UNIVERSIDAD NACIONAL DEL ALTIPLANO - PUNO
FACULTAD DE INGENIERÍA MECÁNICA ELÉCTRICA, ELECTRÓNICA Y SISTEMAS
ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

TESIS

ORGANIZACIÓN DE DATOS MULTIDIMENSIONALES EN UN SISTEMA DE RECOMENDACIONES BASADO EN DATA CLUSTERING E INTELIGENCIA DE ENJAMBRES

PRESENTADO POR:

ALVARO JORGE PACOMPIA LARA

PARA OPTAR EL TÍTULO PROFESIONAL DE:

INGENIERO DE SISTEMAS

PUNO – PERÚ

2017

UNIVERSIDAD NACIONAL DEL ALTIPLANO
FACULTAD DE INGENIERÍA MECÁNICA ELÉCTRICA, ELECTRÓNICA Y SISTEMAS
ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

TESIS

ORGANIZACIÓN DE DATOS MULTIDIMENSIONALES EN UN SISTEMA DE RECOMENDACIONES BASADO EN DATA CLUSTERING E INTELIGENCIA DE ENJAMBRES

PRESENTADO POR:

ALVARO JORGE PACOMPIA LARA

PARA OPTAR EL TÍTULO PROFESIONAL DE:

INGENIERO DE SISTEMAS

FECHA DE SUSTENTACIÓN: 29/12/2017

APROBADA POR:

PRESIDENTE : Dr. MARIO ANTONIO SUAREZ LOPEZ

PRIMER MIEMBRO : M.Sc. HUGO YOSEF GÓMEZ QUISPE

SEGUNDO MIEMBRO : M.Sc. WILLIAM EUSEBIO ARCAYA COAQUIRA

DIRECTOR / ASESOR : M.Sc. EDGAR HOLGUIN HOLGUIN

Área : **Inteligencia Artificial**

Tema : **Data Clustering, Inteligencia de Enjambres**

DEDICATORIA

Dedico esta tesis a quienes inspiraron mi espíritu para la conclusión de esta tesis en tecnología. A mis padres quienes me dieron la vida, educación, apoyo y consejos. A mis compañeros de estudio, a mis maestros y amigos, quienes sin su ayuda nunca hubiera, podido hacer esta tesis. A todos ellos se los agradezco desde el fondo de mi alma. Para todos ellos hago esta dedicatoria.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE ECUACIONES

ÍNDICE DE TABLAS

ÍNDICE DE ANEXOS

ÍNDICE DE ACRÓNIMOS

BRPSO: Boundary Restriction – Particle Swarm Optimization.

PSO: Optimización por Enjambre de Partículas.

RS: Sistema de Recomendaciones.

SC: Subtractive Clustering.

SI: Inteligencia de Enjambres.

RESUMEN

La búsqueda de información relevante dentro de grandes cantidades de datos es conocida como Data Mining. Las herramientas de Data Mining permiten la extracción de cualquier tipo de información oculta pero significativa, la cual permite mejorar el proceso de toma de decisiones. Dentro de estas herramientas se encuentran los Sistemas de Recomendaciones, que están diseñadas para ayudar a diferentes usuarios el navegar dentro de sistemas complejos que presenten una gran cantidad de información; como lo pueden ser los sitios de entretenimiento, tiendas en línea, buscadores, etc.; mediante el empleo de características de los usuarios o de la similitud entre los ítems almacenados dentro de las bases de datos de estos sistemas.

A partir de esta premisa, se puede hacer empleo de una rama del Data Mining conocida como Data Clustering, el cual estudia técnicas de agrupamiento de datos; siguiendo ciertas características o diferencias que puedan existir entre ellas. Estas técnicas emplean diversas formas de realizar esta tarea en particular; pero para mejorar el rendimiento y la eficiencia con la que realizan el trabajo, normalmente se suelen ser combinadas con técnicas de optimización, como lo son las técnicas de la rama de Inteligencia de Enjambres.

La Inteligencia de Enjambres es una rama dentro de la Inteligencia Artificial que estudia el comportamiento colectivo de sistemas descentralizados auto-organizados, suelen ser sistemas bio-inspirados. Estos modelos suelen ser utilizados para resolver problemas de optimización; como en este caso es el agrupamiento de datos; resultando además de tener cierta facilidad de entendimiento e implementación en sistemas computacionales.

Así, la investigación realizada tiene como objetivo principal el desarrollo de un Sistema de Recomendaciones, construido a partir de dos técnicas: Subtractive Clustering; técnica de Data Mining; y Particle Swarm Optimization; técnica de Inteligencia de Enjambres.

Luego de implementar los algoritmos se analizaron el resultado de cada dataset con cada una de las métricas descritas para compararlas entre sí, lo que permitió determinar la eficiencia de cada uno de los algoritmos propuestos y, además, permitió generar las sugerencias necesarias para posibles casos de uso en un entorno real.

Palabras Clave: Data Clustering, Data Standardization, Data Mining, Inteligencia de Enjambres, Sistema de Recomendaciones.

ABSTRACT

The search for relevant information within large amounts of data is known as Data Mining. The Data Mining tools allow the extraction of any type of hidden but significant information, which allows to improve the decision making process. Within these tools are the Recommendations Systems, which are designed to help different users navigate within complex systems that present a large amount of information; like entertainment sites, online stores, search engines, etc.; using the characteristics of the users or the similarity between the items stored within the databases of these systems.

From this premise, it´s posible to use a branch of Data Mining known as Data Clustering, which studies data grouping techniques; following certain characteristics or differences that may exist between them. These techniques employ different ways of performing this particular task; but to improve the performance and efficiency with which they do their work, they are usually combined with optimization techniques, such as swarm intelligence techniques.

Swarm Intelligence is a branch within Artificial Intelligence that studies the collective behavior of self-organized decentralized systems, usually bio-inspired systems. These models are usually used to solve optimization problems; as in this case it is the grouping of data; resulting also to have some ease of understanding and implementation in computer systems.

Thus, the research carried out has as main objective the development of a Recommendations System, built from two techniques: Subtractive Clustering; Data Mining technique; and Particle Swarm Optimization; Swarm Intelligence technique.

After implementing the algorithms, the result of each data set was indicated with each of the measurements, a comparison was made between each of them, which allowed to determine the efficiency of each of the proposed algorithms and, in addition, generated the necessary suggestions for possible use cases in a real environment.

Key Words: Data Clustering, Data Standardization, Data Mining, Swarm Intelligence, Recommender System.

CAPÍTULO I

INTRODUCCIÓN

En trabajo de investigación planteado en la siguiente tesis explora el uso de dos campos de estudio: Data Clustering, que encierra el desarrollo de técnicas que permiten el análisis adecuado de información, del cual se esperan obtener conjuntos; denominados como clústeres; en los cuales existan patrones que los permiten diferenciarse de los demás clústeres.

Así también se hace uso de la Inteligencia de Enjambres, las cuales son técnicas; en su mayoría de optimización, que suelen emplearse en diferentes áreas con el fin de mejorar el rendimiento inicial de técnicas utilizadas anteriormente; en este caso de estudio en particular las tareas de Data Clustering.

Por último, ambas técnicas estarán limitadas al propósito de poder ser incorporadas a un Sistema de Recomendaciones, para lo cual también utilizará parte de los modelos de

esta última al diseño de los algoritmos. Cabe recalcar que el trabajo no está orientado al desarrollo de alguna interfaz como tal.

En el **Capítulo I** se realiza una breve descripción del problema; justificación del problema, formulación del objetivo general y los objetivos específicos.

El **Capítulo II** contiene la revisión literaria, centrándose en el desarrollo de los temas de Data Clustering como también de Inteligencia de Enjambres. Además, contiene los antecedentes utilizados durante la investigación de la tesis, así como las hipótesis planteadas durante el desarrollo del mismo.

El **Capítulo III** explica el tipo de investigación realizado. Así como las muestras que se utilizaron durante el ensayo de los experimentos correspondientes a la investigación y los métodos de tratamiento de datos.

En el **Capítulo IV** se muestra explica el desarrollo de los algoritmos empleados en los experimentos, así como los resultados obtenidos al aplicar cada dataset de muestra en los algorimos desarrollados.

El **Capítulo V** contiene las conclusiones a las que se llegó luego de haber concluido los experimentos planteados dentro de la investigación.

Por último, el **Capítulo VI** contiene recomendaciones que se deberá tener en cuenta a futuras investigaciones y trabajos que se realicen en base a este documento.

1.1 DESCRIPCIÓN DEL PROBLEMA

La importancia de recopilar datos relacionados con actividades empresariales o científicas para lograr una ventaja competitiva es ampliamente reconocida. Potentes sistemas para recopilar y gestionar datos en grandes bases de datos son ampliamente utilizados comercialmente. Los conjuntos de datos cada vez más grandes, complejos y de gran dimensión se almacenan en bases de datos.

Aún con todos esos datos disponibles, surge la duda: ¿Estos datos nos ayudan a tomar decisiones más inteligentes? La respuesta puede ser desconcertante, pero en la mayoría de escenarios es, no. Para tomar mejores decisiones se necesita descubrir y comprender los patrones más profundos que puedan residir en esos datos. Los analistas humanos sin herramientas especiales ya no pueden dar sentido a los enormes volúmenes de datos que requieren procesamiento para tomar decisiones informadas. Impulsados por este antecedente, y con ayuda de los avances tecnológicos, se desarrollan nuevas técnicas y herramientas con la capacidad de analizar de forma inteligente y (semi) automática las grandes cantidades de datos, con el objetivo de poder obtener algún conocimiento útil.

1.2 JUSTIFICACIÓN DE INVESTIGACIÓN

Un proceso crucial dentro del Data Mining es la extracción de conocimiento. Generar conocimiento a partir de datos no clasificados requiere el uso de técnicas que suelen ser medianamente sofisticadas. La extracción de conocimiento necesita del desarrollo de sistemas avanzados, que pueden ser aplicados en múltiples campos, como lo son los

resultados de búsqueda de un sistema, desarrollo de algoritmos evolutivos, sector educativo, sitios web, análisis genético; y en el caso de estudio del presente; sistemas de recomendaciones.

Data Clustering; junto a otras técnicas de extracción de conocimiento, es uno de los pilares de estudio dentro del tratamiento de datos. Gran cantidad de algoritmos desarrollados en esta rama permiten el agrupamiento de datos teniendo en cuenta similaridades; de naturaleza distinta dependiendo de qué algoritmo se use; y diferencias. Estas técnicas por lo general siempre van acompañadas de sistemas optimizadores que permitan un mejor desempeño de estos en situaciones de gran cantidad de datos, de aquí que también se realiza el estudio de la Inteligencia de Enjambres, que no es otra cosa que computarizar modelos sociales observados en la naturaleza.

1.3 PROBLEMA GENERAL

¿En qué medida un Sistema de Recomendaciones basado en Data Clustering e Inteligencia de Enjambres realiza la tarea de clasificación de datos?

1.4 OBJETIVOS DE INVESTIGACIÓN

1.4.1 OBJETIVO GENERAL

Desarrollar un motor de Sistema de Recomendaciones mediante el uso de Data Clustering e Inteligencia de Enjambres en datasets multidimensionales.

1.4.2 OBJETIVOS ESPECÍFICOS

- Analizar el uso de métodos de Data Clustering para determinar los centroides del clúster, así como el máximo de clusters necesarios.

- Analizar las diferentes técnicas de Inteligencia de Enjambres que realicen el agrupamiento posterior de los datos restantes hacia los clusters previamente hallados.

- Implementar los algoritmos pertinentes en el Sistema de Recomendaciones.

- Analizar la cohesión de datos, separación de clústeres y el porcentaje de error en los datasets de muestra.

- Analizar la cantidad de clústeres y número de datos por clúster de la solución.

CAPÍTULO II

REVISIÓN DE LITERATURA

2.1 ANTECEDENTES DE INVESTIGACIÓN

- "AN ANALYSIS OF PARTICLE SWARM OPTIMIZATION WITH DATA CLUSTERING-TECHNIQUE FOR OPTIMIZATION IN DATA MINING" AMREEN KHAN, N.G. BAWANE, SONALI BODKHE. INTERNATIONAL JOURNAL ON COMPUTER SCIENCE AND ENGINEERING VOL. 02, NO. 07 - 2010.

La investigación propone el uso del algoritmo PSO para solucionar tareas de Data Clustering. Dentro del entorno de Data Mining se requiere rapidez y presición para la organización de grandes cantidades de datos. Para ello, el uso de técnicas de Inteligencia de Enjambres; más específicamente el algoritmo PSO; emergen como candidatos que contienen los requisitos computacionales necesarios y que han sido aplicados

satisfactoriamente a problemas de clustering en el mundo real. En la investigación también se hace uso de la técnica Fuzzy C-means, el cual proporciona rendimiento mejorado y mayor diversidad en el enjambre.

El resultado de la investigación afirma que el algoritmo PSO es un eficiente optimizador global en problemas de variable continua, que entre sus ventajas están el uso de pocos parámetros y un gran número de elementos procesados, moverse a través del espacio de solución de forma efectiva. También se demuestra que el rendimiento es mejor en comparación de algoritmos basados en k-means dentro de la rama del clustering.

- "APPLICATION OF PARTICLE SWARM OPTIMIZATION IN DATA CLUSTERING: A SURVEY" SUNITA SARKAR, ARINDAM ROY, BIPUL SHYAM PURKAYASTHA. INTERNATIONAL JOURNAL OF COMPUTER APPLICATIONS VOLUME 65 – NO.25. MARZO 2013.

Se presenta una comparación de 25 variantes del algoritmo PSO en tareas de Data Clustering. Se muestra que la mayoría de estas variantes resultan de combinación del algoritmo PSO con otros algoritmos de clustering, lo que conlleva a mejores resultados en varios problemas de optimización en términos de eficiencia y precisión a comparación de algoritmos evolucionarios.

Además, la investigación determina 03 trabajos futuros a realizar:

- Mejoramiento de las primeras propuestas.
- Analizar y evaluar las variantes de PSO para establecer las fortalezas en sistemas existentes.
- Desarrollar un enfoque de agrupación de documentos de texto basado en la semántica usando Universal Networking Language (UNL) e híbridos del algoritmo PSO - SOM.

- "CLUSTERING MULTIDIMENSIONAL DATA WITH PSO BASED ALGORITHM" JAYSHREE GHORPADE-AHER, VISHAKHA A. METRE. THIRD POST GRADUATE SYMPOSIUM ON COMPUTER ENGINEERING CPGCON - 2014.

La investigación desarrolla la propuesta de una variante del algoritmo PSO para la resolución de problemas de agrupamiento de datos (Data Clustering). Partiendo por el estudio de técnicas de Inteligencia de Enjambres, se desarrolla un algoritmo PSO variante que los autores denominan "Subtractive Clustering based Boundary Restricted Adaptive Particle Swarm Optimization (SC-BR-APSO)", el cual permite el Data Clustering multidimensional con una tasa mínimas de error y tasa máxima de convergencia. En la investigación se compara el algoritmo desarrollado con diversas desarrolladas para el agrupamiento de datos, tales como ACO, GA, PSO y SA; con una descripción breve de las fortalezas y debilidades.

La investigación resuelve el análisis de diferentes aspectos para la aplicación de las diversas variantes PSO en retos de Data Clustering. El algoritmo SC-BR-APSO fue diseñado para solucionar problemas asociados con el agrupamiento de datos en base a

técnicas existentes que incluyen sustracción híbrida y algoritmos de agrupamiento PSO; así como otras metodologías. El desarrollo del algoritmo comprendió 04 aspectos importantes dentro del agrupamiento de datos: tasa de error, tasa de convergencia, número de iteraciones y la suma de distancia de agrupamientos internos (SICD). Para la prueba del algoritmo se hace uso de diferentes datasets.

La investigación concluye que la complejidad en tiempo del algoritmo propuesto es directamente proporcional al tamaño de los datasets usados, haciendo énfasis en la necesidad de minimizar la complejidad del algoritmo para alcanzar un mejor desempeño.

- "METAHEURÍSTICAS APLICADAS A CLUSTERING" VILLAGRA, A. UNIVERSIDAD NACIONAL DE SAN LUIS. ARGENTINA.

La autora de la investigación considera a los trabajos de clustering como problemas de optimización, que requiere la localización de los centroides óptimos de los clusters. En el desarrollo de esta, se combina el uso de PSO con el algoritmo K-means. Esta hibridación se realiza con el objetivo de mejorar el desempeño en tareas de clustering.

Se presentan 02 variantes, el primer resultado de una combinación del algoritmo PSO con la búsqueda local K-means. Esta demuestra mejores resultados que los obtenidos del algoritmo PSO y K-means aplicados de manera individual. La segunda variante; denominada PKD-G; es desarrollada con la intensión de determinar el número óptimo de clusters de manera automática.

- "DISEÑO E IMPLEMENTACIÓN DE ALGORITMOS APROXIMADOS DE CLUSTERING BALANCEADO EN PSO" HAU LAI, C. UNIVERSIDAD DE CHILE - 2012.

En esta investigación se presenta la implementación de 04 variantes del algoritmo PSO: PSO-Absorción-de-Puntos-Cercanos, PSO-Convex-Hull, PSO-Hungaro y PSO-Gale-Shaply, los cuales tienen como objetivo la creación de clusters balanceados. La autora de la investigación hace énfasis en el uso de meta heurísticas como métodos de exploración de espacio de búsqueda, considerando al clustering como un problema de optimización. Concluye con la recomendación de realizar un estudio profundo de los algoritmos usando distintas configuraciones.

- "PARTICLE SWARM OPTIMIZATION RECOMMENDER SYSTEM " Ujjin S; J. Bentley, P. UNIVERSIDAD COLLEGE LONDON - 2012.

La investigación muestra cómo se puede emplear la optimización del enjambre de partículas para afinar un algoritmo de coincidencia de perfiles dentro de un sistema de recomendación, adaptándolo a las referencias de los usuarios individuales. Se compara el algoritmo PSO con un algoritmo GA, en donde el algoritmo de PSO logró la solución significativamente más rápido, por lo que es una forma más eficiente de mejorar el rendimiento donde la velocidad de cálculo juega un parte importante en los RS.

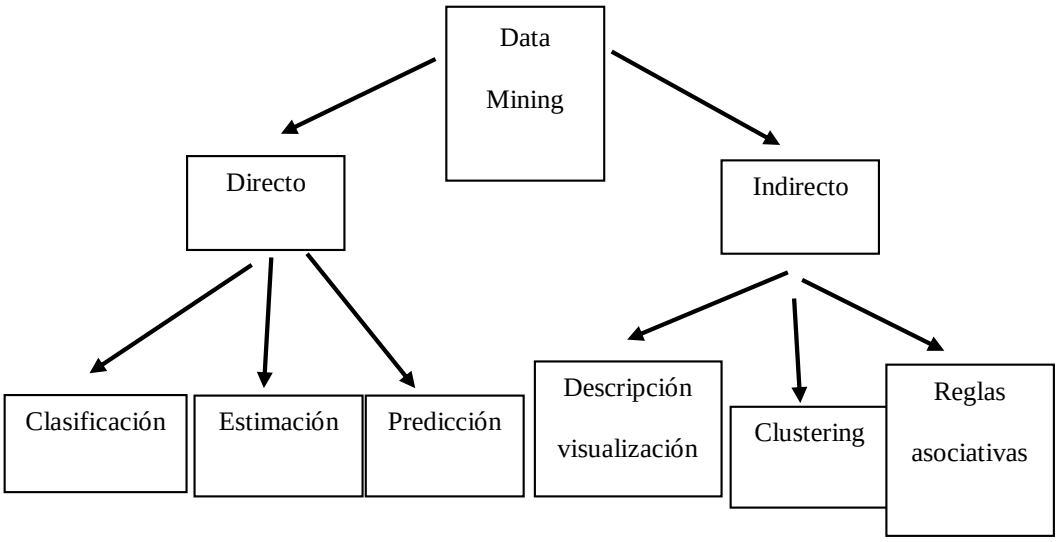
2.2 SUSTENTO TEÓRICO

2.2.1 DATA MINING

Data Mining hace referencia al estudio de recolección, procesado, análisis y extracción de significados relevantes de datos. Existe una amplia variación del concepto en términos de problemas, aplicaciones, formulaciones y representación de datos en el mundo real.

Figura 01: Tareas de Data Mining.

Fuente: Elaboración propia.



La gran cantidad de datos es resultado directo de los avances en la tecnología y la digitalización de cada aspecto de la vida moderna. Es así, que surge la necesidad de examinar la posibilidad de extraer ideas relevantes y posiblemente procesables de los datos disponibles para objetivos específicos de la aplicación. Aquí es donde entra el Data Mining; los datos en bruto suelen ser arbitrarios, no estructurados o incluso en un formato que no es inmediatamente adecuado para el procesamiento automatizado.

Para solucionar esto, análisis de Data Mining usan pipelines de procesamiento, los datos en bruto se recopilan, limpian y transforman en un formato estandarizado. Este proceso puede ser comparado al proceso de minería desde un mineral hasta el producto final refinado.

El proceso que conlleva el Data Mining, en el que se encuentran etapas como limpieza de datos, selección de características, diseño de algoritmos, etc. En general, el flujo de trabajo de una aplicación común de Data Mining conlleva:

a) Recolección de datos: La recolección de datos puede requerir el uso de hardware especializado; como una red sensible; trabajo manual; recopilación de encuestas de usuarios; o herramientas de software como un motor de rastreo de documentos web. Mientras esta etapa es altamente específica de la aplicación y, a menudo fuera del ámbito del análisis de Data Mining, es crítica porque las buenas elecciones en esta etapa pueden tener un impacto significativo en el resto del proceso.

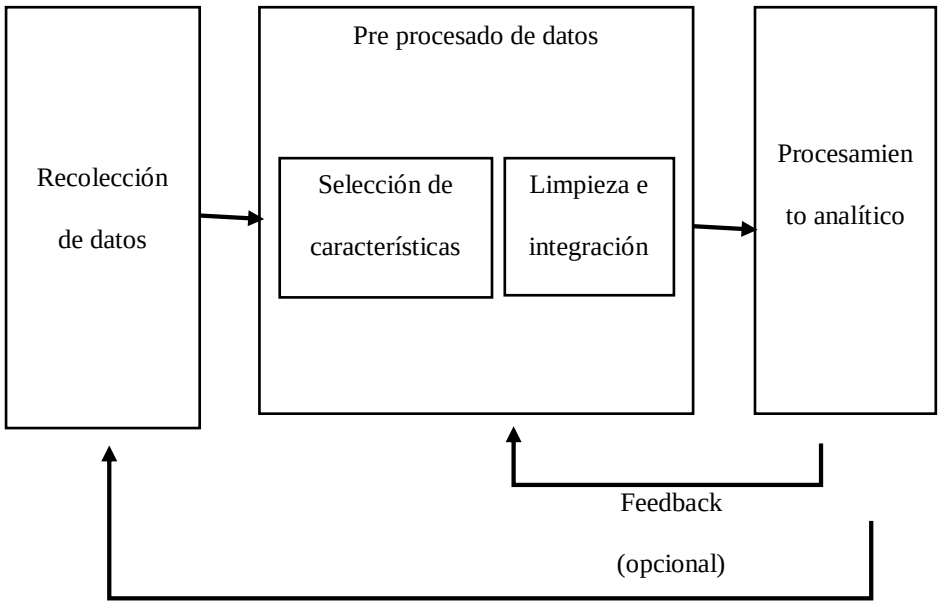
b) Extracción de características: Cuando se recopilan los datos, a menudo no están en una forma adecuada para el procesamiento, ya que esta puede estar; por ejemplo; codificada con llaves complejas, o coexistiendo diferentes tipos de datos. Para que los datos sean adecuados para el procesamiento, es esencial transformarlos en un formato compatible con los algoritmos, como transformación multidimensional, series de tiempo o formato semiestructurado.

La fase de extracción de características a menudo se realiza en paralelo con la limpieza de datos, donde partes faltantes y erróneas de los datos son estimadas o corregidas.

c) Algoritmos y procesamiento analítico: La parte final del proceso de Data Mining es diseñar métodos analíticos efectivos a partir de los datos procesados. En la mayoría de casos, no es posible el uso directo de un problema estándar. En su lugar las aplicaciones son divididas en componentes que realicen tareas específicas de acuerdo a como se quiera trata los datos.

Figura 02: Etapas de Data Mining.

Fuente: Elaboración propia.



2.2.2 REPRESENTACIÓN DE DATOS MULTIDIMENSIONALES

La representación de datos de puntos multidimensionales es un tema central varios campos, incluyendo el diseño de bases de datos, gráficos por computadora, visión artificial, geometría computacional, procesamiento de imágenes, sistemas de información geográfica (GIS), reconocimiento de patrones, integración a gran escala (VLSI) diseño entre otros. Estos puntos pueden representar ubicaciones y objetos en el espacio, así como registros más generales.

Existen diversas representaciones para un conjunto de datos de puntos multidimensionales. La viabilidad de estas representaciones depende, en parte, de la cantidad de atributos y sus dominios. Por ejemplo, existe una representación de mapa de bits pura del espacio de atributos, con un bit reservado para cada registro posible en el espacio de puntos multidimensional para indicar si está presente en el conjunto. Esta representación es útil cuando la cantidad de atributos d es relativamente pequeña ($d \leq 2$). Desafortunadamente, la mayoría de las aplicaciones posee un número de atributos que excede este límite, lo que hace necesario la búsqueda de otras soluciones.

La forma más sencilla de almacenar datos de puntos está en una lista secuencial. En este caso, no se asume ningún orden para ninguno de los atributos.

2.2.3 TIPOS DE DATOS BÁSICOS

Hay dos tipos amplios de datos, de complejidad variable, que se utilizan en el proceso de Data Mining.

a) Datos orientados a la no dependencia: Se refiere a tipos de datos simples como datos multidimensionales o datos de texto. Estos tipos de datos son los más simples y más comunes. En estos casos, los registros de datos no tienen dependencias especificadas entre los elementos de datos o los atributos. Un ejemplo es un conjunto de registros demográficos sobre individuos que contienen su edad, sexo y código postal.

Dentro de los subtipos existentes de datos tenemos:

- Datos multidimensionales cuantitativos: Referidos también como datos numéricos, representan valores que tienen significado numérico, en el sentido de que poseen un orden natural. En Data Mining, este subtipo es considerado común, y la mayoría de algoritmos diseñados trabajan con este subtipo de datos. En rigor, cualquier tipo de datos que se pueden convertir a este subtipo es realizado antes de ser procesado.

- Datos categóricos y mixtos: Diversos datasets en el mundo real contienen algún tipo de atributo categórico. En el caso de datos mixtos, suelen combinar subtipos con atributos categóricos y numéricos.

- Datos binarios: Considerado como un caso especial de datos multidimensionales numéricos o categóricos. En este subtipo se puede tomar uno o más valores discretos.

- Datos de texto: Este subtipo considera a strings o datos multidimensionales, dependiendo de la representación que tengan. Normalmente son representados como vectores, donde la frecuencia de uso de palabras en un documento es utilizada para el análisis.

Tabla 01: Ejemplo de dataset multidimensional (Iris)

Fuente: Elaboración propia

sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	Class
5.3	3.7	1.5	0.2	Iris-setosa
5.5	3.3	0.4	0.2	Iris-setosa
7.0	3.2	4.3	1.4	Iris-versicolor
5.5	2.4	3.7	1.0	Iris-versicolor

b) Datos orientados a la dependencia: En estos casos, pueden existir relaciones implícitas o explícitas entre los elementos de datos. Por ejemplo, un conjunto de datos de redes sociales contiene un conjunto de vértices (elementos de datos) que están conectados entre sí por un conjunto de bordes (relaciones). Por otro lado, las series temporales contienen dependencias implícitas. Por ejemplo, dos sucesivos los valores recolectados de un sensor probablemente estén relacionados entre sí. Por lo tanto, el atributo de tiempo especifica implícitamente una dependencia entre lecturas sucesivas.

Entre los subtipos se tiene:

- Datos de series temporales: Contienen valores que normalmente se generan mediante mediciones continuas a lo largo del tiempo. Por ejemplo, un sensor ambiental medirá la temperatura continuamente. La naturaleza de la dependencia temporal puede variar significativamente en la aplicación.

- Secuencias discretas: Las secuencias discretas se pueden considerar el análogo categórico de los datos de series de tiempo. Como en el caso de los datos de series de tiempo, el atributo contextual es una marca de tiempo o un índice de posición en el orden.

- Datos espaciales: En los datos espaciales, muchos atributos no espaciales; temperatura, presión, intensidad del color del píxel en imágenes; se miden en ubicaciones espaciales. Por ejemplo, los meteorólogos a menudo recogen las temperaturas de la superficie del mar para pronosticar la ocurrencia de huracanes.

- Datos espacio-temporales: Una forma particular de datos espaciales son los datos espacio-temporales, que contienen atributos espaciales y temporales. La naturaleza precisa de los datos también depende de cuáles de los atributos son contextuales y cuáles son conductuales.

- Datos gráficos y de redes: En este, los valores de datos pueden corresponder a nodos en la red, mientras que las relaciones entre los valores de datos pueden corresponder a los bordes en la red. En algunos casos, los atributos pueden estar asociados con nodos en la red. Aunque también es posible asociar atributos con bordes en la red, es mucho menos común hacerlo.

2.2.4 DATA CLUSTERING

Data Clustering (agrupamiento de datos traducido), también llamada Cluster analysis, análisis de segmentación o clasificación no supervisada; es un método para crear grupos de objetos, o clústeres, de tal manera que los objetos en un clúster son muy similares y los objetos en diferentes clústeres sean lo suficientemente distintos.

En este campo, el agrupamiento resultante tiende a producir modelos concisos de datos, que pueden ser interpretados como un resumen o un generativo modelo. A menudo Data Clustering es confundida erróneamente con clasificación, donde los datos son asignados a clases predefinidas. Aquí, también es necesario generar o definir esas clases.

Dentro de los dominios aplicativos del Data Clustering tenemos como ejemplos:

- Filtros colaborativos: En métodos de filtrado colaborativo, proporciona un resumen de las afinidades entre los usuarios. Esto se puede usar para proporcionar recomendaciones en una variedad de aplicaciones.

- Resumen de datos: Muchos métodos de clustering están relacionados a métodos de reducción de dimensionales, que vienen a ser los resúmenes de datos. Este ayuda en la creación de representaciones compactas de datos.

- Detección Dinámica de Tendencia: Se pueden usar muchas formas de algoritmos dinámicos y de transmisión para realizar detección de tendencia en una amplia variedad de aplicaciones de redes sociales. En dichas aplicaciones, los datos se agrupan dinámicamente de forma continua y se pueden usar para determinar patrones importantes de cambios.

- Análisis de redes sociales: La estructura de una red social se usa para determinar las comunidades importantes en la red. La detección comunitaria tiene aplicaciones importantes en el análisis de redes sociales, ya que proporciona una importante comprensión de la estructura de la comunidad en la red.

- Datos de expresión genética: Clustering es uno de los análisis más frecuentes en los datos de expresión genética (Yeung et al., 2003; Eisen et al., 1998). Los datos de expresión genética son un conjunto de mediciones recopiladas a través del microarreglo de ADNc o el experimento del chip oligo-nucleótido (Jiang et al., 2004).

- Segmentación de imágenes: La segmentación de imágenes es la descomposición de imágenes en color o escala en grises, a mosaicos homogéneos (Comaniciu and Meer, 2002). Aquí, el clustering es usado para detectar bordes de formas u objetos en las imágenes.

2.2.5 DISTANCIAS Y SIMILITUDES

Según Everitt (1993), el coeficiente de similitud indica la fuerza del vínculo entre dos puntos de datos. Así, si dos puntos de datos se asemejan entre sí, mayor será su coeficiente de similitud. Si se tiene a $X = (x1, x_2, x_3 \dots x_N)$ y a $Y = (y_1, y_2, y_3 \dots y_N)$ como dos puntos con N-dimensiones, el coeficiente de similitud será el resultado de cierta función entre los atributos de ambos.

$$f\,(X, Y) = f\,(x_1, x_2, x_3 \dots x_N, y_1, y_2, y_3 \dots y_N)$$

La similitud suele ser simétrica. Andeberg (1973) afirma: Una métrica es una función de distancia *f* definida en un conjunto *E* que cumple las siguientes cuatro propiedades:

1. No negatividad : $f\,(x, y) \geq 0$
2. Reflexión : $f\,(x, y) = 0 \Leftrightarrow x = y$
3. Conmutatividad : $f\,(x, y) = f\,(y, x)$
4. Desigualdad triangular : $f\,(x, y) \leq f\,(x, z) + f\,(y, z)$

Donde *x, y, z* son puntos aleatorios. Una función de desemejanza es una medida definida en un conjunto. Pero por una función de similitud, hace referencia a una función $f\,(x, y)$ medida en dos puntos de datos en un conjunto de datos, satisfaciendo las siguientes propiedades:

1. $0 \leq f(x, y) \leq 1$
2. $f(x, x) = 1$
3. $f(x, y) = f(y, x)$

Donde *x*, *y* son puntos aleatorios. Existen otras estructuras que muestran similitud y desemejanza. Si tenemos a *D* como un dataset, Hartigan (1967) lista al menos 12 estructuras de similitud para una medición de similitud *S* defina en *D*:

1. *S* definida en $D \times D$ es una Distancia Euclidiana.
2. *S* definida en $D \times D$ es una métrica.
3. *S* definida en $D \times D$ es simétrica y de valor real.
4. *S* definida en $D \times D$ es un valor real.
5. *S* es un orden completo de similitud $\leq_S$ en $D \times D$, si cada par de objetos pueden ser ordenados.
6. *S* es un orden parcial de similitud $\leq_S$ en $D \times D$, si cada par de objetos comparables pueden ser ordenados, pero no todos los pares de objetos deben ser ordenados.
7. *S* es un árbol en *D*.
8. *S* es un orden completo de similitud relativa $\leq_i$ en *D*; para cada *i* en *D* ($j \leq_i k$ significa que *j* no es más similar a *i* que *k* es a *i*).
9. *S* es una orden de similitud relativa parcial $\leq_i$ en *D*.
10. *S* es una dicotomía de similitud en $D \times D$, en la que $D \times D$ se divide en un conjunto de pares similares y un conjunto de pares diferentes.

11. *S* es una tricotomía de similitud en $D \times D$, en la que $D \times D$ consiste en pares similares, pares diferentes, y los pares restantes.
12. *S* es una partición de *D* en conjuntos de objetos similares.

2.2.6 MEDICIÓN DE DATOS NUMÉRICOS

La elección de las distancias es importante para las aplicaciones, y la mejor opción a menudo se logra mediante una combinación de experiencia, habilidad, conocimiento y suerte. A continuación, se describen algunas distancias comúnmente utilizadas:

Distancia Euclidiana.

Es la distancia más común que se usan en datos numéricos. Para dos puntos x, y en un espacio d-dimensional, la distancia euclidiana es definida por:

Ecuación 01: Distancia Euclidiana.

Fuente: Extraído de Data Clustering: Theory, Algorithms and Applications.

$$d_{euc}(\mathbf{x}, \mathbf{y}) = \left[\sum_{j=1}^{d} (x_j - y_j)^2 \right]^{\frac{1}{2}} = \left[(\mathbf{x} - \mathbf{y})(\mathbf{x} - \mathbf{y})^T \right]^{\frac{1}{2}}$$

Donde x_j , y_j son valores de los *j*-avos atributos de los puntos x, y respectivamente, el Cuadrado Euclidiano es definido por:

Ecuación 02: Cuadrado Euclidiano.

Fuente: Extraído de Data Clustering: Theory, Algorithms and Applications.

$$d_{seuc}(\mathbf{x}, \mathbf{y}) = d_{euc}(\mathbf{x}, \mathbf{y})^2 = \sum_{j=1}^{d} (x_j - y_j)^2 = (\mathbf{x} - \mathbf{y})(\mathbf{x} - \mathbf{y})^T$$

Aquí se debe hacer énfasis en que el Cuadrado Euclidiano no es una métrica en sí, aunque es mencionada en una gran variedad de cálculos.

Distancia Manhattan.

Definida como la suma de la distancia de todos los atributos entre dos puntos x, y d-dimensionales:

Ecuación 03: Distancia Manhattan.

Fuente: Extraído de Data Clustering: Theory, Algorithms and Applications.

$$d_{man}(\mathbf{x}, \mathbf{y}) = \sum_{k=1}^{d} |x_j - y_j|$$

De esta fórmula se obtienen dos variaciones: Si en x o y hubiera datos faltantes, se definiría como:

Ecuación 04: Variación Manhattan para atributos faltantes.

Fuente: Extraído de Data Clustering: Theory, Algorithms and Applications.

$$d_{manw}(\mathbf{x}, \mathbf{y}) = \sum_{k=1}^{d} \frac{w_j |x_j - y_j|}{\sum_{k=1}^{d} w_j}$$

Por otro lado, si solo se quieren usar parte de las d-dimensiones:

Ecuación 05: Variación Manhattan con P-subdimensiones.

Fuente: Extraído de Data Clustering: Theory, Algorithms and Applications.

$$d_P(\mathbf{x}, \mathbf{y}) = \sum_{j \in P} \frac{|x_j - y_j|}{|P|}$$

Distancia Máxima.

Definida como el máximo valor de la distancia de los atributos en dos puntos x, y con d-dimensiones:

Ecuación 06: Distancia Máxima.

Fuente: Extraído de Data Clustering: Theory, Algorithms and Applications.

$$d_{max}(\mathbf{x}, \mathbf{y}) = \max_{1 \le k \le d} |x_j - y_j|.$$

Distancia Minkowski.

Las tres distancias mencionadas anteriormente, son casos particulares de la Distancia Minkowski:

Ecuación 07: Distancia Minkowski.

Fuente: Extraído de Data Clustering: Theory, Algorithms and Applications.

$$d_{min}(\mathbf{x}, \mathbf{y}) = \left(\sum_{j=1}^{d} |x_j - y_j|^r \right)^{\frac{1}{r}}, \quad r \ge 1.$$

Distancia Mahalanobis.

Esta distancia es usada para aliviar la distorsión de distancia causada por combinaciones lineales de atributos:

Ecuación 08: Distancia Mahalanobis.

Fuente: Extraído de Data Clustering: Theory, Algorithms and Applications.

$$d_{mah}(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})\Sigma^{-1}(\mathbf{x} - \mathbf{y})^T}$$

$\sum^{-1}$ es definida como la matriz covarianza de un dataset.

Distancia Promedio.

Según Legendre (1983) mencionan que la distancia euclidiana tiene dos efectos adversos: dos puntos de datos sin valores de atributo en común pueden tener una distancia menor que otro par de puntos de datos que contienen los mismos valores de atributo. En este caso, la distancia promedio debe ser usada. La distancia promedio es una modificación de la distancia euclidiana.

Ecuación 09: Distancia promedio.

Fuente: Extraído de Data Clustering: Theory, Algorithms and Applications.

$$d_{ave}(\mathbf{x}, \mathbf{y}) = \left(\frac{1}{d} \sum_{j=1}^{d} (x_j - y_j)^2 \right)^{\frac{1}{2}}$$

2.2.7 MEDICIÓN DE DATOS CATEGÓRICOS Y MIXTOS

Los datos categóricos son medidos en escalas normales. A diferencia de los datos numéricos, el cálculo de las medidas de asociación entre registros descriptos por variables nominales ha recibido poca atención.

Distancia de emparejamiento simple.

Propuesta por Kaufman and Rousseeuw, esta distancia es una simple y conocida fórmula para medir datos categóricos. Sean *x*, *y* dos valores categóricos, la distancia de emparejamiento simple está definida por:

Figura 03: Valores usados en datos categóricos.

Fuente: Extraído de Data Clustering: Theory, Algorithms and Applications.

$$\delta(x, y) = \begin{cases} 0 & \text{if } x = y, \\ 1 & \text{if } x \neq y. \end{cases}$$

Sean *x*, *y* dos objetos categóricos descritos por d atributos categóricos. Entonces la diferencia entre *x*, *y* medida por la distancia de correspondencia simple se define por:

Ecuación 10: Distancia Emparejamiento Simple.

Fuente: Extraído de Data Clustering: Theory, Algorithms and Applications.

$$d_{sim}(\mathbf{x}, \mathbf{y}) = \sum_{j=1}^{d} \delta(x_j, y_j)$$

Coeficiente de Similitud General.

Propuesto por Gower (1971), el Coeficiente de Similitud General ha sido ampliamente utilizado para medir la similitud de dos puntos de datos mixtos. Este coeficiente también se puede aplicar a puntos de datos con valores faltantes.

Sean *x*, *y* dos puntos d-dimensionales, el Coeficiente de Similitud General está definido por:

Ecuación 11: Coeficiente de Similitud General.

Fuente: Extraído de Data Clustering: Theory, Algorithms and Applications.

$$s_{gower}(\mathbf{x}, \mathbf{y}) = \frac{1}{\sum_{k=1}^{d} w(x_k, y_k)} \sum_{k=1}^{d} w(x_k, y_k) s(x_k, y_k)$$

S (x_k, y_k) y W (x_k, y_k) están definidos de acuerdo al tipo de atributo que sean:

- Para datos cuantitativos, W (x_k, y_k) = 0 si en x, y faltaran atributos, de otra manera W (x_k, y_k) = 1. En la fórmula a continuación R_k representa el rango del k-ésimo atributo.

Ecuación 12: Valores en datos numéricos.

Fuente: Extraído de Data Clustering: Theory, Algorithms and Applications.

$$s(x_k, y_k) = 1 - \frac{|x_k - y_k|}{R_k}$$

- En atributos binarios x_k, y_k; S (x_k, y_k) = 1 si ambos valores están presentes en el k-ésimo atributo, de otra forma S (x_k, y_k) = 0. W (x_k, y_k) = 0 si x_k, y_k tienen el k-ésimo atributo ausente, de otra manera W (x_k, y_k) = 1.

- Para datos categóricos y nominales, S (x_k, y_k) = 1 si x_k = y_k, de otra manera S (x_k, y_k) = 0. W (x_k, y_k) = 0 si x_k, y_k tienen el k-ésimo atributo ausente, de otra manera W (x_k, y_k) = 1.

2.2.8 DATA STANDARDIZATION

Data Standardization vuelve los datos no dimensionales, y es útil al definir índices estándar. En este proceso, es posible que la ubicación y escala de los datos originales se pierda, aunque su uso es necesario en situaciones donde las mediciones de disimilitud sean sensibles a diferencias de magnitudes o escalas de las variables de entrada.

Antes de cualquier tipo de estandarización, es necesario escoger el mejor método dependiendo del tipo de entrada que se tenga y el campo de estudio escogido.

Ecuación 13: Ecuación base de estandarización.

Fuente: Extraído de Data Clustering: Theory, Algorithms and Applications.

$$x_{ij} = \frac{x_{ij}^{*} - L_j}{M_j}$$

x_{ij} denota el valor estandarizado, L_j denota la medida de ubicación y M_j es la escala de medición. De esta fórmula derivan otros métodos de estandarización, entre los más conocidos están mediana, mean desviación estándar, estimación Huber, estimación de onda Andrew, entre otros.

Ecuación 14: Mean, rango y desviación estándar.

Fuente: Extraído de Data Clustering: Theory, Algorithms and Applications.

$$\bar{x}_j^* = \frac{1}{n}\sum_{i=1}^{n} x_{ij}^*,$$

$$R_j^* = \max_{1\le i\le n} x_{ij}^* - \min_{1\le i\le n} x_{ij}^*,$$

$$\sigma_j^* = \left[\frac{1}{n-1}\sum_{i=1}^{n}(x_{ij}^* - \bar{x}_j^*)^2\right]^{\frac{1}{2}}$$

En la siguiente tabla se muestran algunos métodos empleados en el proceso de estandarización:

Tabla 02: Métodos de estandarización.

Fuente: Extraído de Data Clustering: Theory, Algorithms and Applications.

Nombre	L_j	M_j
z-score	$\bar{x}_j^*$	σ_j^*
USTD	0	σ_j^*
Máximo	0	$\max_{1\le i\le n} x_{ij}^*$
Mean	$\bar{x}_j^*$	1
Mediana	$\bar{x}^*_{\frac{n+1}{2}j}$ si n es impar $\frac{1}{2}(x^*_{\frac{n}{2}j} + x^*_{\frac{n+2}{2}j})$ si n es par	1
Suma	0	$\sum_{i=1}^{n} x_{ij}^*$
Rango	$\min_{1\le i\le n} x_{ij}^*$	R_j^*

2.2.9 TIPOS DE DATA CLUSTERING

A continuación, se describirán algunos tipos de Data Clustering existentes:

Algoritmos de Clustering Jerárquico (Hierarchical Clustering).

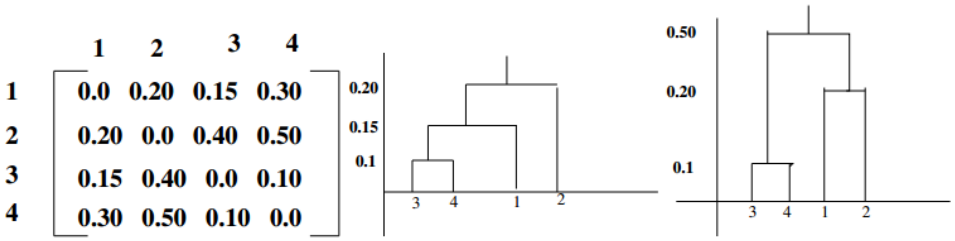
Este pertenece a una rama del Data Clustering llamada Hard Clustering. Los algoritmos jerárquicos se subdividen en algoritmos jerárquicos aglomerativos y algoritmos jerárquicos divisivos. Los algoritmos aglomerativos empiezan con cada objeto en un solo clúster. Luego se repite fusionando el par más cercano de clusters de acuerdo a ciertos criterios de similitud, mientras que la información de los datos está en un clúster. Estos tienen inconvenientes, como que los puntos de datos que se han agrupado incorrectamente en una etapa temprana no se pueden asignar posteriormente y diferentes medidas de similitud para medir la similitud entre los grupos pueden conducir a resultados diferentes.

Los algoritmos jerárquicos de tipo aglomerativo son:

- Enlace simple y completo: Este método intuitivamente da más importancia a las regiones donde los clusters están más cercanos, descuidando la estructura general del clúster. Por lo tanto, este método cae dentro de la categoría de un método de clustering basado en la similitud local.

Figura 04: Ejemplo de clustering aglomerativo.

Fuente: Elaboración propia.



- Clustering de centroide aglomerativo y grupo promedio: Considera la similitud entre todos los pares de puntos presentes en ambos grupos y disminuye los inconvenientes asociados con los métodos de enlace simple y completo.

- Criterio de Ward: Propuesto para calcular la distancia entre dos grupos durante la agrupación aglomerativa. Este proceso de utilizar el criterio de Ward para la unión de conglomerados en la agrupación aglomerativa también se denomina aglomeración de Ward.

Los algoritmos jerárquicos divisionales comienzan con todos los objetos en un grupo y se repite dividiendo los grupos grandes en piezas más pequeñas. La agrupación jerárquica divisiva tiene los mismos inconvenientes que la agrupación jerárquica aglomerativa.

Algoritmos de Clustering Difuso (Fuzzy Clustering).

En Hard Clustering se requiere que cada punto de datos del conjunto de datos pertenezca a un solo clúster. Fuzzy Clustering amplía esta noción para asociar cada punto de datos en el conjunto de datos con cada grupo utilizando una función de pertenencia.

Desde que Zadeh (1965) introdujo el concepto de fuzzy sets, Fuzzy Clustering ha sido estudiado, debatido y aplicado en diferentes áreas.

Algoritmos de Clustering basado en gráficos (Graph-based Clustering).

Los algoritmos de Clustering basados en gráficos primero construirá un gráfico o un hipergráfico y luego aplicará un algoritmo de agrupamiento para dividir el gráfico o el hipergráfico. Un algoritmo de agrupamiento basado en enlaces también se puede considerar como uno basado en gráficos, porque podemos pensar en los enlaces entre los puntos de datos como enlaces entre los nodos del gráfico

- Chameleon: Propuesto por Karypis (1999), el algoritmo camaleón (traducido al español) construye una representación de gráfico disperso de los datos en base al enfoque de gráfico vecino k más cercano utilizado comúnmente. Los datos son provistos a este algoritmo mediante una matriz de similitud, que correspondería a un dataset o una base de datos.

Tabla 03: Características del algoritmo Chameleon.

Fuente: Extraído de Data Clustering: Theory, Algorithms and Applications.

Tipo de datos	Datos Categóricos
Complejidad Temporal	$O(nm + n \log n + m^2 \log m)$
Tipo de Algoritmo	Jerárquico

- CACTUS: Categorical Clustering Using Summaries; desarrollado por Ganti (1999); basa su funcionamiento en la construcción de resúmenes de inter atributos e intra atributos, que luego son usados para la construcción de un gráfico de similitud.

- ROCK: RObust Clustering using linKs es un algoritmo aglomerativo jerárquico que emplea enlaces para unir clusters, usando la medición de similitud basada en enlace para medir la similitud entre dos puntos de datos y entre dos grupos.

Tabla 04: Características del algoritmo ROCK.

Fuente: Extraído de Data Clustering: Theory, Algorithms and Applications.

Tipo de datos	Datos Categóricos
Complejidad Temporal	$O(n2 + nm_m m_a + n^2 \log n)$
Complejidad Espacial	$O(min\{n^2, nm_m m_a\})$
Tipo de Algoritmo	Jerárquico aglomerativo

Algoritmos de Clustering basado en densidad (Density-based Clustering).

Estos algoritmos están diseñados para encontrar clusters de forma arbitraria, donde los clusters se definen como regiones densas separadas por regiones de baja densidad. Estas solo necesitan un escaneo del conjunto de datos original y puede manejar el ruido. La cantidad de clústeres no es necesaria, ya que los algoritmos de clustering basados en densidad pueden detectar automáticamente los clusters, junto con el número natural de clusters.

- DBSCAN: Ester et al. (1996) propuso un algoritmo de clustering basado en la densidad llamado DBSCAN (DensityBased Spatial Clustering of Applications with Noise) para descubrir clusters de forma arbitraria. Solo se requiere un parámetro de entrada, y el algoritmo también es compatible con el usuario para determinar un valor apropiado para este parámetro de entrada

- BRIDGE: Dash et al. (2001) propuso un algoritmo híbrido, llamado BRIDGE, que integra el popular algoritmo k-means y el algoritmo basado en la densidad DBSCAN. BRIDGE permite a DBSCAN manejar bases de datos muy grandes y, al mismo tiempo, mejora la calidad de los clústeres k-means eliminando el ruido.

- DBCLASD: Xu et al. (1998) propone el algoritmo de clustering incremental llamado DBCLASD (DistributionBased Clustering of LArge Spatial Databases) basado en la suposición de que los puntos dentro de un grupo están distribuidos uniformemente. Como en DBSCAN, DBCLASD es capaz de encontrar clusteres con formas arbitrarias, aunque DBCLASD no requiere parámetros de entrada.

  DBCLASD determina dinámicamente el número y la forma apropiados de los clústeres para una base de datos, y el algoritmo es eficiente para las bases de datos más grandes.

2.2.10 EVALUACIÓN DE CLUSTERING

Realizar el agrupamiento de un dataset es un proceso no supervisado; es decir; no existen clases definidas y no hay ejemplos que puedan mostrar que los clusters encontrados por los algoritmos de clustering son válidos. Casi todos los algoritmos de

clustering encontrarán clusters en el conjunto de datos, aun cuando este conjunto no contenga clusters naturales en su estructura.

Para comparar la validez de los clusters generados, es necesario tener en cuenta ciertos aspectos en consideración:

- Determinar la tendencia de agrupamiento de los datos; en otras palabras; distinguir si en los datos existen estructuras no aleatorias.

- Determinar el número de clusters.

- Evaluar cuán bien los resultados del análisis de un cluster se adecúan a los datos sin intervención de información externa.

- Comparar los resultados del análisis con resultados externos conocidos.

- Comparar dos grupos de clústeres, para determinar cuál es el mejor.

Las medidas de evaluación de clústeres suelen agruparse en dos grupos:

- Supervisada, que mide el grado de coincidencia de la estructura de un clúster determinado por el algoritmo, con alguna estructura externa.

- No supervisada, que mide la bondad de la estructura del clúster sin tomar alguna referencia externa.

Las investigaciones experimentales suelen utilizar evaluaciones no supervisadas, dado que en algunos casos aún no existe información externa a la que puedan comparar.

Las medidas no supervisadas a menudo califican dos criterios, la medida de cohesión de los clústeres; que mide la distancia de los puntos de un clúster; y la medida de separación; que mide la distancia entre clústeres. Estos suelen ser llamados índices internos, ya que solo usan la información presente en los datos.

Distancia Intra Clúster

Tabla 05: Distancias Intra Clúster.

Fuente: Extraído de Data Clustering and Similarity. Proceedings of the Twenty-Sixth International Florida Artificial Intelligence Research Society Conference.

Radius	$max(d(x,\mu_{\mathcal{A}})$ where $\mu_{\mathcal{A}}$ is the arithmetic mean of $\mathcal{A}$
Radius (variation)	$\frac{1}{\lvert\mathcal{A}\rvert}\sum_{x\in\mathcal{A}} d(x,\mu_{\mathcal{A}})$ where $\mu_{\mathcal{A}}$ is the arithmetic mean of $\mathcal{A}$
Diameter	$max(d(x,y)), x \in \mathcal{A}, y \in \mathcal{A}, x \neq y$
Diameter (variation)	$\frac{1}{\lvert\mathcal{A}\rvert\cdot(\lvert\mathcal{A}\rvert-1)}\sum_{x\in\mathcal{A}}\sum_{y\in\mathcal{A}} d(x,y)$

Ecuación 15: Fórmula de Distancia Intra Clúster.

Fuente: Extraído de Metaheurísticas Aplicadas a Clustering.

$$Intra\_C = \frac{\sum_{i=1}^{K}(\sum_{x,t\in C_i} dist(x,t)/m_i)}{K}$$

La fórmula de Distancia Intra Cluster permite identificar la cohesión de los elementos de un clúster con respecto al centroide del mismo. *K* representa al número de clústeres, C_i representa al i-ésimo clúster, m_i representa la cantidad de elementos del clúster. Por último, *dist* hace referencia a la distancia euclidiana de dos puntos *x*, *t*.

Distancia Inter Clúster

Tabla 06: Distancias Inter Clúster.

Fuente: Extraído de Data Clustering and Similarity. Proceedings of the Twenty-Sixth International Florida Artificial Intelligence Research Society Conference

Single linkage	$min(d(x,y)), x \in \mathcal{A}, y \in \mathcal{B}$
Complete linkage	$max(d(x,y)), x \in \mathcal{A}, y \in \mathcal{B}$
UPGMA or Average distance	$\frac{1}{\lvert\mathcal{A}\rvert \cdot \lvert\mathcal{B}\rvert} \sum_{x \in \mathcal{A}} \sum_{y \in \mathcal{B}} d(x,y)$
Average linkage (variation)	$d(\mu_{\mathcal{A}}, \mu_{\mathcal{B}})$ where $\mu_{\mathcal{A}}$ and $\mu_{\mathcal{B}}$ are the arithmetic means of the clusters

Ecuación 16: Fórmula de Distancia Inter Clúster.

Fuente: Extraído de Metaheurísticas Aplicadas a Clustering.

$$Inter\_C = \frac{\sum_{i=1}^{K-1} \sum_{j=i+1}^{K} dist(c_i, c_j)}{\sum_{i=1}^{K-1} i}$$

La Distancia Inter Clúster permite evaluar la separación de los clústeres hallados. C_i y C_j representan dos clústeres, *K* es el número de clústeres y *dist* es la distancia euclidiana de C_i y C_j.

2.2.11 SUBTRACTIVE CLUSTERING

Subtractive Clustering es un algoritmo rápido de una sola pasada para estimar el número de clústeres y los centroides de clúster en un conjunto de datos. Nace del refinamiento de otro algoritmo llamado Mountain Clustering.

Para entender mejor el funcionamiento del algoritmo Subtractive Clustering, describiremos primero el algoritmo Mountain Clustering. Este método puede ser usado para determinar centros iniciales que son requeridos por algoritmos de clustering más sofisticados. También puede ser usado como un método rápido único para determinar clústeres aproximados. Utiliza tres pasos para determinar centroides en clústers.

- El primer paso implica formar una cuadrícula del espacio de datos, donde las intersecciones de las líneas de la cuadrícula constituyen los candidatos para los centroides del clúster, al que denotaremos como un conjunto *V*. La cuadrícula generalmente está espaciada de manera uniforme, pero no es un requisito.

- El segundo paso implica construir una función de montaña que represente una medida de densidad de datos.

Ecuación 17: Función Mountain.

Fuente: Extraído de A Comparative Study of Data Clustering Techniques.

$$m(v) = \sum_{i=1}^{N} \exp\left(- \frac{\| v - x_i \|^2}{2\sigma^2} \right)$$

La altura de la función en un punto $v \in V$ es igual donde x_i es el i-ésimo punto de datos y σ es una constante específica de la aplicación. Cada punto de datos x_i contribuye a la altura de la función en *v*, *y* la contribución es inversamente proporcional a la distancia entre x_i y *v*. La constante σ determina la altura y la suavidad de la función resultante.

Los resultados de la agrupación son normalmente insensibles al valor de σ, siempre que el dataset tenga el tamaño suficiente y esté bien agrupado. El centro candidato en V que tenga el mayor valor para la función se convierte en el primer centro de clústeres c_1.

- El tercer paso consiste en seleccionar los centroides del clúster mediante la eliminación secuencial de la función. La obtención del siguiente centro de clúster requiere eliminar el efecto del centro recién identificado, que generalmente está rodeado por una serie de puntos de cuadrícula que también tienen altos puntajes de densidad.

Ecuación 18: Función Mountain de Eliminación.

Fuente: Extraído de A Comparative Study of Data Clustering Techniques.

$$m_{new} = (v) = m(v) - m(c_1)\exp\left(-\frac{\| v - c_1 \|^2}{2\beta^2}\right)$$

- Después de la sustracción, el segundo centro del grupo se selecciona de nuevo como el punto en *V* que tiene el mayor valor para la nueva función. Este proceso

de revisión de la función y búsqueda de los próximos centros de clúster continúa hasta que se alcanza un número suficiente de centros de clúster.

El rendimiento de Mountain Clustering depende en gran medida de la dimensión de la cuadrícula, con cuadrículas más finas que proporcionan un mejor rendimiento. Sin embargo, a medida que aumenta la dimensión de la cuadrícula, el método se vuelve computacionalmente costoso.

Subtractive Clustering es un enfoque alternativo que se puede utilizar para eliminar este problema. En esta, los puntos de datos (no los puntos de cuadrícula) se consideran candidatos para los centros de clúster. Al usar este método, el cálculo es simplemente proporcional al número de puntos de datos e independiente del problema de dimensión como se mencionó anteriormente.

Si se considera una colección de n puntos de datos $\{x_1, ..., X_n\}$ en un espacio de M dimensiones. Se supone que los puntos de datos se han normalizado dentro de un hipercubo. Dado que cada punto de datos es un candidato para centros de clústeres, una medida de densidad en el punto de datos x_i se define como:

Ecuación 19: Función Subtractive Clustering.

Fuente: Extraído de A Comparative Study of Data Clustering Techniques.

$$D_i = \sum_{j=1}^{n} \exp\left(- \frac{\| x_i - x_j \|^2}{(r_a / 2)^2} \right)$$

Donde r_a es una constante positiva. Por lo tanto, un punto de datos tendrá un valor de alta densidad si tiene muchos puntos de datos vecinos. El radio r_a define un vecindario; los puntos de datos fuera de este radio contribuyen solo ligeramente a la medida de densidad.

Después de calcular la medida de densidad de cada punto de datos, se selecciona el punto de datos con la medida de densidad más alta como el primer centroide, representado como x_{c1} y D_{c1} su medida de densidad. La medida de densidad para cada punto de datos x_i se revisa mediante la fórmula:

Ecuación 20: Subtractive Clustering de eliminación.

Fuente: Extraído de A Comparative Study of Data Clustering Techniques.

$$D_i = D_i - D_{c1} \exp\left(- \frac{\| x_i - x_{c1} \|^2}{(r_b / 2)^2} \right)$$

Donde r_b es una constante positiva. Por lo tanto, los puntos de datos cercanos al primer centro de clúster x_{c1} tendrán una medida de densidad significativamente reducida, por lo que es poco probable que los puntos se seleccionen como el siguiente centro de clúster.

La constante r_b define un vecindario que tiene reducciones medibles en la medida de la densidad. La constante r_b es normalmente mayor que r_a para evitar centros de clúster

estrechamente espaciados; generalmente r_b esta entre los valores [1.2, 1.5] r_a. Al igual que en Mountain Clustering,

2.2.12 METAHEURÍSTICAS

Proveniente de la unión de la palabra griega “meta”; que significa “más allá”, “superior”; y “heurística”; significa “encontrar”; las metaheurísticas son métodos heurísticos que sirven para resolver un tipo de problema computacional general, usando parámetros dados por el usuario sobre unos procedimientos genéricos y abstractos de una manera; que se espera eficiente. La mayoría de las metaheurísticas tienen como objetivo problemas de optimización combinatoria (exploración del espacio de soluciones).

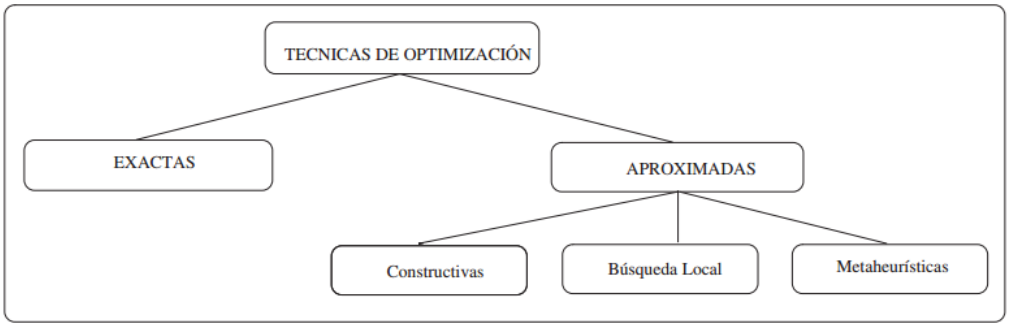
Las metaheurísticas suelen ser menos eficientes que las heurísticas específicas, en varios órdenes de magnitud, en problemas que aceptan este tipo de heurísticas puras, pero suelen ser más fáciles de implementar.

Técnicas de Optimización.

Los problemas de optimización resuelven encontrar la mejor solución o al menos una solución lo suficientemente buena para un problema en la vida real. Constantemente estamos resolviendo pequeños problemas de optimización, como encontrar el camino m: corto para ir de un lugar a otro, la organización de la agenda, etc. Debido a la gran importancia de los problemas de optimización se han desarrollado múltiples métodos para resolverlos. En la siguiente figura se muestra una clasificación simple de estos métodos.

Figura 05: Clasificación de las técnicas de optimización.

Fuente: Metaheurísticas aplicadas al Clustering.



Las técnicas exactas garantizan encontrar la solución óptima, con el inconveniente del crecimiento exponencial en el tiempo necesario para hallar la solución óptima, en algunos casos siendo hasta virtualmente imposible. Por esta razón, los algoritmos aproximados intentan resolver estos problemas, sacrificando la garantía de encontrar el óptimo a cambio de encontrar una mejor solución en un tiempo razonable. Esto no quiere decir que los algoritmos aproximados no sean buenos para encontrar una solución que heurísticas exactas, al contrario, son mejores si lo que se pretende es encontrar un grupo de soluciones y no una única solución.

Como se puede observar, las metaheurísticas corresponden a la categoría de métodos aproximados, junto con métodos de búsqueda local y métodos constructivos. Al desarrollar métodos metaheurísticos, se deben en cuenta una serie de propiedades:

- Simplicidad, las metaheurísticas deben estar basadas en principios sencillos y claros; fácil de comprender.

- Precisión, los pasos y fases de la metaheurística deben estar formulados en términos concretos.

- Coherencia, los elementos de la metaheurística debe deducirse naturalmente de sus principios.

- Efectividad, deben proporcionar soluciones de alta calidad; óptimas o muy cercanas a las óptimas.

- Eficacia, la probabilidad de alcanzar soluciones óptimas de casos reales debe ser alta.

- Eficiencia, la metaheurística debe realizar un buen aprovechamiento de recursos computacionales, tiempo de ejecución y espacio de memoria.

- General, la metaheurística debe ser utilizable con buen rendimiento en una amplia variedad de problemas.

- Adaptabilidad, además de ser general a varios problemas, debe ser capaz de adaptarse a diferentes contextos de aplicación o modificaciones importantes del modelo.

- Robustez, el comportamiento debe ser poco sensible a pequeñas alteraciones del modelo o contexto de aplicación.

- Interactividad, debe permitir que el usuario pueda aplicar sus conocimientos para mejorar el rendimiento del procedimiento.

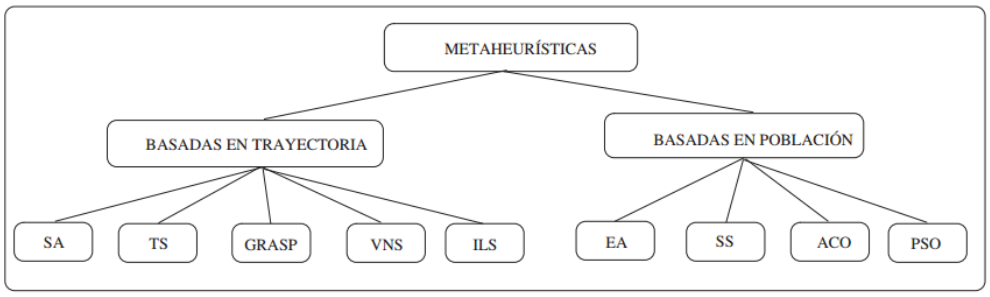
- Múltiple, debe suministrar diferentes soluciones alternativas de alta calidad entre las que el usuario pueda elegir.

- Autonomía, debe permitir un funcionamiento libre de parámetros o que se puedan establecer automáticamente.

Existen varias maneras de subdividir los métodos metaheurísticos, ya sean basados en su naturaleza, basadas en memoria, basadas en el comportamiento de la función objetivo, etc.

Figura 06: Clasificación de metaheurísticas.

Fuente: Metaheurísticas aplicadas al Clustering.



Las metaheurísticas de trayectoria parten de un punto y mediante la exploración del vecindario van variando la solución actual, formando una trayectoria. La mayoría de estos algoritmos surgen como extensiones de los métodos de búsqueda local simple a los que

se les añade alguna característica para escapar de los mínimos locales. Esto implica la necesidad de una condición de parada más compleja que la de encontrar un mínimo local.

Normalmente se termina la búsqueda cuando se alcanza un número máximo predefinido de iteraciones, se encuentra una solución con una calidad aceptable, o alguna restricción ha sido alcanzada.

Las metaheurísticas basadas en población se caracterizan por trabajar con un conjunto de soluciones en cada iteración, a diferencia de los métodos de trayectoria que únicamente utilizan un punto del espacio de búsqueda por iteración. El resultado final proporcionado por este tipo de algoritmos depende fuertemente de la forma en que manipula la población.

2.2.13 INTELIGENCIA DE ENJAMBRES

La Inteligencia de Enjambres, es una rama de la Inteligencia Artificial que estudia el comportamiento colectivo de sistemas descentralizados, auto organizados. Ha atraído gran interés, y muchos algoritmos de optimización basados en SI han ganado una gran popularidad. Hay muchas razones para tal popularidad y atención, y dos razones principales son, probablemente, que estos algoritmos basados en SI son flexibles y versátiles, y que son muy eficientes en la resolución de problemas de diseño no lineal con aplicaciones del mundo real.

La computación de inspiración biológica ha penetrado en casi todas las áreas de las ciencias, la ingeniería y las industrias, desde la minería de datos hasta la optimización, desde la inteligencia computacional hasta la planificación empresarial, y desde la bioinformática hasta las aplicaciones industriales.

Los algoritmos metaheurísticos a menudo se inspiran en la naturaleza, y ahora se encuentran entre los algoritmos más utilizados para modelos de optimización. Los algoritmos metaheurísticos son muy diversos, incluyendo algoritmos genéticos, recocido simulado, evolución diferencial, algoritmos de hormigas y abejas, optimización de enjambre de partículas, búsqueda de armonía, algoritmo de luciérnaga y búsqueda cuckoo.

A continuación, describiremos brevemente algunos de los algoritmos de SI.

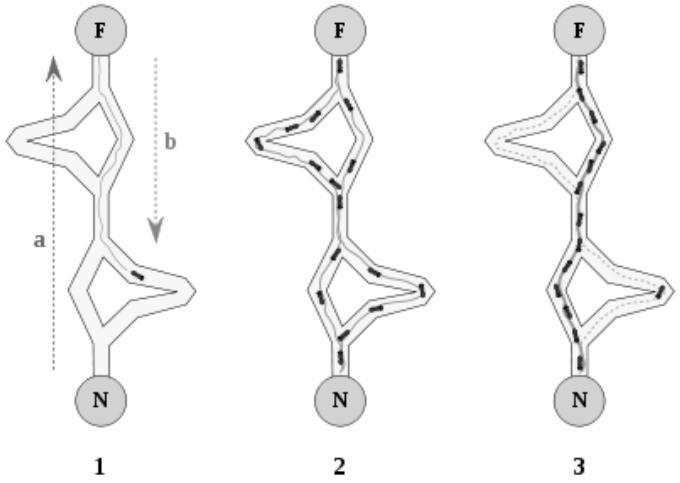
Algoritmo de Hormigas (Ants Algorithms)

Los algoritmos de hormigas, especialmente la optimización de la colonia de hormigas (Dorigo y Stu'tle, 2004), imitan el comportamiento de alimentación de las hormigas sociales. Las hormigas principalmente usan feromonas como mensajero químico, y la concentración de feromonas se puede considerar como el indicador de soluciones de calidad para un problema de interés. Como la solución a menudo está relacionada con las concentraciones de feromonas, los algoritmos de búsqueda a menudo producen rutas y caminos marcados por las concentraciones más altas de feromonas; por

lo tanto, los algoritmos basados en hormigas son particularmente adecuados para problemas de optimización discretos.

Figura 07: Búsqueda de comida de hormigas.

Fuente: Extraído de es.wikipedia.org/wiki/Algoritmo_de_la_colonia_de_hormigas#/media/File:Aco_branches.svg

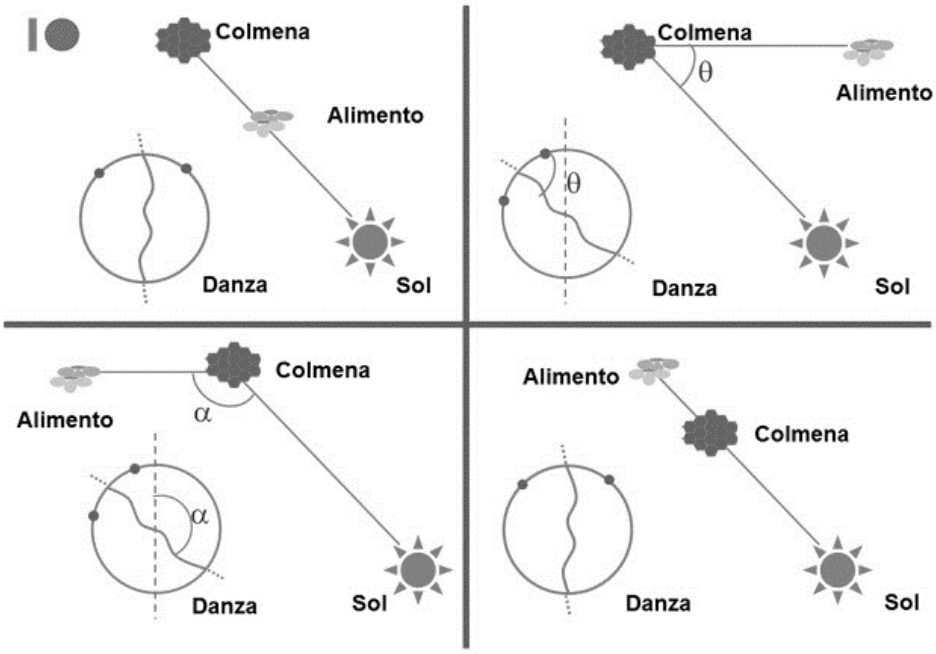


Algoritmo de Abejas (Bee Algorithms)

Estos algoritmos son más diversos, y algunos usan feromonas y la mayoría no. Casi todos los algoritmos de las abejas están inspirados en el comportamiento de alimentación de las abejas melíferas en la naturaleza. Las características interesantes, como la danza del movimiento, la polarización y la maximización del néctar se utilizan a menudo para simular la asignación de las abejas recolectoras a lo largo de parches de flores y, por lo tanto, diferentes regiones de búsqueda en el espacio de búsqueda.

Figura 08: Búsqueda de comida de abejas.

Fuente: Extraído de www.cs.us.es/~fsancho/?e=70



Optimización por Enjambre de Partículas (Particle Swarm Optimization)

Desarrollado por Kennedy and Eberhart (1995), está basado en el comportamiento del enjambre como la educación de peces y aves en la naturaleza. Este algoritmo busca el espacio de una función objetivo ajustando las trayectorias de los agentes individuales, llamadas partículas, como las trayectorias por partes formadas por vectores posicionales de una manera estocástica.

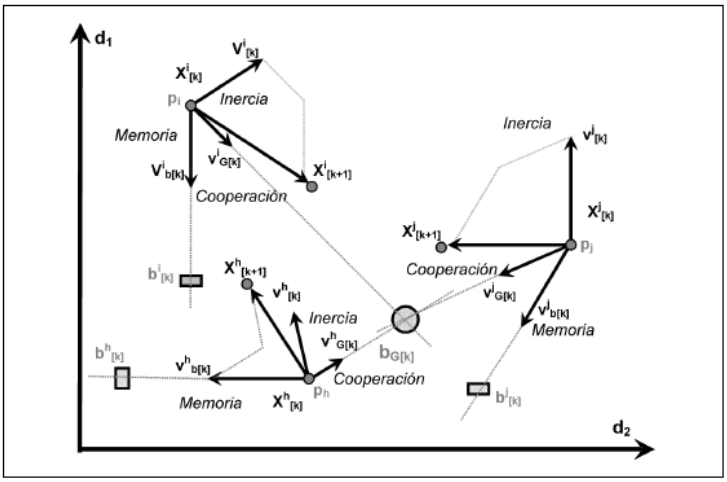
Su principal fortaleza es su rápida convergencia, que se compara favorablemente con muchos otros algoritmos de optimización global.

El movimiento de una partícula consta de dos componentes principales: un componente estocástico y un componente determinista. Cada partícula se ve atraída hacia

la posición de la mejor ubicación global actual y su mejor ubicación local en la historia, mientras que al mismo tiempo tiene una tendencia a moverse aleatoriamente.

Figura 09: Movimiento de Partículas.

Fuente: Extraído de www.cs.us.es/~fsancho/?e=70



Búsqueda Cuckoo (Cuckoo Search)

La búsqueda cuckoo es uno de los últimos algoritmos metaheurísticos inspirados en la naturaleza, desarrollado por Yang y Deb (2009). CS se basa en el parasitismo de cría de algunas especies de cuckoo. En general, la búsqueda cuckoo sigue tres reglas básicas derivadas del comportamiento de reproducción de estas especies:

- Cada cuckoo pone un huevo a la vez y lo tira en un nido elegido al azar.

- Los mejores nidos con huevos de alta calidad se trasladarán a las siguientes generaciones.

- El número de nidos anfitriones disponibles es fijo, y el ave huésped descubre el huevo puesto por un cuckoo con una probabilidad. En este caso, el ave huésped puede deshacerse del huevo o simplemente abandonar el nido y construir un nido completamente nuevo.

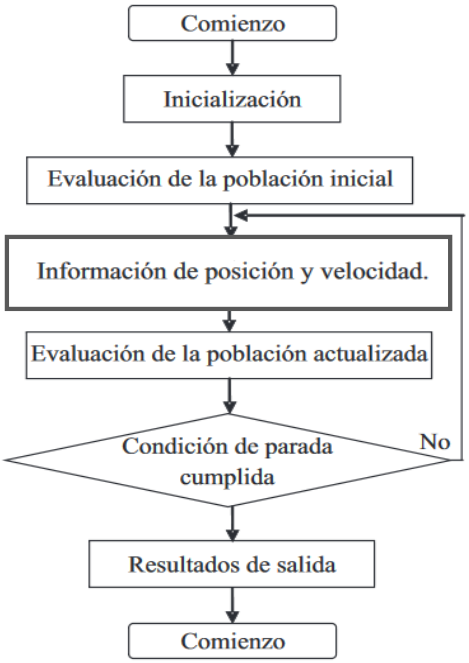
2.2.14 PARTICLE SWARM OPTIMIZATION

Este modelo está inspirado en patrones de comportamiento social de organismos que viven e interactúan dentro de grandes grupos, como el comportamiento social del vuelo de las bandadas de aves, bancos de peces o enjambres de abejas; de donde surge el paradigma de la "inteligencia de enjambre". PSO fue desarrollado por Jammes Kennedy y Russell Eberhart (1995), el modelo clásico de enjambre de partículas consiste en un enjambre de partículas, que se inicializan con una población de soluciones candidatas aleatorias, que representan el espacio de búsqueda.

Cada partícula se desplaza a través del espacio de búsqueda y recuerda la mejor posición que ha encontrado. Cada partícula comunica las buenas posiciones a las demás y dinámicamente ajustan su propia posición y su velocidad con base en las buenas posiciones. La velocidad se ajusta con el comportamiento histórico de las partículas.

Figura 10: Diagrama de flujo PSO

Fuente: Extraído de Algoritmo PSO para identificación de parámetros en un motor DC



Cada partícula se caracteriza por un vector de posición x_i, un vector de velocidad v_i, y una memoria que contiene el vector con la mejor posición de la partícula hasta el momento p_i. El vector de posición codifica una solución al problema de optimización, de modo que cada coordenada corresponde al valor de uno de los parámetros de la función objetivo. En cada iteración, la velocidad y la posición de cada partícula se actualiza. Además, en cada iteración se busca la mejor posición local de cada partícula, así como la mejor posición global del enjambre.

El primer paso, se inician los vectores posición y velocidad con valores aleatorios, además de iniciar la mejor posición local de cada partícula con la posición actual generada. Después, se realiza el cálculo de la función objetivo, o función fitness. Si el valor de la función objetivo actual es mejor que el mejor valor de la función objetivo, el

primero reemplaza al último. Esto es usado para determinar si es necesario la actualización del vector posición.

Luego de calcular el valor de la función objetivo, ambos se actualizan para determinar la siguiente posición de la partícula.

La velocidad de cada partícula se actualiza de la siguiente forma:

Ecuación 21: Actualización de velocidad.

Fuente: Extraído de Swarm Intelligence and Bio-Inspired Computation.

$$v_{ij}(t+1) = wv_{ij}(t) + c_1 r_1 (x_{ij}^{\#}(t) - x_{ij}(t)) + c_2 r_2 (x_j^{*}(t) - x_{ij}(t))$$

v_{ij} representa la velocidad del i-ésimo punto en la j-ésima dimensión, t representa el momento de iteración, c_1 representa el coeficiente de auto reconocimiento; o componente cognitivo; y c_2 representa el coeficiente de componente social, usualmente ambos con valores entre el intervalo [0, 2]. r_1 y r_2 son valores aleatorios entre el intervalo [0, 1] producidos en cada iteración. La variable w se denomina factor de inercia, cuyo valor se establece típicamente para variar linealmente de 1 a cerca de 0 durante el procesamiento iterado. $x_{ij}^{\#}$ es la mejor posición local y x_{ij}^{*} la mejor posición global. Luego de actualizada la velocidad, se actualiza la posición con:

Ecuación 22: Actualización de posición.

Fuente: Extraído de Swarm Intelligence and Bio-Inspired Computation.

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1)$$

x_{ij} representa la velocidad del i-ésimo punto en la j-ésima dimensión y V_{ij} es la velocidad anteriormente calculada. Si existiera alguna restricción programada, esta es revisada antes de seguir con la siguiente iteración.

El cálculo termina cuando se alcanza el número máximo de iteraciones, se alcanzó una solución óptima o se alcanzó alguna restricción.

PSO Binario.

En 1997, Kennedy y Eberhart, 1997 proponen una versión binaria del enjambre de partículas, con el objetivo de extender el campo de aplicación del algoritmo a problemas de optimización combinatoria. La modificación propuesta afecta a la codificación de las partículas, de modo que las coordenadas del vector de posición x_i pueden tomar únicamente los valores 0 o 1, y la fórmula de actualización del mismo, que ya no es la suma vectorial con el vector de velocidad. En su lugar, la posición se determina a partir de una distribución aleatoria de tipo sigmoidea, que depende de la velocidad. Para calcular la posición, primero se genera otro valor aleatorio, al que denominamos ψ, en el intervalo [0, 1], que se compara con la función sigmoidea. Esta distribución esta generada de forma que, a medida que la velocidad crece, la probabilidad de que la posición sea un 1 crece también.

Ecuación 23: Posición para PSO binario.

Fuente: Extraído de Metaheurísticas Aplicadas a Clustering.

$$x_{ij}(t+1) = \begin{cases} 1, & \psi < \dfrac{1}{1+e^{-v_{ij}(t+1)}} \\ 0, & caso\ contrario \end{cases}$$

2.2.15 SISTEMA DE RECOMENDACIONES

Jannach (2010). Los Sistemas de recomendación son herramientas y técnicas de software que proporcionan sugerencias para que ciertos ítems sean útiles para un usuario. Las sugerencias se relacionan con varios procesos de toma de decisiones, como qué artículos comprar, qué música escuchar o qué noticias en línea leer, etc. El término "Item" es utilizado para indicar lo que el sistema recomienda a los usuarios.

Construir un buen motor de recomendación plantea desafíos tanto para los actores del sistema, es decir, los consumidores y vendedores. Desde la perspectiva del consumidor, recibir ítems relevantes de una fuente confiable es fundamental para la toma de decisiones. Por lo tanto, el motor de recomendación debe construirse de tal forma que adquiera la confianza de los consumidores. Desde la perspectiva del vendedor, generar ítems relevantes para los consumidores a un nivel personalizado es muy importante. Con el aumento de las ventas en línea, los grandes jugadores ahora están recopilando grandes volúmenes de registros de interacción transaccional de los usuarios para analizar el comportamiento del usuario más profundamente que nunca. Además, la necesidad de recomendar en tiempo real aumenta el desafío. Con los avances en tecnología e investigación, los motores de recomendación están evolucionando para superar estos desafíos basados en el análisis de grandes datos y la inteligencia artificial.

Sistema de Recomendaciones Colaborativos.

Los RS de filtrado colaborativo, el filtrado de ítems de un gran conjunto de alternativas se realiza de forma colaborativa según las preferencias de los usuarios. La

suposición básica en un RS de filtrado colaborativo es que, si dos usuarios compartían los mismos intereses en el pasado, también tendrán gustos similares en el futuro. Existen dos subtipos dentro de esta clase de RS:

- Filtrado basado en usuarios: Las recomendaciones se generan al considerar las preferencias en el vecindario del usuario. El filtrado colaborativo basado en el usuario se realiza en dos pasos:

    - Identificar usuarios similares basados en preferencias de usuario similares.
    - Recomendar nuevos ítems a un usuario activo en función de la calificación otorgada por usuarios similares en los artículos no calificados por el usuario activo.

- Filtrado basado en ítems: Las recomendaciones se generan utilizando el vecindario de los ítems. A diferencia del filtrado basado en usuarios, primero se buscan similitudes entre los ítems y luego recomendamos los ítems no calificados que son similares a los ítems que el usuario activo calificó en el pasado.

    - Calcular la similitud del artículo en función de las preferencias del artículo.

Sistema de Recomendaciones basados en Contenido.

En el filtrado colaborativo, consideramos solo preferencias de elementos de usuario y construimos los sistemas de recomendación. Aunque este enfoque es preciso, tiene más

sentido si consideramos las propiedades del usuario y las propiedades de los elementos al construir motores de recomendación. A diferencia del filtrado colaborativo, utilizamos las propiedades del elemento y las preferencias del usuario para las propiedades del elemento al crear motores de recomendación basados en el contenido.

Un RS basado en el contenido utiliza la información de contenido de los artículos para construir el modelo de recomendación. Generalmente contiene un paso de generación de perfil de usuario, paso de generación de perfil de artículo y paso de creación de modelo para generar recomendaciones para un usuario activo. El RS basado en el contenido recomienda elementos a los usuarios tomando el contenido o las características de los elementos y perfiles de usuario.

2.2.16 EVALUACIÓN DE SISTEMAS DE RECOMENDACIONES.

Los RS se han evaluado tradicionalmente utilizando experimentos offline que intentan estimar el error de predicción de las recomendaciones utilizando un conjunto de datos existente de transacciones.

Algunos señalan las limitaciones de tales métodos, mientras que otros argumentan que la calidad de un RS nunca se puede medir directamente porque hay demasiadas funciones objetivas diferentes. Sin embargo, el uso generalizado de los RS hace que sea crucial desarrollar métodos para evaluar de forma realista y precisa su verdadero rendimiento y efecto en los usuarios.

Evaluar el rendimiento de los algoritmos de los RS no es fácil, porque diferentes algoritmos pueden ser mejores o peores dependiendo de la base de datos elegida. Según Herlocker (2004) una métrica para la exactitud de un RS mide, empíricamente, que tan cerca está el ordenamiento de ítems predichos para un usuario por el sistema con respecto al ordenamiento verdadero que el usuario haría, según su preferencia, de los mismos ítems.

Error Medio Absoluto (MAE)

Ecuación 24: Ecuación Error Medio Absoluto.

Fuente: Extraído de Metaheurísticas Aplicadas a Clustering.

$$MAE = \sum_{i=1}^{N} \frac{|p_i - q_i|}{N}$$

Permite medir la desviación de las recomendaciones predichas p_i y los valores reales q_i. A menor MAE mejor predice el RS las evaluaciones de los usuarios. El MAE puede dar una idea distorsionada del algoritmo para los sistemas que tienen como objetivo encontrar una lista de buenos elementos recomendables. El usuario tan sólo está interesado en los *N* primeros elementos de la lista. Tampoco es recomendable en sistemas en los que la salida es una decisión binaria de sí o no.

Error Medio Cuadrático

Ecuación 25: Ecuación Error Medio Cuadrático.

Fuente: Extraído de Metaheurísticas Aplicadas a Clustering.

$$EMC = \sqrt{\sum_{i=1}^{N} \frac{(|p_i - q_i|)^2}{N}}$$

El Error Medio Cuadrático proporciona la medida de las diferencias en promedio entre los valores pronosticados y los observados.

Curvas ROC (Receiver Operating Characteristic).

Herlocker (2004). Estos suministran una idea de la fortaleza de diagnóstico de un RS. Las curvas ROC dibujan la exactitud (Probabilidad de que un elemento malo del conjunto sea rechazado por el filtro) y la sensibilidad (probabilidad de que un elemento bueno al azar sea aceptado). Si un elemento es bueno o malo viene dado por las valoraciones de los usuarios. Las curvas se dibujan variando el umbral de predicción a partir del cual se acepta un elemento. El modelo de las curvas ROC trata de medir el grado con el cual un RS puede distinguir de manera exitosa entre relevancia y ruido

Ecuación 26: Proporción ROC de verdaderos positivos.

Fuente: Extraído de Recommender Systems Handbook.

$$\text{Re}call = TP\_rate = \frac{TP}{P}$$

Ecuación 27: Proporción ROC de falsos positivos.

Fuente: Extraído de Recommender Systems Handbook.

$$Fallout = FP\_rate = \frac{FP}{N}$$

Correlación Pearson

Herlocker (2004). Dos variables están correlacionadas si la varianza en una variable puede ser explicada por la varianza de la segunda. La correlación producto-momento de Pearson, la ρ de Spearman y el Tau de Kendall son tres de las medidas de correlación más conocidas La correlación de Pearson mide la extensión que existe en una relación lineal entre dos variables.

Ecuación 28: Correlación Pearson.

Fuente: Extraído de Recommender Systems Handbook.

$$C = \frac{\sum (x - \bar{x})(y - \bar{y})}{(n-1)S_x S_y}$$

2.3 MARCO CONCEPTUAL

Metaheurísticas.

Método heurístico para resolver un tipo de problema computacional general, usando parámetros dados por el usuario con procedimientos genéricos y abstractos de una manera que se espera sea suficientemente eficiente. En su mayoría, estos se usan cuando no existe una técnica heurística específica que dé una solución satisfactoria, cuando no se puede generar un método óptimo, o cuando se requiere de un conjunto de soluciones en lugar de una solución única.

Data Clustering.

Comprende el estudio de técnicas que tienen como finalidad el agrupamiento de objetos de tal manera que elementos de un mismo grupo de objetos; generalmente denominado clúster; tengan relación entre ellos.

Inteligencia de Enjambres.

Es una rama en la IA que estudia el comportamiento colectivo de sistemas descentralizados, autoorganizados; su diseño se inspira de sistemas biológicos o sociales existentes en la naturaleza.

Sistema de Recomendaciones.

Estos son una rama dentro de los sistemas de filtrado de información, que cuentan con la función de presentar distintos tipos de temas o ítems de información que son de interés de un usuario en particular. En general, comparan el perfil del usuario con algunas características de referencia de los temas, y busca predecir el "ranking" o ponderación que el usuario le daría a un ítem que aún el sistema no ha considerado.

Particle Swarm Optimization.

PSO permite optimizar un problema a partir de una población de soluciones candidatas, denotadas como "partículas", moviendo éstas por todo el espacio de búsqueda según reglas matemáticas que tienen en cuenta la posición y la velocidad de las partículas. El movimiento de cada partícula se ve influido por su mejor posición local hallada hasta el momento, así como por las mejores posiciones globales encontradas por otras partículas a medida que recorren el espacio de búsqueda. El fundamento teórico de esto es hacer que la nube de partículas converja rápidamente hacia las mejores soluciones.

Subtractive Clustering.

Pertenece a los algoritmos de clustering basados en densidad de funcionamiento rápido que permite la estimación de clústeres en un nconjunto de objetos, además de los respectivos centros.

2.4 HIPÓTESIS DE INVESTIGACIÓN

2.4.1 HIPÓTESIS GENERAL

El uso de Data Clustering e Inteligencia de Enjambres permite el desarrollo de un motor de Sistema de Recomendaciones que permita el agrupamiento de datasets multidimensionales.

2.4.2 HIPÓTESIS ESPECÍFICAS

- El método de Data Clustering utilizado permite el reconocimiento automático de centroides de clústeres y los clústeres iniciales.

- El método de Inteligencia de Enjambres utilizado realiza el refinamiento de los clústeres provistos por el método de Data Clustering, así como la solución óptima de los centroides.

- Los métodos mencionados son capaces de integrarse al desarrollo de un Sistema de Recomendaciones.

- Existe una óptima cohesión de datos dentro de los clústeres, buena separación de clústeres diferentes y una tasa de error (datos mal agrupados) mínima.

CAPÍTULO III

MATERIALES Y MÉTODOS

3.1 TIPO DE INVESTIGACIÓN

Los tipos de investigación de la tesis son cuasi experimental, exploratorio. Es cuasi experimental debido a que los datasets en algunos casos durante el desarrollo fueron ligeramente modificados para hacer ciertas pruebas. El carácter exploratorio se debe a que el tema de investigación desarrollado tiene escaso estudio en el entorno local. Además, las investigaciones y antecedentes que precedieron esta investigación en su mayoría de casos ocultan ciertas partes de información vital; aunque el autor no cree que haya sido a propósito; por lo que se requirió un gran esfuerzo al completar ciertas tareas.

3.2 POBLACIÓN DE INVESTIGACIÓN

Para la investigación se usaron datasets provistos por UCI Machine Learning Repository (https://archive.ics.uci.edu/ml/datasets.html)

3.3 MUESTRA DE INVESTIGACIÓN

A continuación, se realizará una descripción de los datasets usados:

Tabla 07: Datasets experimentales.

Fuente: Elaboración propia.

Nombre	Características de datos	Características de atributos	Número de instancias	Número de atributos
Annealing Dataset	Multivariado	Entero, Real, Categórico	798	38
Glass Identification Dataset	Multivariado	Real	214	10
Iris Dataset	Multivariado	Categórico, Real	150	4
Meta-data Dataset	Multivariado	Entero, Real, Categórico	528	22
Wine Dataset	Multivariado	Entero, Real	178	13

3.3.1 ANNEALING DATASET

Pertenece a un conjunto de datos de proceso de metalurgia que se utiliza para llevar un material a la consistencia, textura y dureza deseadas mediante un proceso de calentamiento y enfriamiento gradual. Este conjunto de datos contiene 38 atributos que se distribuyen en 6 tipos de valor continuo, 3 de valor entero y 29 de valor nominal, con un total de 798 datos. El listado de atributos es el siguiente:

Tabla 08: Descripción de Annealing Dataset.

Fuente: Elaboración propia.

Atributo	Valor / Descripción	Tipo
Family	--, GB, GK, GS, TN, ZA, ZF, ZH, ZM, ZS	Categórico, Nominal
Product Type	C, H, G	Categórico, Nominal
Steel	-, R, A, U, K, M, S, W, V	Categórico, Nominal
Carbon	Continuo	Real
Hardness	Continuo	Real
Temper_rolling	-, T	Categórico, Nominal
Condition	-, S, A, X	Categórico, Nominal
Formability	-, 1, 2, 3, 4, 5	Entero
Strength	Continuo	Real
Non-ageing	-, N	Categórico, Nominal
Surface-finish	P, M, -	Categórico, Nominal
Surface-Quality	-, D, E, F, G	Categórico, Nominal
Enamelability	-, 1, 2, 3, 4, 5	Categórico, Nominal
Bc	Y, -	Categórico, Nominal

Bf	Y, -	Categórico, Nominal
Bt	Y, -	Categórico, Nominal
Bw/me	B, M, -	Categórico, Nominal
bl	Y, -	Categórico, Nominal
m	Y, -	Categórico, Nominal
Chrom	Y, -	Categórico, Nominal
Phos	Y, -	Categórico, Nominal
CBond	Y, -	Categórico, Nominal
Marvi	Y, -	Categórico, Nominal
Expti	Y, -	Categórico, Nominal
Ferro	Y, -	Categórico, Nominal
Corr	Y, -	Categórico, Nominal
Blue/Bright/Varn/Clean	B, R, V, C, -	Categórico, Nominal
Lustre	Y, -	Categórico, Nominal
Jurofm	Y, -	Categórico, Nominal
S	Y, -	Categórico, Nominal
P	Y, -	Categórico, Nominal
Shape	COIL, SHEET	Categórico, Nominal
Thick	Continuo	Real
Width	Continuo	Real
Len	Continuo	Real
Oil	-, Y, N	Categórico, Nominal
Bore	0000, 0500, 0600, 0760	Real
Packing	-, 1, 2, 3	Entero

3.3.2 GLASS IDENTIFICATION DATASET

Este conjunto de datos representa el estudio de la clasificación de los tipos de vidrio que fue motivado por la investigación criminológica. En la escena del crimen, el vidrio que queda puede usarse como evidencia. Cuenta con 10 atributos y un total de 214 datos:

Tabla 09: Descripción de Glass Dataset.

Fuente: Elaboración propia.

Atributo	Valor / Descripción	Tipo
ID Number	1 – 214	Entero
RI	Continuo	Real
Na	Continuo	Real
Mg	Continuo	Real
Al	Continuo	Real
Si	Continuo	Real
K	Continuo	Real
Ca	Continuo	Real
Ba	Continuo	Real
Fe	Continuo	Real

Type of Glass	- 1: building_windows_float_processed - 2: building_windows_non_float_processed - 3: vehicle_windows_float_processed - 4: vehicle_windows_non_float_processed (valor no utilizado en dataset) - 5: containers - 6: tableware	Nominal

3.3.3 IRIS DATASET

Representa el conjunto de datos de cada una de tres especies de la flor Iris. Tiene 4 atributos con un total de 150 datos:

Tabla 10: Descripción de Iris Dataset.

Fuente: Elaboración propia.

Atributo	Valor / Descripción	Tipo
sepal length	Medida en cm.	Numérico
sepal width	Medida en cm	Numérico

petal length	Medida en cm	Numérico
petal width	Medida en cm	Numérico
class	Iris Setosa Iris Versicolor Iris Virginica	Categórico, Nominal

3.3.4 META-DATA DATASET

Este DataSet trata sobre los resultados del proyecto Statlog. El proyecto realizó un estudio comparativo entre algoritmos de aprendizaje estadísticos, neuronales y simbólicos. Se evaluaron aproximadamente 20 algoritmos diferentes en más de 20 conjuntos de datos diferentes. Las pruebas llevadas a cabo bajo el proyecto produjeron muchos resultados interesantes. Contiene 22 atributos con 528 datos.

Tabla 11: Descripción de Meta Dataset.

Fuente: Elaboración propia.

Atributo	Valor / Descripción	Tipo
DS	Nombre de dataset	Categórico
T	Número de muestras de prueba	Entero
P	Número total de muestras	Entero
n	Número de atributos	Entero
k	Número de clases	Entero
Bin	Número de atributos de tipo binarios	Entero

Cost	Coste (1 = Si, 0 = No)	Nominal
SDRatio	Desviación estándar	Real
Correl	Correlación mean	Real
Cancon1	Primera correlación canónica	Real
Cancon2	Segunda correlación canónica	Real
Fract1	Primer valor propio	Real
Fract2	Segundo valor propio	Real
Skewness	Ecuación |E(X-Mean)|^3/STD^3	Real
Kurtosis	Ecuación | E(X-Mean)|^4/STD^4	Real
Hc	Entropía de atributos	Real
Hx	Entropía de clases	Real
MCx	Entropía de mutua de clases y atributos	Real
EnAtr	Número de atributos equivalente	Entero
NSRatio	Radio de ruido	Real
Alg_Name	Nombre de algoritmo	Categórico
Nom_error	Error normalizado	Real

3.3.5 WINE DATASET

Pertenece a un conjunto de datos de análisis químico de vinos derivados de tres cultivares diferentes. El análisis determina las cantidades de 13 componentes encontrados en cada uno de los tres tipos de vinos. Con un total de 178 datos y con 13 atributos; todos continuos; que son:

Tabla 12: Descripción de Wine Dataset.

Fuente: Elaboración propia.

Atributo	Valor / Descripción	Tipo
Wine	Tipo de Vino	Nominal
Alcohol	Grado de Alcohol	Real
Malic Acid	Continuo	Real
Ash	Continuo	Real
Alcalinity of ash	Continuo	Real
Magnesium	Continuo	Real
Total Phenols	Continuo	Real
Flvanoids	Continuo	Real
Nonflavanoid phenols	Continuo	Real
Proanthocyanins	Continuo	Real
Color Intensity	Continuo	Real
Hue	Continuo	Real

OD280/OD315 of diluted wines	Continuo	Real
Proline	Continuo	Real
Alcohol	Continuo	Real
Malic Acid	Continuo	Real
Ash	Continuo	Real
Alcalinity of ash	Continuo	Real
Magnesium	Continuo	Entero
Total Phenols	Continuo	Real
Flvanoids	Continuo	Real
Nonflavanoid phenols	Continuo	Real
Proanthocyanins	Continuo	Real
Color Intensity	Continuo	Real
Hue	Continuo	Real
OD280/OD315 of diluted wines	Continuo	Real
Proline	Continuo	Entero

3.4 MATERIAL EXPERIMENTAL

Para el desarrollo de la investigación se utilizaron los siguientes recursos:

- Sistema Operativo Ubuntu 16.04, preferencias de autor.

- Editor de textos Sublime Text 3 como entorno de desarrollo.

- El lenguaje de programación escogido para el desarrollo es Python 3.6, debido a preferencias de autor.

- Se hizo el uso de la herramienta Git para el seguimiento del proyecto.

- Datasets experimentales mencionados en el apartado de muestras.

3.5 MÉTODOS DE TRATAMIENTO DE DATOS

Para el tratamiento de datos se utilizarán los siguientes indicadores:

- Cuantificación de error:

Ecuación 29: Prueba de error.

Fuente: Extraído de Metaheurísticas Aplicadas a Clustering.

$$Error - C = \frac{\sum_{i=1}^{K}(\sum_{z \in C_i} d(c_i, z)/ \mid C_i \mid)}{K}$$

- Distancia Intra Clúster:

Ecuación 30: Prueba Intra Clúster.

Fuente: Extraído de Metaheurísticas Aplicadas a Clustering.

$$Intra - C = \frac{\sum_{i=1}^{K}(\sum_{z,t \in C_i} d(z,t)/ \mid C_i \mid)}{K}$$

- Distancia Inter Clúster

Ecuación 31: Prueba Inter Clúster.

Fuente: Extraído de Metaheurísticas Aplicadas a Clustering.

$$Inter - C = \frac{\sum_{i=1}^{K-1}\sum_{j=i+1}^{K} d(c_i, c_j)}{\sum_{i=1}^{K-1} i}$$

- Coeficiente Silhouette

Ecuación 32: Prueba Silhouette.

Fuente: Extraído de Metaheurísticas Aplicadas a Clustering.

$$s(i) = \frac{b - a}{max(a, b)}$$

En cada uno de estos indicadores la función *d* refiere a la distancia euclidiana, que corresponde a una manera de medición de similitud dentro de los RS.

Dentro de la implementación del algoritmo SC, se ajustaron los valores del radio de aceptación y radio de rechazo dentro del rango de [0, 1], esto para hallar el número óptimo de clústeres. Luego, para comprobar el buen agrupamiento de los datos hacia sus centroides se usó el coeficiente silhouette con la salida del algoritmo PSO.

La modificación de estos valores impactó directamente en las medidas de error, intra clúster, inter clúster y coeficiente silhouette. La interpretación de los resultados se dará de la siguiente forma:

La distancia intracluster muestra la media de separación entre datos de un clúster, valores más elevados significan que se están aceptando elementos probablemente más distantes que aún guardan cierta correlación con los centroides. Los valores más pequeños indican que existe una mayor cohesión de datos, aunque también puede implicar que existe una menor cantidad de elementos por clúster.

La distancia intercluster muestra la media de separación de los clústeres, valores más altos indican una mayor separación de clústeres.

La tasa de error, si este está más cerca de 0, existe una mayor probabilidad de que los elementos estén bien agrupados. Los valores de r_a y r_b tienen un impacto directo en la cantidad de clústeres. Se mostrará estos valores al ajustar r_a y r_b.

El coeficiente silhouette muestra buen agrupamiento de todos los datos en los datasets a sus respectivos centroides; este resultado siempre está dentro del intervalo [-1, 1]. Valores cercanos a 1 indican un agrupamiento óptimo, mientras que valores cerca de -1 indican incapacidad de determinar clústeres bien definidos y por lo tanto aleatoriedad de agrupamiento en los datos. Esta medida es la más relevante de las anteriormente descritas.

CAPÍTULO IV

RESULTADOS Y DISCUSIÓN

4.1 IMPLEMENTACIÓN DEL ALGORITMO DE NORMALIZACIÓN

Luego de haber analizado y comprendido la estructura de cada uno de los datasets de prueba, se optó por construir un módulo de estandarización y optimización, el cual cuenta con funciones de Transformación Nominal (numérica) y Transformación Categórica. La entrada de este módulo es el dataset en crudo.

La Transformación Nominal hizo uso de la técnica rescaling (re escalado), que permite a los datos de tipo numérico (real, entero, fraccionario) quedar con valores entre el rango de intervalo [0, 1].

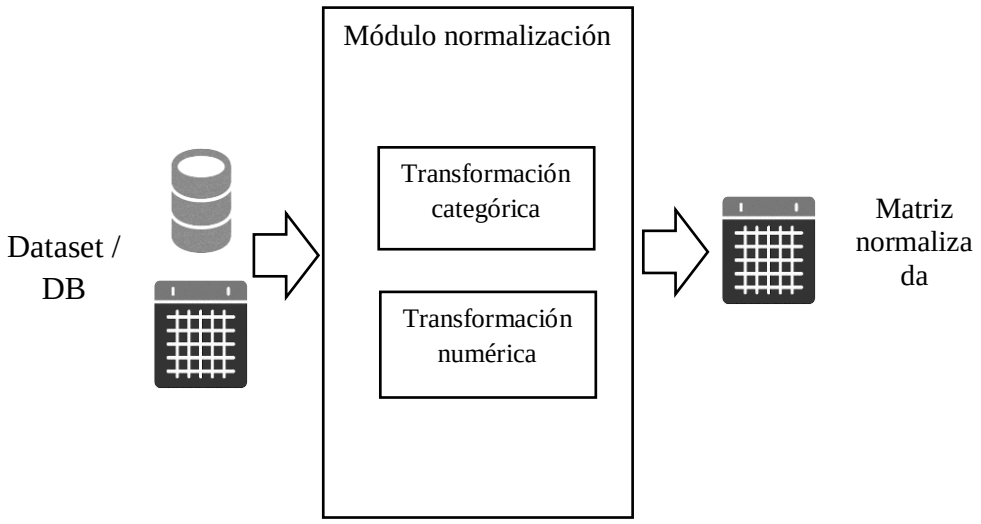
Para la transformación categórica, se usó la tecnica Dummy Transformation (transformación tonta), el cual crea una matriz de valores existentes. Ya que en la mayoría

de datasets de prueba no existen muchos campos con datos categóricos, se eligió esta técnica por la facilidad de implementación.

Este módulo se diseñó para una mayor comodidad y facilidad de entendimiento, además de permitir ajustar valores más fácilmente en el módulo Subtractive Clustering. La salida de este módulo es una matriz con valores normalizados entre los valores [0, 1] para todos los atributos.

Figura 11: Módulo de normalización.

Fuente: Elaboración propia.



4.2 IMPLEMENTACIÓN DEL ALGORITMO SUBTRACTIVE CLUSTERING

Para la implementación del algoritmo Subtractive Clustering, se hicieron los siguientes ajustes al algoritmo original:

- Los valores r_a y r_b fueron ajustados a 0.8 y 0.5 respectivamente.
- Los valores de *accept_ratio* y *reject_ratio* fueron ajustados manualmente para determinar los elementos por clúster.

- Se usó la distancia euclidiana para determinar los valores de densidad entre los vectores de datos.

- Se usaron límites máximos y mínimos de aceptación para posibles candidatos a clúster. Además, esta restricción ayudo a determinar la cantidad de clústeres.

La entrada de este módulo corresponde a la matriz normalizada del módulo anterior. La salida es una lista de clústeres, donde el primer elemento de cada clúster corresponde al centroide del mismo.

4.3 IMPLEMENTACIÓN DEL ALGORITMO PARTICLE SWARM OPTIMIZATION

El módulo de PSO refina el valor del centroide de clúster, además tiene la función de repartir datos que se encuentren entre dos o varios clústeres a la vez. Este módulo utilizó una variante de PSO llamada BRPSO (Boundary Restriction – Particle Swarm Optimization) que limita el movimiento de las partículas cuando estas salen de un rango determinado. Las modificaciones hechas al algoritmo original son:

- Las partículas están definidas con los siguientes elementos:

  - Data: Array de centroides posibles de clústeres (se incluye una partícula con los centros encontrados en el algoritmo SC)

  - IClass: Cluster al que pertenece la partícula (definida por la función fitness).

- Position: Posición actual de la partícula con respecto al centro

- Velocity: Velocidad de movimiento de la partícula.

- fvalue: Valor de la función fitness para la partícula.

- LBest: Mejor posición local, esto permitirá determinar el mejor centro del cluster de la partícula.

- Las velocidades iniciales de cada partícula fueron puestas a 0.
- Las posiciones iniciales de cada partícula se determinaron de manera aleatoria dentro del intervalo [0, 1]

- El peso inercial *w* en cada iteración fue recortada para evitar movimientos de posición bruscos en el espacio de búsqueda. Se utilizó la siguiente variante:

Ecuación 33: Variante de peso inercial.

Fuente: Extraído de Clustering Multidimensional Data with PSO based Algorithm.

$$w = maxw * \left(exp(-iter)\right)$$

Donde *w* es el valor de peso inercial actual en cada iteración, *maxw* es el peso inercial inicial, e *iter* es el número de iteración al momento.

- La función fitness está definida por la menor distancia euclidiana del punto de dato hacia los centroides encontrados en SC.

- Cuando se inicia la evaluación del proceso de agrupamiento, algunas partículas tienden a viajar más allá del área del espacio de búsqueda, lo que puede ocasionar un clustering inexacto. Para evitar esto, se puso una restricción de la siguiente forma:

if position ≤ upper_bound **and** position ≥ lower_bound:

then position = position

else:

then position = position - velocity

El valor para upper_bound será 2, el valor para lower_bound será 0.

4.4 IMPLEMENTACIÓN DEL MÓDULO DE PRUEBAS (Measures)

El módulo de pruebas define los métodos de tratamiento descritos anteriormente. Este módulo está compuesto por 3 funciones:

- ErrorQuantify, que define la tasa de error en los clústeres hallados, se usa la función de la figura 39.
- IntraCluster, que define la media de la similitud entre elementos de un clúster, se usa la función de la figura 40.

- InterCluster, que define la media de la separación entre clústeres, se usa la función de la figura 41.
- Coeficiente silhouette, que indica el número óptimo de clústeres, se usa la función de la figura 42.

Luego de implementar el RS con los módulos diseñados, se procedió a probar el sistema en los datasets de prueba. En cada uno de las pruebas realizadas, se incorporó además de las medidas, los valores de prueba de *accept_ratio* y *reject_ratio*, el número de clústeres y la cantidad de elementos por clúster.

4.5 ANÁLISIS DE RESULTADOS EN ANNEALING DATASET

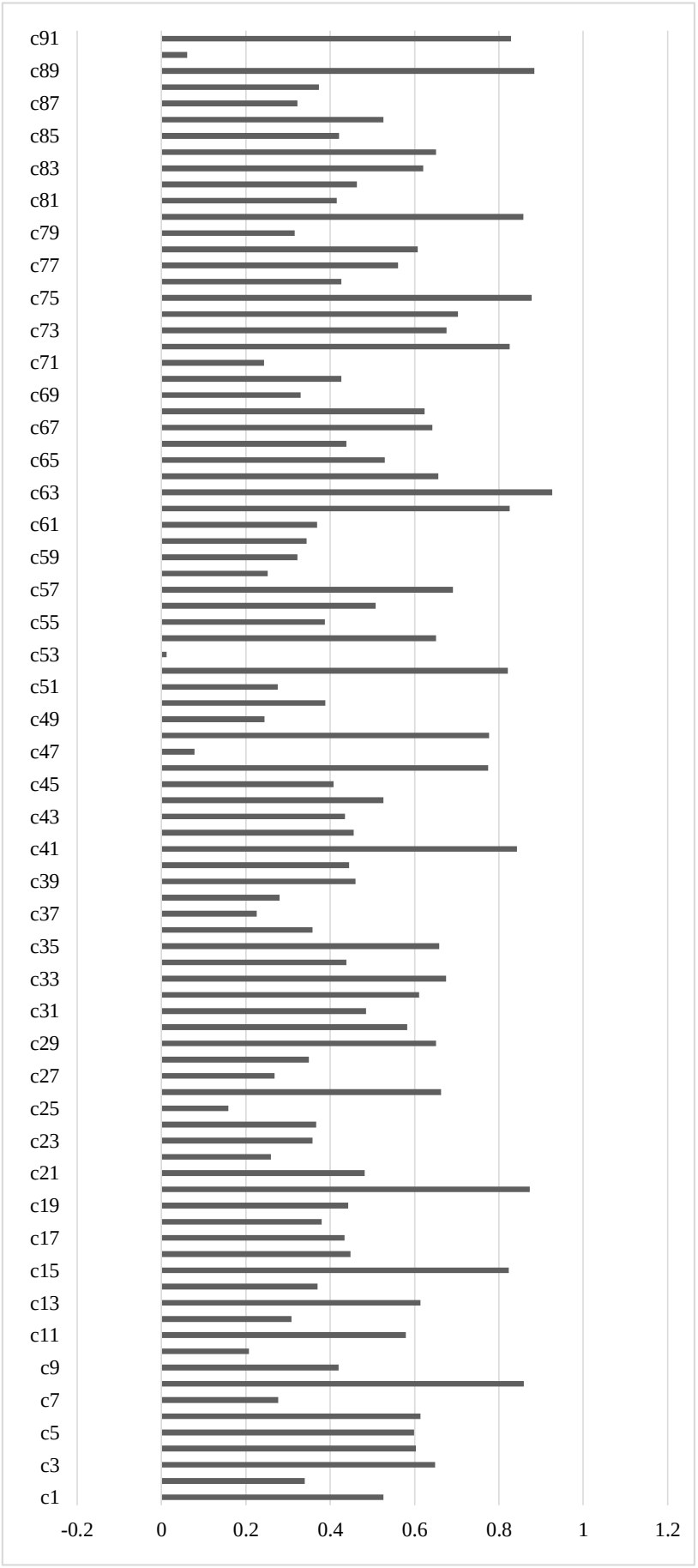
Tabla 13: Resultados en Annealing Dataset.

Medida	**Valor obtenido**
Radio de aceptación	0.1
Radio de rechazo	0.1
Promedio tasa de error	0.73264873
Distancia intracluster media	9.5460565
Distancia intercluster media	6.9330413
Coeficiente Silhouette	0.44109906
Número de clústeres	91
Instancias por clúster	0, 19, 13, 17, 14, 9, 12, 8, 12, 11, 10, 17, 8, 13, 7, 9, 11, 19, 11, 5, 7, 9, 12, 11, 11, 7, 12, 17, 7, 8, 10, 7, 5, 7, 6, 6, 11, 19, 9, 11,

4, 9, 8, 6, 9, 4, 13, 4, 19, 9, 8, 4, 11, 8, 10, 7, 4, 17, 9, 12, 8, 4, 3, 5, 5, 6, 6, 5, 5, 7, 7, 3, 7, 5, 3, 7, 6, 4, 11, 3, 6, 13, 6, 4, 6, 4, 10, 7, 3, 14, 3

En este dataset se usó la totalidad de atributos en la prueba. El análisis del valor obtenido del coeficiente silhouette muestra una estructura débil de clústeres, que se debe a la variabilidad de los datos dentro del dataset. La siguiente figura muestra los valores individuales de coeficiente silhouette por clúster, con los clústeres 53 y 90 con valores bajos en comparación al resto de los clústeres. En promedio se puede observar que los valores no superan el rango de 0.5, por lo cual estos clústeres tienen un enlace débil entre sus datos.

Figura 12: Valores Silhouette en Annealing.



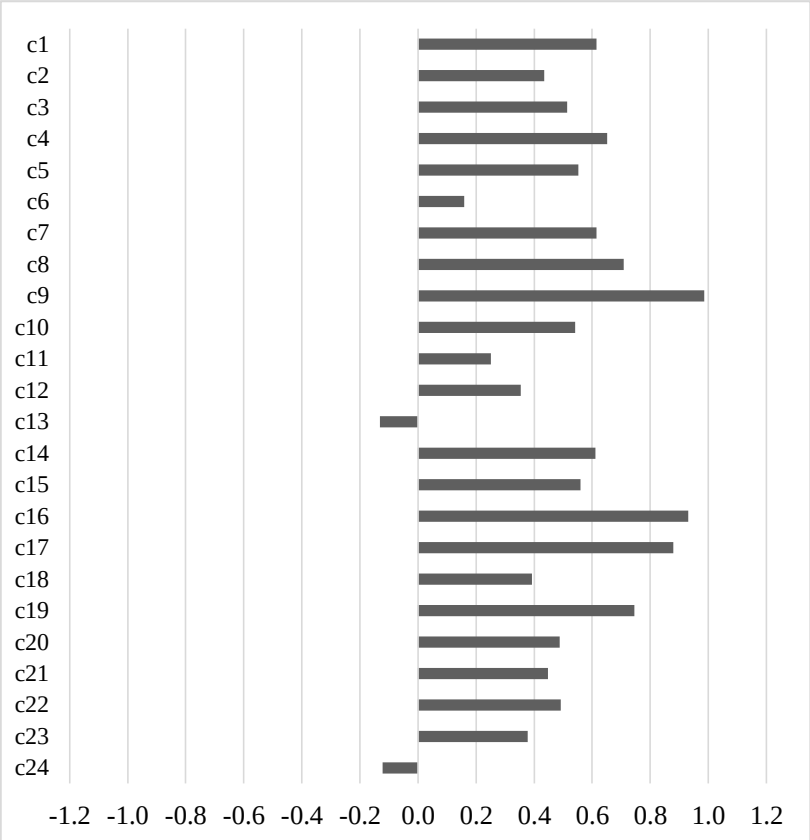
4.6 ANÁLISIS DE RESULTADOS EN GLASS CLASSIFICATION DATASET

Tabla 14: Resultados en Glass Dataset.

Medida	**Valor obtenido**
Radio de aceptación	0.1
Radio de rechazo	0.1
Promedio tasa de error	1.8757865
Distancia intracluster media	50.0113
Distancia intercluster media	16.810680
Coeficiente Silhouette	0.38590939
Número de clústeres	24
Instancias por clúster	10, 13, 11, 7, 10, 10, 5, 6, 4, 8, 10, 6, 36, 6, 8, 14, 3, 8, 4, 7, 4, 6, 4, 14

En este dataset se usó la totalidad de atributos en la prueba. El análisis del valor obtenido del coeficiente silhouette muestra que se formó una agrupación artificial, por lo que los datos de todos los clústeres obtenidos pueden no estar correctamente agrupados.

Figura 13: Valores Silhouette en Glass.



Como se puede observar en la imagen, los clústeres 13 y 24 presentan valores negativos, que indican un mal agrupamiento de los datos. De todos los clústeres, el número 9 presenta mejor agrupamiento.

4.7 ANÁLISIS DE RESULTADOS EN IRIS DATASET

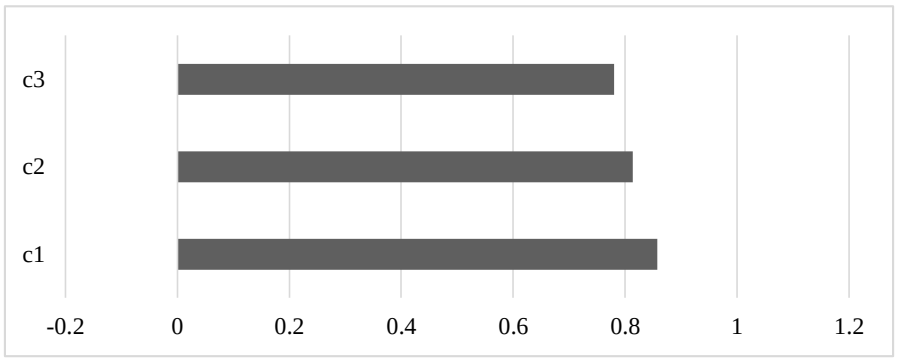
Tabla 15: Resultados en Iris Dataset.

Medida	**Valor obtenido**
Radio de aceptación	0.2
Radio de rechazo	0.1
Promedio tasa de error	0.20738545

Distancia intracluster media	14.132204
Distancia intercluster media	3.2247519
Coeficiente Silhouette	0.81716687
Número de clústeres	3
Instancias por clúster	50, 50, 50

En este dataset se usó la totalidad de atributos en la prueba. El análisis del valor obtenido del coeficiente silhouette muestra que se una estructura estable de lo clústeres.

Figura 14: Valores Silhouette en Iris.



Se puede apreciar que los coeficientes personales de cada clúster están dentro del rango óptimo de agrupamiento.

4.8 ANÁLISIS DE RESULTADOS EN META-DATA DATASET

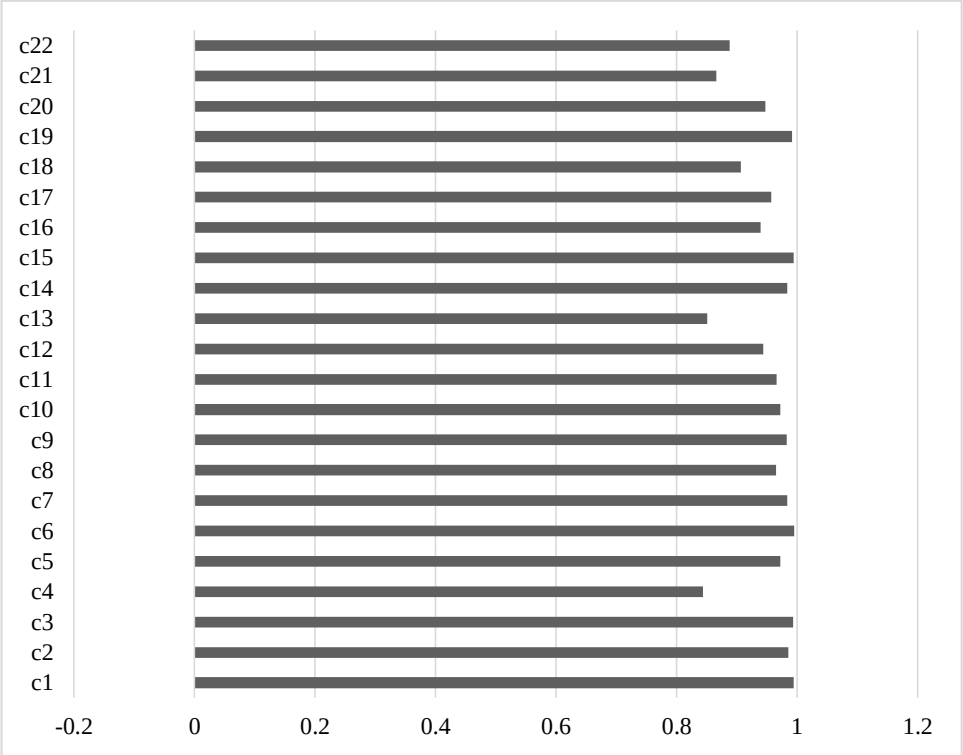
Tabla 16: Resultados en Meta Dataset.

Medida	**Valor obtenido**
Radio de aceptación	0.3
Radio de rechazo	0.3
Promedio tasa de error	0.93632445
Distancia intracluster media	37.816959
Distancia intercluster media	292.96974
Coeficiente Silhouette	0.95134653
Número de clústeres	22
Instancias por clúster	24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 19, 17, 11, 24, 13, 7, 524, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 17, 11, 24, 13, 724, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24

En este dataset se usó la mayoría de atributos en la prueba, exceptuando los atibutos correspondientes al Nombre de dataset (1) y al Nombre del algoritmo (21), ya que estos generaban pésimos resultados (exceso de clústeres hallados). El análisis del valor obtenido del coeficiente silhouette muestra que se una estructura estable de lo clústeres.

La siguiente figura muestra el coeficiente individual de los clústeres, que en su mayoría está dentro del rango de estructuras robustas (≥ 0.75), con el menor valor en clúster 4.

Figura 15: Valores Silhouette en Meta.



4.9 ANÁLISIS DE RESULTADOS EN WINE DATASET

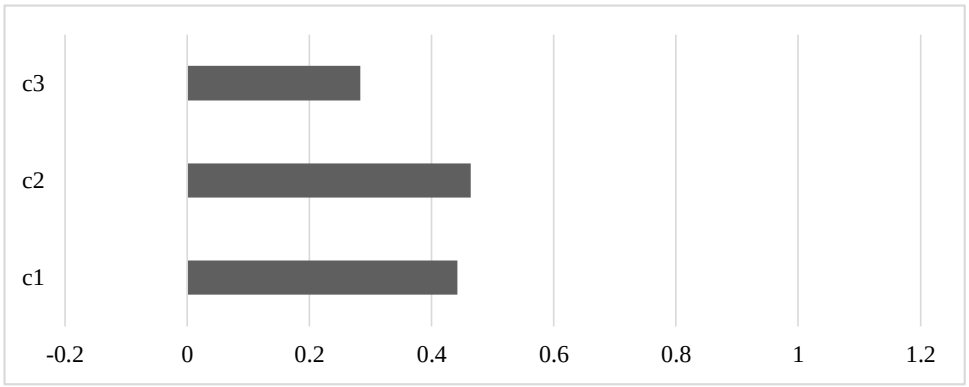
Tabla 17: Resultados en Wine Dataset.

Medida	Valor obtenido
Radio de aceptación	0.3
Radio de rechazo	0.3
Promedio tasa de error	14.383991
Distancia intracluster media	892.8187
Distancia intercluster media	46.210163

Coeficiente Silhouette	0.41527125
Número de clústeres	3
Instancias por clúster	25, 108, 45

En este dataset se usó la totalidad de atributos en la prueba. El análisis del valor obtenido del coeficiente silhouette muestra que se una estructura débil de lo clústeres, por lo cual es necesario hacer ajustes de los atributos a usar. La siguiente figura muestra los valores individuales del coeficiente, que en su totalidad se encuentran en el rango [0.26, 0.5]

Figura 16: Valores Silhouette en Wine.



CONCLUSIONES

Primero: El algoritmo Subtractive Clustering es una técnica rápida para estimar el número de clústeres y los centros de clúster en un conjunto de datos. Las estimaciones de clúster obtenidas a partir de esta pueden utilizarse para inicializar métodos de agrupación basados en la optimización iterativa, como lo es el algoritmo PSO.

Segundo: El algoritmo PSO permite una rápida convergencia de los puntos de datos a los centroides encontrados por el algoritmo SC. Además, de entre todas las técnicas de SI existentes, PSO es una técnica relativamente fácil de implementar para distintos escenarios, como en este caso el de clustering.

Tercero: En el análisis de cohesión de datos se muestra que la distancia intracluster aumenta conforme el radio de aceptación aumente y el radio de rechazo disminuye, ya que se consideran datos distantes. Reducir el radio de aceptación y aumentar el radio de rechazo disminuye esta distancia, aumentando la cohesión de datos a la vez que excluye otros elementos que se toman por menos coherentes con respecto al clúster. En cuanto al análisis de separación de clústeres la distancia intercluster no muestra un patrón sucesivo en los resultados. Esto puede se debe a que, si existen más clústeres, es posible que los centroides empiecen a alejarse o acercarse indiferentemente; esto es debido a la densidad de los datos. Cuyo caso la distancia intercluster guarda mayor relación respecto a la densidad de los centroides y la cantidad de centroides.

Cuarto: Analizando los valores del coeficiente silhouette se puede determinar que no para todos los casos se puede obtener un óptimo resultado de agrupamiento utilizando todos los atributos del dataset, teniendo como consecuencia una estructuración débil de los clústeres. Esto tiene un impacto negativo a la hora de desarrollar el RS, ya que se agrupan datos que posiblemente no tengan mucha coherencia entre sí. Comparando los valores obtenidos en la tasa de error realizada en cada prueba muestra que este aumenta conforme a menos clústeres se identifican dentro del dataset. Esto se relaciona directamente con la distancia la media de distancia intracluster hallada, ya que al aumentar el radio de aceptación de clústeres es posible incluir datos que no tengan suficiente semejanza entre ellos.

Quinto: Un mayor número de clústeres conlleva a reducción de tasa de errores y mayor coherencia de datos, aunque no signifique necesariamente una menor separación entre clústeres. Hallar el balance de número de clústeres ideal dependerá significativamente de las modificaciones a los parámetros de radio de aceptación y rechazo; y cuyo caso; dependerá mayormente del entorno de datos y de un analista para determinar el número de clústeres ideal. Respecto a la cantidad de elementos por clúster es variable, esto es porque el RS no busca un balanceo de elementos entre clústeres, sino la agrupación de elementos comunes, que depende; en este caso, de la distancia euclidiana de cada uno de los elementos.

RECOMENDACIONES

Primero: Para próximas investigaciones, se recomienda un estudio más profundo de las técnicas de normalización y estandarización de datos.

Segundo: En casos de exceso de atributos categóricos, se recomienda el uso de otras técnicas de normalización, como el análisis de componentes principales, que reduce valores categóricos a 0 con el fin de brindar a otros valores una escala numérica sin necesidad de aumentar columnas de atributos.

Tercero: A cualquier investigador que desee profundizar en el tema, el algoritmo PSO puede ser optimizado utilizando técnicas de programación concurrente o paralela, que permitirá reducir el tiempo de convergencia de los puntos de datos hacia sus respectivos centroides.

Cuarto: A cualquier investigador que desee profundizar en el tema, se recomienda que en caso de que el dataset o base de datos cuente con demasiados atributos (dimensiones), se haga una selección de atributos a usar, descartando atributos no relevantes.

Quinto: A cualquier investigador que desee profundizar en el tema, se recomienda que si no se encuentra una estructura lo suficientemente robusta (silhouette ≥ 0.51), se haga una selección de atributos relevantes para el proceso de clustering en el RS.

REFERENCIAS

Hammouda K. & Karray F.(2000). *A comparative study of data clustering techniques. Tools of Intelligent Systems Design*. 5.

Bataineh K. & Naji M. & Saqer M.(2011). *A Comparison Study between Various Fuzzy Clustering Algorithms*. Jordan Journal of Mechanical and Industrial Engineering, 5(4), 335-343.

Ahmed K. & Razak T.(2016). *A Study on Metaheuristic Optimization Approach for Data Clustering*. International Journal of Emerging Trends

Sidhu et al.. (2010). *A Subtractive Clustering Based Approach for Early Prediction of Fault Proneness in Software Modules*. International Journal of Computer and Systems Engineering, 4(7), 1165-1169.

JunYing C. & Zheng Q. & Ji J.(2008). *A Weighted Mean Subtractive Clustering Algorithm. Information Technology Journal*. 7.Science and Technology, 3(8), 94-101.

Khan A. & Bawane N.G. & Bodkhe S.(2010). *An Analysis of Particle Swarm Optimization with Data Clustering-Technique for Optimization in Data Mining*. International Journal on Computer Science and Engineering, 2(7), 2223-2226.

Chuang L.-Y. & Lin Y.-D. & Yang, C.-H.(2012) *An Improved Particle Swarm Optimization for Data Clustering*. In: Proceedings of the International Multiconference of Engineers and Computer Scientists Hong Kong, vol. I, pp. 440–445.

Sarkar S. & Roy A. & Purkayastha, B.(2013). *Application of Particle Swarm Optimization in Data Clustering: A Survey*. International Journal of Computer Applications , 65(25), 38-46.

Gorakala S.(2017). *Building Recommendation Engines*. Packt Publishing.

Agresti A.(2012). *Categorical Data Analysis, 3rd Edition*. Wiley.

Azab S. & Hefny H.(2017). *Center of Gravity PSO for Partitioning Clustering*. International Journal of Advanced Research in Computer Science and Software Engineering. 7.

Rovira A.(2009). *Clasificación mediante Enjambre de Prototipos*. Universidad Carlos III de Madrid.

Soler J. & Gaubert L. & Tencé F. & Buche C.(2013). *Data Clustering and Similarity*. 492-495.

Zhu S. & Wang D. & Li T.(2010). *Data Clustering with Size Constraints*. Knowledge-Based Systems, 23(8), 883-889.

Reddy C. & Aggarwal C.(2013). *Data Clustering: Algorithms and Applications*. Chapman and Hall/CRC.

Ma C. & Gan G. & Wu J.(2007). *Data Clustering: Theory, Algorithms, and Applications*. SIAM.

Tan Y. & Shi Y.(2016). *Data Mining and Big Data*. Springer.

Sarafis I.(2005). *Data Mining Clustering of High Dimensional Databases ith Evolutionary Algorithms*. Heriot-Watt University.

Aggarwal C.(2015). *Data Mining: The Textbook*. Springer.

Zaki M. & Meira W.(2014). *Data Mining and Analysis: Fundamental Concepts and Algorithms*. Cambridge University Press.

Pierson L.(2015). *Data Science For Dummies*. For Dummies.

Abubaker M.(2011). *Efficient Data Clustering Algorithms*. Islamic University of Gaza.

Pradeep S. & Sinha P. & Kulhare R.(2017). *Enhanced Clustering Based on K-means Clustering Algorithm and Proposed Genetic Algorithm with K-means Clustering*. Current Trends in Technology and Science, 6(2), 710-713.

Izakian Z. & Mesgari M.(2015). *Fuzzy clustering of time series data: A particle swarm optimization approach*. Journal of AI and Data Mining, 3(1), 39-46.

Chiu S.(1994). *Fuzzy Model Identification Based on Clustering Estimation*. Journal of Intelligent and Fuzzy Systems, 2(1), 267-278.

Muñoz M. & López J. & Caicedo E.(2008). *Inteligencia de enjambres: sociedades para la solución de problemas (una revisión)*. Revista Ingeniería e Investigación, 28(2), 119-130.

Moertini V.(2002). *Introduction to Five Data Clustering Algorithms*. Integral, 7(2), 87-96.

Villagra, A.(2007). *Metaheurísticas aplicadas a Clustering*. Universidad Nacional de la Patagonia Austral.

Joaquín L. & Benjamín B.(2006). *Optimización de Enjambre de Partículas aplicada al Problema del Cajero Viajante Bi-objetivo*. Revista Iberoamericana de Inteligencia Artificial, 10(32), 67-76.

Chen C. & Ye F.(2004). *Particle swarm optimization algorithm and its application to clustering analysis*. IEEE International Conference on Networking, Sensing and Control, 2004, 2, 789-794.

Sok-ping K. & Salim N.(2006). *Optimized Subtractive Clustering for Cluster-Based Compound Selection*. Proceedings of the 1st International Conference on Natural Resources Engineering & Technology 2006, 1(1), 492-499.

Ujjin S & Bentley P.(2003). *Particle Swarm Optimization Recommender System*. Proceedings of the IEEE Swarm Intelligence Symposium 2003, 1, 124-131.

Ghorpade-aher J. & Metre, V.(2014). *PSO based Multidimensional Data Clustering*: A Survey. International Journal of Computer Applications, 87(16), 41-48.

Idris I. & Felipe Martins L. & Persson, M. & Czygan, M. & Vothihong P.(2017). *Python: End-to-end Data Analysis*. Packt Publishing.

Felfernig A. & Jannach D. & Friedrich G. & Zanker M.(2010). *Recommender Systems: An Introduction*. Cambridge University Press.

Ricci F. & Rocach L. & Shapira B. & Kantor P.(2010). *Recommender Systems Handbook*. Springer.

Holguin, E. (2013). *Recuperación Semántica de la Información usando la Similitud Distribucional*. Universidad Nacional del Altiplano.

Andritsos P.(2004). *Scalable Clustering of Categorical Data and Applications*. University of Toronto.

Kuan-Ching L. & Jiang H. & Yang L. & Cuzzocrea A.(2015). *Singular Value Decomposition, Clustering, and Indexing for Similarity Search for Large Data Sets in High-Dimensional Spaces. En Big Data: Algorithms, Analytics, and Applications(32)*. Chapman and Hall/CRC.

Rolando, E.(2012). *Sistemas de Recomendación de Contenidos para Libros Inteligentes*. Universidad de Oviedo.

Seguido, M.(2009). *Sistemas de recomendación para webs de información sobre la salud*. Universidad Politécnica de Cataluña.

Rao U. & Sood, Y & Jarial R.(2015). *Subtractive clustering Fuzzy Expert System for Engineering Applications*. Procedia Computer Science, 48, 77-83.

Cadie F.(2017). *The Data Science Handbook*. Wiley.

ANEXOS

Anexo 1: Módulo normalización de datos "normalize.py".

```
from decimal import *

import csv



getcontext().prec = 8



## Transformacion de datos numericos - Rescaling

## Output Range [0,1]

def NumericalTransform(row):



    trow = []



    min = row[0]

    max = row[0]



    for value in row:

            if value < min:
```

```
                min = value

            if value > max:

                max = value



    fract = Decimal(max) - Decimal(min)

    for value in row:

            upper = Decimal(value) - Decimal(min)

            trow.append(upper / fract)



    return trow



## Transformacion de datos categoricos - Dummy Transformation

## Matrix values

## ('a')   (1, 0)

## ('b') = (0, 1)

## ('a')   (1, 0)

def CategoricalTransform(row):
```

```
    cvalues = []

    for value in row:

        if value not in cvalues:

            cvalues.append(value)



    trow = []

    for i in range(len(row)):

        trow.append([ Decimal(0) for j in range(len(cvalues)) ])



    for i in range(len(row)):

        pos = cvalues.index(row[i])

        trow[i][pos] = Decimal(1)



    return trow


## Funcion de normalizacion

def Normalize(dataset):
```

```
print("Iniciando normalizacion de dataset.")

normalize_matrix = []

with open(dataset, newline='') as data_input:

        entries = csv.reader(data_input, delimiter=',')

        matrix = []

        for row in entries:

                matrix.append(row)

        dim = len(matrix[0])

        reverse = [ [row[i] for row in matrix] for i in range(dim)]

        del(matrix)



        for row in reverse:

                flag = 0

                for value in row:

                        try:

                                Decimal(value)

                        except InvalidOperation:

                                flag = 1
```

```
                    break


            if flag == 0:
                nrow = NumericalTransform(row)
            else:
                nrow = CategoricalTransform(row)
            normalize_matrix.append(nrow)


dim = len(normalize_matrix[0])
matrix = [ [row[i] for row in normalize_matrix] for i in range(dim)]
del(normalize_matrix)
normalize_matrix = []


for i in range(len(matrix)):
    tmp = []
    for value in matrix[i]:
        if isinstance(value, list):
            tmp.extend(value)
```

```
                else:
                    tmp.append(value)
            normalize_matrix.append(tmp)
        del(matrix)
    print("Normalización finalizada.")
    return normalize_matrix
```

Anexo 2: Algoritmo Subtractive Clustering "clustering.py"

```
from decimal import *

getcontext().prec = 8

class SubCluster:

    def __init__(self, dataset):

        self.dataset = dataset

        self.density_matrix = [ Decimal('0') for row in dataset ]

        self.centroids = []

        self.data_centroids = []

        self.clusters = []

        self.clusters_index = []

        self.accept_ratio = Decimal('0.25')

        self.reject_ratio = Decimal('0.12')

        self.ra = Decimal('0.8')

        self.rb = Decimal('1.5')
```

```
    def EuclideanDistance(self, vector1, vector2):


            total = Decimal(0)

            context = Context()

            parcial = Decimal(0)


            for i in range(len(vector1)):

                    parcial = Decimal(vector1[i] - vector2[i]).normalize()

                    total += context.power(parcial, 2).normalize()


            total = context.sqrt(total)

            return total


## Calculo de densidades y primer centroide

    def FirstCenter(self):


            print(" -> Busqueda de primer centroide.")

            for i in range(len(self.dataset)):

                    for j in range(len(self.dataset)):
```

```
                    if i != j:
                        context = Context()
                        density =
context.power(self.EuclideanDistance(self.dataset[i], self.dataset[j]), 2).normalize()
                        density *= Decimal('-4')
                        density /= context.power(self.ra, 2)
                        density = density.exp()
                        self.density_matrix[i] += density

            max = Decimal('0')
            count = 0

            for i in range(len(self.density_matrix)):
                if max.compare(self.density_matrix[i]) == Decimal('-1'):
                    max = self.density_matrix[i]
                    count = i

            self.centroids.append(count)
            self.data_centroids.append(self.dataset[count])
            return
```

```
## Calculo de densidades y Diferenciacion de siguientes clusters
    def NextCenters(self):

        print(" -> Busqueda de siguiente centroide.")
        lcentroid = self.dataset[self.centroids[-1]]
        ldensity = self.density_matrix[self.centroids[-1]]

        for i in range(len(self.density_matrix)):
            if i not in self.centroids:
                context = Context()
                density =
context.power(self.EuclideanDistance(self.dataset[i], lcentroid), 2).normalize()
                density *= Decimal('-4')
                nra = self.ra * self.rb
                density /= context.power(nra, 2)
                density = density.exp() * ldensity
                self.density_matrix[i] -= density
```

```
max = Decimal('0')

count = 0


for i in range(len(self.density_matrix)):

        if i not in self.centroids:

                if max.compare(self.density_matrix[i]) == Decimal('-1'):

                        max = self.density_matrix[i]

                        count = i


upper_limit = self.accept_ratio * self.density_matrix[self.centroids[0]]

lower_limit = self.reject_ratio * self.density_matrix[self.centroids[0]]


if max.compare(upper_limit) == Decimal('1'):

        self.centroids.append(count)

        self.data_centroids.append(self.dataset[count])

        self.NextCenters()


elif max.compare(lower_limit) == Decimal('-1'):

        print(" -> Maxima cantidad de centroides hallados.")

        self.IdentifyClusters()
```

```
                return

            else:

                min_density = []

                for i in range(len(self.centroids)):

    min_density.append(self.EuclideanDistance(self.dataset[count], self.dataset[i]))

                min_density.sort()

                value1 = min_density[0] / self.ra

                value2 = self.density_matrix[count] / self.density_matrix[self.centroids[0]]

                if (value1 + value2) >= 1:

                    self.centroids.append(count)

                    self.data_centroids.append(self.dataset[count])

                else:

                    self.density_matrix[count] = Decimal('0')
```

```
                self.NextCenters()


        ## Asignar los puntos de datos al cluster más cercano

            def IdentifyClusters(self):


                print("Iniciando diferenciacion de data points.")

                distances = []


                for i in range(len(self.dataset)):

                    row = []

                    for j in range(len(self.data_centroids)):

                        distance = self.EuclideanDistance(self.dataset[i],
self.data_centroids[j]).normalize()

                        row.append(distance)

                    distances.append(row)


                for i in range(len(self.data_centroids)):

                    row, irow = [], []

                    row.append(self.data_centroids[i])

                    irow.append(self.centroids[i])
```

```
                self.clusters.append(row)

                self.clusters_index.append(irow)



        for i in range(len(distances)):


                if i not in self.centroids:

                        index = 0

                        min = distances[i][0]


                        for j in range(1, len(distances[i])):

                                if min.compare(distances[i][j]) == Decimal('1'):

                                        index = j

                                        min = distances[i][j]


                        self.clusters[index].append(self.dataset[i])

                        self.clusters_index[index].append(i)


        print("Diferenciacion de data points finalizada.")

        return
```

```
def Subtract(self):

        print("Iniciando busqueda de centroides.")

        self.FirstCenter()

        self.NextCenters()
```

Anexo 3: Algoritmo Particle Swarm Optimization "optimizer.py".

```
#!/usr/bin/env python3


from common import *

from decimal import *

from random import uniform


getcontext().prec = 8


## Acceleration coefficient at interval [0,2]

c1 = Decimal('1.5')

c2 = Decimal('1.5')


class Particle:


    def __init__(self, dimension, data, iclass):


            self.data = data

            self.iclass = iclass
```

```
        self.fvalue = Decimal('0')
        self.velocity = [ Decimal('0') for j in range(dimension) ]
        self.position = []
        self.LBest = []

        for i in range(dimension):
            self.position.append(Decimal(uniform(0,2)).normalize())
            self.LBest.append(self.position[i])

    def UpdatePosition(self):

        for i in range(len(self.position)):
            self.position[i] += self.velocity[i]
        return

    def UpdateVelocity(self, weight, GBest):

        r1, r2 = Decimal(uniform(0,1)).normalize(),
Decimal(uniform(0,1)).normalize()
        for i in range(len(self.velocity)):
```

```
                    inertia = weight * self.velocity[i]

                    cognitive = c1 * r1 * (self.LBest[i] - self.position[i])

                    social = c2 * r2 * (GBest[i] - self.position[i])

                    self.velocity[i] = inertia + cognitive + social


class SwarmOptimizer:


    def __init__(self):

            self.swarm = []

            self.centroids = []

            self.solution = []

            self.upper_bound = Decimal('2')

            self.lower_bound = Decimal('0')

            self.GBest = 0


    def Initialize(self, data):


            for i in range(len(data)):

                    row = []

                    for j in range(len(data[i])):
```

```
                    dimension = len(data[i][j])
                    particle = Particle(dimension, data[i][j], i)
                    row.append(particle)
                self.swarm.append(row)

            for i in range(len(self.swarm)):
                self.centroids.append(self.swarm[i][0])

            return

        def Fitness(self, vector):

            total = Decimal('0')
            parcial = Decimal('0')

            for i in range(len(self.centroids)):
                for j in range(len(self.swarm[i])):
                    parcial += EuclideanDistance(self.centroids[i].data,
self.swarm[i][j].data).normalize()
                parcial /= Decimal(len(self.swarm[i]))
```

```
                total += parcial

            total /= Decimal(len(self.centroids))


            return total


def Optimize(self, num_iter = 100):


            ## Inertia weight coefficient at interval [0.8,1.2]

            ## Lower values speed up convergence

            ## High values improve search space

            max_weight = Decimal('0.9')


            for i in range(num_iter):

                ## Calculate next inertial weight

                GBest = self.swarm[0].position

                weight = max_weight * Decimal(-i).exp()


                for j in range(len(self.swarm)):

                    LocalFit = self.swarm[j].Fitness

                    GlobalFit = GBest.Fitness
```

```
                    ## Evaluate local best
                    if self.swarm[j].Fitness <= LocalFit:
                        LocalFit = self.swarm[j].Fitness
                        self.swarm[j].LBest = self.swarm[j].position

                    ## Evaluate global best
                    if self.swarm[j].Fitness <= GlobalFit:
                        GlobalFit = self.swarm[j].Fitness
                        GBest = self.swarm[j].position

                    if self.constraint() == True:
                        return self.solution

                    self.swarm[j].UpdateVelocity(weight, GBest)
                    self.swarm[j].UpdatePosition()
                self.solution.append(GBest)
            return self.solution
```

Anexo 4: Módulo de mediciones "measures.py"

```
#!/usr/bin/env python3


from common import *

from decimal import *


getcontext().prec = 8



def ErrorQuantify(clusters):


    error = Decimal('0')


    for i in range(len(clusters)):


            parcial = Decimal('0')

            for j in range(1, len(clusters[i])):

                    distance = EuclideanDistance(clusters[i][0],
clusters[i][j]).normalize()
```

```
                parcial += distance


                fraction = len(clusters[i])
                parcial /= fraction
                error += parcial


    error /= Decimal(len(clusters))
    return error


def IntraCluster(clusters):


    intra = Decimal('0')


    for i in range(len(clusters)):


                parcial = Decimal('0')
                for j in range(len(clusters[i])):
                        for k in range(len(clusters[i])):
                                if j != k:
```

```
                                        distance = EuclideanDistance(clusters[i][j],
clusters[i][k]).normalize()
                                        parcial += distance

                fraction = len(clusters[i])
                parcial /= fraction
                intra += parcial

        intra /= Decimal(len(clusters))
        return intra


    def InterCluster(clusters):

        inter = Decimal('0')

        for i in range(len(clusters)):

                parcial = Decimal('0')
                for j in range(len(clusters)):
```

```
                    if i != j:
                        distance = EuclideanDistance(clusters[i][0],
clusters[j][0]).normalize()
                        parcial += distance
                inter += parcial

        fraction = 0
        for k in range(1, len(clusters)):
            fraction += k

        inter /= Decimal(fraction)
        return inter

    def ClusterDistances(data_centroids):

        dist_list = list()

        for i in range(len(data_centroids)):
            row = list()
            for j in range(len(data_centroids)):
```

```
                    if i != j:
                        distance = EuclideanDistance(data_centroids[i],
data_centroids[j]).normalize()
                        row.append(distance)
                    if i == j:
                        row.append(Decimal(0))

            count = -1
            min = 99999
            for k in range(len(row)):
                if row[k].compare(Decimal('0')) != Decimal('0'):
                    if min > row[k]:
                        count = k
                        min = row[k]

            dist_list.append(count)
        return dist_list

    def ElemDistance(elem, cluster):
```

```
    total = Decimal('0')


    for i in range(len(cluster)):
            distance = EuclideanDistance(elem, cluster[i]).normalize()
            total += distance


    total /= Decimal(len(cluster))
    return total


def SilhouetteCoefficient(clusters, near_cluster):


    coefficient = Decimal('0')
    c_list = list()


    for i in range(len(clusters)):
            a_x = Decimal('0')
            b_x = Decimal('0')
            row = list()
            for j in range(len(clusters[i])):
                    s_i = Decimal('0')
```

```
                elem = clusters[i][j]

                a_x = ElemDistance(elem, clusters[i])

                b_x = ElemDistance(elem, clusters[near_cluster[i]])

                if a_x.compare(b_x) == Decimal('-1'):

                        s_i = a_x / b_x

                        s_i = Decimal('1') - s_i

                elif a_x.compare(b_x) == Decimal('0'):

                        s_i = Decimal('0')

                elif a_x.compare(b_x) == Decimal('1'):

                        s_i = b_x / a_x

                        s_i = s_i - Decimal('1')

                row.append(s_i.normalize())

        c_list.append(row)


sum = Decimal('0')

count = Decimal('0')


for i in range(len(c_list)):

        count += len(c_list[i])

        for j in range(len(c_list[i])):
```

```
                sum += c_list[i][j]

    coefficient = sum / count

    return coefficient
```