Un modelo de Clustering Temporal

María Daniela Navas
Facultad de Ingeniería
Universidad de Buenos Aires
mnavas@mnavas.com.ar

Juan M. Ale
Facultad de Ingeniería
Universidad de Buenos Aires
ale@acm.org

Resumen: Clustering consiste en particionar el conjunto de datos en colecciones de objetos de manera que dentro de cada partición los objetos sean "similares" entre sí, y a su vez se "diferencien" de los objetos contenidos en otras particiones. En la literatura han sido propuestos muchos algoritmos para realizar el proceso de clustering, pero la mayoría de ellos tiene un enfoque estático, por lo tanto, estas soluciones no pueden ser aplicadas correctamente para datos más complejos, como colecciones de objetos espacio-temporales. En muchos casos, la información guardada en las bases de datos tiene una naturaleza espacial dinámica: además de tener datos espaciales, a menudo se asocian los mismos con información temporal, como marcas de tiempo (time-stamp) ,manejo de versiones, fechas o rango de fechas. En el presente trabajo se propone un método de Clustering Temporal que realiza el proceso de clustering sólo teniendo en cuenta los atributos espaciales, pero para distintos momentos de tiempo (dato aportado por los atributos temporales). Esto nos permite ver cómo varían los clusters durante el transcurso del tiempo, observar la trayectoria de los objetos, y obtener distintas estadísticas sobre el movimiento de clusters y objetos, que no se podrían obtener aplicando un algoritmo de clustering estándar.

Palabras clave: clustering temporal, data mining, k-d-Median.

1.Introducción

Se denomina proceso de descubrimiento de conocimiento en bases de datos (Knowlegde Discovery in Databases-KDD) al proceso de descubrir conocimiento interesante, como por ejemplo, patrones, asociaciones, cambios, anomalías y estructuras significativas en grandes cantidades de datos guardados en bases de datos u otros repositorios de información.

Data Mining se refiere a la aplicación de algoritmos para la extracción de patrones utilizando los datos disponibles, o sea, que es un componente del proceso de Knowlegde Discovery.

Clustering es una de las principales tareas en Data Mining, y consiste en particionar el conjunto de datos (dataset) en colecciones de objetos o instancias de manera que dentro de cada partición los objetos sean "similares" entre sí, y a su vez se "diferencien" de los objetos contenidos en otras particiones. Similitud y disimilitud son expresadas a través de funciones de distancia/similitud y a las particiones resultantes se las denomina clusters.

En la literatura han sido propuestos hasta el momento, muchos algoritmos para realizar el proceso de clustering (Han et al.,[8]). Sin embargo, hasta ahora, la mayoría de los resultados conciernen a datos simples, compuestos por elementos representados como puntos individuales en un espacio multidimensional (posiblemente con una combinación de dimensiones discretas y continuas).

Las soluciones desarrolladas bajo estas restricciones no pueden ser aplicadas correctamente para datos más complejos, como colecciones de objetos espacio-temporales. En muchos casos, la información guardada en las bases de datos tiene una naturaleza espacial dinámica: no solamente las bases tienen datos espaciales, sino que a menudo asocian los datos espaciales con información temporal, como marcas de tiempo (time-stamp) o manejo de versiones, e incluso en muchos casos contienen información temporal dada por una fecha o rango de fechas.

En estos casos algunas de las peculiaridades de los datos deben ser sacrificadas para poder utilizar los procesos de clustering convencionales. Por lo tanto, se requieren soluciones capaces de manejar las nuevas nociones de "distancias" que pueden desprenderse de datos tan complejos.

Para resolver esto, algunas propuestas proponen utilizar nuevos conceptos de distancias y métricas adaptadas a un contexto espacio-temporal (Nanni, [13,14]), esto permite agrupar en un mismo cluster a objetos que tienen similitudes tanto en sus atributos espaciales como en sus atributos temporales.

En el presente trabajo se resuelve el problema desde otro enfoque: se propone un método que realiza el proceso de clustering sólo teniendo en cuenta los atributos espaciales, pero para distintos momentos de tiempo (dato aportado por los atributos temporales). Esto nos permite ver cómo varían los clusters durante el transcurso del tiempo. Lo que se hizo fue crear un modelo temporal que determina a qué tipo de datos se puede aplicar este proceso de clustering, cómo se deben especificar los mismos, cómo se procesa la información y cómo se muestran los resultados.

Este modelo se comporta como un framework o marco; es un "molde" que puede basarse en cualquier algoritmo de clustering "convencional" (que no maneje datos temporales), para crear el algoritmo temporal que provee todas las ventajas que se especifican en el presente trabajo, por lo tanto se investigó que algoritmo podía ser el más adecuado para este propósito.

Para elegir un algoritmo de clustering adecuado para una determinada aplicación deben considerarse muchos factores, según se indica en (Han et al,[8]). Los mismos son:

- *Objetivo de la aplicación*: El objetivo de la aplicación a menudo afecta la elección de un algoritmo de clustering. Por ejemplo, si se desea encontrar la ubicación óptima de las sucursales de una cadena de supermercados, el objetivo será encontrar la mínima distancia entre los clientes y cada sucursal. En el caso de reconocimiento de imágenes, a menudo es deseable hallar los clusters que se forman naturalmente, en donde los clusters deberán tener cierta uniformidad de color, densidad, etc.
  Los algoritmos de particionamiento (partitioning algorithms) como k-means o k-medoids, son útiles para el primer caso, y tienden a descubrir clusters con forma esférica y tamaño similar. En cambio para el segundo caso responden mejor los algoritmos basados en densidad (density-based algorithms)

- *Elección entre calidad y velocidad:* Existe siempre un problema al elegir entre velocidad de procesamiento y calidad de los clusters obtenidos. Un algoritmo adecuado debe cumplir con estas dos características, pero a veces la cantidad de instancias a procesar juega un papel importante en el tiempo de ejecución del algoritmo de clustering. Un algoritmo que produce clusters de calidad, por lo general es incapaz de manejar grandes cantidades de información. A su vez, cuando se trabaja con grandes volúmenes de datos se realiza una especie de compresión sobre la información inicial, perdiendo así calidad.

- *Características de los datos*: Las características de los datos a los cuáles se quiere aplicar el clustering, también son un factor importante para la elección del método adecuado.

  - *Tipos de datos de los atributos*: La similitud entre dos objetos es medida según la diferencia entre los valores de sus atributos. Cuando todos los atributos son numéricos las medidas de distancia como Euclídea o Manhattan, pueden ser fácilmente computadas. En cambio, cuando los atributos son binarios, categóricos u ordinales, este cálculo se complica. Por lo tanto, la mayoría de los algoritmos se aplican sólo con datos numéricos.
  - *Dimensionalidad*: la dimensionalidad se refiere al número de atributos de cada objeto. Muchos algoritmos funcionan bien con pocos atributos, pero al aumentar las dimensiones los resultados se degeneran. La degeneración puede darse por la disminución de la velocidad, o el empobrecimiento de la calidad de los clusters.
  - *Cantidad de ruido en los datos*: Algunos algoritmos son muy sensibles a ruido o elementos aislados, no pudiendo funcionar correctamente ante la presencia de los mismos.

El objetivo del presente trabajo es implementar un nuevo método de clustering temporal el cual estará basado en algún algoritmo ya existente. Cómo se especificó con anterioridad, existen numerosos algoritmos, y la elección del adecuado depende del tipo de problema que se quiera encarar.

Por lo tanto, es necesario buscar un área determinada, en la cuál sea útil la información que puede proveer este clustering temporal.

Se ha decidido utilizar bases de datos con información demográfica, por ejemplo, datos que provee el INDEC (Instituto Nacional de Estadística y Censos), ya que sería muy útil poder determinar como varían los resultados obtenidos en censos durante el transcurso de los años.

Estas bases de datos poseen la particularidad de ser muy grandes, por lo que se necesita un algoritmo que sea lo suficientemente rápido para poder analizar grandes volúmenes de datos.

La información recolectada es representada por una cantidad mediana de atributos, que pueden ser numéricos o nominales. La cantidad de ruido o de objetos aislados no se puede determinar, pero puede estar presente.

Según las división de algoritmos de clustering explicada anteriormente, los algoritmos que mejor se adaptan a este tipo de problema son los de particionamiento.

Comparando los distintos algoritmos existentes dentro de este rubro se seleccionó un algoritmo llamado k-D-median, presentado por sus autores (Estivill-Castro, et al. [6]) como un algoritmo de clustering de propósito general rápido y robusto.

Lo que se hizo en el presente trabajo fue implementar un nuevo algoritmo en Java para integrarlo al sistema WEKA, que realiza el formateo de los datos y el proceso de clustering en sí. También se creó en Matlab una aplicación que permite analizar gráficamente los resultados del clustering.

El resto del trabajo está organizado de la siguiente forma: en la sección 2 se detallan algunos trabajos relacionados, así como también la elección del algoritmo de clustering que sirve como base al clustering temporal. En la sección 3 se explica el modelo temporal de datos utilizado, y su adaptación para realizar el clustering temporal. En la sección 4 se describe como se implementó el algoritmo. En la sección 5 se describen una serie de experimentos que muestran la performance y escalabilidad del algoritmo. Por último la sección 6 remarca las conclusiones.

2.Trabajos Relacionados

En la bibliografía actual existen varios intentos para tratar los objetos temporales. Por ejemplo, en (Roddick et. al, [17]) se propone tratar las coordenadas temporales como geográficas, utilizando el espacio cartesiano), pero prácticamente ninguno aplica estos conocimientos específicamente al proceso de clustering.

En (Nanni, [13,14]) se propone una familia de medidas de similitud definida sobre objetos espacio-temporales, que se pueden utilizar con cualquier algoritmo de clustering para mejorar su performance. Primero se propone una definición de lo que es una distancia espacio-temporal, enfocándose en las propiedades de las métricas. Luego se analiza el comportamiento de algoritmos de clustering al aplicar alguna distancia de las definidas en su esquema, y se proponen optimizaciones a dichos algoritmos.

En (Han et al, [8]) se presenta un estudio sobre distintos tipos de algoritmos de clustering, que se consultó para evaluar la mejor forma de implementar el presente trabajo.

La diferencia del presente trabajo con otros trabajos o algoritmos implementados es justamente que no se trata sólo de un algoritmo, sino de un framework que se puede aplicar a cualquier algoritmo de clustering convencional, para obtener información temporal sobre prácticamente cualquier campo de estudio (dependiendo del algoritmo base).

3.Modelo Temporal

Cuando se piensa en la forma de representar valores que varían con el tiempo, la principal pregunta es cómo asociar los hechos con el tiempo. Según (Jensen et.al.,[11]) existen numerosos modelos de datos temporales que de alguna forma asocian los hechos con un tiempo válido.

Enfocándose solo en la semántica (Jensen et al.,[11]) determinaron que los modelos existentes, incluso los desarrollados por ellos, eran demasiado complicados. Entonces crearon un modelo muy simple, Modelo Conceptual Bitemporal de Datos (Bitemporal Conceptual Data Model – BCDM), cuyo objetivo es capturar cuando los hechos son válidos en la realidad (tiempo válido) y cuando son almacenados en la base de datos (tiempo de transacción).

El BCDM emplea el mismo modelo para los dos dominios de tiempo (válido y de transacción): una secuencia finita de cronos. Un crono es la más corta duración de tiempo soportada por una base de datos temporal, es una unidad de tiempo no descomponible, un subintervalo de la línea de tiempo real, de duración fija. Puede estar medido en segundos, en días, en años, etc. dependiendo de las necesidades particulares de los datos procesados.

Un intervalo de tiempo es definido como el tiempo entre dos instantes, un instante de inicio y otro de fin. Un intervalo de tiempo es representado por una secuencia de cronos consecutivos, donde cada crono representa todas las instancias que ocurren durante la duración del mismo. También se puede representar una secuencia de cronos, simplemente por el par de cronos inicial y final. La unión de estos intervalos se denomina elemento temporal.

En este modelo conceptual, una tupla (A_1, A_2, ..., A_n | t^b) consiste en un número de atributos A_1, A_2, ..., A_n , asociados con una marca de tiempo bitemporal t^b .

Combinación y Eliminación de Información repetida

En el modelo conceptual BCDM se observa que existe una cierta flexibilidad en la representación de las tuplas.

Existen dos transformaciones que pueden cambiar la forma de ver los datos sin afectar los resultados de las consultas que se hacen a la base de datos (Jensen et al.,[11])

La primera transformación se denomina *Combinación*. Significa que si dos tuplas, con los valores de atributos explícitos iguales, están temporalmente superpuestas o son temporalmente adyacentes, se pueden juntar en una sola tupla. Esta transformación permite reducir el número de tuplas necesarias para representar una relación bitemporal, útil, por ejemplo, si se desea optimizar el espacio físico de la base de datos.

Esta transformación se puede aplicar tanto para el tiempo válido como para el tiempo de transacción.

Para introducir la otra transformación, antes se debe indicar que una relación tiene información repetida, si contiene dos tuplas distintas con valores equivalentes, o sea que los valores de sus atributos explícitos (no temporales) son idénticos, y tienen marcas de tiempo superpuestas en el tiempo. Esta transformación es empleada para eliminar información temporal redundante, a expensas de agregar tuplas extras. Lo que hace es transformar dos tuplas con valores equivalentes con superposición temporal, en tres tuplas con valores equivalentes, en donde no existe la superposición temporal.

Una relación bitemporal conceptual es estructuralmente simple, es un conjunto de hechos asociados con un conjunto de cronos bitemporales. Pero este modelo conceptual es sólo a nivel del diseño, precisamente por eso es conceptual. Para representar este modelo en una base de datos real existen muchas formas distintas, enumeradas en (Jensen et al., [12]).

El objetivo del clustering temporal es realizar el proceso de clustering sólo teniendo en cuenta los atributos espaciales (utilizando algún algoritmo convencional), pero para distintos momentos de tiempo (dato aportado por los atributos temporales). Esto nos permite ver cómo varían los clusters durante el transcurso del tiempo.

Para realizar este trabajo se tuvo que determinar cuál es el formato más adecuado que deben tener los datos de entrada para obtener los mejores resultados al aplicar el algoritmo de clustering temporal.

Estos datos provienen de una base de datos temporal. Conceptualmente se puede representar a los datos con el modelo BCDM de (Jensen et al.,[11]), pudiendo aplicarles las transformaciones de combinación y eliminación de información repetida.

De los cinco esquemas físicos para representar los datos propuestos por (Jensen et al., [12]), nos basamos en la *Tupla con marca de tiempo de Snodgrass* para obtener la forma de representar los datos a los que se les aplicará el clustering temporal.

Utilizamos el siguiente modelo:

$$R = (A_o, A_1, A_2, \ldots, A_n, V_i, V_f)$$

donde

- R es la relación, o sea es un conjunto de muchos objetos que cumplen con las mismas características.
- A_o: es el atributo que identifica al objeto, es el que se utiliza para determinar como el objeto cambia a través del tiempo.
- $A_1, A_2, \ldots, A_n$: son los atributos explícitos del objeto, pueden ser categóricos o numéricos.
- V_i, V_f : es el tiempo válido de inicio y de fin. El intervalo durante el cual los atributos mencionados son verdaderos en el mundo real.

Sólo se utiliza el tiempo válido porque es el que interesa al proceso de clustering, ya que se estudia como varían los objetos en la realidad. En el caso específico que se quieran agrupar los datos según el momento en que fueron almacenados en la base de datos, se deben reemplazar V_i, V_f por los respectivos tiempos de transacción, pero solo se trabaja con un dominio de tiempo.

Un proceso de clustering temporal abarca cierto período de tiempo, por ejemplo, un año calendario (desde Enero hasta Diciembre). Un intervalo es un subconjunto de ese período de tiempo. El algoritmo de clustering divide el período total de tiempo en T subconjuntos o intervalos. La duración de estos intervalos se calcula teniendo en cuenta el *crono* (que es la mínima unidad de medida utilizada para este proceso). El crono lo determina la unidad con que están almacenados los rangos de tiempo válido de las tuplas. Las medidas disponibles en principio, son: segundos, minutos, horas, días, meses y años. Por ejemplo, en el caso anterior el crono a utilizar sería el "día". Entonces el período completo se vería como 360 días, y al dividir por T=12 se analizarían intervalos de 30 días.

Es importante determinar el crono por un tema de efectividad en los cálculos, ya que hacer el cálculo del ejemplo en segundos implicaría números muy grandes que pueden producir un desbordamiento de memoria. Otro caso: si el período total a observar es cuestión de segundos, medirlo en años implicaría una pérdida de precisión enorme.

Una vez determinada la unidad de medida, el período total y la cantidad de intervalos que se desean analizar nos enfrentamos a que los rangos de tiempo válido definidos para cada tupla no necesariamente coinciden con los intervalos que se desean analizar.

Es importante que para cada intervalo a analizar exista sólo un individuo representativo del mismo, ya que para poder calcular distintas estadísticas y hacer comparaciones, el número de individuos en cada intervalo debe ser el mismo. Es aquí donde acudimos a los conceptos de combinación y eliminación de información repetida para obtener tuplas cuyos rangos de tiempo válido coincidan totalmente con los intervalos a analizar.

Existen dos casos en los que será necesario aplicar alguna transformación a las tuplas:

1) Las tuplas cuyo rango de tiempo válido no está contenido íntegramente en un intervalo a analizar se particionan en dos más tuplas, con los mismos atributos explícitos, pero con distintos intervalos de tiempo (acorde a los intervalos que se desean analizar)

2) Existe más de una tupla para un mismo intervalo de tiempo. En este caso se juntan dichas tuplas para formar una sola: los tiempos se suman y los valores de los atributos se promedian (si son numéricos) o se busca su moda (si son nominales).

4.Implementación

La implementación del Clustering Temporal se divide en dos partes. Una aplicación implementada en Java, denominada Weka (Ian et al,[10]), que es una herramienta para Data Mining, en la que se agregaron funciones específicas para realizar el proceso de Clustering Temporal. Y otra aplicación, implementada en Matlab, que se utiliza para analizar gráficamente los resultados del proceso de clustering.
A su vez, el proceso de clustering temporal implementado en Java consta de 2 etapas:

- Generación del archivo temporal, donde se transforma un archivo de entrada en un archivo que contemple el formato adecuado para procesarlo temporalmente.
- Clustering temporal, donde los datos de entrada se dividen en grupos o clusters, según su unidad de tiempo.

Generación del archivo temporal

Para comenzar a procesar, se deben ingresar los datos en un formato válido. Dicho formato debe cumplir con las normas para los archivos arff (utilizados con la herramienta WEKA):

- Indicar el nombre de la relación, precedido por los caracteres @relation
- Listar nombre y tipo de datos de los atributos.
  Los atributos pueden ser:
  - Categóricos: son valores discretos.
    - Nominales: pueden ser descripciones, nombres, por ejemplo “Sexo”, que puede ser Femenino o Masculino.
    - Ordinales: son categóricos, pero con cierto orden, por ejemplo: “junior, semi senior, senior”
  - Numéricos: reales o enteros. Por ejemplo Edad, Ingresos, etc.

  Si el atributo es categórico, se deberán indicar todos los valores posibles del mismo.

- Listar los valores de los atributos separados por coma (,), precedidos por los caracteres @data

Además, para poder realizar el clustering temporal cada registro debe tener un atributo que lo identifique y un rango de tiempo en el que sean válidos sus atributos especificados. Por lo tanto, cada registro del archivo debe tener estas características:

- Un atributo que indique Fecha de Inicio de validez del registro, la misma debe tener el formato “yyyy-mm-dd hh:mm:ss” o “yyyy/mm/dd hh:mm:ss”. Por ejemplo “2004-12-01 21:35:00”
- Un atributo que indique Fecha de Fin de validez del registro, la misma debe tener el formato “yyyy-mm-dd hh:mm:ss” o “yyyy/mm/dd hh:mm:ss”. Por ejemplo “2004-12-05 21:35:00”
- Un atributo que sea un ID único, que identifique al registro (puede haber mas de un registro para el mismo ID, aunque deben tener distintas fechas)

- Uno o más atributos (numéricos o categóricos) que se compararán para realizar el clustering.

A este archivo se le aplica un filtro, que transforma los rangos de fechas en unidades temporales (intervalos) según los parámetros especificados (cantidad de intervalos, crono, etc.). Este filtro combina o separa los registros según sea necesario, cómo se indicó en la sección anterior.

Clustering temporal

K-D-Median

Los algoritmos de clustering de particionamiento se caracterizan porque miden distancias. La distancia ideal a utilizar es la distancia Euclídea , ya que tiene sentido físico. Algunos métodos, como K-means, utilizan la distancia Euclídea al cuadrado para simplificar los cálculos, pero sólo se obtiene un resultado físicamente correcto si la distribución de los individuos es normal o uniforme. Además, es altamente sensible ante la presencia de puntos muy aislados, alterando los resultados.

El método propuesto, k-D-Median, especificado en (Estivill-Castro,V et. al, [6]) utiliza la distancia Euclídea, por lo que se da a los puntos aislados un correcto tratamiento.

La idea de todos los algoritmos de particionamiento es encontrar el punto mínimo de la función

$$FW(C) = \sum_{i=1}^{n} \omega_i . Euclid(s_i, REP[s_j, C])$$

Donde:

- $S= \{s_1,s_2,....,s_n\}$ es un conjunto de n items en un espacio real D-dimensional.
- El peso $\omega_i > 0$ refleja la relevancia de la observación s_i y $Euclid(x, y) = \sqrt{\sum_{i=1}^{D} | x_i - y_i |^2}$
- $C=\{c_1,c_2,....,c_k)\}$ es el conjunto de k centroides de los clusters.
- $REP[s_i,C]$ es el punto en C más cercano s_i

O sea que si buscamos los puntos representativos para el cluster $Cj = \{x_1,x_2,...,x_{nj}\}$ debemos encontrar el mínimo para esta función:

$$FW(x) = \sum_{i=1}^{nj} \omega_i . Euclid(x, x_i) = \sum_{i=1}^{nj} \omega_i . \sqrt{(x - x_i)^T . (x - x_i)}$$

Resumidamente el k-D-Median funciona de la siguiente forma para hallar el centroide de un cluster:

1. Se encuentra la mediana discreta: La mediana continua es el punto que minimiza la distancia Euclídea entre todos los individuos. La mediana discreta (o simplemente mediana) es el punto que está más cercano a la mediana continua.
2. El método más directo para buscar la mediana sería evaluar la función FW en todos los puntos, y determinar en que punto es mínima pero para clusters muy grandes este proceso es ineficiente y se hace imposible de usar.
3. El método propuesto en k-D-Median tiene en cuenta que la función FW es convexa. Entonces, si se obtiene para cualquiera de sus individuos una recta normal a la función FW, se garantiza que los puntos cuyo producto escalar con esta recta es negativo estarán fuera del hiperplano tangente a FW en dicho punto y la función evaluada en ellos será mayor que la del punto seleccionado, por lo que pueden ser

excluidos del análisis. En particular si se elige el punto más cercano a la media (centro de masa de la distribución) se garantiza que se filtrará aproximadamente el 50% de los puntos en cada iteración. El método se repite en forma iterativa hasta reducir la población de individuos a 6 o menos. Por último se evalúa la función FW en cada uno de estos puntos y se encuentra el centroide.

Obtención de la recta normal: En la mayoría de los casos la recta normal de un campo escalar (función que va desde $R^n \rightarrow R^1$, como es el caso de FW) se obtiene en forma directa a partir de su gradiente.

$$N(fn) = -\nabla(fn)$$

Para la función FW en particular, el gradiente no está definido para los puntos del cluster por lo que la solución no es tan directa. Sin embargo se puede definir una función llamada gradiente extendido de la siguiente forma:

$$\nabla_E FW(x) = \begin{cases} \nabla \mathrm{FW(x)} & if\ \mathrm{x} \notin \mathrm{C_j} \\ \max\left\{1 - \dfrac{1}{\left\|\nabla FW_{\neg m}(x)\right\|}, 0\right\}.\nabla FW_{\neg m}(x_m) & if\ \mathrm{x} \in \mathrm{C_j} \end{cases}$$

Se demuestra en (Estivill-Castro,V et.al,[6]) que el gradiente extendido $\nabla_E(FW)$ es normal al hiperplano tangente a FW para cualquier punto x. Esta propiedad permite utilizarlo para filtrar los individuos que están de un lado u otro del mismo.

Atributos Categóricos en K-D-Median

Si se utilizan atributos categóricos la función FW pasa a estar definida como:

$$FW(C) = \sum_{i=1}^{n} \omega i. \sqrt{\sum_{j=AtNum} (x_j - xi_j)^2 + \sum_{k=AtCateg} (1 \text{ si } \mathrm{x_j} \neq \mathrm{x_{ij}}, 0 \text{ si } \mathrm{x_j} = \mathrm{x_{ij}})}$$

Esta función deja de ser convexa, por lo que el algoritmo KDMedian resulta imposible de utilizar. Lo que se propone en este trabajo (que no está especificado en el trabajo de Estivill-Castro) es reemplazar los atributos categóricos por un conjunto de tuplas numéricas, de forma tal que la distancia entre dichas tuplas sea 1 si son distintas o 0 si son iguales (esta es la forma en que se tratan los atributos categóricos).

Siendo n los posibles valores de un atributo categórico, se generan n tuplas de dimensión (n-1), de la siguiente forma:

$$\begin{pmatrix} A_1 \\ A_2 \\ A_3 \\ A_4 \\ \dots \\ A_n \end{pmatrix} = \begin{pmatrix} 1/2 & 0 & 0 & \dots & 0 & 0 \\ -1/2 & 0 & 0 & \dots & 0 & 0 \\ 0 & \sqrt{1-a_{11}{}^2} & 0 & \dots & 0 & 0 \\ 0 & \dfrac{2a_{32}{}^2-1}{2a_{32}} & \sqrt{a_{32}{}^2-a_{42}{}^2} & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & 0 \\ 0 & a_{(n-1)2} & a_{(n-1)(n-3)} & \dots & \dfrac{2\left[a_{32}{}^2-\sum(a_{42}{}^2..a_{(n-1)(n-3)}{}^2)\right]-1}{2x_{(n-1)(n-2)}} & \sqrt{a_{(n-1)(n-2)}{}^2-x_{n(n-2)}{}^2} \end{pmatrix}$$

Esto significa que, por ejemplo, si un atributo tiene 3 valores posibles, cada vez que en el archivo de entrada figure el valor 1, se reemplazará dicho valor por el vector $A_1 = (1/2,0)$, el valor 2 se reemplazará por el vector $A_2 = (-1/2,0)$, y el tercer valor se reemplazará por el vector $A_3 = (0, \sqrt{1-1/2^2})$.

La particularidad de los vectores así definidos es que la distancia entre dos de ellos es siempre 1. Cada valor categórico entonces se reemplaza por uno de estos vectores, resultando así un conjunto numérico de X cuya función de FW (puramente numérica) evaluada en cada punto será igual a la función de FW (numérica y categórica) del dominio original. De esta manera se puede obtener la mediana discreta por el método K-D-Median propuesto, y aplicar el resultado al dominio original de valores.

Algoritmo k-d-Median Temporal

Este algoritmo ejecuta los siguientes pasos:

1) Se reemplazan los atributos categóricos por atributos numéricos según la fórmula anterior y se normalizan dichos valores.
2) Se dividen las instancias (o registros) del archivo de entrada, según su unidad temporal (ya pasaron por el filtro principal)
3) Se elige entre los individuos de cada unidad temporal (UT), tantos individuos como clusters se desee obtener. Estos individuos serán los centroides originales.
4) Se calcula la distancia Euclídea de cada individuo respecto de los centroides. Dicho individuo se asigna al cluster cuyo centroide esté más cercano (mínima distancia).
5) Se toman los individuos de cada cluster, en cada UT, y se aplica el algoritmo K-D-Median para obtener su nuevo centroide (mínimo de la función FW)
6) Con los nuevos centroides se repiten los pasos 4 y 5 hasta que ningún individuo cambie de cluster o los cambios sean recurrentes (por ej, un individuo que oscila entre 2 clusters)
7) En este punto, se tiene la distribución final de cada cluster en cada UT. Ahora se determina la relación entre los clusters entre distintos tiempos, según se explica en el siguiente apartado.
8) Por último, se genera un archivo con la distribución de los clusters obtenidos en cada UT, sus centroides, e información adicional para calcular las estadísticas.

Relación entre clusters de distintos tiempos

Como se pudo observar, el algoritmo de clustering temporal aplica el algoritmo convencional para cada intervalo de tiempo. De esto se obtiene una distribución de clusters en cada intervalo de tiempo, pero lo que no se sabe es si el cluster denominado “cluster 1” en el primer

intervalo es el mismo “cluster 1” en el segundo intervalo, porque el algoritmo da el número de orden de cada cluster en forma totalmente aleatoria.

Por lo tanto, para saber cómo se llamó a cada cluster en los distintos intervalos de tiempo, hay que seguir la trayectoria del mismo.

Para hacer esto, primero se cuentan cuántos individuos pasaron de cada cluster en el primer intervalo a cada cluster en el segundo intervalo. Luego se computan todas las combinaciones posibles de pasar de un cluster a otro al cambiar de tiempo. Un cluster es la evolución de otro en un momento anterior si resulta de la combinación con el número máximo de individuos.

Análisis de Resultados

La aplicación implementada en Matlab analiza los resultados obtenidos a partir del archivo generado en la aplicación de clustering temporal. La misma posee las siguientes opciones:

- Cluster 2 D : Solo se habilita si el archivo de entrada tiene 2 atributos. Se muestra gráficamente la representación de todos los clusters en cada UT.
- Cluster 2 D Tiempo : Solo se habilita si el archivo de entrada tiene 2 atributos. Se muestra gráficamente la evolución de cada cluster a través de las distintas UT.
- Cluster 3 D : Solo se habilita si el archivo de entrada tiene 3 atributos. Se muestra gráficamente la representación de todos los clusters en cada UT.
- Cluster 3 D Tiempo : Solo se habilita si el archivo de entrada tiene 3 atributos. Se muestra gráficamente la evolución de cada cluster a través de las distintas UT.
- Análisis por Atributo: esta opción está habilitada siempre. Muestra atributo por atributo, la evolución del mismo a través del tiempo, mediante un gráfico de líneas, y a su vez compara el mismo atributo en los distintos clusters.
- Estadísticas: Este es un gráfico de barras, que muestra cuántos individuos cambiaron de cluster en el pasaje de un tiempo al siguiente. Cluster Inicial muestra la cantidad de individuos para cada cluster en un determinado tiempo, y Cluster Final indica la cantidad de individuos que tienen esos clusters en el tiempo siguiente.
- Individuos Perdidos: Muestra en un gráfico de barras, los individuos que pertenecían a algún cluster en un tiempo dado, y no pertenecían a ninguno en el tiempo siguiente.

Cada pantalla permite editar los gráficos, agregar textos o etiquetas, ampliar o reducir las imágenes, rotar las figuras e imprimir o guardar los resultados.

5.Experimentación

A continuación se detallan los juegos de datos utilizados para probar el algoritmo de Clustering Temporal:

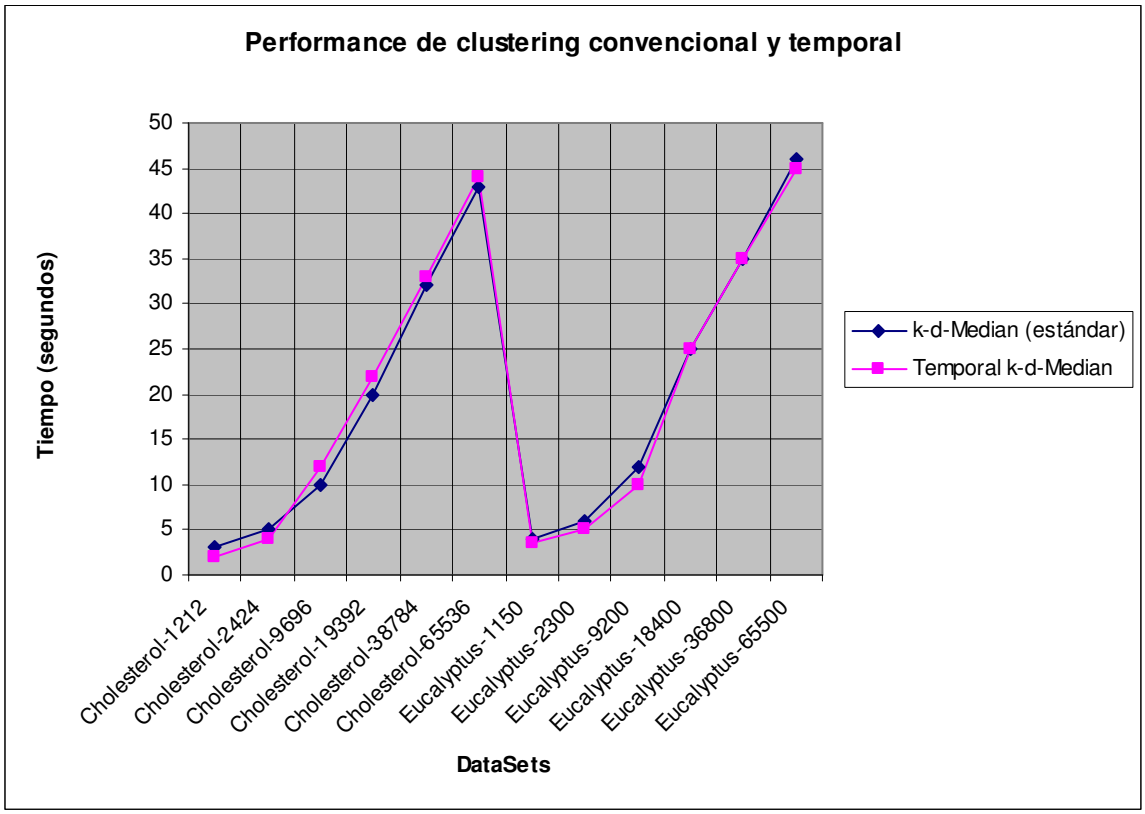
Dataset	Instancias	Atributos	At. Nominales	At. Numéricos	Valores Desconocidos	Unidades Temporales
Cholesterol-1212	1212	16	9	5	si	4
Cholesterol-2424	2424	16	9	5	si	4
Cholesterol-9696	9696	16	9	5	Si	4
Cholesterol-19392	19392	16	9	5	si	4
Cholesterol-38784	37784	16	9	5	si	4
Cholesterol-65536	65536	16	9	5	si	4
Eucalyptus-1150	1150	22	6	16	Si	5
Eucalyptus-2300	2300	22	6	16	Si	5
Eucalyptus-9200	9200	22	6	16	Si	5

Eucalyptus-18400	18400	22	6	16	Si	5
Eucalyptus-36800	36800	22	6	16	Si	5
Eucalyptus-65500	65500	22	6	16	Si	5

Los resultados que se describen a continuación, se obtuvieron luego de ejecutar el sistema WEKA en una PC con procesador Pentium III 733 Mhz, con 256 Mb de memoria RAM.

Las pruebas consistieron en aplicar el algoritmo Temporal_k-D-Median, a los juegos de datos y observar el tiempo transcurrido hasta la finalización del mismo. El algoritmo se ejecutó para formar distintas cantidades de clusters. En particular, se crearon 2,3,4,5,6 y 7 clusters, y los tiempos del algoritmo no variaron considerablemente entre un número de clusters y otro, por lo que a continuación se transcribe el promedio de los tiempos obtenidos. Al utilizar distintas cantidades de individuos, sí se observó una variación de la duración del procesamiento.

También se aplicó el algoritmo k-D-Median (algoritmo no temporal) a cada una de las unidades temporales, para observar las ventajas de ejecutar el algoritmo temporal en forma completa. Por un lado, la suma de los tiempos si se ejecuta por separado, es similar o incluso mayor al tiempo que se demora en ejecutar el algoritmo temporal, pero lo más importante, con el Clustering Temporal se puede obtener la relación de los clusters a través de los distintos tiempos, y se pueden conocer estadísticas o trayectorias de los individuos.



6.Conclusiones y Trabajo Futuro

Conclusiones

El método de Clustering Temporal propuesto utiliza los atributos temporales de una base de datos para relacionar los clusters en los distintos momentos de tiempo. Esto nos permite observar la trayectoria de los objetos, ver cómo varían los clusters durante el transcurso del tiempo, y obtener distintas estadísticas sobre el movimiento de clusters y objetos. La aplicación de nuestro algoritmo a

datos temporales brinda información muy difícil de obtener aplicando sólo un algoritmo de clustering convencional.

Trabajo Futuro

a) Adaptación del algoritmo para clustering de grandes bases de datos temporales.
b) Desarrollo de herramientas de visualización que faciliten el análisis de los resultados obtenidos.

7.Bibliografía

[1] Chen,C. & Kong,J. & Zaniolo,C., Design and Implementation of a Temporal Extension of SQL. 19th International Conference on Data Engineering, 5-8 Marzo de 2003, Bangalore, India (ICDE 2003)

[2] Estivill-Castro,V. Computational Geometry Provides Techniques for Approximately Solving the p-Median Problem, 5 Enero de 2001.

[3] Estivill-Castro,V. Convex Group Clustering of Large Geo-referenced Data Sets. The Pacifc Institute for the Mathematical Sciences, The University of British Columbia, Canada, Vancouver, Canada, 1999.

[4] Estivill-Castro,V. & Houle,M., Robust Distance-Based Clustering with Application to Spatial Data Mining, Algorithmica, 30, 216-242, Springer-Verlag, 2001.

[5] Estivill-Castro,V. & Murray,A. , Hybrid Optimization for Clustering in Data Mining, Pacific Rim International Conference on Artificial Intelligence, pp 424-434, 2000.

[6] Estivill-Castro,V. & Yang,J. A Fast and Robust General Purpose Clustering Algorithm, Pacific Rim International Conference on Artificial Intelligence, pp 208-218, 2000.

[7] Grabmeier,G. & Rudolph,A. Techniques of Cluster Algorithms in Data Mining, Data Mining and Knowledge Discover, 6, 303-360, 2002

[8] Han,J. & Kamber,M. & Tung,A.K.H. Spatial Clustering Methods in Data Mining: A Survey, H. J. Miller and J. Han (eds.), Geographic Data Mining and Knowledge Discovery, NY: Taylor and Francis, 2001.

[9] Huang, Z.,A Fast Clustering Algorithm to Cluster Very Large Categorical Data Sets in Data Mining, Research Issues on Data Mining and Knowledge Discovery, pp 0-, 1997.

[10] Ian, H. & Witten,I. & Frank,E. Data Mining: Practical machine learning tools with Java implementations, Morgan Kaufmann, San Francisco, 2000.

[11] Jensen, C.S. & Snodgrass, R.T., Semantics of Time-Varying Information, 2000, Information Systems, Vol. 21, No. 4, pp. 311-352, 1996.

[12] Jensen, C.S. & Soo, M.D. & Snodgrass, R.T., Unifying Temporal Data Models, Information Systems, v.19 n.7, p.513-547, Oct. 1994.

[13] Nanni,M. Distances for Spatio-Temporal Clustering, Proceedings of Sistemi Evoluti per Basi di Dati (SEBD'02), Junio 2002.

[14] Nanni,M. Clustering methods for spatio-temporal data. PhD Thesis. Dipartimento di Informatica, Università di Pisa, 2002.

[15] Ng, R. T. & Han, J. ,Efficient and Effective Clustering Methods for Spatial Data Mining, 20th International Conference on Very Large Data Bases, September 12--15, 1994, Santiago, Chile proceedings, Morgan Kaufmann Publishers, pp 144-155, 1994.

[16] Roddick,J.F. & Hornsby,K. & Spiliopoulou,M. Yet Another Bibliography of Temporal, Spatial and Spatio-Temporal Data Mining Research, Proc. SIGKDD Temporal Data Mining Workshop, San Francisco, CA. Unnikrishnan, K. P. and Uthurusamy, R., Eds., ACM. 167-175, 2001.

[17] Roddick,J.F. & Lees,B. Paradigms For Spatial and Spatio-Temporal Data Mining, H.G. Miller and J. Han (eds), Geographic Data Mining and Knowledge Discovery. London: Taylor & Francis, 2001