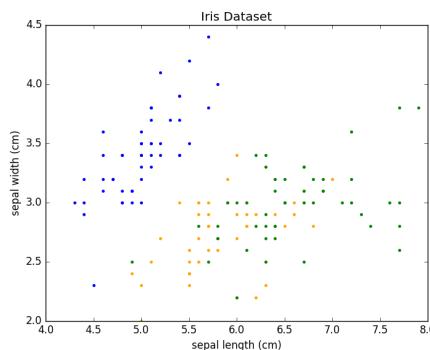


Figura 6.1: Distribución de las dos primeras características de *Iris Dataset*

6.1.2 Conjuntos de datos artificiales

En el caso de los conjuntos de datos artificiales el objetivo es generar clusters con una geometría concreta, para ello sólo son necesarios dos atributos (x e y) y un número de muestras arbitrario.

La Figura 6.2 muestran los 4 conjuntos de datos generados de manera artificial. Por simplicidad nos referiremos a cada uno de ellos por su identificador en inglés, que aparece en la anotación bajo cada imagen.

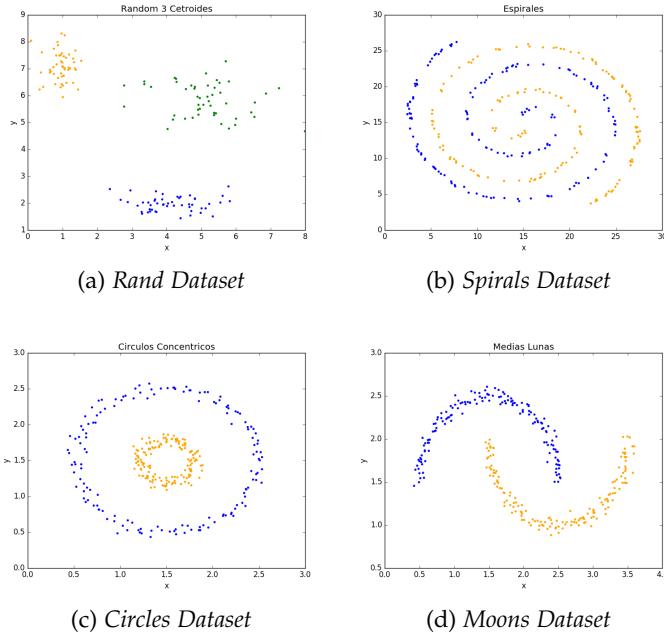


Figura 6.2: Conjuntos de datos artificiales.

6.2 MEDIDA DEL ERROR

Tomamos la medida del error que Wagstaff et al. (2001) [7], los autores de COP-K-medias, proponen en su trabajo para evaluar los resultados.

Dado que en la experimentación disponemos de las etiquetas verdaderas asociadas a cada uno de los conjuntos de datos, podemos hacer uso de las mismas en el post-procesado para evaluar los resultados que proporciona cada método.

Para calcular la exactitud de las predicciones resultado de cada método emplearemos *Rand Index* [19], que calcula el grado de similitud entre dos particiones dadas P_1 y P_2 del mismo conjunto de datos X .

Interpretamos cada partición como una colección de $n(n - 1)/2$ decisiones de emparejamiento, donde n es el número de instancias en X . Para cada par de instancias x_i y x_j en X , P_i las asigna al mismo

cluster o a clusters diferentes. Tomamos a como el número de emparejamientos donde x_i está en el mismo cluster que x_j en P_1 y P_2 , y tomamos b como el suceso contrario. Entonces, el grado de similitud entre P_1 y P_2 se calcula como:

$$Rand(P_1, P_2) = \frac{a + b}{n(n - 1)/2} \quad (6.1)$$

Esta será la medida del error que emplearemos en todos los experimentos.

6.3 GENERACIÓN DE LAS RESTRICCIONES

Para generar las restricciones haremos uso de la función propuesta para ello en la Sección 5.3.1. La cantidad de restricciones viene definida como un porcentaje aplicado sobre el cardinal de conjunto de datos. Dicho porcentaje se especificará en la sección correspondiente al análisis de cada algoritmo. No obstante, podemos obtener una representación gráfica de las restricciones haciendo uso de las funcionalidades de la biblioteca Matplotlib, la Figura 6.3 muestra las restricciones generadas sobre los conjuntos de datos *Iris Dataset* e *Rand Dataset*, especificando que el número de restricciones debe ser un 30 % del total de los datos disponibles en cada caso.

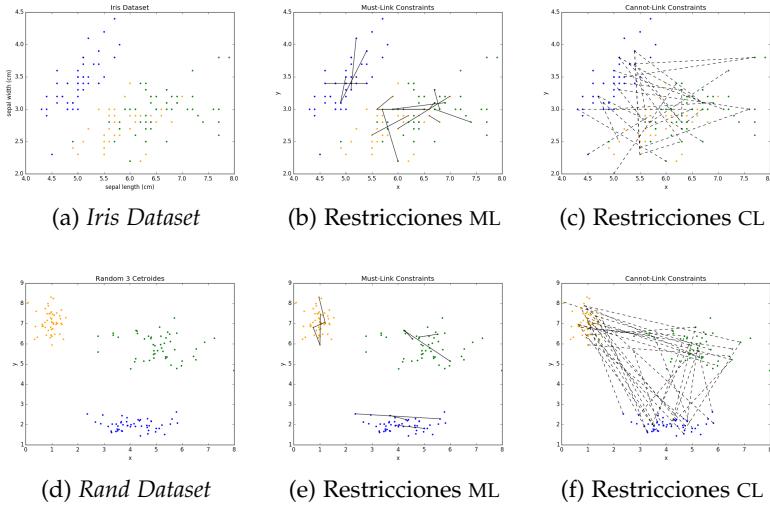


Figura 6.3: Visualización de las restricciones.

6.4 RESULTADOS OBTENIDOS CON COP-K-MEDIAS

La Tabla 6.2 muestra los resultados obtenidos al aplicar el algoritmo COP-K-Medias (Sección 4.2) a los conjuntos de datos presentados en la Sección 6.1. En ella se indican: el valor para *Rand Index*, el tiempo que tarda el algoritmo en proporcionar salida, y el porcentaje de

restricciones ML y CL. El número de restricciones generado para cada conjunto de datos es un 10 % del total de datos disponibles en cada caso, por tanto la suma del porcentaje de restricciones de los dos tipos debe ser aproximadamente 0,1.

Resultados de COP-K-Medias con restricciones CL y ML				
Dataset	RandIndex	Tiempo	% ML	% CL
Iris	0,499	0,0532	0,033	0,066
Wine	0,38	0,087	0,056	0,039
Glass	0,25	0,1449	0,15	0,04
Breast Cancer	NA	NA	0,028	0,07
Digits	0,632	13,306	0,008	0,092
Rand	0,960	0,0346	0,033	0,066
Spirals	0,0245	0,13	0,05	0,05
Circles	-0,002	0,16	0,046	0,053
Moons	NA	NA	0,053	0,046

Tabla 6.2: Resultados obtenidos con COP-K-Medias empleado restricciones CL y ML

De entre los resultados obtenidos destacan los asociados a los conjuntos de datos *Glass Dataset* y *Moons Dataset*. Encontramos que en ninguno de los dos casos el algoritmo COP-K-Medias es capaz de encontrar una solución factible, a pesar de que es seguro que esta existe, ya que las restricciones se generan en base a las etiquetas verdaderas. Esto es consecuencia directa de lo expuesto en la Observación 3.4, puesto que en la resolución del problema se emplean tanto restricciones de tipo ML como CL. Para obtener resultados para *Glass Dataset* y *Moons Dataset* podemos aplicar la solución más simple propuesta en la sección 3.6.3, esto es, suprimir las restricciones de tipo CL y aumentar las de tipo ML. La Tabla 6.3 muestra los resultados con esta nueva configuración.

Resultados de COP-K-Medias con restricciones ML				
Dataset	RandIndex	Tiempo	% ML	% CL
Iris	0,489	0,0423	0,366	0
Wine	0,337	0,0762	0,353	0
Glass	0,175	0,112	0,555	0
Breast Cancer	0,45	0,556	0,224	0
Digits	0,582	10,847	0,105	0
Rand	1,0	0,031	0,393	0
Spirals	-0,0028	0,1302	0,52	0
Circles	0,091	0,1301	0,463	0
Moons	0,126	0,116	0,513	0

Tabla 6.3: Resultados obtenidos con COP-K-Medias empleado solo restricciones ML

Como cabía esperar, los resultados de la Tabla 6.3 muestran que empleado solo restricciones de tipo ML el algoritmo COP-K-Medias es capaz de dar una partición para todos los conjuntos de datos. Sin embargo la precisión de las predicciones decrece en algunos casos, esto se debe la pérdida de la información que proporcionaban las restricciones CL.

Además de obtener los resultados numéricos podemos representar algunas de las particiones obtenidas con COP-K-Medias para obtener una mejor idea del comportamiento del algoritmo. En la Figura 6.4 se muestran los resultados obtenidos para *Iris Dataset* y *Rand Dataset*, los cuadros (a) y (b) muestran los resultados considerando restricciones ML y CL, mientras que los (c) y (d) solo consideran restricciones ML.

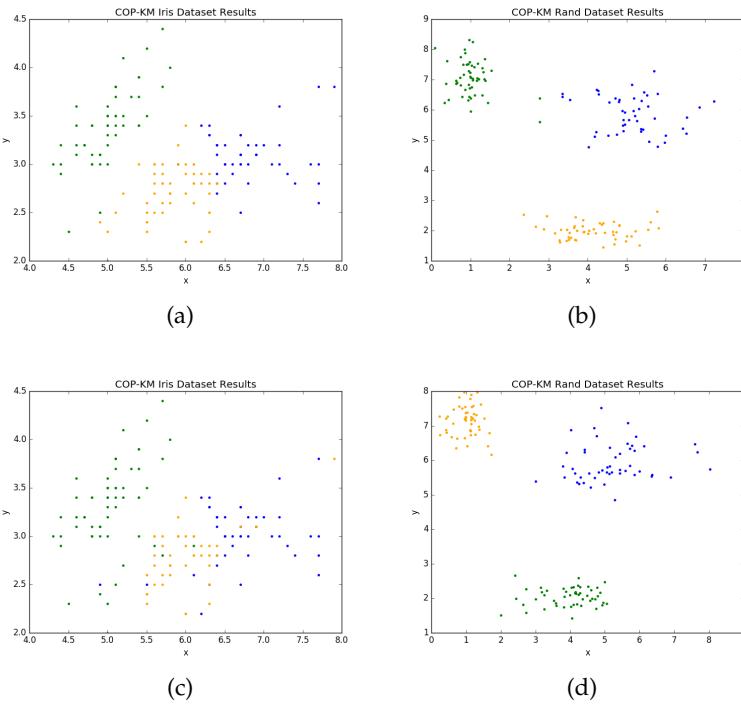


Figura 6.4: Visualización de los resultados de COP-K-Medias

En la Figura 6.4 destacan las instancias del conjunto de datos *Iris Dataset* que parecen estar dentro de un cluster distinto al que pertenecen. Es en estos casos en los que destaca la influencia de las restricciones en el proceso de asignación de instancias a clusters, el algoritmo K-Medias (Apéndice A) nunca proporcionaría esta partición como resultado.

6.5 RESULTADOS OBTENIDOS CON CEKM

La Tabla 6.4 muestra los resultados obtenidos al aplicar el algoritmo Constrained Evidential K-means (CEKM) (Sección 4.3) a los conjuntos de datos presentados en la Sección 6.1. Emplearemos un número de restricciones igual al 25 % del total de ejemplos disponibles para cada conjunto de datos.

Resultados obtenidos con CEKM		
Dataset	RandIndex	Tiempo
Iris	0,554	9,567
Wine	0,395	14,48
Glass	NA	NA
Breast Cancer	0,491	64,22
Digits	NA	NA
Rand	0,950	9,411
Spirals	0,061	9,839
Circles	-0,0030	12,588
Moons	0,247	11,191

Tabla 6.4: Resultados obtenidos con CEKM

En ella destacan los resultados obtenidos para los conjuntos de datos *Glass Dataset* y *Breast Cancer Dataset*. El algoritmo CEKM no es capaz de dar resultado para estos dos conjuntos de datos, debido a que las estructuras de memoria que lo soportan crecen de manera exponencial con K . La figura 6.5 muestra los resultados obtenidos con CEKM para los conjuntos de datos *Iris Dataset* y *Rand Dataset*.

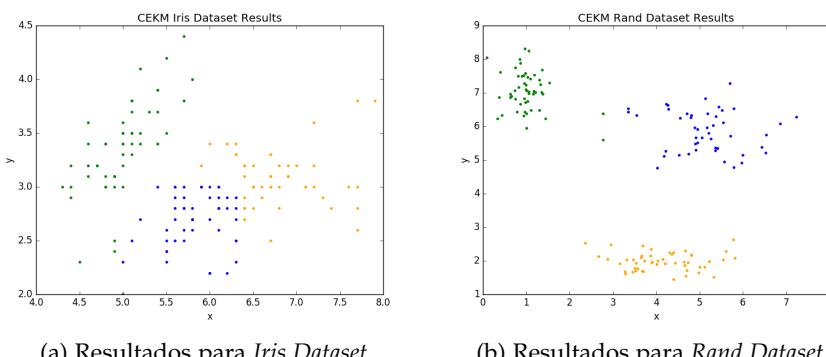


Figura 6.5: Visualización de los resultados de CEKM

6.6 RESULTADOS OBTENIDOS CON LCVQE

La Tabla 6.4 muestra los resultados obtenidos al aplicar el algoritmo Linear Constrained Vector Quantization Error (LCVQE) (Sección 4.4) a los conjuntos de datos presentados en la Sección 6.1. Daremos como parámetro para los centroides iniciales los calculados con el algoritmo K-Medias (Apéndice A). Respecto al número de restricciones empleadas, será un 25 % del total de ejemplos disponibles para cada conjunto de datos.

Resultados obtenidos con LCVQE		
Dataset	RandIndex	Tiempo
Iris	0,663	0,141
Wine	0,390	0,002
Glass	0,197	0,002
Breast Cancer	0,616	0,003
Digits	0,680	0,062
Rand	0,989	0,001
Spirals	0,057	0,002
Circles	-0,003	0,005
Moons	0,215	0,002

Tabla 6.5: Resultados obtenidos con LCVQE

De nuevo, podemos obtener una representación de algunos de los mejores resultados obtenidos con LCVQE. La figura 6.6 muestra los resultados obtenidos con LCVQE para los conjuntos de datos *Iris Dataset* y *Rand Dataset*.

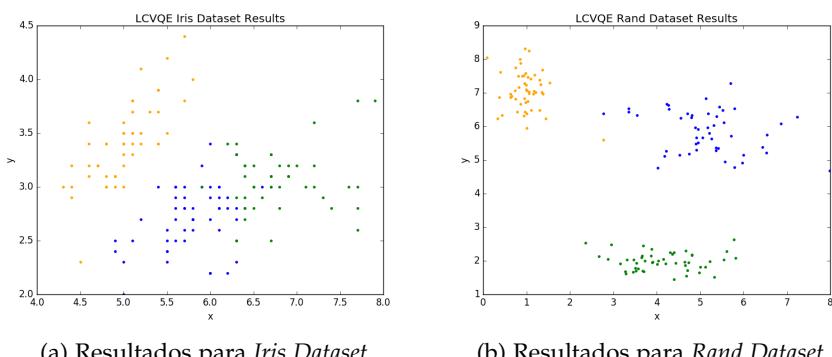


Figura 6.6: Visualización de los resultados de LCVQE

6.7 RESULTADOS OBTENIDOS CON RDP-MEANS

La Tabla 6.6 muestra los resultados obtenidos al aplicar el algoritmo Relational Dirichlet Process - Means (RDPM) (Sección 4.5) a los conjuntos de datos presentados en la Sección 6.1. Para aplicar RDPM es necesario especificar el parámetro λ , en este caso este parámetro se calcula de manera experimental y en base a las distancia medias entre instancias dentro del conjunto de datos. De igual forma que en el caso anterior, especificaremos el número de restricciones como un 25 % del total de ejemplos disponibles para cada conjunto de datos.

Resultados obtenidos con RDPM				
Dataset	RandIndex	Tiempo	λ	K_{out}
Iris	0,479	0,508	1,7	3
Wine	0,342	0,598	438,11	3
Glass	0,255	0,785	2,214	6
Breast Cancer	0,501	14,096	2833,35	3
Digits	0,665	177,12	44,14	17
Rand	0,98	0,365	4	3
Spirals	0,0147	2,683	13,83	3
Circles	-0,003	1,27	1,5	2
Moons	0,234	1,563	2	2

Tabla 6.6: Resultados obtenidos con RDPM

En la Tabla 6.6 la columna K_{out} representa el número de clases de la partición de salida proporcionada por LCVQE. Podemos estudiar la influencia del parámetro λ sobre este resultado, para ello ejecutamos el algoritmo especificando en cada caso un λ mayor. La Tabla 6.7 muestra los resultados obtenidos con esta nueva configuración.

Resultados obtenidos con RDPM				
Dataset	RandIndex	Tiempo	λ_{mayor}	K_{out}
Iris	0,321	0,683	2	2
Wine	0,313	0,691	538,11	3
Glass	0	0,505	4,214	3
Breast Cancer	0,491	11,236	3833,35	2
Digits	0,268	144,96	60,14	4
Rand	0	0,26	7	1
Spirals	0	0,969	20,5	1
Circles	0	0,965	4,5	1
Moons	0	0,97	5	1

Tabla 6.7: Resultados obtenidos con RDPM especificando un mayor para λ

A la vista de los resultados expuestos en la Tabla 6.7 podemos decir que escoger λ es equivalente a escoger K en los algoritmos que toman como parámetro el número de clusters de la partición de salida. Por tanto es crucial realizar un estudio sobre este parámetro para resolver el problema concreto al que nos enfrentemos. En la Figura 6.7 representamos los resultados obtenidos con las diferentes elecciones de λ para los conjuntos de datos *Iris Dataset* y *Rand Dataset*.

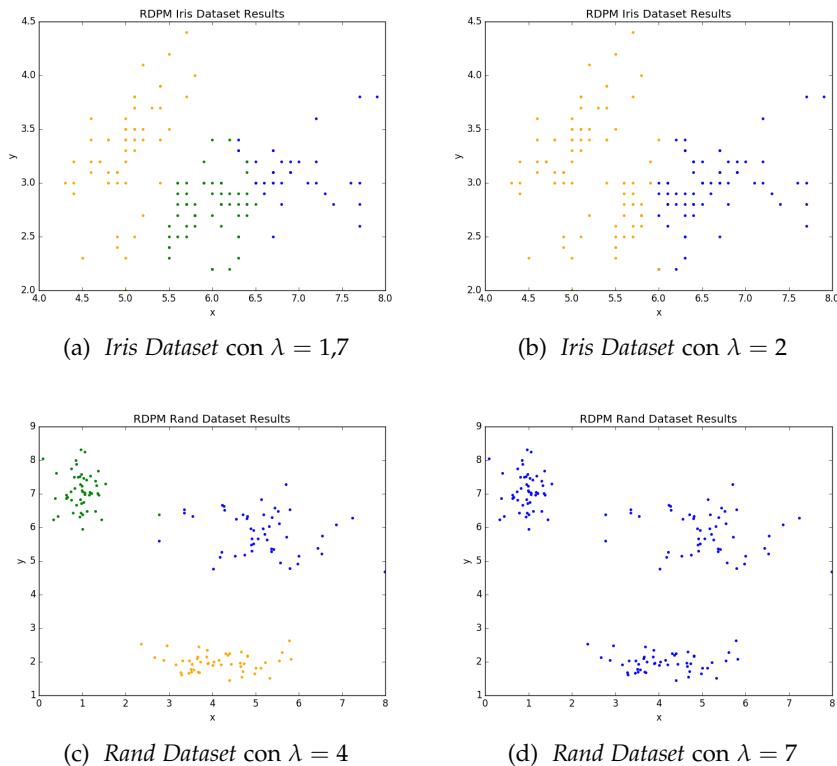


Figura 6.7: Visualización de los resultados de RDPM con diferentes valores de λ

La Figura 6.7 refleja claramente la influencia del parámetro λ en la partición que RDPM proporciona como resultado del proceso de clustering que lleva a cabo. En el caso del conjunto de datos *Iris Dataset*, bastan 3 décimas de diferencia en λ para que el resultado degenera en una partición de dos clusters en lugar de 3.

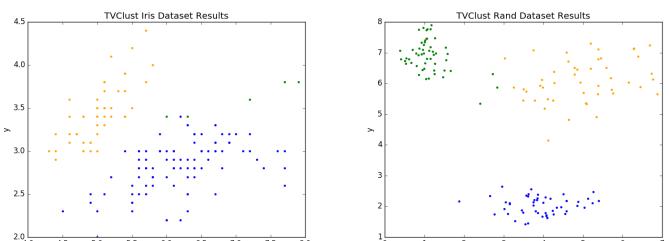
6.8 RESULTADOS OBTENIDOS CON TVCLUST

La Tabla 6.7 muestra los resultados obtenidos al aplicar el algoritmo Two Views Clustering (TVClust) (Sección 4.5) a los conjuntos de datos presentados en la Sección 6.1. Especificamos el número de restricciones como un 25 % del total de ejemplos disponibles para cada conjunto de datos, tal y como hicimos en los casos anteriores.

Resultados obtenidos con TVClust		
Dataset	RandIndex	Tiempo
Iris	0,532	0,342
Wine	0,176	0,5
Glass	0,26	3,332
Breast Cancer	0	1,3
Digits	0,680	78,46
Rand	0,941	0,83
Spirals	0,042	0,3
Circles	0,075	1,7
Moons	0,576	5,59

Tabla 6.8: Resultados obtenidos con TVClust

Destaca el resultado obtenido para el conjunto de datos *Breast Cancer Dataset*, para el que TVClust no es capaz de encontrar una partición mínimamente acertada. Esto puede deberse a la alta dimensionalidad del conjunto de datos. La Figura 6.7 muestra los resultados obtenidos con TVClust para los conjuntos de datos *Iris Dataset* y *Rand Dataset*.



(a) Resultados para *Iris Dataset* (b) Resultados para *Rand Dataset*

Figura 6.8: Visualización de los resultados de TVClust

6.9 COMPARATIVA GENERAL DE RESULTADOS

La Tabla 6.9 muestra los resultados obtenidos por cada algoritmo para cada conjunto de datos, en lo que a precisión en las predicciones se refiere. Salvo algunas excepciones, todos los algoritmos se comportan de manera similar bajo las condiciones descritas al inicio de la Sección 6. Destaca la precisión en las predicciones sobre el conjunto de datos *Rand Dataset*, así como la pobreza de las mismas sobre el conjunto *Circles Dataset*.

Comparativa General de Resultados (RandIndex)					
Dataset	COPKM	CEKM	LCVQE	RDPM	TVClust
Iris	0,489	0,554	0,663	0,479	0,532
Wine	0,337	0,395	0,390	0,342	0,176
Glass	0,175	NA	0,197	0,255	0,26
Breast Cancer	0,45	0,491	0,616	0,501	0
Digits	0,582	NA	0,680	0,665	0,680
Rand	1,0	0,950	0,989	0,98	0,941
Spirals	-0,0028	0,061	0,057	0,0147	0,042
Circles	0,091	-0,003	-0,003	-0,003	0,075
Moons	0,126	0,247	0,215	0,234	0,576

Tabla 6.9: Comparativa general de resultados (RandIndex)

En lo que a tiempo de ejecución se refiere, la Tabla 6.10 muestra las marcas obtenidas por cada algoritmo. Cabe destacar que, en los casos en los que un método toma menos de 1 segundo en proporcionar resultado, cualquier otro proceso ejecutando en la máquina puede perturbar las mediciones, por tanto no han de tomarse como medidas exactas, simplemente como indicadores generales.

Comparativa General de Resultados (Tiempo)					
Dataset	COPKM	CEKM	LCVQE	RDPM	TVClust
Iris	0,0423	9,567	0,141	0,508	0,683
Wine	0,0762	14,48	0,002	0,598	0,691
Glass	0,112	NA	0,002	0,785	0,505
Breast Cancer	0,556	64,22	0,003	14,096	11,236
Digits	10,847	NA	0,062	177,12	144,96
Rand	0,031	9,411	0,001	0,365	0,26
Spirals	0,1302	9,839	0,002	2,683	0,969
Circles	0,1301	12,588	0,005	1,27	0,965
Moons	0,116	11,191	0,002	1,563	0,97

Tabla 6.10: Comparativa general de resultados (Tiempo)

7

CONCLUSIONES

7.1 TRABAJO FUTURO

A

EL ALGORITMO K-MEDIAS

El algoritmo K-medias (KM) es uno de los métodos más básicos para aplicar clustering. Fue ideado por Hugo Steinhaus en 1957, aunque no fue aplicado a un caso real hasta 1967, por James MacQueen [31].

El procedimiento que aplica K-medias es simple: asignar cada instancia al cluster cuyo centroide se encuentre más cercano a ella, y calcular los centroides en base a las instancias asignadas al cluster asociado al mismo. De esta manera, siguiendo la notación introducida en la sección 4, la regla de actualización de los centroides es:

$$v_i = \frac{1}{|c_i|} \sum_{x_i \in c_i} x_i \quad (\text{A.1})$$

Así, si cada instancia se asigna al cluster correspondiente al centroide más cercano, la función de error que minimiza K-medias es:

$$ERR = \frac{1}{2} \sum_{j=1}^k \sum_{x_i \in c_j} (v_j - x_i)^2 \quad (\text{A.2})$$

Formalizando la regla de asignación obtenemos:

$$c^i = \operatorname{argmin}(\|x_i - v_j\|^2) \quad t.q. \quad j \in \{1, \dots, k\} \quad (\text{A.3})$$

donde c^i representa el cluster seleccionado para la instancia x_i . Con todo, el proceso que K-medias sigue para obtener una partición de los datos queda resumido en el Algoritmo 5.

Algoritmo 5: K-medias

Entrada: Conjunto de datos X , numero de cluster resultantes k .

Salida: Partición P del conjunto de datos X , centroides V .

función K-medias(X, k) **begin**

- 1. Inicialización aleatoria de los centroides
 $V = \{v_1, \dots, v_k\}$.
- 2. Asignar cada instancia $x_i \in X$, al centroide más cercano c_j siguiendo la ecuación A.3.
- 3. Para cada cluster c_i , actualizar su centroide aplicando la ecuación A.1
- 4. Iterar entre (2.) y (3.) hasta converger.
- 5. **return** C, V

end

BIBLIOGRAFÍA

- [1] Ryszard S Michalski, Jaime G Carbonell y Tom M Mitchell. *Machine learning: An artificial intelligence approach*. Springer Science & Business Media, 2013.
- [2] Brian Everitt S., Sabine Landau, Morven Leese y Stahl Daniel. *Custer Analysis*. Berlin, Germany: WILEY, 2011.
- [3] Ian Davidson y Sugato Basu. «A survey of clustering with instance level». En: *Constraints* 1 (2007).
- [4] David Cohn, Rich Caruana y Andrew McCallum. «Semi-supervised clustering with user feedback». En: *Constrained Clustering: Advances in Algorithms, Theory, and Applications* 4.1 (2003), págs. 17-32.
- [5] Kiri Wagstaff y Claire Cardie. *Clustering with Instance-level Constraints*. 2000, págs. 1103-1110.
- [6] Ian Davidson y SS Ravi. «Hierarchical clustering with constraints: Theory and practice». En: vol. 14. 1. Springer, 2007, págs. 25-61.
- [7] Kiri Wagstaff, Claire Cardie, Seth Rogers, Stefan Schrödl y col. «Constrained k-means clustering with background knowledge». En: *Proceedings of the Eighteenth International Conference on Machine Learning*. 2001, págs. 577-584.
- [8] Ian Davidson y SS Ravi. «Agglomerative hierarchical clustering with constraints: Theoretical and empirical results». En: *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer. 2005, págs. 59-70.
- [9] Sugato Basu, Arindam Banerjee y Raymond J Mooney. «Active semi-supervision for pairwise constrained clustering». En: *Proceedings of the 2004 SIAM international conference on data mining*. SIAM. 2004.
- [10] Eran Segal, Haidong Wang y Daphne Koller. «Discovering molecular pathways from protein interaction and gene expression data». En: *Bioinformatics* 19.suppl_1 (2003), págs. i264-i272.
- [11] Ian Davidson y SS Ravi. «Clustering with constraints: Feasibility issues and the k-means algorithm». En: *Proceedings of the 2005 SIAM international conference on data mining*. SIAM. 2005, págs. 138-149.
- [12] Martin HC Law, Alexander Topchy y Anil K Jain. «Model-based clustering with probabilistic constraints». En: *Proceedings of the 2005 SIAM international conference on data mining*. SIAM. 2005, págs. 641-645.

- [13] Ayhan Demiriz, Kristin P Bennett y Mark J Embrechts. «Semi-supervised clustering using genetic algorithms». En: *Artificial neural networks in engineering (ANNIE-99)* (1999), págs. 809-814.
- [14] Janne Sinkkonen y Samuel Kaski. *Semisupervised clustering based on conditional distributions in an auxiliary space*. Helsinki University of Technology, 2000.
- [15] Sugato Basu, Arindam Banerjee y Raymond Mooney. «Semi-supervised clustering by seeding». En: *In Proceedings of 19th International Conference on Machine Learning (ICML-2002)*. Citeseer. 2002.
- [16] Rong Yan, Jian Zhang, Jie Yang y Alexander G Hauptmann. «A discriminative learning framework with pairwise constraints for video object classification». En: *IEEE transactions on pattern analysis and machine intelligence* 28.4 (2006), págs. 578-593.
- [17] Ioannis Xenarios, Esteban Fernandez, Lukasz Salwinski, Xiaoqun Joyce Duan, Michael J Thompson, Edward M Marcotte y David Eisenberg. «DIP: the database of interacting proteins: 2001 update». En: *Nucleic acids research* 29.1 (2001), págs. 239-241.
- [18] Aharon Bar-Hillel, Tomer Hertz, Noam Shental y Daphna Weisz-hall. «Learning distance functions using equivalence relations». En: *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*. 2003, págs. 11-18.
- [19] William M Rand. «Objective criteria for the evaluation of clustering methods». En: *Journal of the American Statistical association* 66.336 (1971), págs. 846-850.
- [20] Kiri Lou Wagstaff y Claire Cardie. «Intelligent clustering with instance-level constraints». En: (2002).
- [21] Ian Davidson, SS Ravi y Kiri Wagstaff. 2006.
- [22] Ian Davidson y SS Ravi. «Identifying and generating easy sets of constraints for clustering». En: *AAAI*. Vol. 6. 2006, pág. 336341.
- [23] Violaine Antoine, Benjamin Quost, M-H Masson y Thierry Denoeux. «CECM: Constrained evidential C-means algorithm». En: *Computational Statistics & Data Analysis* 56.4 (2012), págs. 894-914.
- [24] Dan Pelleg y Dorit Baras. *K-means with large and noisy constraint sets*. 2007, págs. 674-682.
- [25] Daniel Khashabi, John Wieting, Jeffrey Yufei Liu y Feng Liang. «Clustering With Side Information: From a Probabilistic Model to a Deterministic Algorithm». En: *arXiv preprint arXiv:1508.06235* (2015).
- [26] Ke Jiang, Brian Kulis y Michael I Jordan. «Small-variance asymptotics for exponential family Dirichlet process mixture models». En: (2012), págs. 3158-3166.

- [27] David Blackwell y James B MacQueen. «Ferguson distributions via Pólya urn schemes». En: *The annals of statistics* (1973), págs. 353-355.
- [28] Arindam Banerjee, Srujana Merugu, Inderjit S Dhillon y Joydeep Ghosh. «Clustering with Bregman divergences». En: *Journal of machine learning research* 6.Oct (2005), págs. 1705-1749.
- [29] Jürgen Forster y Manfred K Warmuth. «Relative expected instantaneous loss bounds». En: *Journal of Computer and System Sciences* 64.1 (2002), págs. 76-102.
- [30] Babaki Behrouz. «COP-Kmeans version 1.5». En: (2017). DOI: [10.5281/zenodo.831850](https://doi.org/10.5281/zenodo.831850). URL: <https://doi.org/10.5281/zenodo.831850>.
- [31] James MacQueen y col. «Some methods for classification and analysis of multivariate observations». En: *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. Vol. 1. 14. Oakland, CA, USA. 1967, págs. 281-297.

DECLARACIÓN

Put your declaration here.

Granada, Junio 2018

Germán González Almagro

COLOFÓN

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede and Ivo Pletikosić. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". `classicthesis` is available for both L^AT_EX and LyX:

<https://bitbucket.org/amiede/classicthesis/>

Happy users of `classicthesis` usually send a real postcard to the author, a collection of postcards received so far is featured here:

<http://postcards.miede.de/>

Thank you very much for your feedback and contribution.

Final Version as of 31 de mayo de 2018 (`classicthesis` version 4.4).