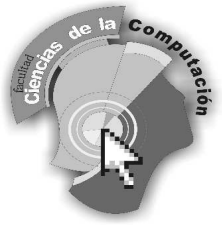

Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

Sistema de Minería de Datos Basado en Grafos con Soporte para Etiquetas Numéricas Continuas

TESIS

Que presenta

MIGUEL ÁNGEL ÁLVAREZ CARMONA

Para obtener el grado de

LICENCIADO EN INGENIERÍA EN CIENCIAS DE LA COMPUTACIÓN

Asesores

Dr. Ivan Olmos Pineda - BUAP
Dr. Jesús González Bernal - INAOE

Puebla, Puebla, México, Otoño 2012

Dedicado a
mi madre y mi abuelo

Agradecimientos

A mi madre, porque sin ella yo no sería nada. Mamá, lo logramos.

A mis asesores, el Dr. Ivan Olmos Pineda y el Dr. Jesús Antonio González Bernal, por aceptarme como asesorado en este trabajo, por su apoyo recibido y el conocimiento transmitido de manera incondicional en la realización de mi tesis.

A los profesores que han marcado mi vida académica hasta el día de hoy, al profesor Jehú Martínez Espinoza, al M.C. Miguel Rodríguez Hernández, al Dr. César Bautista Ramos y al Dr. Manuel Martín Ortíz.

Al laboratorio de software libre/CEDEP de la Facultad de Ciencias de la Computación de la BUAP. Al profesor encargado, el Dr. Mario Rossainz López, así como a Rodrigo Alberto Cuevas Vede, alumno responsble de este espacio. Y por supuesto a cada uno de sus integrantes por permitirme aprender tantas cosas a su lado. Este trabajo nunca se hubiese podido realizar sin cada uno de ellos.

A mis amigos, que son sin duda pilares de vida, a Alan, Alfredo, Gibran, Gustavo, Julio y sus familias. El destino decidió que nuestras vidas se interceptaran y nosotros decidimos crear una hermandad, que espero dure mucho tiempo.

A cada persona que se interese en las ciencias y se tome el tiempo de leer este trabajo.

Miguel Ángel Álvarez Carmona

Índice general

Índice de figuras

Índice de cuadros

Resumen

En muchas áreas del conocimiento es cada vez más común almacenar datos que poseen inherentemente dominios estructurados. Estos tipos de datos pueden ser representados con grafos, los cuales pueden, de forma natural, representar entidades, sus atributos y su relación con otras entidades. Así fue como se inició con las investigaciones para generar tecnologías que manejaran información con grafos.

Al conjunto de técnicas utilizadas para extraer información de una colección de grafos se le conoce como minería de datos basada en grafos.

Una manera de realizar minería de datos basada en grafos es buscando los subgrafos frecuentes. La minería de subgrafos frecuentes es un problema bastante estudiado en el área de minería de grafos. Esta técnica consiste en retornar todos los subgrafos que tengan una frecuencia considerable.

Una de las problemáticas en la actualidad surge en el momento de manejar datos que por naturaleza contienen características continuas como atributos que manejen distancias, grados, temperatura, peso, altura, entre otras características. Estos atributos ocasionan que la mayoría de los algoritmos existentes para minería de datos sean incapaces de establecer una relación entre datos muy cercanos entre sí, a menos que sean estrictamente iguales.

En este trabajo de tesis se propone un sistema de minería de grafos con la capacidad de detectar los atributos con características continuas y agrupar cada uno de los elementos de estos atributos en rangos, de tal forma que dicho sistema tenga la propiedad de soportar etiquetas numéricas continuas.

Para resolver este problema el sistema propuesto se dividió en 3 módulos pincipales. En la primera parte se atacó el problema de la discretización de datos. Se discriminó entre un conjunto de técnicas para llevar acabo la elección de las mejores formas de discretizar conjuntos para la minería de datos.

En la segunda parte se utilizó la minería de datos basada en grafos tradicional.

En la tercera parte de este sistema se organizó la información para que el usuario sea capaz de interpretar los nuevos resultados de manera fácil y eficiente.

Capítulo 1

Introducción

En la actualidad se ha vuelto imprescindible el manejo de grandes volúmenes de información. Debido a los rápidos avances en diversas áreas del conocimiento es muy notable el crecimiento de la creación y el almacenamiento de datos. Esto ha generado la necesidad de desarrollar nuevas técnicas capaces de procesar estos grandes volúmenes de datos y transformarlos en conocimiento para toma de decisiones.

Al principio, las bases de datos eran simples tablas, lo cual generaba bastantes dificultades para las solicitudes en las búsquedas y otras operaciones sobre dichos conjuntos. Debido a los pocos recursos computacionales con los que se contaba en la época eran una buena opción, pero evidentemente no podrían ser suficiente por mucho tiempo.

Después se contó con la teoría relacional introducida por Edgar Frank Codd [6] que definió el modelo relacional y publicó una serie de reglas para la evaluación de administradores de sistemas de datos relacionales y así nacieron las bases de datos relacionales. Si combinamos esta evolución teórica con los avances de las tecnologías de procesamiento tenemos como resultado un manejo eficiente de datos, pero como todo, siempre llega el momento en que la tecnología se ve rebasada y es necesario continuar con la evolución de las bases de datos.

En los últimos años se ha experimentado con diferentes propuestas para desarrollar nuevas técnicas que permiten la representación de datos a través de grafos etiquetados capaces de trabajar con diferentes dominios estructurados como estructuras proteínicas, cadenas de ADN, estructuras químicas, reportes de accidentes de transporte aéreos, análisis de códigos fuente, topologías de redes de computadoras, verificación toxicológica predictiva, mapas [27], entre otros, por lo que los grafos han resultando ser una estructura bastante noble con el manejo de datos y relaciones.

El conjunto de las características de los datos que son analizados bajo estas técnicas puede ser agrupado en 2 categorías: **discretas** y **continuas**. En las características discretas podemos contemplar los conjuntos con un número finito de elementos como valores de tipo alfanumérico, valores booleanos o conjuntos numéricos que representen categorías. Algunos ejemplos de características

discretas pueden ser nombre, sexo, domicilio, etc. En cambio, las características continuas contienen datos cuantitativos que representan mediciones como altura, distancia, velocidad, etc.

Existen algoritmos que trabajan con representaciones basadas en grafos etiquetados como, AGM [18], FSG [20, 21], gSpan [32], gRed [15], creados con la intención de atacar el problema de minería de datos y otros que se enfocan en la detección de patrones frecuentes y de encontrar subgrafos isomorfos como SI-COBRA[22]. El problema con estos algoritmos es que todos dan por hecho que las etiquetas tanto de aristas como de vértices de cualquier grafo son de tipo discreto, lo cual no forzosamente es cierto puesto que en el mundo real existen numerosos dominios estructurados con características discretas y continuas, esto ocasiona que sin importar que un valor sea continuo se le va a procesar como si fuera un valor discreto. Esto provoca que los algoritmos sean incapaces de generar relaciones entre las estructuras que contengan datos numéricos muy similares a menos que sean estrictamente iguales.

Por ejemplo, suponga que está trabajando con un dominio que contiene solo características continuas como peso, altura y distancia, representado en la figura 1.1 en la cual podemos observar que existen estructuras muy similares a la estructura que se desea buscar. Si consideramos todos los valores como un conjunto de etiquetas discretas es claro ver que las estructuras coinciden, en caso contrario no se encontrará ningún patrón.

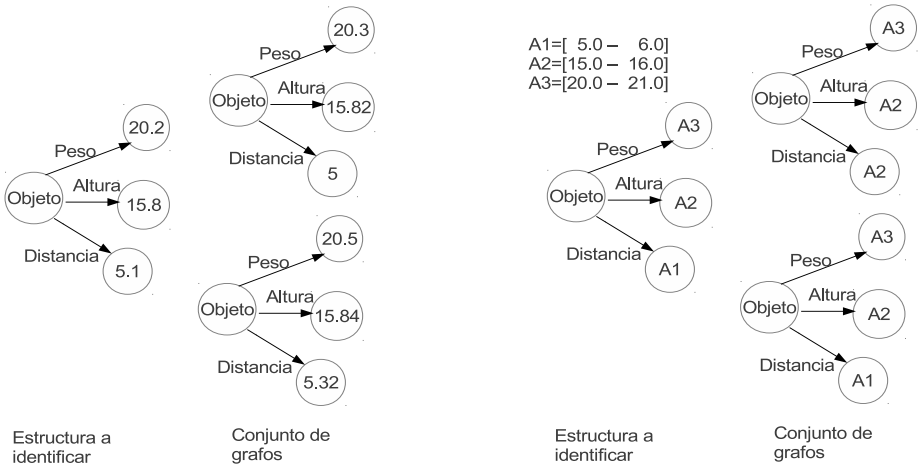

Figura 1.1: Ejemplo de un proceso de discretización aplicado a grafos.

En nuestro enfoque, las etiquetas de las aristas representan una característica o un tipo de relación entre un par de vértices, por lo que asumiremos que las etiquetas de las aristas tendrán asociados valores discretos. Mientras que la información contenida en las etiquetas asociadas a los vértices podrán ser tanto discretas como continuas, pues representan cada uno de los valores de las

características o relaciones que pueden tomar.

En esta propuesta de tesis se aborda el problema de minería de datos utilizando grafos etiquetados con soporte para dominios estructurados que contengan características continuas (valores numéricos continuos).

1.1. Objetivos

1.1.1. Objetivo General

Diseñar un algoritmo y desarrollar un sistema para minería de datos con grafos etiquetados que cuente con soporte para trabajar con dominios estructurados que por su naturaleza contengan datos tanto discretos como continuos. Este soporte medirá a través de un proceso de discretización aplicado a los campos con datos continuos.

El resultado será un conjunto finito de etiquetas sobre el cual trabajará el algoritmo de minería de datos para la búsqueda de patrones comunes de tal modo que se implemente un sistema capaz de procesar dominios estructurados de cualquier tipo y arroje mejores resultados que otros algoritmos existentes.

1.1.2. Objetivos particulares

1. Seleccionar una o varias técnicas de discretización para desarrollar una estrategia de soporte basada en la adaptación o combinación de la o las técnicas seleccionadas.

2. Hacer un cuadro comparativo de las diferentes formas de crear las codificaciones de un grafo con el propósito de mostrar las ventajas y desventajas para elegir la mejor manera o combinación de técnicas para manejar estas estructuras y así aprovecharlas lo mejor posible en tiempo y espacio.

3. Diseñar un algoritmo para minería de datos basado en gafos con soporte para etiquetas numéricas continuas.

4. Realizar un análisis del sistema implementado con diversos dominios estructurados para verificar la calidad de los resultados arrojados.

1.2. Estado del arte

Este trabajo tiene dos partes fundamentales. Primero se hablará de algoritmos de minería de datos que utilizan una representación basada en grafos etiquetados. La segunda etapa consiste en nombrar diferentes métodos para discretizar un conjunto con elementos continuos.

1.2.1. Algoritmos basados en grafos etiquetados

La minería de subgrafos frecuentes consiste en encontrar todos los grafos que ocurren frecuentemente en una colección de grafos. A estas colecciones se les llama bases de grafos o conjuntos de grafos[1].

En los últimos años se han desarrollado métodos especializados en minería de subgrafos frecuentes sobre colecciones de grafos. Algunos ejemplos son:

AGM (**A**priori based **G**raph **M**ining) [18] es el primer algoritmo de minería de subgrafos frecuentes. Se diseñó para buscar los subgrafos conexos frecuentes sin tener en cuenta la conexidad y basándose en la generación de candidatos utilizando una matriz de adyacencia llamada CAM [2] para la representación de grafos. En CAM se ordenan los vértices del grafo en un orden lexicográfico, lo cual establece un control para la generación de candidatos. Esta representación se utiliza para la generación de nuevos candidatos por medio de la fusión de CAM de grafos comunes. Cada uno de estos grafos candidatos tiene que ser procesado para saber si es soportado[3] o no. Este proceso es una de las desventajas de este algoritmo ya que genera un gran número de candidatos por lo que en tiempo de ejecución puede llegar a ser muy costoso.

FSG (**F**requent **S**ub**G**raph discovery) [20, 21] surge en respuesta a los inconvenientes con los que cuenta AGM puesto que este algoritmo puede acelerar el conteo de frecuencia de los subgrafos candidatos utilizando recursos de memoria para mantener listas de identificadores.

El algoritmo comienza enumerando todos los subgrafos simples (de una o dos aristas) que cumplan con la restricción de soporte, descartando así aquellos grafos que no sean soportados. Después a los grafos que se mantienen como candidatos se les va agregando una arista a la vez y de forma iterativa el proceso se repite hasta que ya no sea posible generar ningún grafo candidato.

En FSG los grafos se representan a través de etiquetas canónicas, las cuales son códigos que se generan de manera única para un grafo, de esta forma es posible ordenar de los vértices de acuerdo al código generado y al grado de los vértices. Esto significa que si dos o más grafos tiene la misma etiqueta canónica se dice que son isomorfos.

gSpan (**g**raph-based **S**ubstructure pattern mining) [32] es el primer algoritmo basado en crecimiento de patrones. A diferencia de los anteriores algoritmos, gSpan no realiza generación de candidatos, lo cual es una de las fortalezas de este enfoque ya que la generación de candidatos representa un costo muy alto en tiempo y memoria.

Este algoritmo usa una representación basada en una lista de códigos donde cada arista es asociada con un código formado por la combinación de su etiqueta

[1] En textos en inglés se pueden encontrar como graph databases o graph datasets
[2] Code of Adjancency Matrix
[3] Para este contexto el termino *soportado* se refiere al número mínimo de repeticiones que debe presentar un grafo candidato para ser considerado como frecuente

y las etiquetas de sus vértices adyacentes. gSpan emplea una estrategia de exploración en DFS (Depth First Search) para generar la lista de códigos y posteriormente realiza un ordenamiento lexicográfico para evitar que un mismo grafo sea representado por dos diferentes códigos. De esta forma se construye un árbol de búsqueda en profundidad para encontrar los grafos comunes.

SUBDUE [8, 27] es un algoritmo de minería de datos basada en grafos que descubre subestructuras interesantes en datos estructurales. Su nombre es un acrónimo de la palabra inglesa Substructure Discovery (descubrimiento de subestructuras).

Este algoritmo no sólo se enfoca en la búsqueda de subestructuras frecuentes en conjuntos de datos, sino que también realiza la búsqueda de las subestructuras que mejor comprimen[4] al grafo que se analiza utilizando el principio de longitud de descripción mínima (MDL) para evaluar las subestructuras interesantes.

Sin embargo, se puede pasar por alto algunos patrones significativos ya que sus estrategias de búsqueda son voraces (greedys) aunque la ventaja de usar esta estrategia es la reducción en tiempo.

FFSM (**F**ast **F**requent **S**ubgraph **M**ining) [17] es otro ejemplo de algoritmos desarrollados bajo el paradigma de crecimiento de patrones al igual que gSpan. Este algoritmo mantiene en memoria estructuras de correspondencia que permite realizar un conteo eficiente de la frecuencia de los candidatos.

MoFa (mining **Mo**lecular **Fra**gments) [4] es un algoritmo de minería de datos basado en grafos especializado en estructuras moleculares. Este enfoque utiliza la codificación BFS (Breadth First Search) para representar a los candidatos.

MoFa brinda muchas funcionalidades para su aplicación en conjuntos de datos moleculares, sin embargo ha mostrado demoras en los tiempo de respuesta en estudios comparativos realizados previamente.

1.2.2. Discretización

El principal objetivo de usar uno o varios métodos de discretización es crear a partir de un conjunto de datos un nuevo conjunto más pequeño de rangos a los que se les conoce como bins, en los cuales se agrupan los valores numéricos, procurando que exista similitud entre todos los elementos en un intervalo.

Así podemos dividir el problema de discretizar un conjunto numérico en 2 partes: *A)* Número de intervalos que deben generarse a los cuales llamaremos bins o clusters y *B)* Los límites para cada intervalo que se genere.

Existen diversas maneras de clasificar los métodos de discretización [11], por ejemplo:

[4] Con el termino *comprimir* se esta haciendo referencia a que el algoritmo cuando encuentra un subgrafo frecuente, para la siguiente evaluación toma toda la subestructura como si fuera un nodo en el grafo, de esta forma disminuye considerablemente el tamaño de dicho grafo

- Supervisados. Utilizan información de las características de los elementos (atributo clasificador).
- No supervisados. No necesitan obtener ningún tipo de información de las características.
- Estáticos. El proceso de discretización se hace de manera independiente para cada una de las características.
- Dinámicos. Realizan la discretización considerando todas las características simultáneamente.
- Splitting ó top-down. Crean una o varias particiones iniciales y luego de forma recursiva se divide cada partición.
- Mergin o bottom-up. Inicialmente cada uno de los valores en el rango se toma como una partición independiente a las demás, las cuales se van fusionando de acuerdo a sus similitudes para formar parte de un solo intervalo.

Selección del número de intervalos

Uno de los puntos fundamentales para poder llevar a cabo el proceso de discretización es la elección del número de intervalos (que de ahora en adelante a ese número lo representaremos con la letra K) que deben generarse para un conjunto numérico en general.

Aunque la elección de K es muy importante para el proceso de discretización, aún no existe una estrategia definitiva para determinar el K óptimo, por lo que se han propuesto diversos enfoques. Estos enfoques se listarán a continuación tomando en cuenta que C es el conjunto con los valores numéricos continuos, n representa la cardinalidad del conjunto C y C^{Max} y C^{Min} representan el valor máximo y mínimo en C respectivamente.

- **Ten bins** [11, 33]. Este enfoque es el más simple de todos, consiste en asignar 10 a K sin importar en número de elementos en C y sin tomar en cuenta a C^{Max} y a C^{Min}. Aunque es un enfoque muy simple varios autores han realizado un número importante de experimentos con esta idea.
- **Dougherty** [5]. Consiste en determinar el número de intervalos como: $K = max(1, \lfloor 2\log(n) \rfloor)$, esto es, K se determina a partir de la función piso del doble del logaritmo de n considerando un solo intervalo si el logaritmo es menor que 1.
- **Sturges** [26]. Al igual que el enfoque anterior, Sturges utiliza el logaritmo para establecer el valor de K, así $K = \lfloor 1 + \log_2(n) \rfloor$.
- **Scott** [9]. Para este enfoque primero se debe calcular el ancho w que será utilizando para todos los intervalos. Este valor se calcula como:

$$w = \frac{3{,}5\sigma}{n^{1/3}}$$

Donde σ se refiere a la desviación estándar del conjunto C. Por último K se define como $K = \lfloor \frac{C^{Max} - C^{Min}}{w} \rfloor$.

- **Freedman-Diaconis** [14]. En este enfoque primero se calcula el ancho para cada uno de los intervalos y posteriormente se determina el número de intervalos de acuerdo al ancho calculado. Para este enfoque el ancho se calcula con la siguiente fórmula:

$$w = 2\frac{IQR(x)}{n^{1/3}}.$$

El valor $IQR(x)$ es conocido como el rango intercuartil de un conjunto de datos y una medida de dispersión estadística que se calcula de la siguiente manera: se ordenan los datos de manera ascendente y se identifica al primer cuartil (Q1) y al tercer cuartil (Q3) que representa al 25 % y 75 % respectivamente.

Así $IQR(x) = Q3 - Q1$. Finalmente K se calcula de la siguiente forma: $K = \lfloor \frac{C^{Max} - C^{Min}}{w} \rfloor$.

- **Birgé** [2]. Esta propuesta consiste en explorar todos los posibles valores para K, entre 1 y $\frac{n}{\log(n)}$, seleccionando el valor de K entero que maximice la función $L_n(K)$–$pen(K)$, donde $L_n(k)$ es conocido como el *loglikelihood* del histograma con K intervalos y está definido como:

$$L_n(K) = \sum_{i=1}^{K} n_i \log\left(\frac{n_i K}{n}\right)$$

Donde n_i = número de datos que caen en el intervalo i y $pen(K) = K - 1 + \log(K)$[2,5] es la penalización.

- **Cross-validated** log-likelihood [10]. La técnica de validación cruzada es ampliamente utilizada en Machine Learning para evaluar que tan bien se ajustan los modelos de clasificación, regresión o clusterización a las distribuciones reales [25].

La validación cruzada consiste en repartir el conjunto de datos en n folds[5] del mismo modo de manera aleatoria y entrenar n veces el modelo a validar utilizando $n - 1$ folds para entrenamiento y el fold restante para evaluación. El criterio de evaluación se obtiene del promedio de las repeticiones realizadas. Comúnmente se utilizan 10 repeticiones (10-fold cross-validation) para la evaluación del modelo propuesto. Por otro lado, la medida *likelihood* es ampliamente utilizada para evaluar estimadores de densidad y comúnmente se utiliza el logaritmo de esta medida. Por lo tanto, la fórmula de Cross-validate log-likelihood se define como:

$$log - likelihood = \sum_{j=1}^{n} n_{j-test} \log\left(\frac{n_{j-train}}{wN}\right)$$

Donde:

[5]Un $fold$ es un conjunto que se particiona en K subconjuntos donde un subconjunto es entrenado con los demás.

- n_{j-test} = número de instancias de evaluación que caen en el bin j.
- $n_{j-train}$ = número de instancias de entrenamiento que caen en el bin j.
- N = número total de instancias.
- w = ancho del bin.

- **K-proportional** [34]. Esta técnica es de las más simples, K se calcula a partir de la función piso de la raíz cuadrada de n, es decir, $K = \lfloor\sqrt{n}\rfloor$.

Métodos de discretización no supervisados

Equal-width[23]. Este método como su nombre lo indica consiste en crear K intervalos de la misma longitud, donde K está definido por algunas de las estrategias antes mencionadas. La longitud w de cada intervalo se calcula dividiendo la diferencia de C^{Max} y C^{Min} entre K.

Este método podría funcionar muy bien en conjuntos cuya distribución es uniforme. Por lo contrario, si la distribución de dicho conjunto no es uniforme, es posible que se generen intervalos vacíos o con número muy pequeño de elementos, también tiene el riesgo de que dos elementos muy cercanos entre sí queden en distinto intervalo.

Equal-frequency [23]. Este método consiste en dividir el número de elementos del conjunto de datos en K intervalos, donde cada intervalo contiene aproximadamente el mismo número de instancias. Esta estrategia resuelve el problema de generar intervalos vacíos, sin embargo puede generar intervalos de longitud muy grandes cuando existan datos muy dispersos.

Para determinar el número de elementos para cada intervalo se divide el total de elementos entre el número de intervalos ($\frac{n}{K}$). Al igual que el método anterior, el valor de K es calculado de manera previa.

Métodos basados en clusterización

La clusterización es el proceso de agrupar datos en conjuntos a los que por lo regular se les conoce como clusters, de tal forma que los elementos de un mismo cluster tengan una cercanía o similitud muy alta. Algunas de las medidas de similitud más utilizadas pueden ser: distancia euclidiana, Manhattan, Minkowski entre otras.

K − means es uno de los métodos de clusterización más utilizado, consiste en crear K clusters con los siguientes pasos:

1. Se seleccionan K elementos de manera aleatoria, los cuales representan el centro de cada cluster inicial.
2. Cada uno de los elementos restantes se asignan al cluster con el que tengan mayor similitud, es decir, se asignan al centroide más cercano.
3. Se calcula el nuevo centroide para cada cluster utilizando la media (means) y se reasigna cada elemento al cluster más cercano.

4. Se repite el paso 3 hasta que ningún elemento tenga que ser reasignado, o bien, el número de reasignaciones no supere un umbral preestablecido.

También es posible utilizar la moda ($K-modes$) o mediana ($K-medoids$) para seleccionar los centroides de cada cluster. Debido a que los centroides son elegidos de forma aleatoria, es posible obtener más de un esquema de agrupamiento, por lo cual es necesario realizar varios experimentos para seleccionar el esquema con mayor frecuencia.

EM (**E**xpectation **M**aximization) es otro método de clusterización muy común. Este enfoque calcula las probabilidades de que cada elemento pertenezca a un cluster y usa esa probabilidad para volver a estimar los parámetros de las probabilidades hasta que se logre una convergencia.

El cálculo de las probabilidades de los clusters forma la fase de esperanza (expectation) y el paso de calcular los valores de los parámetros de las distribuciones tratando de maximizar la verosimilitud, constituye la fase de maximización.

Es posible que la convergencia lograda sea un máximo local, por lo que es recomendable repetir el proceso varias veces.

Discretización basados en estimadores de densidad

La estimación de la densidad es el proceso de construir una aproximación de la función de densidad de un conjunto de datos dado. Existen dos tipos de funciones de densidad:

1. **Paramétricas**. Se asume que la función de densidad de los datos pertenece a algunas de las familias de distribuciones conocidas (gaussianas, cuadráticas, etc)

2. **No paramétricas**. Contrario al punto anterior no se asume una función específica para la distribución de los datos. Los histogramas son un estimador de densidad no paramétrico muy conocidos, en los cuales la densidad de cada intervalo corresponde al número de elementos contenido.

Capítulo 2

Notación y definiciones

En esta sección se mencionarán los términos y definiciones necesarias para la familiarización con este trabajo de tesis.

Al igual que el estado del arte de el capítulo anterior podemos dividir esta sección en 2 partes fundamentales. En primera instancia haremos mención de los términos que abarca la parte de los algoritmos de minería de datos con grafos etiquetados para posteriormente hablar de las notaciones y definiciones que tienen que ver con los procesos de discretización.

2.1. Notación y definiciones de grafos etiquetados

Comenzaremos con una introducción a conceptos básicos de teoría de grafos. En general, un grafo $G = (V, E)$ consiste en en un conjunto finito V de elementos llamados vértices y un conjunto finito E de elementos llamados aristas, donde una arista se asocia con un par de vértices [16]. Pero esta definición ya no es suficiente cuando se trata de un grafo etiquetado (una etiqueta es una cadena asociada a cada vértice y cadena en un grafo). Por lo que en este trabajo un grafo está definido de la siguiente manera:

Definición 1 ***Grafo:*** *Un grafo G es una 6-tupla $G = (V, E, L_V, L_E, \alpha, \beta)$ Donde.*

- $V = \{v_i : i \in [1, n]\}$*, es el conjunto finito de vértices,* $V \neq \emptyset$*.*
- $E \subseteq \{\{v_i, v_j\} \in V \times V : v_i, v_j \in V\}$
- L_V *es el conjunto de las etiquetas de los vértices.*
- L_E *es el conjunto de las etiquetas de las aristas.*
- $\alpha : V \to L_V$ *es una función que asigna etiquetas a los vértices.*
- $\beta : E \to L_E$ *es una función que asigna etiquetas a las aristas.*

Esta definición solo describe a los grafos no dirigidos. Un grafo dirigido está denotado por:

$G^{\rightarrow} = (V, E, L_V, L_E, \alpha, \beta)$ es un grafo donde las aristas están representadas por (v_i, v_j), esto es, v_i y v_j son pares ordenados de vértices.

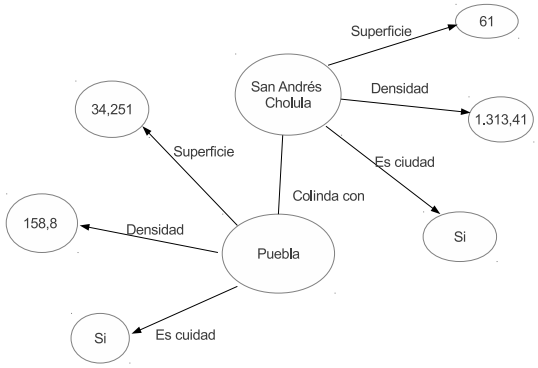

Figura 2.1: Ejemplo de un grafo etiquetado.

Como un ejemplo, la figura 2.1 muestra un grafo etiquetado $G = (V, E, L_V, L_E, \alpha, \beta)$ donde:

- $V = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$
- $E = \{\{v_1, v_2\}, \{v_1, v_3\}, \{v_1, v_4\}, \{v_1, v_5\}, \{v_5, v_6\}, \{v_5, v_7\}, \{v_5, v_8\}\}$
- $L_V = \{Puebla, San\ Andres\ Cholula, Si,$ "158,8", "34,251", "61", "1,313, 41"$\}$
- $L_E = \{Colinda\ con, Es\ ciudad, Densidad, Superficie\}$
- α
  - $\alpha(v_1) = Puebla$
  - $\alpha(v_2) = Si$
  - $\alpha(v_3) =$ 158,8
  - $\alpha(v_4) =$ 34,251
  - $\alpha(v_5) = San\ Andres\ Cholula$
  - $\alpha(v_6) = 61$
  - $\alpha(v_7) =$ 1,313, 41
  - $\alpha(v_8) = Si$
- β
  - $\beta(\{v_1, v_2\}) = Es\ ciudad$
  - $\beta(\{v_1, v_3\}) = Densidad$
  - $\beta(\{v_1, v_4\}) = Superficie$

- $\beta(\{v_1, v_5\}) = Colinda\ con$
- $\beta(\{v_5, v_6\}) = Superficie$
- $\beta(\{v_5, v_7\}) = Densidad$
- $\beta(\{v_5, v_8\}) = Es\ ciudad$

Un grafo no etiquetado se puede definir como $G = (V, E, L_V, L_E, \alpha, \beta)$, donde $|L_V| = |L_E| = 1$.

Dos vértices $v_i, v_j \in G$ son **adyacentes** o vecinos, denotado por $v_i \sim v_j$. Si $\exists\ e \in E$ donde $e = \{v_i, v_j\}$ y e es una arista incidente para v_i y v_j. Por ejemplo $v_1 \sim v_2$ en la figura 2.1 son adyacentes.

Finalmente, el **grado** de un vértice v denotado por $deg(v)$ es igual al número de vértices adyacentes al vértice v. Por ejemplo el $deg(v_1)$ en la figura 2.1 es 4 por que es adyacente con v_2, v_3, v_4 y v_5. El concepto de grafo cambia cuando se trata de grafos dirigidos, porque ahora tenemos un **grado entrante**, el cual es el número de aristas por las cuales pueden llegar directamente al vértice v y un **grado saliente**, se refiere al número de aristas que salen directamente del vértice v.

Definición 2 ***Grafo completo:*** *Un grafo G es completo si $\forall\ v_i, v_j \in V, v_i \neq v_j : v_i \sim v_j$.*

Por lo general, un grafo completo se denota por K_n, donde n es el número de vértices en el grafo. Por otro lado, si $G = (V, E, L_V, L_E, \alpha, \beta)$ es completo, entonces $\forall\ v \in V : deg(v) = n - 1$, donde $|V| = n$. Como un ejemplo, el grafo que se muestra en la figura 2.2 es un grafo completo con 4 vértices y para cada vértice v en el grafo, $deg(v) = 3$.

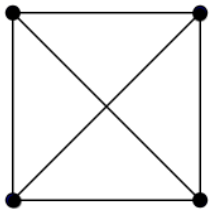

Figura 2.2: Ejemplo de un grafo completo.

Definición 3 ***Subgrafo:*** *Dado un grafo G, donde $G = (V, E, L_V, L_E, \alpha, \beta)$. Un subgrafo G^S de G, denotado por $G^S \subseteq G, G^S = (V^S, E^S, L_V^S, L_E^S, \alpha^S, \beta^S)$ es un grafo tal que $V^S \subseteq V, E^S \subseteq E, L_V^S \subseteq L_V, L_E^S \subseteq L_E, \alpha^S \subseteq \alpha$ y $\beta^S \subseteq \beta$.*

Definición 4 ***Grafo inducido:*** *Dado un grafo G^S, subgrafo de G. G^S es un grafo inducido de G si $\forall\ v_i, v_j \in V^S : \{v_i, v_j\} \in E^S \Leftrightarrow \{v_i, v_j\} \in E$.*

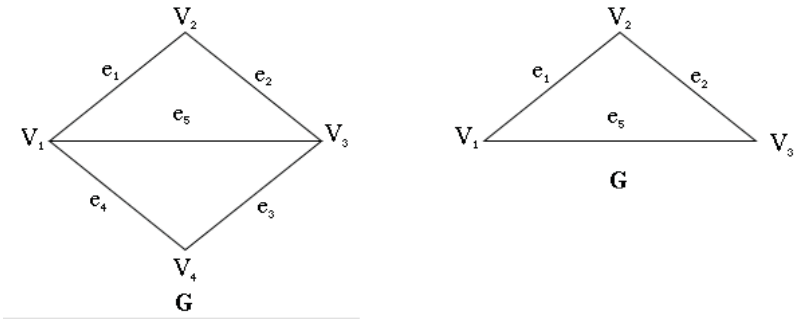

Figura 2.3: Ejemplo de un subgrafo

Definición 5 ***Isomorfismo:*** *Dados dos grafos* $G = (V, E, L_V, L_E, \alpha, \beta)$ *y* $G' = (V', E', L'_V, L'_E, \alpha', \beta')$, G' *es isomorfo a* G*, denotado por* $G' \cong G$ *si existen funciones biyectivas* $f : V' \to V$ *y* $g : E' \to E$*, donde:*

- $\forall\ v' \in V', \alpha'(v') = \alpha(f(v'))$
- $\forall\ \{v'_i, v'_j\} \in E', \beta'(\{v'_i, v'_j\}) = \beta(g(\{v'_i, v'_j\}))$

Es decir, G' y G son isomorfos si y solo si:(1) si los vértices v'_x y v'_y en G' corresponden a los vértices v_i y v_j en G respectivamente, entonces una arista en G' definida como $\{v'_x, v'_y\}$ debe corresponder a una arista $\{v_i, v_j\}$ en G y viceversa; (2) la forma de etiquetado de ambos grafos es exactamente la misma.

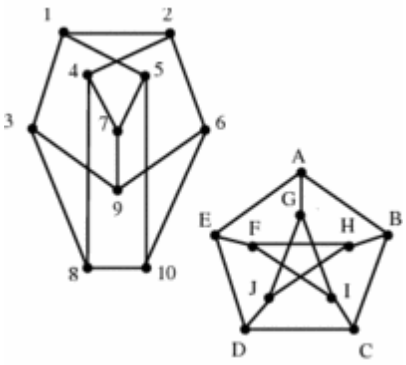

Figura 2.4: Ejemplo de grafos isomorfos

Se debe notar que si $|L'_v| = |L'_E| = 1$, entonces tenemos una definición tradicional de isomorfismo.

Definición 6 ***Subgrafo isomorfo:*** *Sean* G *y* G' *dos grafos.* G' *es subgrafo isomorfo de* G *si existe* $G^S \subseteq G$ *tal que* $G' \cong G^S$*.*

Definición 7 ***Grafo conectado:*** *Un grafo* G *es conectado o conexo si existe una secuencia de aristas distintas* $\{v_i, v_j\}, \{v_j, v_k\}, \cdots, \{v_x, v_y\}, \{v_y, v_z\}$ *para todo par de vértices* v_i *y* v_z *en* G*, donde* $v_i \neq v_z$*.*

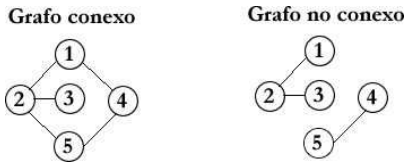

Figura 2.5: Ejemplo de grafo conexo y no conexo respectivamente

En otras palabras, del vértice v_i podemos ir a cualquier otro vértice en el grafo.

Definición 8 ***Anillo:*** *Un grafo G es un ciclo o anillo si existe una secuencia de distintos vértices* $< v_1, v_2, \ldots, v_n >, |V| = |E| = n$, *donde* $\forall\ v_i, v_{i+1} : v_i \sim v_{i+1}, v_n \sim v_1$.

Definición 9 ***Circuito:*** *un circuito en un grafo G es un subgrafo* G^S *donde* G^S *es un anillo.*

Definición 10 ***Árbol:*** *Un grafo G es un árbol si G no tiene circuitos.*

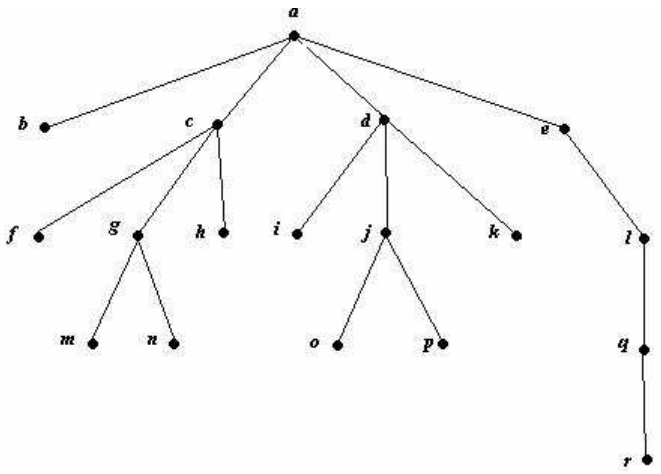

Figura 2.6: Ejemplo de un árbol

Definición 11 ***Árbol de expansión:*** *Un árbol de expansión de un grafo G es un árbol de G que tiene todos los vértices de G.*

En la figura 2.7 se puede apreciar uno de los diversos árboles de expansión conformado por las aristas: {1,2}, {2,3}, {3,6}, {6,5}, {5,4}

2.2. Notación y definiciones de discretización

Un **proceso de discretización** se encarga de crear, a partir de un conjunto de datos, un nuevo conjunto más pequeño de rangos a los que se les conoce como

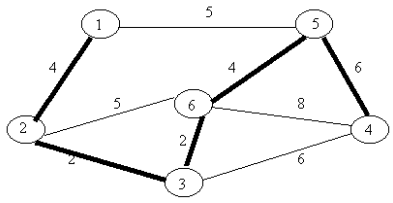

Figura 2.7: Ejemplo de árbol de expansión

bins o clusters, en los cuales se agrupan los valores numéricos, procurando que exista cierta similitud entre todos los elementos en un intervalo.

Un proceso de discretización consta de dos partes fundamentales:

1. Un número K: El cual se refiere al número de bins en que se particionará el conjunto original.
2. Límites: Podemos definirlos como las fronteras mínimas (límite inferior) y máximas (límite superior) de cada uno de los bins.

Podemos enunciar matemáticamente la descripción anterior como:

Definición 12 ***Partición:*** *Dado un conjunto C, tal que $C \subseteq \mathbb{R}$, una partición de C es la familia de subconjuntos $\{C_i : i \in I\}$ que cumple las siguientes condiciones:*

1. $C_i \neq \emptyset$, *para toda* $i \in I$
2. $\bigcup_{i \in I} C_i = C$
3. *Si* $C_i \neq C_j$ *entonces* $C_i \cap C_j = \emptyset$

Un proceso de discretización crea conjuntos mutuamente excluyentes. Es importante aclarar que la cardinalidad de un conjunto C_i no necesariamente es igual a la cardinalidad de un conjunto C_j.

Cada conjunto C_i tiene un valor máximo y mínimo, definidos de la siguiente manera:

Definición 13 ***Elemento mayor:*** *Denotaremos al mayor elemento de un cluster i como:*

$$C_i^{max} = x \in C_i : \forall y \in C_i : x \geq y$$

Definición 14 ***Elemento menor:*** *Denotaremos al menor elemento de un cluster i como:*

$$C_i^{min} = x \in C_i : \forall y \in C_i : x \leq y$$

Podemos inferir a partir de las definiciones 13 y 14 que los límites de un cluster C_i estan definidos como C_i^{min} y C_i^{max}.

Es posible definir un orden $\prec$ entre las particiones en C como

Definición 15 ***Orden:***

$$C_i \prec C_{i+1} \; si \; C_i^{max} < C_{i+1}^{min}, \forall i \in [1, K-1]$$

Por lo que el orden de todos los clusters de C es de la siguiente forma:

$$C_1 \prec C_2 \prec \cdots \prec C_k$$

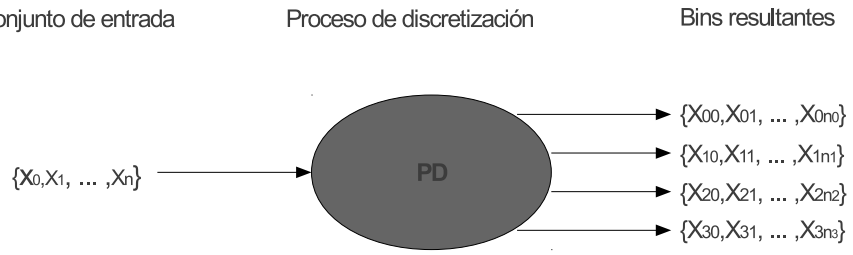

Figura 2.8: Funcionamiento de un proceso de discretización

En la *figura 2.8* se muestra el comportamiento básico de un proceso de discretización, donde la entrada es un conjunto de cardinalidad n y la salida consta de 4 bins los cuales no necesariamente tendrán la misma cardinalidad.

Una vez que se tienen bien identificados los límites de cada uno de los K clusters, se agrupan los elementos en el cluster que corresponde.

Es muy importante mencionar que esto no garantiza que el 100 % de los elementos del conjunto de entrada después de un proceso de discretización estén ubicados en alguno de los bins resultantes.

Se identifica como **Ruido** al porcentaje de los elementos que se encuentren fuera de todas las fronteras de los bins resultantes de un proceso de discretización. Es decir:

Definición 16 ***Ruido:***

$$Ruido = \{x \in C : x \notin C_i, \forall i \in I\}$$

En la figura 2.9 se propone un conjunto C que esta constituido de la siguiente manera:

$$\mathrm{C} = \{0.1, 0.15, 0.23, 0.234, 0.28, 1, 1.45, 1.456, 1.47, 1.8, 2.01, 2.02, 2.021, 3.01, 4.0\}$$

En el ejemplo se describe 3 bins:

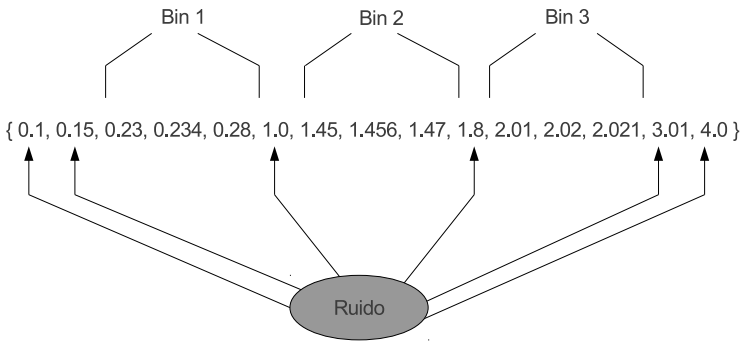

Figura 2.9: Elementos que no se encuentran en algún bin

$$C_1 = \{0.23, 0.234, 0.28\}$$
$$C_2 = \{1.45, 1.456, 1.47\}$$
$$C_3 = \{2.01, 2.02, 2.021\}$$

Se puede ver que los elementos 0.1, 0.15, 1.0, 1.8, 3.01 y 4.0 no se encuentran en alguno de los 3 bins existentes. La cardinalidad del conjunto original en este ejemplo es de 15 elementos, de los cuales 6 están fuera de todas la fronteras, por lo tanto este ejemplo tiene un 40 % de ruido.

Si después de un proceso de discretización este ruido no se asigna a ningún cluster, es claro ver que ya no se esta cumpliendo con la definición 12, por lo que se necesita por fuerza adjuntar todos los datos del ruido a los clusters que más se adapten a dichos elementos. Claro que esto implica mayor procesamiento pero es un proceso que no nos podemos dar el lujo de evitar.

Para asignar cada elemento del ruido a los bins existentes podemos intentar asignar cada elemento del ruido al bin más cercano. Esto es, tendríamos que encontrar el bin que tenga la menor distancia entre los elementos de cada bin y los elementos del ruido. Por lo que después de un proceso de reacomodamiento, los bins podrían quedar de la siguiente forma:

$$C_1 = \{0{,}1, 0{,}15, 0{,}23, 0{,}234, 0{,}28\}$$
$$C_2 = \{1{,}0, 1{,}45, 1{,}456, 1{,}47\}$$
$$C_3 = \{1{,}8, 2{,}01, 2{,}02, 2{,}021, 3{,}01, 4{,}0\}$$

Capítulo 3

Propuesta

En este capítulo se explicará como se atacó el poblema de discretización aplicada a la minería de datos. En pimer lugar se explica la estructura general de del sistema que se esta prroponiendo y el formato del arrchivo de entrada para su funcionalidad.

Despúes se profundiza en cada componente de dicho sistema. De igual manera se expresa la elección de las técnicas de discretización para ser implementadas.

3.1. Estructura general del sistema

El sistema propuesto se descompone en 3 módulos fundamentales donde cada uno se encarga de un problema específico. Los módulos son:

- Módulo de discretización
- Módulo de minería de datos
- Módulo de interpretación de datos.

En la figura 3.1 se muestra el flujo básico del sistema propuesto.

A continuación se describen con más detalle cada uno de los módulos.

1. **Discretización:** Se extraen del grafo de entrada todas las etiquetas de los vértices que sean de tipo continuo y se agrupan según sus características (por ejemplo, las etiquetas de velocidad con las de velocidad, las de altura con las de altura, etc.). Para cada uno de estos grupos se le aplica un proceso de discretización, en el cual el usuario es libre de elegir entre diferentes técnicas para discretizar.

   Este proceso retorna un nuevo conjunto de n etiquetas que representan los n clusters resultantes de dicho proceso.

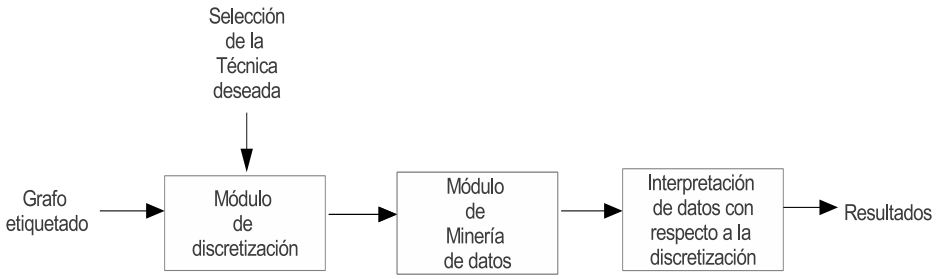

Figura 3.1: Estructura básica del comportamiento del sistema

Antes de terminar este proceso se etiquetan de nuevo todos los vértices que tenían características continuas con las nuevas etiquetas correspondientes.

2. **Minería de datos con grafos:** Una vez que el grafo ya cuente con las etiquetas generadas en el paso anterior, es posible aplicar algún algoritmo de minería de datos con grafos y de esta forma recibir las subestructuras frecuentes.

3. **Interpretación de los resultados:** Cuando se cuente con los resultados arrojados por el paso anterior se debe de volver a etiquetar para que el usuario interprete de manera correcta los resultados y no se confunda por las etiquetas con las que se trabajó de forma interna.

Antes de explicar el funcionamiento interno de cada uno de los módulos es prudente mencionar el formato del archivo de entrada con el cual se representarán los grafos a los que se les pretende aplicar minería de datos.

3.2. Formato del archivo de entrada

Un formato muy conocido de minería de datos basado en grafos etiquetados es el formato utilizado por el sistema SUBDUE [8]. SUBDUE es un sistema ampliamente utilizado alrededor del mundo en diversas áreas del conocimiento.

Con el objetivo de que el sistema propuesto tenga una mayor aceptación se planteó la idea de modificar de forma mínima el formato de SUBDUE con la intención de que los usuarios de dicho sistema puedan mapear sus archivos al formato de nuestro sistema de la forma más fácil posible.

Lo que se propone es agregar un nuevo comando al formato tradicional de SUBDUE, el cual conoceremos como "C" y el formato de este comando es :

$$C < DatoContinuo >$$

Donde $< DatoContinuo >$ es el nombre de la arista que contenga en uno de sus extremos un dato continuo.

Supongamos que se desea representar el grafo de la figura 3.2 en formato de SUBDUE. Podemos ver en la figura 3.3 el formato tradicional de SUBDUE para representar dicho grafo.

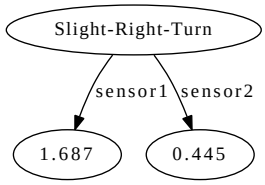

Figura 3.2: Grafo que se desea representar

```
% estrella 1
v 1 Slight-Right-Turn
v 2 1.687
v 3 0.445
e 1 2 sensor1
e 1 3 sensor2
```

Figura 3.3: Formato tradicional de SUBDUE

Para mapear del formato de SUBDUE al formato propuesto en este trabajo primero se necesita identificar las aristas que en uno de sus extremos contienen un elemento continuo. El grafo de la figura 3.2 tiene 2 aristas y las 2 tienen en uno de sus extremos un dato continuo. Por lo tanto se deben agregar 2 veces el comando "C" al formato tradicional de SUBDUE como se muestra en la figura 3.4.

En general, si en un grafo en el formato tradicional de SUBDUE existieran N etiquetas de aristas que en uno de sus extremos contengan datos continuos, se debe agregar N veces el comando "C" al principio del archivo.

Si por el contrario se desea regresar al formato de SUBDUE solo se debe de comentar los N comandos "C". Por lo que mapear de un formato a otro es muy sencillo.

```
C sensor1
C sensor2

% estrella 1
v 1 Slight-Right-Turn
v 2 1.687
v 3 0.445
e 1 2 sensor1
e 1 3 sensor2
```

Figura 3.4: Formato de SUBDUE agregando el comando C

Si el usuario desea una mayor información acerca de la estructura de los archivos de SUBDUE se recomienda consultar el apéndice A de esta tesis.

3.3. Módulo de discretización

Existen diversos algoritmos para discretizar un conjunto de datos numéricos que son capaces de calcular los límites de cada uno de los bins resultantes. Algunos de estos algoritmos también pueden calcular el número de clusters K si es que no se establece antes como uno de los parámetros, el inconveniente de este cálculo es que necesita de técnicas de aprendizaje supervisado.

El problema de utilizar aprendizaje supervisado es que no podemos suponer que para cada dominio estructurado que se va a procesar tendremos información de las características de los atributos clasificadores, lo que hace inviable esta opción. Por lo tanto, para elegir K, usaremos técnicas que sólo necesiten datos básicos como la cardinalidad del conjunto original, la desviación estándar, entre otras.

En el cuadro 3.1 podemos ver las técnicas que se acoplan a las especificaciones que acabamos de mencionar.

Cuadro 3.1: Técnicas para elegir K

Técnica	**Parámetros**
Dougherty	Cardinalidad del conjunto
Sturges	Cardinalidad del conjunto
Freedman-Diaconis	Elemento menor del conjunto Elemento mayor del conjunto Cardinalidad del conjunto
Scott	Elemento menor del conjunto Elemento mayor del conjunto Cardinalidad del conjunto Desviación estándar
K-proportional	Cardinalidad del conjunto

Para la elección del algoritmo para la selección de los límites podemos elegir entre dos de los algoritmos más usados en la actualidad para lograr un proceso

de discretización y que se acoplan a las mismas características que las técnicas previamente enumeradas. Estos algoritmos son EM y KMeans.

3.3.1. Elección de técnicas y algoritmos

Existen muchas posibles combinaciones de técnicas y algoritmos de discretización que se pueden aplicar a un mismo conjunto. Pero evidentemente se desea encontrar la mejor combinación entre las técnicas de discretización y los algoritmos para encontrar los límites de cada bin.

Existen dos posibles formas de encontrar estas combinaciones. La primera de ellas es haciendo un experimento empírico, mientras que la segunda opción es realizando un experimento teórico.

Se podría pensar que un experimento teórico es la mejor opción, el problema es que un experimento de esta índole escapa a los tiempos de esta tesis, por lo que para este trabajo se optó por hacer experimentos empíricos.

Basándonos en las técnicas ya enumeradas y suponiendo que cada combinación de los 2 algoritmos (EM, KMeans) con las 6 técnicas (Ten Bins, Dougherty, Sturges, Freedman-Diaconis, Scott, K-proportional) nos arrojan resultados distintos podemos ver que el número de combinaciones posibles como máximo es 12.

Ahora queda hacer una discriminación entre estas 12 posibilidades para elegir la combinación o combinaciones que se adapten mejor a nuestras necesidades. Para lograr este objetivo se propone un experimento empírico que consta de 2 etapas.

Primera Etapa

Esta etapa consiste en aplicar cada una de las combinaciones a 20 conjuntos de números reales, diferentes tanto en contenido como en cardinalidad. Estos conjuntos son hechos al azar y sus longitudes oscilan entre los 8 y los 322196 elementos.

Pero aún teniendo los resultados en mano de todos estos procesos de discretización, ¿cómo sabemos cual es el mejor resultado?

Podríamos basarnos en diversos objetivos, por ejemplo, podemos decir que el mejor resultado es el que tenga un menor tiempo de ejecución o el que gaste menor memoria, pero son parámetros que no nos dicen nada en cuanto al valor del contenido del resultado mismo. Es por esto que nos basaremos en 2 factores básicos:

1. El número K que arroje cada técnica.

2. El porcentaje de ruido de cada combinación.

Evidentemente se necesita que el porcentaje de ruido sea el menor posible, esto para que el proceso donde se asigna cada elemento del ruido al bin más cercano sea menos costoso.

También necesitamos minimizar tanto como sea posible el valor de K, esto para hacer que cada bien tenga una cantidad considerable de elementos y de esta forma maximizar los patrones frecuentes en el módulo de minería de datos.

Se debe entender que no podemos nombrar directamente a un proceso de discretización como el mejor si su valor de K es el menor de todos puesto que se puede dar el caso que el porcentaje de ruido sea muy elevado. También se corre el riesgo de que ocurra el caso contrario, es decir, que el porcentaje de ruido sea el mas cercano a cero pero que maneje un valor muy alto para el número de clusters, por lo que necesitamos un equilibrio entre ambos factores, lo que hace que la discriminación más más complicada.

Función de evaluación de rendimiento

En este caso se propone una función que nos permita observar numéricamente el rendimiento de cada combinación definida de la siguiente forma:

$$Rendimiento_{kij} = f_1(K_i) * \alpha + f_2(Ruido_j) * \beta$$

Donde:

- $Rendiminto_{kij}$ es el resultado de rendimiento de la combinación del algoritmo j con la técnica i sobre el conjunto k.
- α y β son factores de equilibrio de K y el ruido respectivamente. Se asume que $\alpha + \beta = 1$ y podemos elegir el peso que se le dará a cada factor, es decir que si $\alpha > \beta$ entonces tendrá más peso la minimización de K que la del ruido. A su vez si $\beta > \alpha$ podemos asegurar que se dará mayor importancia a la minimización del ruido. Por último, si $\alpha = \beta$ se asegura que se le dará la misma importancia a ambas minimizaciones. Aunque lo obvio es preferir el equilibrio se aplicaran los 3 casos a cada combinación para observar que tanto puede cambiar el valor del rendimiento para cada caso según el peso que se le de a cada factor.
- $f_1(K_i)$ es una función que asigna un valor entre 0 y 1 según su ubicación en el rango $[K_m, K_M]$ al valor resultante K de la técnica i para el conjunto k.
- K_m es el menor K para el conjunto k.
- K_M es el mayor K para el conjunto k.
- $f_2(Ruido_j)$ es una función que asigna un valor entre 0 y 1 según su ubicación en el rango $[R_m, R_M]$ al valor resultante del porcentaje de ruido de la combinación del algoritmo j y la técnica i para el conjunto k.
- R_m es el menor porcentaje de ruido para el conjunto k.

Técnica	K	f_1(K)
Dougherty	7	0
Ten Bins	10	0.073170732
Sturges	12	0.12195122
Scott	24	0.414634146
Freedman-Diaconis	44	0.902439024
K-proportional	48	1

Figura 3.5: Ejemplo de f_1

Combinación	Ruido	f_2(Ruido)
EM con Ten Bins	0.00%	0
EM con Scott	0.00%	0
EM con Sturges	0.30%	0.005681818
EM con K-proportional	0.51%	0.00974026
EM con Freedman-Diaconis	4.04%	0.07711039
Kmeans con K-proportional	17.52%	0.334415584
Kmeans con Freedman-Diaconis	17.60%	0.336038961
Kmeans con Dougherty	27.72%	0.529220779
Kmeans con Scott	28.78%	0.549512987
Kmeans con Sturges	30.99%	0.591720779
Kmeans con Ten Bins	31.25%	0.596590909
EM con Dougherty	52.38%	1

Figura 3.6: Ejemplo de f_2

- R_M es el mayor porcentaje de ruido para el conjunto k.

- $i \in \mathbb{N}$ y $1 \leq i \leq 6$
- $j \in \mathbb{N}$ y $1 \leq j \leq 2$
- $k \in \mathbb{N}$ y $1 \leq k \leq 20$

Observando los ejemplos para la asignación de f_1 y f_2 respectivamente no sería difícil calcular el rendimiento de cualquiera de las combinaciones. Por ejemplo si se desea obtener el rendimiento de la combinación del algoritmo EM con la técnica Dougherty solo se necesita encontrar en la figura 3.5 el valor de f_1 correspondiente a esta técnica el cual es de 0, para posteriormente encontrar en la figura 3.6 el valor de f_2 de la combinación de EM y Dougherty la cual es de 1. Suponiendo que $\alpha = \beta$ entonces el $Rendimiento = 0 * 0{,}5 + 1 * 0{,}5$. Es evidente entonces que el rendimiento de este algoritmo y esta técnica es de 0,5.

Necesitamos calcular la sumatoria de los rendimientos de cada combinación con todos los conjuntos, por ejemplo, para conocer el valor total del rendimiento global de la combinación Kmeans con la técnica Ten Bins necesitamos calcular su valor para cada conjunto y sumarlos. Así definimos a la mejor combinación como la pareja que tenga el menor valor en su sumatoria.

Para finalizar la primera etapa se necesita calcular la sumatoria de los rendimientos de todas las parejas entre los 2 algoritmos y las 6 técnicas con las 3 diferentes combinaciones de factores de equilibrio($\alpha = \beta$, $\alpha < \beta$, $\alpha > \beta$). En la figura 3.10 se muestra la suma de los rendimientos de todas la combinaciones

con el algoritmo EM con los factores equilibrados con $\alpha = \beta = 0{,}5$.

Equilibrado		
	Sturges con EM	**3.3086382699**
	Dougherty con EM	**3.8927917222**
	Scott con EM	**4.2022882111**
	Ten Bins con EM	4.6887989318
	Sturges con SKMeans	7.7813599326
	Dougherty con SKMeans	7.8330844567
	Ten Bins con SKMeans	8.0718590776
	Freedman-Diaconis con EM	8.7643235188
	Scott con SKMeans	9.179216895
	K-proportional con EM	9.3319925089
	Freedman-Diaconis con SKMeans	12.6413269493
	K-proportional con SKMeans	14.2367718575

Figura 3.7: Etapa uno con factores de equilibrio iguales

Cargado a K		
	Dougherty con EM	**3.1390058715**
	Sturges con EM	**3.1470278541**
	Scott con EM	**4.1243727248**
	Ten Bins con EM	4.4359834702
	Dougherty con SKMeans	6.3075665896
	Sturges con SKMeans	6.7252051842
	Ten Bins con SKMeans	7.1424315869
	Scott con SKMeans	8.11414352
	Freedman-Diaconis con EM	9.3187090576
	K-proportional con EM	10.457823342
	Freedman-Diaconis con SKMeans	12.420311802
	K-proportional con SKMeans	14.3816468208

Figura 3.8: Etapa uno con factores de equilibrio cargados a K

Las figuras 3.7 y 3.9 muestran como la mejor combinación con los factores equilibrados y cargados al ruido es la técnica Sturges con el algoritmo EM. Mientras que la figura 3.8 muestra que la mejor combinación con los factores cargados a K es la técnica Dougherty con el algoritmo EM.

Podemos observar que independientemente de los factores de equilibrio las combinaciones Sturges con EM, Scott con EM y Dougherty con EM se mantienen como las 3 mejores opciones, lo que marca una cierta continuidad que habla de lo constantes y en cierta parte confiables que pueden ser estas parejas. Se podría decir que estas 3 combinaciones pueden contemplarse como las mejores opciones. Aunque tenemos que esperar a los resultados de la segunda etapa para poder sacar mejores conclusiones.

Cargado al ruido		
	Sturges con EM	**3.4702486857**
	Scott con EM	**4.2802036975**
	Dougherty con EM	**4.646577573**
	Ten Bins con EM	4.9416143934
	K-proportional con EM	8.2061616759
	Freedman-Diaconis con EM	8.2099379799
	Sturges con SKMeans	8.8375146809
	Ten Bins con SKMeans	9.0012865684
	Dougherty con SKMeans	9.3586023237
	Scott con SKMeans	10.24429027
	Freedman-Diaconis con SKMeans	12.8623420965
	K-proportional con SKMeans	14.0918968942

Figura 3.9: Etapa uno con factores de equilibrio cargados al Ruido

Segunda Etapa

A pesar de que en este punto ya contamos con resultados interesante tenemos que hacer una segunda ronda de evaluaciones para corroborar que los datos arrojados en la etapa anterior se mantengan.

Esta etapa es parecida a la etapa anterior, pero esta vez se tomarán 20 bases de datos reales extraídas de el sitio web UCI Machine Learning Repository [1].

Las bases seleccionadas comprenden información de distintas disciplinas como información de enfermedades, agrupaciones de vocales japonesas, información de flora, etc. Lo que enriquece el experimento al tratar con resultados de la vida real. En el cuadro 3.2 se encunetran las especificaciones de cada una de estas bases de datos.

En esta ocasión la cardinalidad de los conjuntos oscila entre 103 y 164860 elementos.

El siguiente paso, una vez seleccionadas las bases de datos y con los campos de tipo real extraídos, consiste en aplicar a esta información todas las combinaciones de técnicas y algoritmos para medir el rendimiento de cada una de estas parejas con la misma función de la etapa uno.

Los resultados de la segunda etapa se pueden ver en las figuras 3.11, 3.12 y 3.13.

Los resultados de las 2 etapas no cambian sustancialmente pues aunque las combinaciones no necesariamente ocupan las mismas posiciones en ambas pruebas se logra observar que no cambian mucho entre ellas, lo que nos habla que estos resultados se mantienen a pesar de la diversidad de los conjuntos.

El comportamiento gráfico de las técnicas Dougherty y Sturges se pueden

[1] UCI Machine Learning es una colección de bases de datos, teorías de dominio y generadores de datos que son utilizados por la comunidad de aprendizaje automático para el análisis empírico de algoritmos [29]

Base	Técnica 1 con EM	Técnica 2 con EM	Técnica 3 con EM	Técnica 4 con EM	Técnica 5 con EM	Técnica 6 con EM
1	0.0819094595	0.078465652	0.0323362735	0.718120152	0.025959679	0.075796730
2	0.7777777778	0.186363636	0.1666666667	0.845454545	0.186363636	0.845454545
3	0.0365853659	0.5	0.0638165188	0.489774707	0.207317073	0.504870129
4	0.0333333333	0.019114681	0.044444444	0.532193159	0.055555556	0.420120724
5	0.0074259193	0.009685712	0.0139442231	0.484388543	0.268089868	0.652981078
6	0.2913933968	0.240933145	0.1143328044	0.248176032	0.215810105	0.5
7	0.0026823431	0.004185055	0.0080789946	0.454973963	0.237419605	0.591578568
8	0.2624948	0.222376452	0.2630041855	0.541154097	0.095492788	0.235412474
9	0.3774427694	0.211055276	0.3774427694	0.224036851	0.224036851	0.5
10	0.5	0.493055556	0.2291666667	0.5	0.5	0.493055556
11	0.1006355111	0.444664032	0.1413237232	0.164341626	0.08114392	0.5
12	0.2388160017	0.191956124	0.2855145715	0.487337346	0.417625551	0.5
13	0.2448979592	0	0.2448979592	0.612244898	0.367346938	0.362244898
14	0.3260612993	0.332116788	0.3367614482	1	0.222542397	0.256708283
15	0.109375	0.142857142	0.1183035714	0.5	0.102678571	0.118303571
16	0.75	0.147058823	0.2970588235	0.1	0.079411764	0.558823529
17	0.0418543563	0.076987799	0.0304878049	0.130788450	0.095460049	0.623895666
18	0.253968254	0.285714285	0.253968254	0.285714285	0.384920634	0.571428571
19	0.1564067487	0.306201550	0.1564067487	0.285681714	0.285681714	0.5
20	0.0957386364	0	0.1306818182	0.159943181	0.149431818	0.519318181
Suma	**4.6887989318**	**3.8927917222**	**3.3086382699**	**8.7643235188**	**4.2022882111**	**9.3319925089**
Promedio	0.2344399466	0.194639586	0.1654319135	0.438216175	0.210114410	0.466599625
Desviación Est	0.2140169202	0.157646790	0.1115984621	0.236059247	0.127704791	0.177820204

Figura 3.10: Técnicas y algoritmo EM con factores equilibrados

Cuadro 3.2: Especificaciones de las bases de datos para la segunda etapa

Nombre	Instancias	Atributos	Atributos continuos
Breast Cancer Wisconsin	569	32	30
Cloud	1024	10	10
Communities and Crime	1994	128	123
Communities and Crime Unnormalized	2215	147	115
Concrete Slump Test	103	10	9
Connectionist Bench	208	60	60
Connectionist Bench (Vowel)	528	10	10
Glass Identification	214	10	8
Hill-Valley	606	101	100
Ionosphere	351	34	34
Japanese Vowels	640	12	12
Libras Movement	360	91	90
Localization Data for Person Activity	164860	8	4
MAGIC Gamma Telescope	19020	11	10
Ozone Level Detection	2536	73	24
Parkinsons	197	23	22
Parkinsons Telemonitoring	5875	26	23
Pima Indians Diabetes	768	8	2
SECOM	1567	591	589
Spambase	4601	57	56

Equilibrado		
	Dougherty con EM	**2.5737460727**
	Ten Bins con EM	**4.4273942462**
	Sturges con EM	**4.632098906**
	Scott con EM	6.2314378343
	Dougherty con SKMeans	7.8270129688
	Sturges con SKMeans	9.3706832155
	Freedman-Diaconis con EM	9.4152795129
	Ten Bins con SKMeans	9.6961061792
	K-proportional con EM	10.8863147305
	Scott con SKMeans	10.9472383199
	Freedman-Diaconis con SKMeans	13.5706562096
	K-proportional con SKMeans	15.0027614601

Figura 3.11: Etapa dos con factores de equilibrio iguales

ver en las figuras 3.14 y 3.15 respectivamente. En estas gráficas se observa como crece lentamente el valor de K con respecto al número de elementos de un conjunto, lo cual las hace ideales para nuestra propuesta.

Para el uso del sistema, el usuario podrá elegir entre las cuatro técnicas que más se adecúan a las característica antes mencionadas que son:

- Dougherty
- Sturges

Cargado a K		
	Dougherty con EM	**2.0689968582**
	Ten Bins con EM	**4.1832039534**
	Sturges con EM	**4.2689727844**
	Scott con EM	6.1883868915
	Dougherty con SKMeans	6.2861826748
	Sturges con SKMeans	8.0598402319
	Ten Bins con SKMeans	8.3981734998
	Scott con SKMeans	10.0210272799
	Freedman-Diaconis con EM	10.1879577649
	K-proportional con EM	11.7593413818
	Freedman-Diaconis con SKMeans	13.5122591222
	K-proportional con SKMeans	15.0331439268

Figura 3.12: Etapa dos con factores de equilibrio cargados a K

Cargado al ruido		
	Dougherty con EM	**3.0784952873**
	Ten Bins con EM	**4.6715845389**
	Sturges con EM	**4.9952250277**
	Scott con EM	6.2744887772
	Freedman-Diaconis con EM	8.6426012609
	Dougherty con SKMeans	9.3678432627
	K-proportional con EM	10.0132880791
	Sturges con SKMeans	10.681526199
	Ten Bins con SKMeans	10.9940388586
	Scott con SKMeans	11.8734493599
	Freedman-Diaconis con SKMeans	13.6290532969
	K-proportional con SKMeans	14.9723789934

Figura 3.13: Etapa dos con factores de equilibrio cargados al Ruido

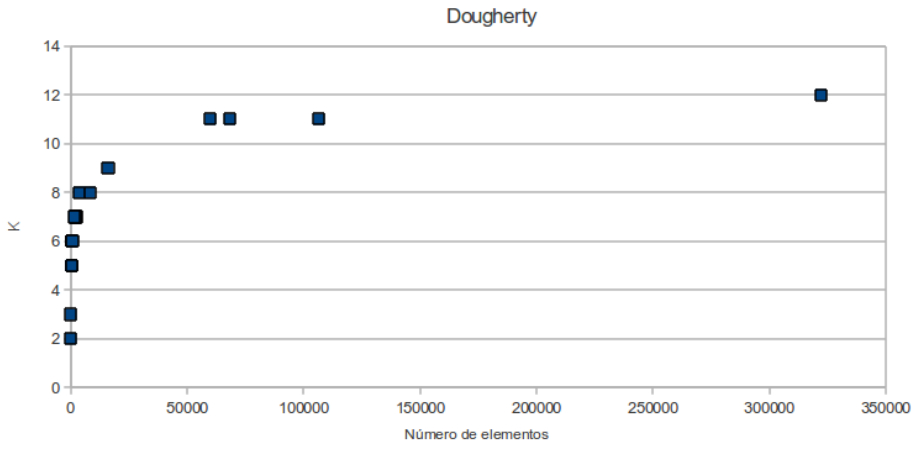

Figura 3.14: Comportamiento de Dougherty

- Scott

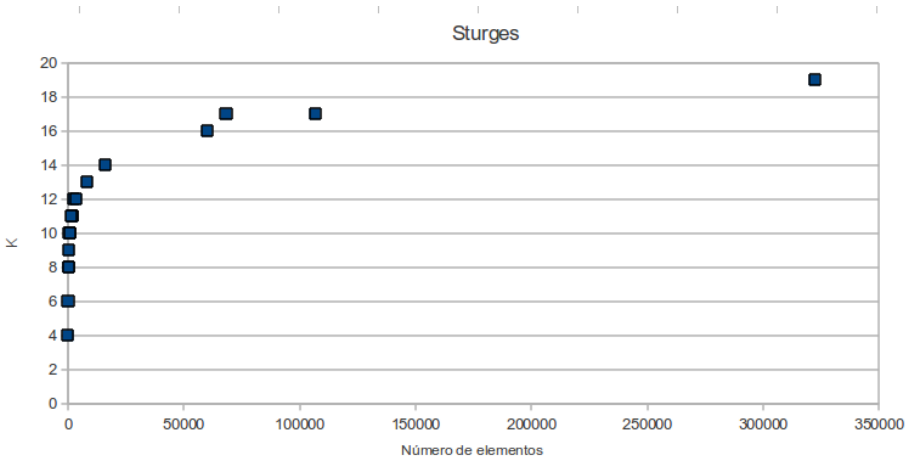

Figura 3.15: Comportamiento de Sturges

- Ten bins

Todas ellas combinadas con el algoritmo EM. De esta forma el usuario tiene la flexibilidad de elegir la técnica que particularmente más le convenga sin que necesite quedarse con solo una forma de discretización.

Una vez que el este módulo termine su tarea, estamos listos para pasar al siguiente módulo.

3.4. Módulo de minería de datos

Para este módulo hará uso del algoritmo SUBDUE. SUBDUE es un algoritmo de aprendizaje basado en el descubrimiento patrones utilizando grafos para la representación de estructuras y relaciones de los datos y de esta forma convertirlas en conocimiento. Esto significa que la fase de preparación de datos incluye una transformación de los datos a un formato de grafo.

El espacio de búsqueda el algoritmo basado en grafos consiste en todos los subgrafos que se pueden derivar a partir del grafo de entrada. Esto quiere decir que el espacio de búsqueda es exponencial de la misma manera que es el tiempo de ejecución de estos algoritmos al menos que se restrinjan de alguna manera para que corran en tiempo polinomial.

Una parte muy importante del algoritmo de minería de datos es el criterio de evaluación. Este criterio se utiliza para determinar cuales subgrafos del espacio de búsqueda son relevantes y pueden ser considerados como parte de los resultados. En el método basado en grafos, SUBDUE utiliza el principio de longitud de descripción mínima (MDL) para evaluar los subgrafos descubiertos. El principio MDL dice que la mejor descripción del conjunto de datos es aquella que minimiza la longitud de la descripción de todo el conjunto de datos. En el método basado en grafos, el principio MDL se utiliza para determinar que tan

bien un grafo comprime al grafo de entrada. De esta manera, todos los subgrafos que se generan durante el proceso de búsqueda se evalúan de acuerdo al principio MDL y los mejores subgrafos se eligen como parte del resultado.

SUBDUE se ha aplicado con éxito en muchas áreas del conocimiento, incluyendo la bioinformática (compuestos químicos), visión por computadora, recuperación de texto, análisis de archivos web-logs, entre otros [27]. Por estas razones se decidió usar SUBDUE como motor de minería de datos.

3.5. Módulo de interpretación de datos

Cuando el módulo de minería de datos temine su tarea, encontrado las subestructuras comunes en la colección de grafos etiquetados es necesario ofrecerle al usuario una forma sencilla de interpretar los nuevos resultados, que distan un poco de los resultados tradicionales, como los de SUBDUE.

Los resultados hasta este punto contienen etiquetas que para el usuario son desconocidas por eso necesitamos dar a conocer los rangos con los que trabajó el módulo de discretización y así los resultados del módulo de minería de datos tomarán más sentido.

Primero se ofrece una copia del archivo original de entrada agregando en forma de comentarios los rangos de cada elementos que fue sujeto a un proceso de discretización. Es decir, todos los elementos que el usuario marcó como continuos.

```
v 14 100
v 15 97.2
```

Figura 3.16: Ejemplo de 2 vértices con valores continuos

```
v 14 100 % velocidad 0 [ 96.8134 , 100 ]
v 15 97.2 % velocidad 0 [ 96.8134 , 100 ]
```

Figura 3.17: Ejemplo de 2 vértices con los rangos una vez discretizados

En las figuras 3.16 y 3.17 se muestra el proceso de discretización. En la primer imagen se encuentran 2 vértices numerados como 14 y 15 respectivamente y con los valores 100 y 97.2.

En la segunda imagen se muestran los mismos vértices con los mismos valores pero ya cuentan con una etiqueta que indica el conjunto en el que se encuentra ese vértice, es decir los 2 vértices se encuentran en el conjunto “velocidad”. También cuentan con un número entero el cual indica el número del cluster en el que cayó el valor del del vértice, es decir, para el ejemplo, los 2 vértices cayeron en el cluster 0. Por último se muestra el rango del cluster en el que

cayó dicho vértice.

Esta copia del archivo original se usará para modificar los archivos de salida de nuestro motor de minería de datos y crear un último archivo de salida más sencillo de entender para el usuario.

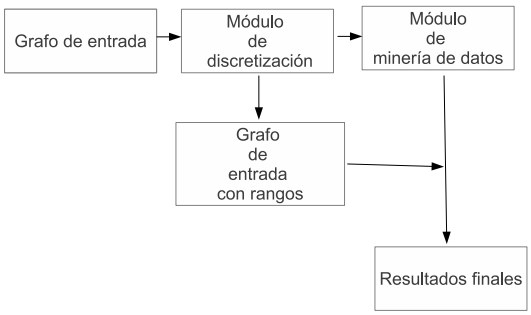

Figura 3.18: Diagrama de bloques para interpretar los resultados

En la figura 3.18 se muestra el funcionamiento de la propuesta para interpretación de datos.

Capítulo 4

Experimentos y resultados

Para experimentar con el sistema se propone hacer 2 fases:

1. Fase de experimentación teórica.

   En esta fase se propone experimentar con casos sintéticos con la intención de introducir diferentes tipos de casos con los que nuestra propuesta se pueda enfrentar.

2. Fase de experimentación con ejemplos reales.

   Para esta fase se experimentó con algunas bases de datos reales tomadas de la página de UCI machine learning [29].

4.1. Experimento teórico y resultados

En esta fase se propone una base de datos de 1000 tuplas con 5 distintos atributos. Cada uno de estos atributos tendrá un comportamiento espacial. De los 5 atributos 4 son continuos mientras que el otro será una etiqueta que representará una clase.

El primer atributo tendrá una distribución uniforme, a este atributo lo conoceremos con el nombre de distribuido (D), donde para cada elemento i, en el conjunto de este atributo D se cumple que $D_{i+1} = D_i + \frac{1}{2}$.

Utilizando el graficador de weka[1] [31], podemos ver una distribución uniforme en la figura 4.1.

Para el segundo atributo intencionalmente se hará una división muy clara entre un grupo de números muy cercanos. Esto con el fin de observar el comportamiento de la clusterización cuando visiblemente se espera un resultado.

[1] weka es una plataforma de software para aprendizaje automático y minería de datos escrito en Java y desarrollado en la Universidad de Waikato. También es un software libre distribuido bajo licencia GNU-GPL

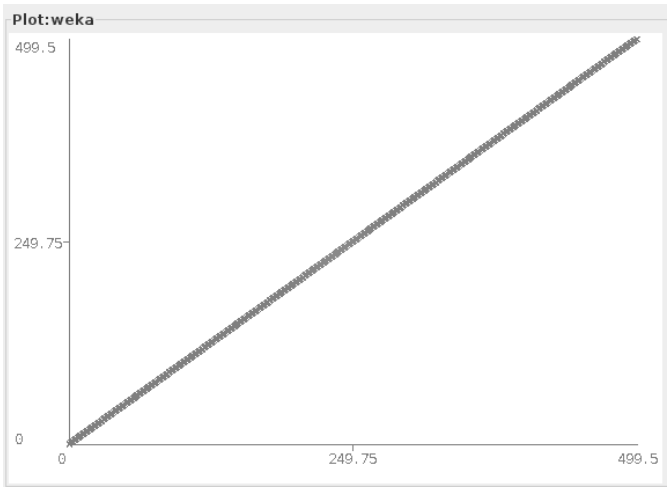

Figura 4.1: Comportamiento gráfico del atributo D.

A este atributo lo conoceremos con el nombre de clusterizado (C).

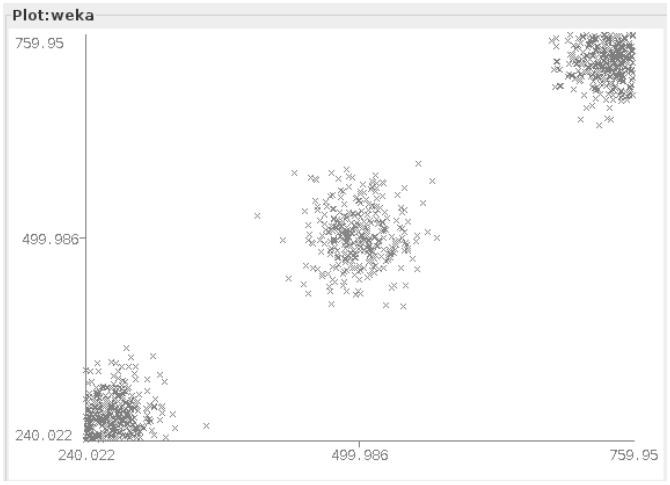

Figura 4.2: Gráfica de la división del atributo C en 3 grupos.

En la figura 4.2 podemos ver claramente la división en 3 grupos con el graficador que nos ofrece weka.

También es importante recalcar que no precisamente se esperan 3 clusters ya que el número de clusters depende de muchas más cosas, de las cuales se hizo mención en la sección 3.3.1.

También haremos la introducción de un atributo donde todos y cada uno de sus elementos esten muy cercanos entre ellos. Es decir, que visualmente se le considere a todos los elementos de este atributo como un solo cluster.

A este atributo lo conoceremos como uncluster (U). En la figura 4.3 podemos ver como cada elemento esta muy cercano a los demás.

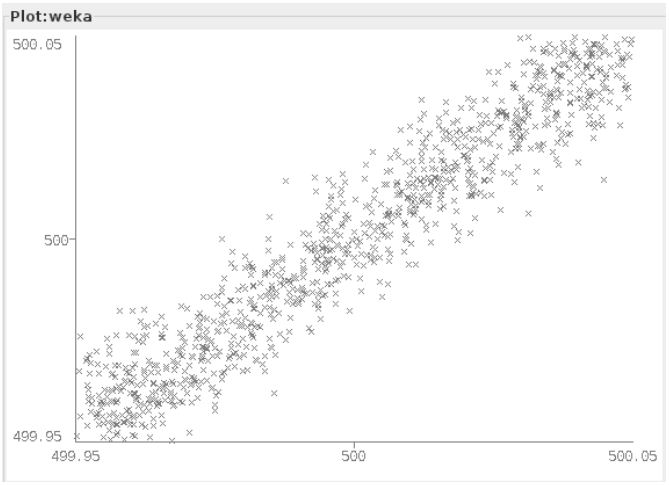

Figura 4.3: Gráfica del comportamiento del atributo U.

También introduciremos un atributo totalmente al azar. No se utilizó ninguna regla ni patrón para la construcción de este conjunto. A este atributo lo conoceremos como random (R).

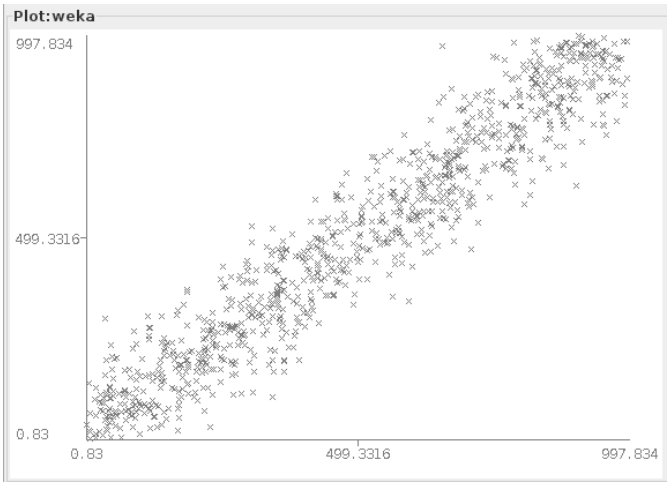

Figura 4.4: Comportamiento gráfico del atributo R.

Para el último atributo se elegirá al azar una letra minúscula del alfabeto inglés para representar la clase del conjunto de parámetros antes mencionados.

En el cuadro 4.1 se puede observar las características más importantes

Cuadro 4.1: Características de los atributos

Atributo	Característica	Elemento menor	Elemento mayor
Clase	Discreto	a	z
D	Continuo	0	499.5
C	Continuo	240.022	759.95
U	Continuo	499.55	500.05
R	Continuo	0.83	997.834

de cada uno de los atributos de la base de datos propuesta. Es importante mencionar que cada atributo cuenta con 1000 elementos.

Atributo distribuido (D)

Lo que se espera de este atributo, es que al pasar por un proceso de discretización, cada uno de los clusters cuenten relativamente con un número similar de elementos en cada uno de ellos.

Los resultados del proceso de discretización del atributo D están representados en el cuadro 4.2

Cuadro 4.2: Resultados del atributo D

Cluster	Rango	Elementos
0	[0 , 52.5]	209
1	[52.8007 , 143]	211
2	[143.5 , 247.5]	184
3	[248.0 , 353.0]	109
4	[353.5 , 445.062]	106
5	[445.5 , 499.5]	181
Total de elementos		1000

Podemos ver que si bien el número de elementos en cada cluster varía, también se puede observar que el valor que se espera tener es el de los 1000 elementos dividido entre los 6 clusters, es decir, aproximadamente 166.66 elementos por cluster (en promedio).

Con estos datos vemos que la desviación estándar es aproximadamente 43.3384. Lo que significa un 4.333 % con respecto al total de elementos. Por lo que se puede argumentar que los resultados están dentro de lo que se esperaban para este atributo.

Atributo clusterizado (C)

Lo que se espera de este atributo es que se note claramente la división de los 3 grupos significativos de los cuales se mencionó en la sección 4.1 y que claramente se identifican en la figura 4.2.

Cuadro 4.3: Resultados del atributo C

Cluster	Rango	Elementos
0	[240.022 , 249.699]	163
1	[249.978 , 259.912]	170
2	[490.059 , 500.571]	181
3	[500.645 , 509.955]	152
4	[740.034 , 747.702]	125
5	[747.806 , 759.95]	209
Total de elementos		1000

Las características que aparecen en el cuadro 4.3 nos muestran el comportamiento de nuestro sistema con respecto al atributo C. En este cuadro vemos 6 clusters donde existen 3 grupos con 2 clusters muy cercanos entre sí.

Observamos que nuestra propuesta divide en 2 cada uno de los grupos que vemos en la figura 4.2. Así que el igual que el atributo anterior, los resultados son los esperados.

Atributo uncluster (U)

Lo que se espera de este atributo es que al ser todos los elementos de este conjunto tan cercanos entre sí, los clusters se adapten al mínimo. Es decir, el mejor resultado sería un cluster (de ahí el nombre del atributo).

En el cuadro 4.4 notamos que los resultados son los esperados.

Cuadro 4.4: Resultados del atributo U

Cluster	Rango	Elementos
0	[499.95 , 500.05]	1000

Atributo random (R)

De este atributo no se puede esperar mucho, solo podríamos mencionar las características que definen a un proceso de discretización, como el hecho de que no exista ruido o que no exista intersección entre clusters y otras reglas antes mencionadas en la sección 2.2. Podemos ver los resultados en el cuadro 4.5.

De estos resultados podemos concluir que efectivamente nuestra propuesta se esta comportando como se espera y no esta rompiendo ninguna regla de las definiciones mencionadas.

Para esta sección nos enfocamos a la parte de discretización y su comportamiento. En la siguiente sección nos enfocaremos más en la parte de minería de datos basada en grafos.

Cuadro 4.5: Resultados del atributo R

Cluster	Rango	Elementos
0	[0.829203 , 101.497]	96
1	[104.219 , 275.966]	168
2	[280.256 , 489.161]	224
3	[491.89 , 715.364]	231
4	[717.577 , 900.307]	184
5	[900.934 , 997.834]	97
Total de elementos		1000

4.2. Experimentos reales y resultados

En esta fase de experimentación se utilizaron 5 bases de datos con diversas características, tomando en cuenta número de tuplas, el número de atributos y el número de atributos continuos de cada una de ellas.

Los detalles de cada una de las bases de datos con las que se experimentó se encuentran en el cuadro 4.6.

Cuadro 4.6: Bases de datos

BD	Tuplas	Atributos	Atributos continuos
Iris	150	5	4
wall-following robot navigation	5457	3	2
vertebral column	311	7	6
abalone	4178	9	7
echocardiogram	133	11	5

Para cada experimento se hará una comparación con los resultados del sistema SUBDUE para ver si existen mejoras con respecto a este sistema.

Es importante aclarar que para las pruebas a continuación se corrió el sistema SUBDUE con los parámetros que ofrece dicho sistema por default.

4.2.1. Base de datos iris

Nuestro primer ejemplo real con el que se experimento fue la base de datos iris. Esta es quizás la mejor base de datos conocida que se encuentran en la literatura de reconocimiento de patrones [19].

El conjunto de datos consta de 3 clases, todas de tipo nominal, de 50 casos cada uno, donde cada una de ellas se refiere a un tipo de planta de iris.

Esta base de datos cuenta con 4 tipos de datos numéricos, los cuales los conoceremos como: longitud del sépalo (sepallength: continuo), anchura del

sépalo (sepalwidth: continuo), longitud del pétalo (petallength: continuo) y anchura del pétalo (petalwidth: continuo).

Resultados de SUBDUE para la base de datos iris

Las subestructuras frecuentes que SUBDUE encuentra con esta base de datos se pueden ver en las figuras 4.5, 4.6 y 4.7.

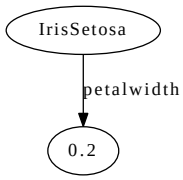

Figura 4.5: Experimento iris con SUBDUE. Número de instancias: 29

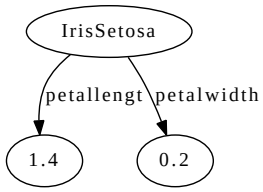

Figura 4.6: Experimento iris con SUBDUE. Número de instancias: 8

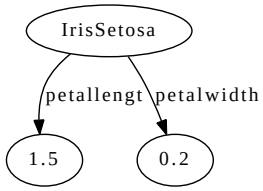

Figura 4.7: Experimento iris con SUBDUE. Número de instancias: 7

Resultados de nuestra propuesta para la base de datos iris

Las subestructuras frecuentes que nuestra propuesta encuentra con esta base de datos se pueden ver en las figuras 4.8, 4.9 y 4.10.

Se puede ver que nuestra propuesta encuentra un número mayor de subestructuras frecuentes y también es visible el hecho de que cada subestructura es de mayor tamaño que las que encuentra SUBDUE.

Vemos que SUBDUE establece su mayor relación en el ancho del pétalo de iris en el valor 0.2, mientras que nuestra propuesta establece la mayor relación

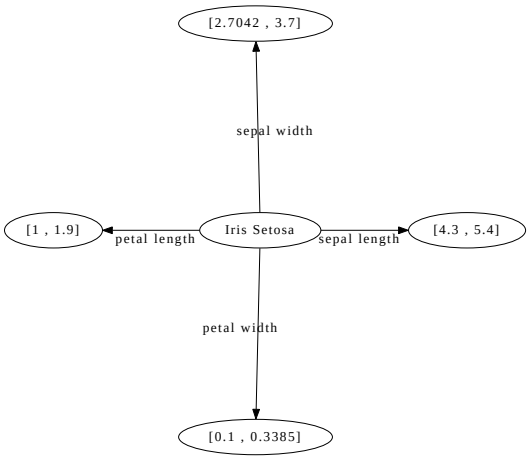

Figura 4.8: Experimento iris con propuesta. Número de instancias: 33

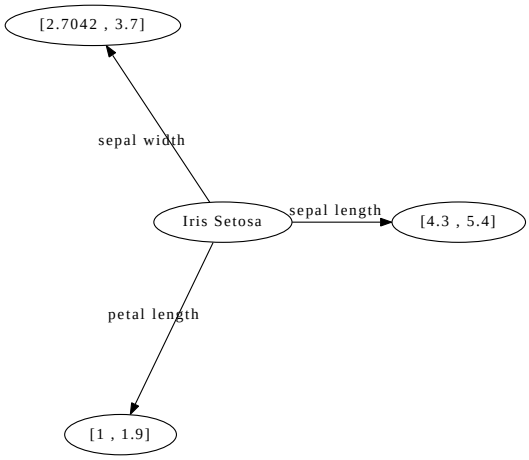

Figura 4.9: Experimento iris con propuesta. Número de instancias: 38

en el rango de [0.1 , 0.3385].

Un inconveniente es que las 3 subestructuras frecuentes, encontradas con nuestra propuesta, tienen intersección entre ellas.
En general podemos ver que las subestructuras de las figuras 4.9 y 4.10 son subgrafos de la subestructura de la figura 4.8.

4.2.2. Base de datos wall-following robot navigation

Esta base de datos contiene cuatro lecturas de los sensores llamados sensor 1 (sensor1: continuo) y sensor 2 (sensor2: continuo)y la simplificación de la etiqueta de clase correspondiente. Consisten en las lecturas de los sensores mínimas dentro de los 60 grados con respecto a la parte delantera, izquierda,

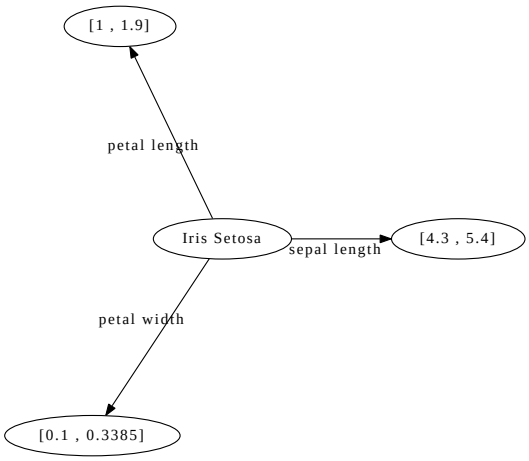

Figura 4.10: Experimento iris con propuesta. Número de instancias: 37

derecha del robot que por naturaleza son de tipo continuo [24].

Resultados de SUBDUE para la base de datos wall-following robot navigation

Las subestructuras frecuentes que SUBDUE encuentra con esta base de datos se pueden ver en las figuras 4.11, 4.12 y 4.13.

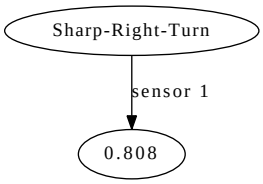

Figura 4.11: Experimento robot con SUBDUE. Número de instancias: 32

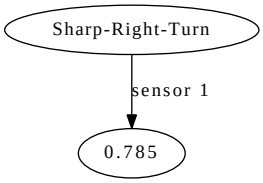

Figura 4.12: Experimento robot con SUBDUE. Número de instancias: 33

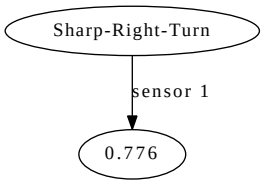

Figura 4.13: Experimento robot con SUBDUE. Número de instancias: 32

Resultados de nuestra propuesta para la base de datos wall-following robot navigation

Las subestructuras frecuentes que nuestra propuesta encuentra con esta base de datos se pueden ver en las figuras 4.14, 4.15 y 4.16.

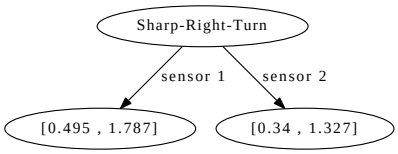

Figura 4.14: Experimento robot con propuesta. Número de instancias: 2080

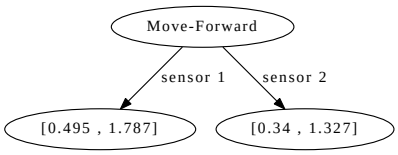

Figura 4.15: Experimento robot con propuesta. Número de instancias: 1644

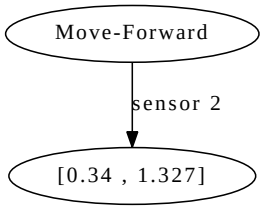

Figura 4.16: Experimento robot con propuesta. Número de instancias: 2205

Se puede ver que nuestra propuesta encuentra un número mayor de subestructuras frecuentes que el sistema SUBDUE.

Notamos que SUBDUE establece su mayor relación cuando al girar a la derecha el sensor 1 registra una diferencia en grados desde 0.776, pasando por la diferencia de 0.785 y terminando en la diferencia de 0.808.

Sin embargo nuestra propuesta encuentra 2 subestructuras frecuentes con un número considerablemente mayor de repeticiones cuando hay un movimiento hacia el frente y cuando hay un movimiento a la derecha, donde el sensor 1 esta en el rango [0.495 , 1.787] y el sensor 2 esta en el rango [0.34 , 1.327].

Al igual que el experimento anterior existen intersecciones entre las subestructuras frecuentes encontradas por nuestra propuesta pero en menor proporción.

4.2.3. Base de datos vertebral column

Esta base cuenta con datos biomédicos construidos por el Dr. Henrique da Mota, durante un período de residencia médica en el Grupo de Investigación Aplicada en Ortopedia (GARO) del Centro Médico-Quirúrgica de readaptación des Massues, Lyon, Francia. Los datos se han organizado en dos tareas de clasificación diferentes, pero relacionados.

La primera tarea consiste en clasificar a los pacientes que pertenecen a una de tres categorías: Normal (100 pacientes), la hernia de disco (60 pacientes) o espondilolistesis (150 pacientes).

Para la segunda tarea, las categorías de hernia de disco y espondilolistesis se fusionaron en una sola categoría etiquetada como “anormal”. Por lo tanto, la segunda tarea consiste en clasificar a los pacientes como pertenecientes a uno de cada dos categorías: Normal (100 pacientes) o anormal (210 pacientes).

Cada paciente se representa en el conjunto de datos por seis atributos biomecánicos derivados de la forma y orientación de la espina dorsal y lumbar, la pelvis (en este orden): La incidencia de la pelvis (inpelvis), inclinación de la pelvis (incpelvis), el ángulo de la lordosis lumbar (anglumbar), sacra pendiente (sacral), radio de la pelvis (radiopelvis) y el grado de espondilolistesis (espondilolistesis). Todos estos atributos son de tipo continuo.

La siguiente convención se utiliza para las etiquetas de clase: DH (hernia discal), espondilolistesis (SL), Normal (NO) y anormal (AB) [7].

Resultados de SUBDUE para la base de datos vertebral column

Las subestructuras fecuentes que SUBDUE encuentra con esta base de datos se pueden ver en las figuras 4.17, 4.18 y 4.19.

Resultados de nuestra propuesta para la base de datos vertebral column

Las subestructuras frecuentes que nuestra encuentra con esta base de datos se pueden ver en las figuras 4.20, 4.21 y 4.22.

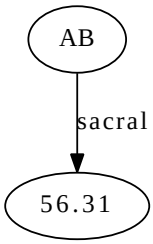

Figura 4.17: Experimento column con SUBDUE. Número de instancias: 4

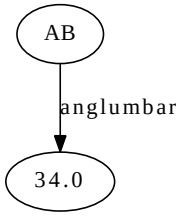

Figura 4.18: Experimento column con SUBDUE. Número de instancias: 3

Una vez mas nuestra propuesta encuentra un número mayor de subestructuras frecuentes que el sistema SUBDUE.

Podemos observar que SUBDUE establece su mayor relación entr la clase anormal y el dato de sacra pendiente en el valor 56.31.

Sin embargo nuestra propuesta establece que la mayor subestructura encontrada es en la clase anormal relacionada con la inclinación de la pelvis en el rango [11.92 , 15.4].

Nuestra propuesta coincide con SUBDUE al encontrar la subestructura frecuente de la clase anormal relacionada con el atributo de el ángulo de la lordosis lumbar sin embargo SUBDUE solo encuentra la frecuencia en el valor 52.2 mientras que nuestro sistema encuentra el lo encuentra en el rango [46.16 , 54.55] que es justamente donde se encuentra el valor que SUBDUE relacionó.

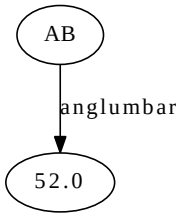

Figura 4.19: Experimento column con SUBDUE. Número de instancias: 3

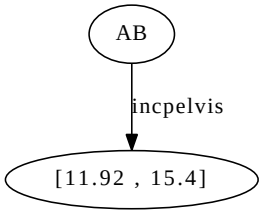

Figura 4.20: Experimento column con propuesta. Número de instancias: 39

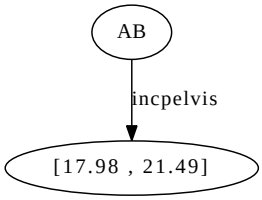

Figura 4.21: Experimento column con propuesta. Número de instancias: 34

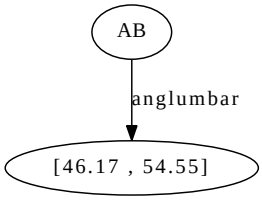

Figura 4.22: Experimento column con propuesta. Número de instancias: 33

A diferencia de los experimentos pasados nuestra propuesta no nos arrojo subestructuras frecuentes con intersecciones.

4.2.4. Base de datos abalone

Esta base de datos aborda la predicción de la edad del abulón a partir de mediciones físicas. La edad del abulón se determina por el corte de la carcasa a través del cono, la tinción y contando el número de anillos a través de un microscopio, una tarea que consume mucho tiempo. Otras medidas, que son más fáciles de obtener, se utilizan para predecir la edad. Para más información, como los patrones del clima y la ubicación (por lo tanto la disponibilidad de alimentos) puede ser necesaria para resolver el problema.

Teniendo en cuenta es el nombre del atributo, tipo de atributo, la unidad de medida y una breve descripción. El número de anillos es el valor para predecir: ya sea como un valor continuo o como un problema de clasificación.

Los atributos que ofrece esta base de datos son: Sexo (M, F y I), longitud (Length: continuo), diámetro (Diameter: continuo), altura (Height: continuo),

todo el peso (Wholeweight: continuo), el peso sin concha (Shuckedweight: continuo), peso de las vísceras (Viscera weigh: continuo), shell peso (Shellweight: continuo) y anillos (Rings) [1].

Resultados de SUBDUE para la base de datos abalone

Las subestructuras frecuentes que SUBDUE encuentra con esta base de datos se pueden ver en las figuras 4.23, 4.24 y 4.25.

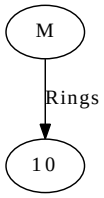

Figura 4.23: Experimento abalone con SUBDUE. Número de instancias: 294

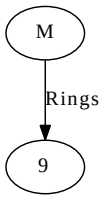

Figura 4.24: Experimento abalone con SUBDUE. Número de instancias: 278

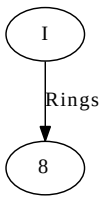

Figura 4.25: Experimento abalone con SUBDUE. Número de instancias: 274

Resultados de nuestra propuesta para la base de datos abalone

Las subestructuras frecuentes que nuestra propuesta encuentra con esta base de datos se pueden ver en las figuras 4.26, 4.27 y 4.28.

Una vez más nuestra propuesta encuentra un número considerablemente mayor de subestructuras frecuentes que el sistema SUBDUE.

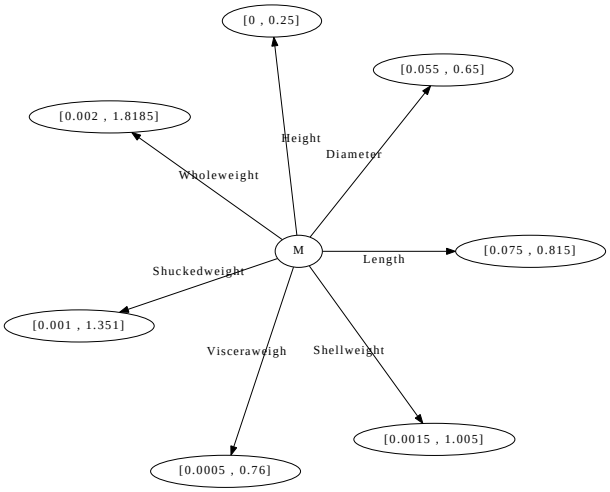

Figura 4.26: Experimento abalone con propuesta. Número de instancias: 1450

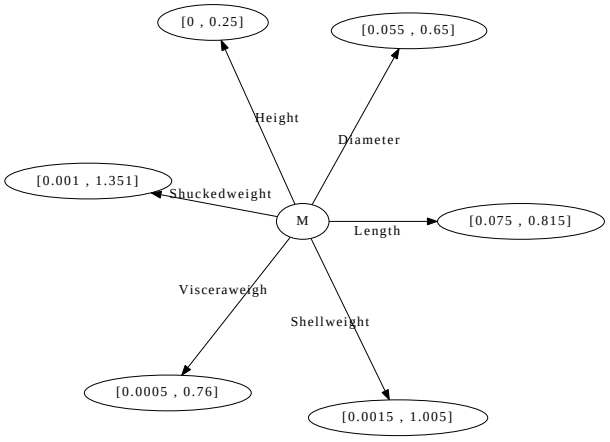

Figura 4.27: Experimento abalone con propuesta. Número de instancias: 1528

Notemos que SUBDUE establece su mayor relación en la clase masculino relacionada con la etiqueta de anillos en 10 y 9. Otra subestructura frecuente encontrada por SUBDUE es la de la clase infantil relacionada con la etiqueta de anillo en el valor 8.

Sin embargo nuestra propuesta establece que la mayor subestructura encontrada es en la clase masculino relacionada con todos los atributos de tipo continuo en diversos rangos.

Una vez más nuestra propuesta nos da como resultado intersecciones entre las subestructuras.

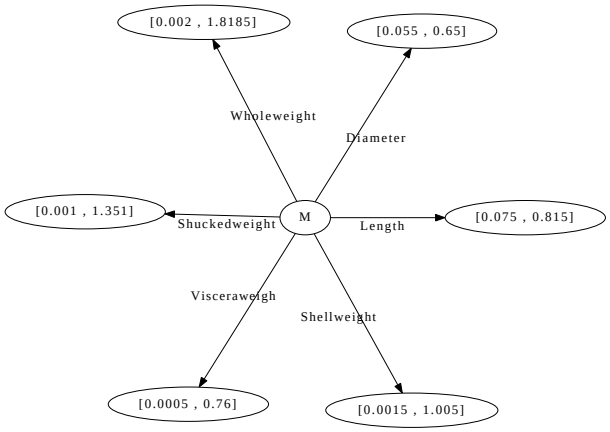

Figura 4.28: Experimento abalone con propuesta. Número de instancias: 1450

4.2.5. Base de datos echocardiogram

Todos los pacientes sufrieron ataques al corazón en algún momento en el pasado. Algunos todavía están vivos y otros no. Las variables de supervivencia y aún con vida, en conjunto, indican si un paciente sobrevivió por lo menos un año después del ataque al corazón.

El problema abordado por los investigadores anteriores era el predecir a partir de las otras variables si el paciente sobrevive al menos un año. La parte más difícil de este problema es predecir correctamente que el paciente no va a sobrevivir. (Parte de la dificultad parece ser el tamaño del conjunto de datos.)

La base cuenta con el atributo de supervivencia (survival: continuo), es el número de meses los pacientes sobrevivientes (ha sobrevivido, si el paciente sigue vivo). Debido a que todos los pacientes tenían sus ataques al corazón en momentos diferentes, es posible que algunos pacientes han sobrevivido menos de un año, pero todavía están vivos. Tales pacientes no se puede utilizar para la tarea de predicción mencionado anteriormente [12].

También existe el atributo aún con vida (still-alive) una variable binaria. 0 para muerte al final del período de supervivencia, 1 significa que sigue vivo.

Edad en situación de ataque al corazón (age-at-heart-attack) es la edad en años, cuando se produjo un ataque al corazón.

Derrame pericárdico (pericardial-effusion), binario. El derrame pericárdico es líquido alrededor del corazón. 0 si no hay flujo, 1 si hay líquido.

Fracción de acortamiento (fractional-shortening: continuo), los números inferiores del corazón son cada vez más anormales.

EPSS (epss), los números más altos son cada vez más anormales.

DDVI (lvdd: continuo), ventrículo izquierdo al final de la dimentasión de la diástole. Esta es una medida del tamaño del corazón al final de la diástole. Corazones grandes tienden a ser los corazones enfermos.

Puntuación de movimiento de la pared (wall-motion-score), una medida de cómo los segmentos del ventrículo izquierdo se mueve.

Índice de la puntuación de movimiento de la pared (wall-motion-index: continuo), igual que el anterior, dividido por el número de segmentos vistos. Por lo general, los segmentos 12-13 se observan en un ecocardiograma. Se puede utilizar esta variable en lugar de la puntuación de movimiento de la pared.

Multi-variable (mult: continuo), esta variable puede ser ignorada.

Nombre del paciente (se han sustituido por "name").

Resultados de SUBDUE para la base de datos echocardiogram

Las subestructuras frecuentes que SUBDUE encuentra con esta base de datos se pueden ver en las figuras 4.29, 4.30 y 4.31.

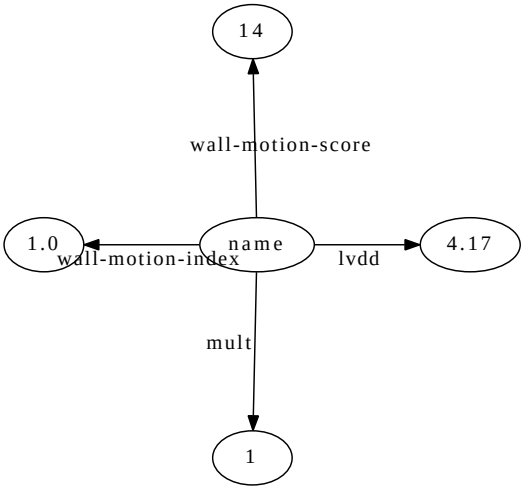

Figura 4.29: Experimento echo con SUBDUE. Número de instancias: 18

Resultados de nuestra propuesta para la base de datos echocardiogram

Las subestructuras frecuentes que SUBDUE encuentra con esta base de datos se pueden ver en las figuras 4.32, 4.33 y 4.34.

De nuevo nuestra propuesta encuentra un número mayor de subestructuras frecuentes que el sistema SUBDUE.

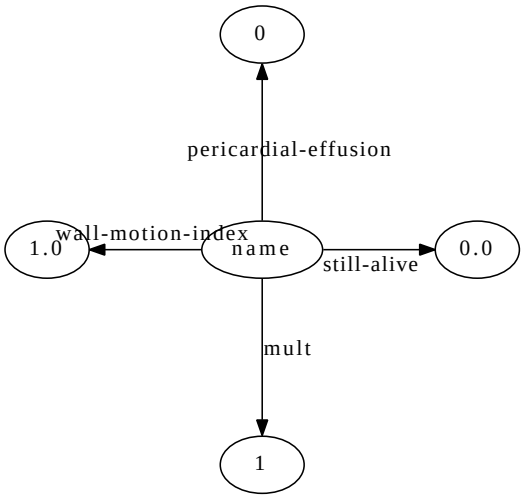

Figura 4.30: Experimento echo con SUBDUE. Número de instancias: 17

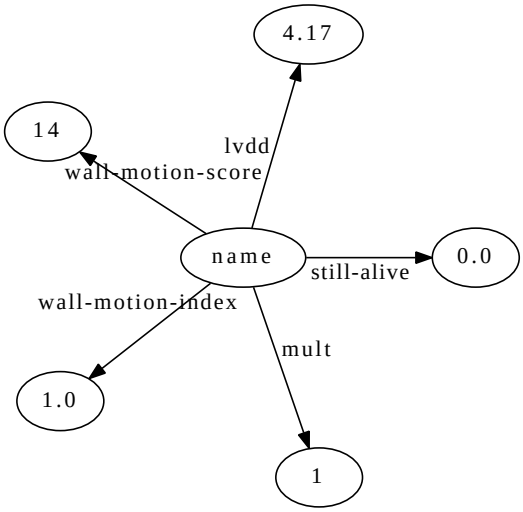

Figura 4.31: Experimento echo con SUBDUE. Número de instancias: 13

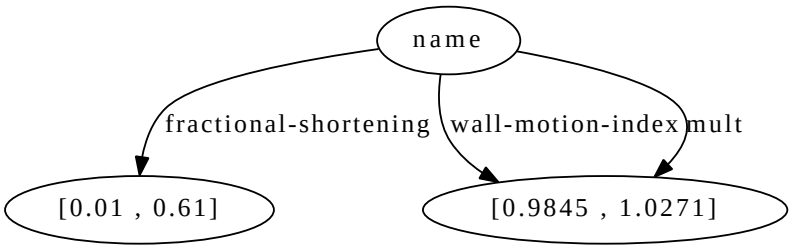

Figura 4.32: Experimento echo con propuesta. Número de instancias: 34

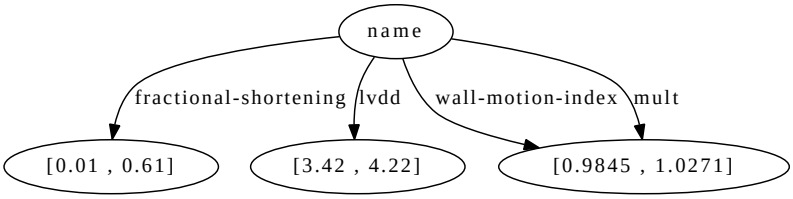

Figura 4.33: Experimento echo con propuesta. Número de instancias: 24

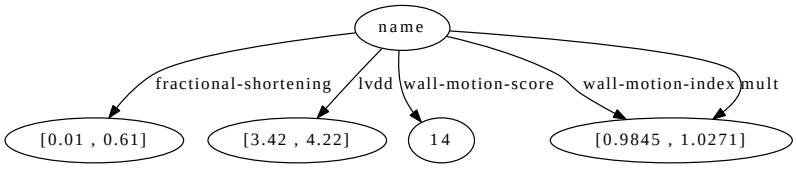

Figura 4.34: Experimento echo con propuesta. Número de instancias: 18

Una vez más nuestra propuesta nos da como resultado intersecciones entre las subestructuras, pero a diferencia de los experimentos anteriores SUBDUE encontró subestructuras frecuentes con intsersecciones.

4.3. Discusión de resultados

En primer lugar debemos mencionar la consistencia de los resultados de nuestro sistema en comparación con SUBDUE, por ejemplo, en el experimento de la base de datos iris apreciamos en la figura 4.7 que SUBDUE encuentra 7 repeticiones de la estructura donde está relacionada el tipo de iris setosa con el largo del pétalo en 1.5 y con el ancho del pétalo en 0.2. Por otro lado en la figura 4.8 se muestra una estructura con 33 repeticiones de la relación de la iris setosa con respecto al largo del pétalo en el rango [1 , 1.9] y con respecto al ancho del pétalo en el rango [0.1 , 0.3385]. los 2 valores con lo que esta relacionada la estructura encontrada por SUBDUE se encuentran en los rangos encontrados por nuestro sistema lo que habla de congruencia en los resultados.

Así como en este ejemplo, los resultados guardan una cierta lógica, también en los demás experimentos tienen esa congruencia. Por lo que de este experimento podemos mencionar que los resultados son confiables.

En segundo lugar debemos mencionar la diferencia en el número de estructuras frecuentes que encuentra cada sistema.

Tomando de nuevo las figuras 4.7 y 4.8 se aprecia que aparte de que nuestro sistema encuentra un estructura más grande, el número de repeticiones es mayor. En el cuadro 4.7 se muestra la comparación del número de instancias encontradas. Solo se tomarán las estructuras más grandes de cada experimento ya que existen muchas intersecciones en los resultados lo que podría generar redundancias.

Cuadro 4.7: Comparación de resultados

Experimento	Número de estructuras frecuentes encontradas por SUBDUE	Número de estructuras frecuentes encontradas por esta propuesta	Mejora
Iris	8	33	412.5 %
wall-following robot navigation	33	2205	6681.8 %
vertebral column	4	39	975 %
abalone	294	1528	519.7 %
echocardiogram	18	34	188.8 %
Promedio			1755.5 %

Capítulo 5

Conclusiones y trabajo futuro

En el presente trabajo de tesis se abordó el problema de la minería de datos basada en grafos con etiquetas continuas. De acuerdo a los experimentos realizados y los resultados alcanzados se pueden hacer las siguientes conclusiones:

SUBDUE es un sistema muy popular usado en todo el mundo, incluso hay diferentes sistemas que utilizan la misma estructura que SUBDUE. Los archivos de entrada para SUBDUE requieren un mínimo de modificaciones para poder ser procesados por el sistema aquí propuesto, por lo que puede llegar a ser de gran impacto.

El sistema es fácil de utilizar ya que la parte de clusterización es transparente para el usuario lo que permite que este sistema pueda ser usado por usuarios de otros sistemas que abarcan minería de datos con grafos. Los resultados que arroja el sistema propuesto se representan de forma típica para estos usuarios por lo que la interpretación de resultados tampoco será un problema.

En el capítulo 4 se presentan los resultados experimentales obtenidos. El sistema propuesto logró encontrar patrones ocultos que por la misma naturaleza de los datos continuos le era imposible encontrar a otros sistemas. El número de instancias de patrones frecuentes creció considerablemente con respecto a los patrones encontrados por SUBDUE, al mismo tiempo de encontrar estructuras más grandes. Por lo que podemos decir que se cumplieron con los objetivos planteados.

Para este trabajo se realizó un experimento empírico utilizando varias técnicas y algoritmos ya conocidos tanto para clusterizar como para la minería de datos. Para ninguno de estos se contempló su complejidad, por lo que es posible que el sistema reaccione de forma lenta para grafos con grandes volúmenes de datos.

Como trabajo futuro se plantea trabajar con diversas técnicas de clusterizado

como máquinas de soporte vectorial, redes neuronales, máquinas de núcleos dispersos, entre otras [3] para mejor la etapa de clusterizado. Así como mejorar la parte de minería de datos con grafos con la intención de bajar el costo computacional experimentado con apendizaje de conceptos [13].

Apéndice A: Formato de archivos de SUBDUE

El archivo de entrada a SUBDUE consiste en uno o más grafos representados en formato de texto con extensión ".g". Cada grafo es precedido por la línea "XP"que indica que se trata de un ejemplo positivo, o "XN"que indica que el grafo describe un ejemplo negativo [28]. Si el primer grafo en el archivo es positivo, entonces su "XP"se puede omitir. Cada grafo es una secuencia de vértices y arcos definidos de la siguiente manera.

Los vértices se definen por una 'v'seguida por el número identificador del vértice y la etiqueta del vértice:

$$\text{v} < \# >< label >$$

Dónde $< \# >$ es una identificación única del vértice para el grafo y $< label >$ es cualquier cadena o número real. Las cadenas que contienen espacios en blanco o el carácter de comentario deben ser encerradas por dobles comillas. Las identificaciones de los vértices para un grafo deben empezar en 1 y deben aumentar en 1 para cada vértice sucesivo.

Los arcos son definidos por una letra que especifica su tipo, el número de identificación de los vértices que conecta y finalmente su etiqueta.

$$\text{e } <\text{v1}><\text{v2} >< label >$$
$$\text{d } <\text{v1}><\text{v2} >< label >$$
$$\text{u } <\text{v1}><\text{v2} >< label >$$

Dónde <v1 > y <v2 > son la identificación del vértice fuente y el vértice destino respectivamente (previamente definidos) y $< label >$ es cualquier cadena o número real. Las cadenas que contienen espacios en blanco o el carácter de comentario deben ser encerrados por doble comillas. Los arcos que comienzan con é'son asumidos como dirigidos a menos que la opción '-undirected' sea especificada en la línea de comando. Los arcos que comienzan con 'd'siempre son dirigidos y los arcos que empiezan con ú'son siempre no dirigidos.

Todo lo que este delante del carácter '%' será considerado como comentario y se ignorará.

5.1. Ejemplo de un archivo en SUBDUE

Supongamos que se desea expresar en archivo con formato de SUBDUE la figura 5.1.

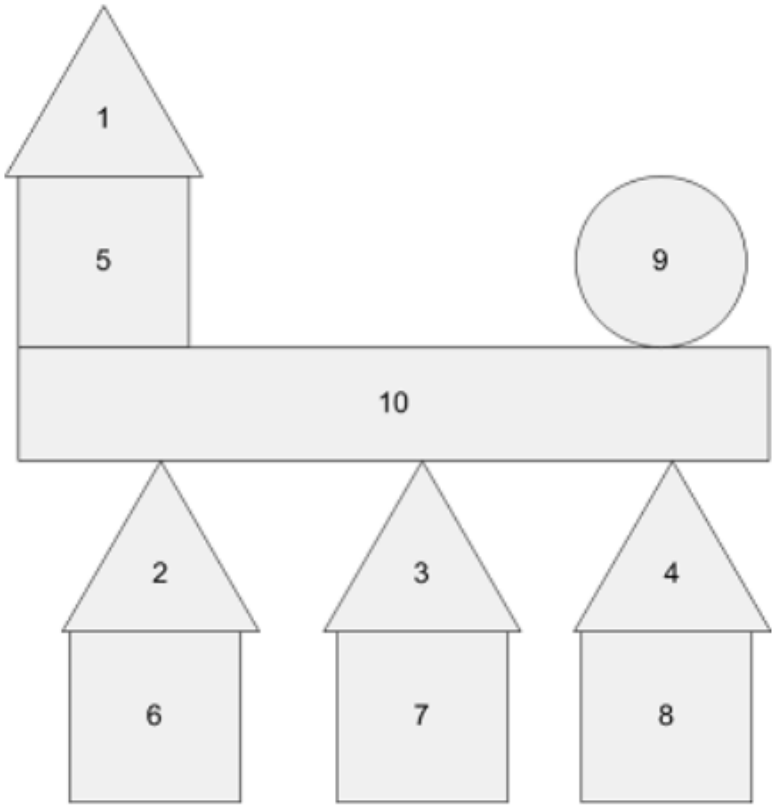

Figura 5.1: Figura que se desea expresar en formato SUBDUE

Lo primero que se debe hacer es representar dicha figura en forma de un grafo. Para esto primero necesitamos definir las partes del grafo. Cada elemento numerado en la figura será un objeto (object). También se asignará la forma (shape) de cada objeto de la figura. Es decir, las partes 1, 2, 3 y 4 son triángulos (triangle), las partes 5, 6, 7 y 8 son cuadrados (square), la parte 9 es un circulo (circle) y por último la parte 10 es un rectángulo (rectangle). Por último se definirá que objetos están sobre (on) otros objetos.

En la figura 5.2 podemos observar el grafo resultante de la figura 5.1. Este grafo finalmente debe ser representado en el formato de SUBDUE, el cual se muestra en el cuadro 5.1

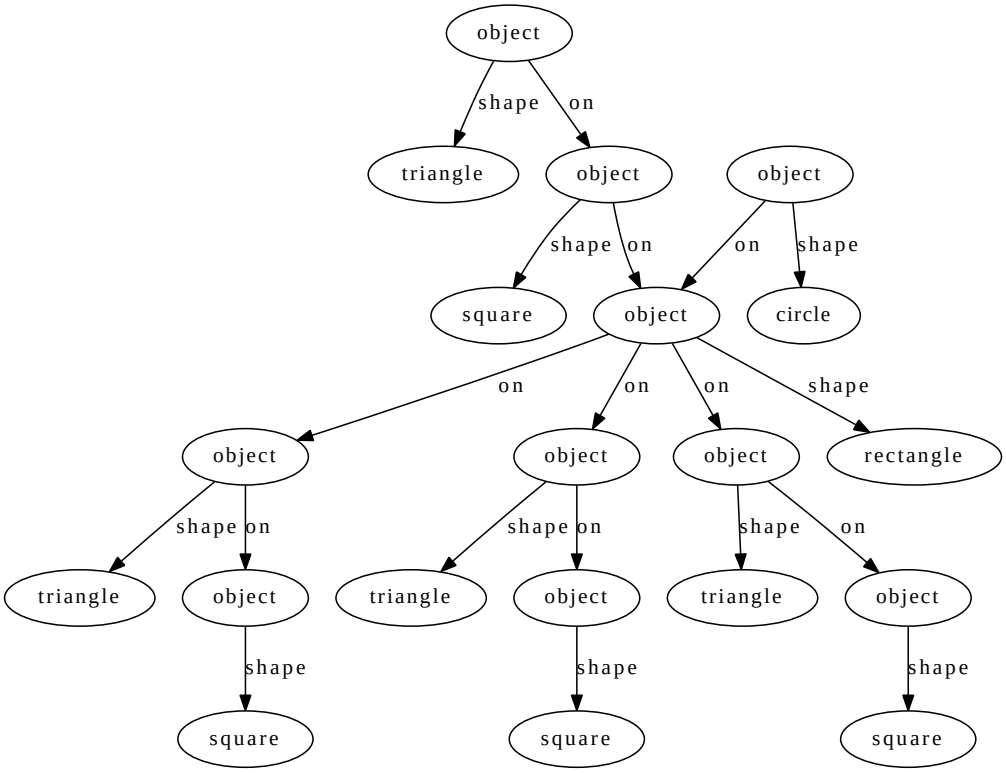

Figura 5.2: Grafo equivalente de la figura 5.1

5.2. Convertir un archivo de SUBDUE

Vamos a plantear un caso en el que se desea convertir de un archivo tradicional de SUBDUE a un archivo con nuestra propuesta, es decir, que dado un ejemplo con valores continuos se pueda agregar el nuevo comando "C" al formato original.

Supongamos que se desea hacer un estudio de 4 de los estados de México. Estos estados son Puebla, Veracruz, Oaxaca y Guerrero.

Lo que a la investigación interesa son datos como temperatura promedio anual medida en grados centígrados, número de habitantes y densidad poblacional medida en hab/km^2.

También se agregarán otros datos, como, saber de estos estados cuales son colindantes entre sí, al mismo tiempo de aclarar en la investigación que los 4 son estados y no son ciudades ni municipios.

Esto nos generaría un grafo como el de la figura 5.3.

A partir de este grafo podemos generar un archivo en formato de SUBDUE. Lo primero que se necesita en enumerar cada uno de los nodos con su respectiva

Cuadro 5.1: Archivo en formato de SUBDUE

Comando	Primer parámetro	Segundo parámetro (en su caso)	Etiqueta
v	1		object
v	2		object
v	3		object
v	4		object
v	5		object
v	6		object
v	7		object
v	8		object
v	9		object
v	10		object
v	11		triangle
v	12		triangle
v	13		triangle
v	14		triangle
v	15		square
v	16		square
v	17		square
v	18		square
v	19		circle
v	20		rectangle
e	1	11	shape
e	2	12	shape
e	3	13	shape
e	4	14	shape
e	5	15	shape
e	6	16	shape
e	7	17	shape
e	8	18	shape
e	9	19	shape
e	10	20	shape
e	1	5	on
e	2	6	on
e	3	7	on
e	4	8	on
e	5	10	on
e	9	10	on
e	10	2	on
e	10	3	on
e	10	4	on

etiqueta. los cuales podrían quedar más o menos así:

v 1 Guerrero
v 2 Oaxaca
v 3 Puebla
v 4 Veracruz

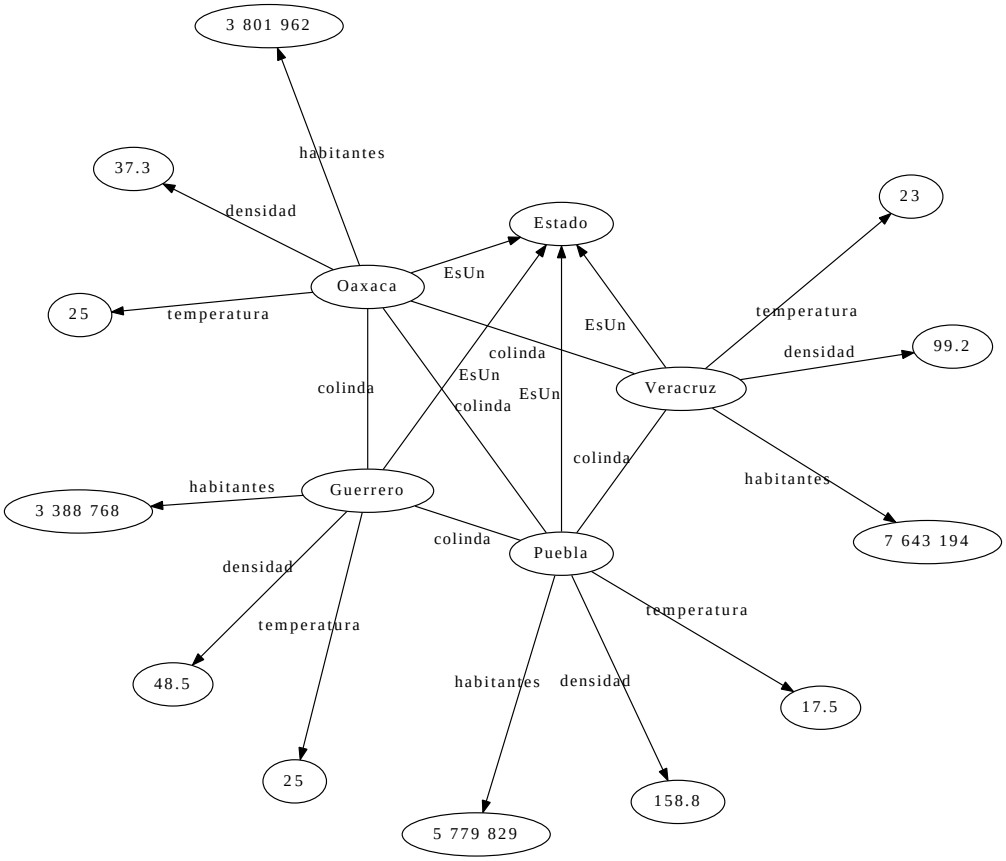

Figura 5.3: Relación en forma de grafo de los 4 estados

v 5 Estado
v 6 3388768
v 7 48.5
v 8 25
v 9 3801962
v 10 37.3
v 11 25
v 12 5779829
v 13 158.8
v 14 17.5
v 15 7643194
v 16 99.2
v 17 23

Ahora se necesita agregar las relaciones, es decir, agregar las aristas. Primero agregaremos la relación de estado. Esta relación es unidireccional y se agregarían de la siguiente forma:

e 1 5 EsUn

e 2 5 EsUn
e 3 5 EsUn
e 4 5 EsUn

Ahora se agregaran las aristas de las relaciones de colindancia. Es importante mencionar que la relación de colindancia es bidireccional, por lo que estas aristas se agregaran con el comando "u" de la siguiente forma:

u 1 2 colinda
u 1 3 colinda
u 2 3 colinda
u 2 4 colinda
u 3 4 colinda

Estas aristas describen que Guerrero colinda con Oaxaca y con Puabla pero no con Veracruz, que Oaxaca colinda con Guerrero, Puebla y Veracruz, que Puebla colinda con Guerrero, Oaxaca y Veracruz y finalmente que Veracruz colinda con Oaxaca y Puebla.

Por último queda agregar los atributos de habitantes, densidad y temperatura. Estas relaciones se expresan de la siguiente forma:

e 1 6 habitantes
e 1 7 densidad
e 1 8 temperatura
e 2 9 habitantes
e 2 10 densidad
e 2 11 temperatura
e 3 12 habitantes
e 3 13 densidad
e 3 14 temperatura
e 4 15 habitantes
e 4 16 densidad
e 4 17 temperatura

Con estos datos quedaría completo la representación del grafo en SUBDUE, pero la intención es mapearlo para el sistema propuesto, por lo que se debe de identificar las aristas que en uno de sus extremos contengan un dato continuo.

Existen 5 diferentes etiquetas en este grafo. Evidentemente que las etiquetas "EsUn" y "colinda" no contienen datos continuos por lo que fácilmente se pueden descartar. Las etiquetas "temperatura" y "densidad" si cuentan con datos continuos en uno de sus extremos, por lo que se puede aplicar el comando "C" a estas etiquetas.

Pero ¿cómo podemos clasificar a la etiqueta "habitantes"?, es claro que siempre sera un número entero ya que en los censos nunca contarán a $\frac{1}{4}$ de persona o a 2.3 personas. Este atributo es estrictamente entero, pero eso no significa que sea un atributo discreto ya que es posible que se desee agrupar los datos de los habitantes de los estado en rangos, aunque por supuesto

esa es decisión del usuario, pero para este ejemplo agregaremos la etiqueta "habitantes" a las etiquetas continuas.

De esta forma finalmente el archivo esta listo para ser procesado por nuestro sistema y se muestra en el cuadro 5.2.

Cuadro 5.2: Archivo con el comando C

Comando	Primer parámetro	Segundo parámetro (en su caso)	Etiqueta
C			habitantes
C			densidad
C			temperatura
v	1		Guerrero
v	2		Oaxaca
v	3		Puebla
v	4		Veracruz
v	5		Estado
v	6		3388768
v	7		48.5
v	8		25
v	9		3801962
v	10		37.3
v	11		25
v	12		5779829
v	13		158.8
v	14		17.5
v	15		7643194
v	16		99.2
v	17		23
e	1	5	EsUn
e	2	5	EsUn
e	3	5	EsUn
e	4	5	EsUn
u	1	2	colinda
u	1	3	colinda
u	2	3	colinda
u	2	4	colinda
u	3	4	colinda
e	1	6	habitantes
e	1	7	densidad
e	1	8	temperatura
e	2	9	habitantes
e	2	10	densidad
e	2	11	temperatura
e	3	12	habitantes
e	3	13	densidad
e	3	14	temperatura
e	4	15	habitantes
e	4	16	densidad
e	4	17	temperatura

Bibliografía

[1] Abalone: `http://archive.ics.uci.edu/ml/datasets/Abalone`

[2] Birgé, L. and Rozenholc, Y.: How Many Bins Should Be Out In a Regular Histogram. Technological Report, Laboratoire Probabilits et Modles Alatoires, Universit Pierre et Marie Curie, Paris, France, (2002).

[3] Bishop, C.: Pattern Recognition and Machine Learning. Springer Editorials, (2006).

[4] Borgelt, C. and Berthold, M.: Mining Molecular Fragments: Finding Relevant Substructures of Molecules. In Proccedings of the 2002 International Conference on Data Mining (ICDM'02), Maebashi, Japan, (2002).

[5] Boulle, M.: Optimal Bin Number for Equal Frequency Discretizations Supervised Leraning. Intelligent Data Analysis, (2005)175-188.

[6] Codd, E.F.: A relational model of data for large shared data banks. Commun. ACM 13, 377–387 (1970).

[7] Column:
`http://archive.ics.uci.edu/ml/datasets/Vertebral+Column`

[8] Cook, D., Holder, L. and Djoko, S.: Knowledge Discovery From Structural Data. Journal of Intelligence and Information Sciences,(1995) 229-245.

[9] David W. Scott. On optimal and data-based histograms. Biometrika, 66(3):605-610, 1979.

[10] Devijver, P. A., and J. Kittler.: Pattern Recognition: A Statistical Approach, Prentice-Hall, London, 1982.

[11] Doughetry, J., Kohavi, R., and Sahami, M.: Supervized and Unsupervised Discretization of Continiuos Features. In Proccedings of the 20^{th} International Conference on Machine Leraning, (1995) 194-202.

[12] Echocardiogram:
`http://archive.ics.uci.edu/ml/datasets/Echocardiogram`

[13] Fonseca, R.: Diseño de un algoritmo de minería de datos basado en grafos para la tarea de aprendizaje de conceptos. Tesis para Obtener el grado de Maestría en Ciencias de la Computación, 2012.

[14] Freedman, D., and Diaconis, P.: On the histogram as a Density Estimator : L2 Theory. Zeitschrift for Wahrscheinlichkeitstheorie und verwandte Gebiete, (1981) 453-476.

[15] Gago, A., Carrasco, J. and Medina, J.: Minería de Subgrafos Conexos Frecuentes en Colecciones de Grafos Etiquetados. Technical Report, Coordinación de Ciencias Computacionales, Instituto Nacional de Astrofísica, Óptica y Electrónica, 2009.

[16] Giudici, R.:Introducción a la teoría de grafos, 1997.

[17] Huan, J., Wang, W. and Prins, J.: Efficient Mining of Frequent Subgraph in the Presence of Isomorphism. In Proccedings of the 2003 International Conference on Data Mining (ICDM'03), Melbourne, FL, (2003) 549-552. 211-218.

[18] Inokuchi, A., Washio T. and Motada, H.: An Apriori-Based Algorithm for Mining Frequent Substructures from Graph Data. In Proccedings of the 2000 European Symposium on the Principle Of Data Mining and Knowledge Discovery (PKDD'00), Lyon, France, (2000) 13-23.

[19] Iris: `http://archive.ics.uci.edu/ml/datasets/Iris`

[20] Kuramochi, M. and Karypi, G.: An Efficient Algoritmh for Discovering Frequent Subgraphs. Technical Report, Departament of Computing Scince, University of Minnesota, 2002.

[21] Kuramochi, M. and Karypis, G.: Frequent Subgraph Discovery . In Proceedings of the 2001 international Conference on Data Mining (ICDM'01), San Jose, CA, (2001) 313-320.

[22] Olmos, I., Gonzalez J,. and Osorio, M.: Subgraph Isomorphism Detection Using a Code Based Representation.

[23] Onofre, G.: Identificación de subtgrafos Isomorfos con soporte para Características Continuas. Tesis para Obtener el grado de Maestría en Ciencias de la Computación en la Benemérita Universidad Autónoma de Puebla, 2010.

[24] Robot: `http://archive.ics.uci.edu/ml/datasets/`

[25] Schmidberger, G. and frank, E.: Unsupervised Discretization Using Tree-Based Density Estimation. In Proccedings 9th Europan Conference on Principles and Practice of Knowledge Discovery in Databases, (2005) 240-251.

[26] Sturges, H.: The Choice of a Class Interval. J. American Statistical Association, (1926) 65-66.

[27] SUBDUE: `http://ailab.wsu.edu/subdue/`

[28] SUBDUEA: `http://cygnus.uta.edu/SUBDUE`

[29] UCI Machine Learrning: `http://archive.ics.uci.edu/ml/`

[30] Vesanto, J., and Alhoniemi, E. (2000). Clustering of the Self-Organizing Map. IEEE Transactions on Neutral Networks, 11 (3), 586-600.

[31] Weka: `http://www.cs.waikato.ac.nz/ml/weka/`

[32] Xifeng, Y. and Han, J.: gSpan: Graph Based Substructure Pattern Mining.Proceedings of the IEEE, International Conference on Data Mining, (2002) 51-18. In Proccedings of the 18^{th} International FLAIRS Conference, (2005) 474-479.

[33] Yang, Y. and Webb, G.: A Comparative Study of Discretization Methods for Naïve-Bayes Classifiers. (2002) 159-173. In Proceedings of PKAW 2002, The 2002 Pacific Rim Knowledge Acquisition Work-shop, Tokyo, Japan, pp. 159-173.

[34] Y. Yang and G. I. Webb.: Proportional k-interval Discretization for Naive-Bayes Classifiers. In Proceedings of Twelfth European Conference on Machine Learning, pages 564-575, 2001.