UNIVERSIDAD DE SEVILLA

DEPARTAMENTO DE

LENGUAJES Y SISTEMAS INFORMÁTICOS

Índice para la Comparación Cualitativa de Series Temporales

Trabajo presentado por

D. Francisco Javier Cuberos García-Baquero

para la obtención del grado de doctor, dirigido por

Dr. Juan Antonio Ortega Ramirez

y **Dr. Rafael Martinez Gasca.**

Sevilla, Abril de 2005

Yo, D. Francisco Javier Cuberos García-Baquero con D.N.I. nº 28.486.648K, Licenciado en Informática por la Universidad de Sevilla,

DECLARO BAJO JURAMENTO

ser el autor de la tesis doctoral titulada

Índice para la Comparación Cualitativa de Series Temporales

Que presento en la Universidad de Sevilla para optar al grado de Doctor en Informática.

Sevilla, Abril de 2.005

Fdo. Francisco Javier Cuberos García-Baquero

COMISIÓN DE DOCTORADO

Sevilla, 27 de Junio de 2005
N/Ref.: Negociado de Tesis EL/CAR
Asunto: Enviando Tesis Doctoral Leída

ILMO. SR. DIRECTOR DE LA BIBLIOTECA
DE LA E.T.S. DE INGENIERÍA
INFORMÁTICA
UNIVERSIDAD DE SEVILLA

Adjunto le remito ejemplares de Tesis Doctorales leídas en Departamentos vinculados a esa Facultad a fin de que pasen a formar parte de fondos bibliográficos de consulta de ese Centro.

AUTORES DE LAS TESIS LEÍDAS

- CHÁVEZ GONZÁLEZ, ANTONIA M.
- CUBEROS GARCÍA-BAQUERO, FRANCISCO JAVIER

LA JEFA DE SECCIÓN DE DOCTORADO

Fdo.: Yolanda Díaz Rolando.

D. Juan Antonio Ortega Ramirez y D. Rafael Martínez Gasca, Titulares de universidad del Departamento de Lenguajes y Sistemas Informáticos,

HACEN CONSTAR

que D. Francisco Javier Cuberos García-Baquero, Licenciado en Informática por la Universidad de Sevilla, ha realizado bajo nuestra supervisión el trabajo de investigación titulado:

Índice para la Comparación Cualitativa de Series Temporales

Una vez revisado, autorizo el comienzo de los trámites para su presentación como Tesis Doctoral al tribunal que ha de juzgarlo.

Sevilla, Abril de 2.005

Fdo. Juan Antonio Ortega Ramirez
Titular de Universidad
Dpto. de Lenguajes y Sistemas Informáticos

Fdo. Rafael Martínez Gasca
Titular de Universidad
Dpto. de Lenguajes y Sistemas Informáticos

A Mary, Fran y Miguel,
por haber soportado mis ausencias.

Agradecimientos

En primer lugar a mis directores, Juan Antonio Ortega y Rafael Martínez, por su tutela y comentarios. Especialmente a Juan Antonio por su perseverancia, constancia y ánimo sin los cuales ni siquiera habría comenzado esta tarea.

A Luis González y Paco Velasco por su colaboración sincera y generosa, sus ideas brillantes, elegantes planteamientos y su discusión inagotable.

A Radio Televisión de Andalucía por las facilidades que me ha proporcionado para la asistencia a congresos y actos relacionados con mis tareas de investigación.

Para finalizar, a mis padres, que me proporcionaron siempre todo lo que necesité a cambio de su sacrifio y gracias a ello he podido llegar hasta aquí. Espero que estén tan orgullosos de mí como yo les estoy agradecido.

ÍNDICE DE CONTENIDO

3

Comparación de Series

4

Índice Cualitativo de Similitud - QSI

5
Etiquetados. El problema de la Discretización

6

Kernel entre literales

7

Metodologia

8

Aplicaciones

9

Conclusiones y trabajo futuro

A

DFT

B
Lenguaje de Definición de Formas (SDL)

C
CAIM

D
Software desarrollado

E
Publicaciones

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

CAPÍTULO 1

INTRODUCCIÓN

Las series temporales son conjuntos de datos complejos de una gran importancia. Aparecen en aplicaciones científicas, financieras, biológicas, estadísticas,..., y como ejemplos de éstas se incluyen índices de precios de acciones, volúmenes de ventas de un producto, datos en telecomunicaciones, señales médicas unidimensionales o sucesiones de medidas medioambientales. Esta pequeña enumeración demuestra que en cualquier campo de aplicación se utilizan series temporales.

Si bien el concepto de serie temporal, como un conjunto de valores para los que tiene relevancia el momento temporal en que cada uno de ellos ha sido registrado, es muy simple y todo el mundo puede recordar múltiples ejemplos prácticos, el tratamiento automatizado de estos datos presenta innumerables problemas.

Este proyecto de tesis se centra en el problema de la comparación de series con la intención de proporcionar mecanismos que, haciendo uso de características cualitativas, permitan identificar el grado de parecido o similitud entre series.

1.1. Generalidades

El estudio de los sistemas que evolucionan en el tiempo es un área de investigación muy actual y cómo puede entenderse fácilmente, la evolución de cualquier variable de uno de estos sistemas genera de forma directa una serie temporal.

Una serie temporal contiene, en su versión más simple, una secuencia de números reales, cada número representando el valor de una magnitud en un instante de tiempo.

Normalmente, estas series son almacenadas en bases de datos y es necesario desarrollar algoritmos para su análisis con la intención de poder extraer conocimiento de la información almacenada.

El primer problema para el tratamiento de esa información viene dado por la dimensión de las bases de datos generadas que presentan tamaños que hacen complicado un procesamiento eficiente.

Por otro lado puede considerarse que una serie temporal es un objeto complejo almacenado en la base de datos, siendo necesaria la definición de algoritmos que permita operar con estos objetos complejos de una forma imposible de abarcar con los algoritmos clásicos diseñados para bases de datos relacionales.

Dentro de esta problemática de la manipulación de los datos almacenados el hecho de cuantificar el grado de similitud o disimilitud entre objetos es un tema de gran trascendencia para muchas aplicaciones de minería de datos, aprendizaje automático,....

Sin embargo, para estos objetos complejos ni la definición de qué se entiende por similitud es en ningún modo evidente y única.

Una posibilidad poco utilizada hasta el momento es explotar la información obtenida desde un punto de vista cualitativo. La idea radica en abstraer la información puramente numérica por medio de la definición de características cualitativas que la representen.

Los objetivos y las tareas realizadas a lo largo de la presente tesis para aportar soluciones a algunos de estos problemas se desglosan en el siguiente apartado.

1.2. Objetivos de la tesis

El objetivo fundamental que ha orientado la realización de los trabajos que se incluyen en esta memoria ha sido la definición de una metodología que de una forma automática y desde una perspectiva cualitativa pueda aplicar los métodos de comparación de series temporales, para tareas de clasificación, de forma satisfactoria sobre el mayor número posible de conjuntos de datos.

El desarrollo de este objetivo nos ha llevado a aportar nuevas ideas sobre el estudio cualitativo de series temporales.

El abordar el problema de la clasificación de series temporales desde una visión cualitativa permite un cambio del dominio del problema que tiene dos consecuencias interesantes:

- Posibilita la utilización novedosa de un grupo de algoritmos, muy conocidos, a la comparación de series. Un ejemplo son los algoritmos de comparación de cadenas.

- Acerca al usuario la información disponible mediante una abstracción de la misma, permitiendo un mayor grado de comprensión. Es más fácil para una persona entender valores cualitativos que cuantitativos; ejemplo: "si la serie crece mucho durante cinco periodos y se mantiene estable durante los diez siguientes es de tipo A"es más simple que si se escribe "si la serie toma valores en el intervalo [2,5] durante cinco periodos y se encuentra en el intervalo [0,0.5] en los diez siguientes es de tipo A".

Reconocida la importancia de realizar la comparación de series de forma cualitativa el primer objetivo de este proyecto de tesis ha sido el definir un mecanismo

que permita la transformación de los datos originales en conjuntos de informaciones cualitativas.

Para ello se ha desarrollado un algoritmo de discretización que permite convertir la serie temporal en una cadena de símbolos como resultado de asignar una letra a cada valor numérico de la serie de tal forma que se pierda la menor información posible, que se minimice el número de símbolos diferentes a utilizar y que este mecanismo será robusto frente al ruido.

Para conseguir esta discretización se considera un conjunto de intervalos de clase y se asigna una etiqueta a cada valor de la serie en función del intervalo de clase en que se encuentre. Es muy importante destacar que el conjunto de etiquetas utilizadas, al proceder de una partición de intervalos disjuntos, proporciona una ordenación de estas según un determinado orden de magnitud (escala ordinal).

Una vez que las series han sido traducidas se plantea la aplicación de algoritmos definidos para cadenas de caracteres sobre dichas series.

Analizados los resultados obtenidos con los algoritmos conocidos y los inconvenientes que presentan se aborda la tarea de la definición de una nueva medida de similitud basada en el concepto de núcleo.

Con esta medida se presenta un concepto concreto de similitud entre series de igual longitud.

Finalmente todos estos diferentes trabajos se integran de una manera natural en la metodología que se propone y se implementa una herramienta.

1.3. Estructura de la tesis

La memoria del trabajo desarrollado en la tesis se organiza como sigue.

El presente capítulo sirve de introducción y descripción general de los objetivos que se intentan alcanzar.

En el capítulo 2 se presentará una visión general de la propuesta de investigación que se ha realizado, se describe el problema planteado y se hace la descripción de las técnicas usadas para resolverlo.

A continuación en el capítulo 3 se realiza la descripción de las técnicas que existen en la literatura para solucionar el problema de la comparación de series temporales desde múltiples perspectivas.

La primera técnica desarrollada para la comparación cualitativa de series temporales se presenta en el capítulo 4, donde también se discuten algunas posibles variaciones y cómo estás afectan a la sensibilidad que el método presenta a la aparición del ruido.

Los problemas inherentes a la técnica anterior provocan una particular atención al problema de discretización de variables continuas en intervalos. Los algoritmos publicados en la literatura, su discusión y comparación con nuestra propuesta se incluyen en el capítulo 5

Por la intención de aprehender la mayor cantidad de información en el proceso de comparación se llega al desarrollo de un kernel entre literales que se define en el capítulo 6 junto con sus implicaciones generales.

El capítulo 7 muestra el compendio de todo lo anterior que se plasma en una metodología que permite abordar de una forma automática la comparación de series con un alto grado de identificación correcta en problemas de clasificación.

Finalmente se incluyen los capítulos donde se muestran los resultados de las aplicaciones de la metodología sobre diferentes conjuntos de datos así como el resumen de conclusiones del trabajo realizado y las líneas que permanecen abiertas para el futuro.

Una colección de apéndices, la referencia de los trabajos publicados en el proceso de investigación y las referencias bibliográficas utilizadas completan el presente proyecto de tesis.

CAPÍTULO 2

PLANTEAMIENTO DEL PROBLEMA Y SOLUCIONES

En este capítulo se realiza una presentación general de la investigación realizada, detallando de manera especial el problema planteado, antecedentes y motivación, así como la solución que se propone.

2.1. Antecedentes

Como ya se indicó en el capítulo anterior es posible encontrar series temporales en casi cualquier campo de la ciencia y su almacenamiento produce grandes bases de datos.

En nuestro caso estamos interesados en bases de datos obtenidas como resultado del estudio de sistemas que evolucionan en el tiempo.

En anteriores trabajos, como [OGT99], se ha estudiado el comportamiento de modelos con restricciones, en especial de modelos semicualitativos, aquellos cuya descripción viene dada simultáneamente por valores cuantitativos y descriptores

cualitativos de su estado. El mecanismo para automatizar el análisis y el estudio de estos sistemas se basa en la realización de un modelo semicualitativo con restricciones que se transforma en una familia de modelos cuantitativos que son simulados generándose una base de datos. Esta base de datos puede utilizarse para realizar consultas sobre las propiedades de la evolución del sistema y para aprendizaje automático, una vez etiquetada en función de algún criterio particular.

Desde el punto de vista de los valores tomados por las magnitudes que son representadas por cada serie podemos hablar de series discretas, aquellas formadas por un conjunto limitado de valores, y serie continuas, donde los valores sólo están limitados por las características propias de la magnitud dentro del dominio del problema.

Si además de los valores concretos que conforman la serie es relevante la información temporal del momento en que se ha producido cada uno de ellos, estaremos hablando entonces de *series temporales*, centrándose nuestra investigación en bases de datos compuestas de series temporales continuas.

Una vez que se dispone de bases de datos como las anteriormente descritas es natural plantear su procesamiento con la intención de obtener nuevos conocimientos. Esta tarea de extracción de conocimiento desde grandes bases de datos implica múltiples pasos desde la manipulación de los datos y su recuperación a procesos de inferencia matemática y estadística, búsqueda y razonamiento.

Declarada la búsqueda, que podemos generalizar como sistema de consulta, como uno de los pasos básicos para la extracción de conocimiento, por la necesidad básica de poder localizar en una base de datos elementos que coincidan con un patrón determinado o dilucidar si dos elementos cualquiera responden a un mismo patrón, se hace evidente la necesidad de utilizar un concepto de similitud.

Considerando que la similitud venga determinada por una función de distancia entre las series y siendo ϵ un valor de distancia fijado por el usuario, podemos catalogar las diferentes consultas para la manipulación de una base de datos de series temporales en tres tipos:

- Consulta de rango: donde se desean localizar todas las series que se distancian en menos de un ϵ concreto de una serie determinada, lo que permite encontrar las series similares a una dada admitiéndose la tolerancia indicada.

- Vecino más próximo: Dada una serie ha de localizarse aquella que está mas cerca a ella que ninguna otra, obteniéndose la serie de la base de datos que es más similar a la utilizada como patrón de búsqueda.

- Pares cercanos: Con este tipo consulta se desea encontrar todos los pares de series presentes en la base de datos que están a una distancia menor que un ϵ indicado.

Desde la perspectiva de la comparación realizada, la búsqueda de similaridad puede clasificarse en:

- coincidencia completa, que busca series similares a la serie de consulta teniendo en cuenta toda su longitud,

- localización de subsecuencias, que intenta descubrir subsecuencias, contenidas en las series de datos, que sean similares a la secuencia, de longitud arbitraria, presentada como consulta.

2.2. Similitud

Centrada la discusión sobre la importancia de la similitud, el mayor problema que se encuentra al intentar proporcionar un criterio para evaluar la similitud entre dos series temporales está en la definición del propio concepto de *similitud*. Así, dependiendo de la aplicación, del concepto que tenga el propio usuario del sistema, de la respuesta buscada,... , se pueden manejar diferentes definiciones de la similitud.

Se puede considerar que las siguientes transformaciones como desplazamientos con respecto al eje Y, desplazamientos sobre el eje X (retrasos o adelantos), cambios

de escala o variaciones en la longitud de la serie, afectan o no a la similitud entre dos series.

Además ha de considerarse que estas transformaciones pueden aplicarse de forma simultánea. En la figura 2.1 se presenta una serie y estas posibles variaciones aplicadas de forma individual.

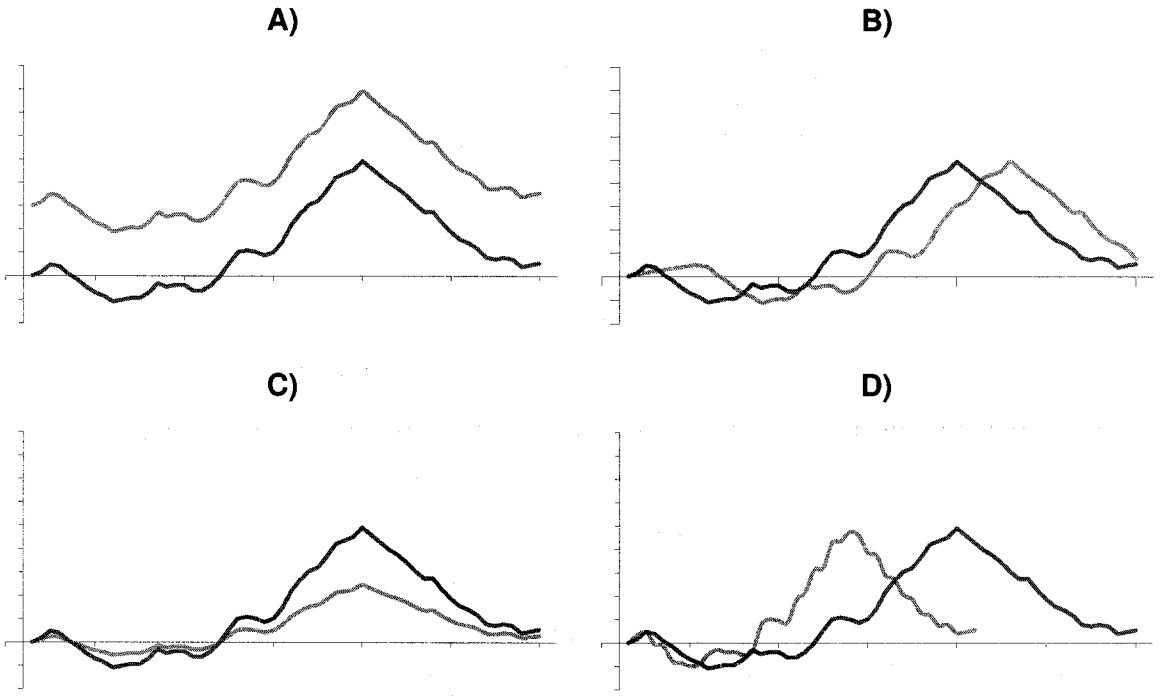


Figura 2.1: Comparación gráfica de diferentes transformaciones de una serie. La serie original, con trazo negro, junto a: A) desplazamiento sobre el eje Y, B) retraso inicial, C) cambio de escala y D) variación de la longitud.

También es posible, en algunas aplicaciones, considerar sólo alguna sección concreta de las series que se comparan. En algunas aplicaciones de diagnosis dos series se tienen por similares por el hecho de presentar un comportamiento parecido en el estacionario independientemente del régimen que hayan tenido para llegar a él, mientras que en otras se considera sólo el transitorio.

A la luz de la flexibilidad que tiene el concepto subjetivo de similitud se antoja

imposible la definición de un método que abarque todas las posibles percepciones de una forma intrínseca. Es por tanto fundamental que cualquier método que se desarrolle defina claramente el tipo de similitud que considera, aunque éste pueda ser alterado o precisado por el usuario del mismo.

Nuestra aproximación considera que dos series son similares aunque una de ellas se obtenga mediante un desplazamiento sobre el eje Y y/o un cambio de escala con respecto de la otra. Por otro lado, las variaciones de longitud y los posibles desplazamientos sobre el eje X conducen a menores índices de similitud.

2.3. Soluciones

Hasta ahora todas las soluciones presentadas se han desarrollado fundamentalmente desde el punto de vista numérico, por medio de la definición de distancias y/o mecanismos de indexación de características de las series. Una revisión de estas técnicas se incluye en el Capítulo 3 de la presente tesis.

Nuestra solución se presenta como una metodología que incluye un proceso de autodescubrimiento y otro de aplicación, como se observa en la figura 7.1.

Se tiene una base de datos o conjunto de aprendizaje y se quiere entrenar el sistema para que sea capaz de identificar la clase de las nuevas series que se le presenten. Se entiende que cada serie incluirá una etiqueta que indicará la clase a la que pertenece, ya sea por razón de su origen o de un procedimiento de etiquetado previo por parte de un experto.

De este conjunto de series se deben seleccionar los componentes que formarán tanto el subconjunto de aprendizaje como el de verificación. Esta división se realizará aleatoriamente.

El paso posterior consiste en transformar las series en cadenas de símbolos. Como no conocemos ningún método que garantice las mejores prestaciones para cualquier conjunto de datos se realiza una comparación entre varios, siendo uno de ellos un

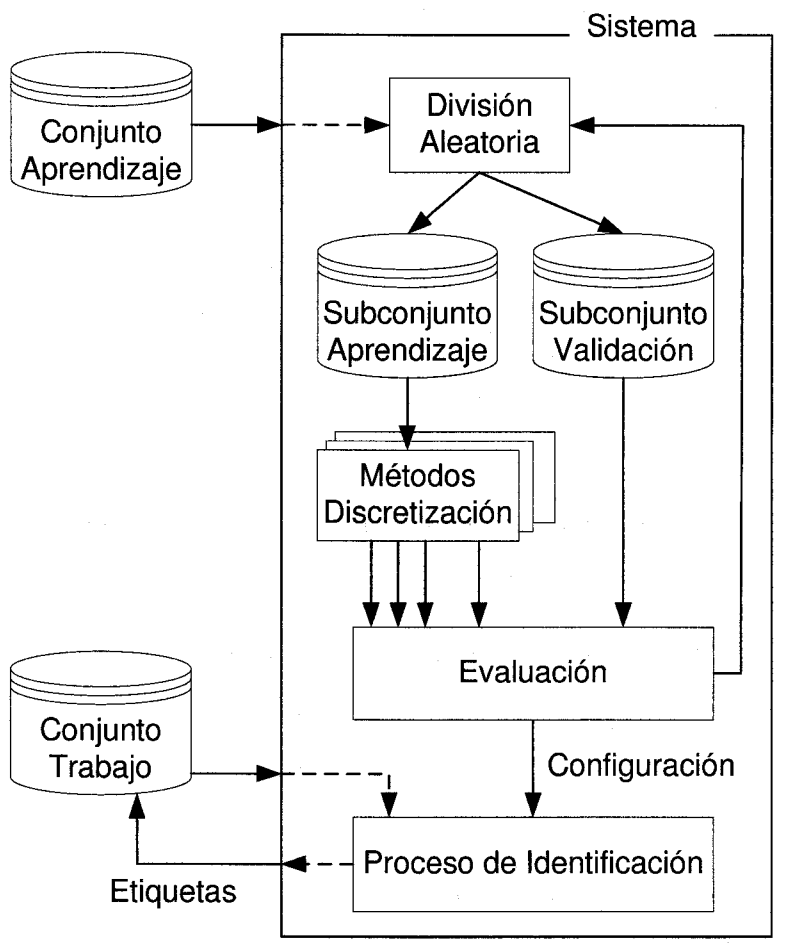


Figura 2.2: Metodología propuesta

sistema de discretización que se presenta en este proyecto de tesis y que presenta muy buenas características.

Tras la aplicación de los diferentes métodos utilizaremos los conjuntos de marcas proporcionadas para convertir las series en un conjunto de secuencias de símbolos.

El sistema valorará la bondad de los métodos de discretización por medio de contabilizar las identificaciones correctas del conjunto de validación utilizando la la aplicación de una distancia intervalar [GAVO04] sobre las series de símbolos.

Como la división inicial de la base de datos en los conjuntos de aprendizaje y verificación se realiza por medio de un muestreo estratificado simple, es necesario repetir todo el procedimiento varias veces para poder promediar los resultados con

respecto a las posibles divisiones.

Una vez finalizado, el sistema determinará el mejor método para identificar las series que componen la base de datos original, quedando preparado para realizar identificar las nuevas series que se le vayan proponiendo.

Las aportaciones originales que se incluyen en esta metodología se pueden resumir en:

- Un nuevo enfoque. El comparar las series temporales desde una perspectiva cualitativa es una aportación original de este proyecto de tesis.
- Cambio del dominio del problema. El punto anterior provoca la necesidad de hacer una conversión de las series que posibilita aplicar algoritmos definidos para otros objetos como las cadenas de símbolos.
- Un nuevo sistema de discretización. Ante la problemática que presentan los diferentes sistemas de discretización definidos hasta la fecha se presenta un nuevo algoritmo que se caracteriza por necesitar un número muy reducido de símbolos, comparado con los algoritmos existentes, obteniendo unos resultados comparables en tareas de clasificación.
- Nuevas medidas de similitud. Se han desarrollado durante la investigación dos nuevas medidas. La primera debe considerarse un índice de similitud y ha sido definido partiendo de la similitud entre cadenas de símbolos. La segunda, por el contrario, está definida para cadenas de igual longitud, se basa en el concepto de núcleo y verifica las condiciones para ser una distancia.

CAPÍTULO 3

COMPARACIÓN DE SERIES

La búsqueda de un sistema que permita identificar el grado de parecido entre dos o mas series ha sido abordado por múltiples autores en los últimos años. Estas múltiples aproximaciones se han realizado desde tan diferentes enfoques, en lo que se refiere a la metodología como a los antecedentes teóricos en que se apoyan, que es muy difícil realizar una clasificación que los reúna todos en un orden lineal. Muchos intentos fusionan técnicas o fundamentos de varias aproximaciones lo que permite su clasificación en varias secciones.

Con éstos antecedentes se presenta una revisión de los trabajos previos que se agruparán por medio de los diferentes enfoques existentes con el apoyo de un gráfico que permita una visión completa que no es posible de realizar en un texto.

Las técnicas que se revisan en esta sección se centran en la creación de índices, en la definición de medidas de similitud y la proposición de distancias que permitan realizar la comparación entre series.

No se incluyen las técnicas provenientes del campo del aprendizaje automático como modelos ocultos de Markov o redes de neuronas, pudiéndose encontrar una revisión de estos métodos en [Die04].

3.1. Comparación de Series Temporales

Ya ha quedado descrito, ver capítulos 1 y 2, el amplio rango de situaciones y ámbitos en que se producen series temporales y para los cuales es necesaria una medida de similitud de series.

Como ejemplo de esta diversidad puede citarse el trabajo de [GW02] que se ocupa del caso de consultas continuas que son las que se realizan sobre series a las que se están añadiendo elementos de forma indefinida.

En la bibliografía se han presentado diferentes aproximaciones para comparar series temporales. La mayoría proponen la creación de un índice con un pequeño conjunto de valores extraídos de los datos originales.

Los índices presentados en estos trabajos proporcionan una comparación eficiente de las series temporales consiguiendo una velocidad de comparación muy superior a la obtenida con la comparación de todos los datos originales.

Para la generación del índice se ha optado mayoritariamente por dos enfoques diferentes: el realizar una transformación de los valores de la serie temporal a un espacio de menor dimensión y la de reducir directamente los datos originales de la serie temporal seleccionando un subconjunto de ellos.

Siguiendo estas dos importantes líneas de investigación dedicaremos secciones diferentes a los trabajos que se encuadran en cada una de ellas. Además se incluirán otras aproximaciones y prestaremos una especial atención a los algoritmos sobre los que definiremos nuestra aproximación.

Así, en las siguientes secciones incorporaremos una revisión en profundidad aproximaciones que pueden encontrarse, de una forma visual, en el gráfico 3.1 en sentido horario comenzando por la esquina superior izquierda:

- Los trabajos basados en la generación de una firma que represente las propiedades fundamentales de cada serie. Especialmente los que se basan en espectros,

que utilizan la distribución de las frecuencias de cada serie como base para una representación característica.

- Las aproximaciones que proponen una reducción de datos, donde se realiza una selección de los datos originales de forma directa o por medio de agrupaciones de los mismos

- Los sistemas de máximos y mínimos, que pretenden asimilar el mecanismo psicológico humano de comparación.

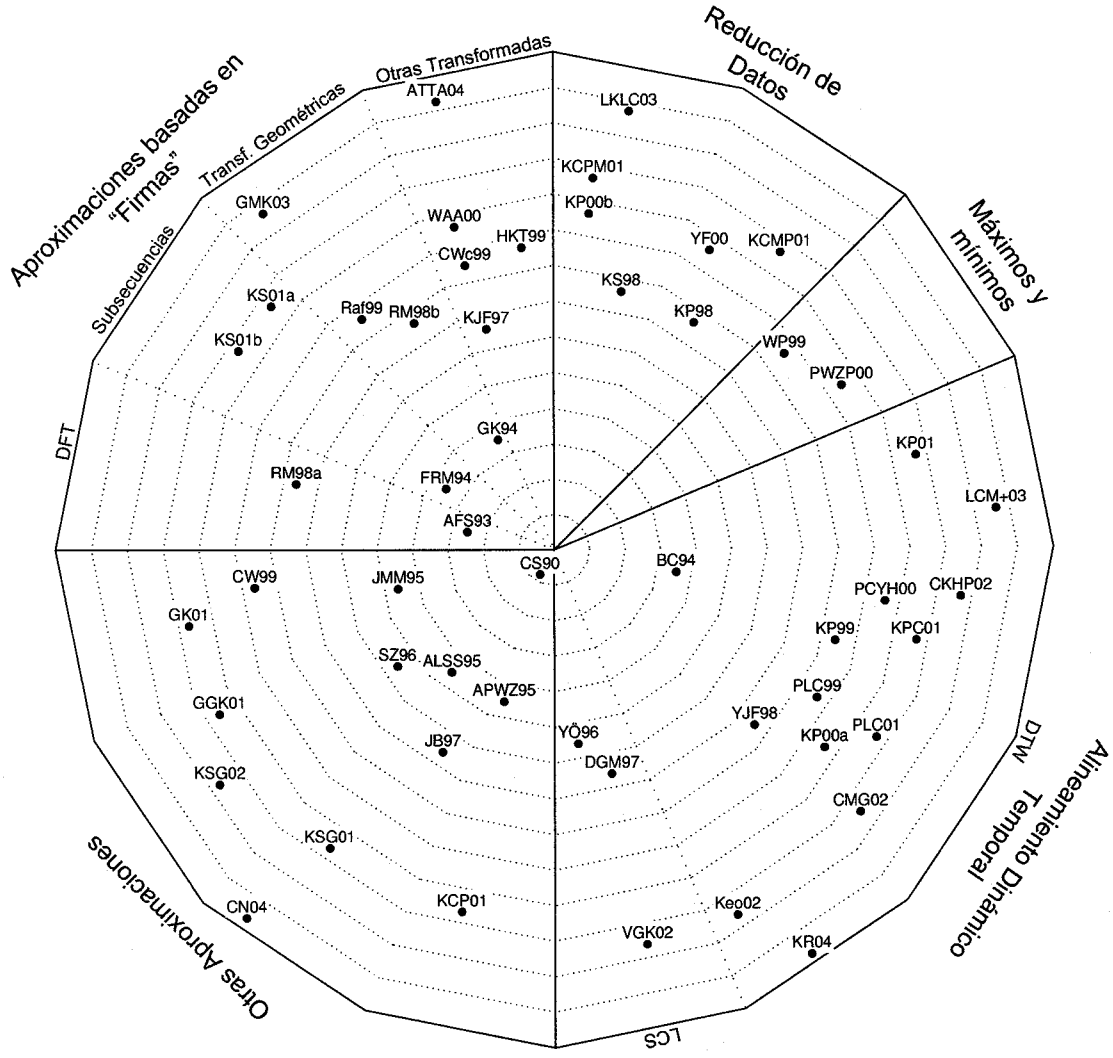


Figura 3.1: Representación de trabajos previos agrupados por enfoque

- Los basados en el *Alineamiento Temporal* que permite que la similitud incorpore desplazamientos temporales y engloba a:

  el algoritmo *Alineamiento Dinámico del Tiempo*, o *DTW*, y los trabajos que lo aplican, amplían y modifican, junto con

  el algoritmo *Subsecuencia Común Máxima*, o *LCS*, originario del mundo de la comparación de cadenas de caracteres, y sus aplicaciones a las series temporales.

- Y finalmente repasaremos todos aquellos trabajos que por sus características no son encuadrables en los grupos anteriores.

Esta clasificación difiere de la realizada por otros autores, como [HKT99], que encuadran las aproximaciones de reducción de datos y los sistemas de cotas como un subconjunto de los métodos basados en firmas. Nosotros los vamos a considerar como aproximaciones diferentes partiendo de la idea de que en estos trabajos siempre se opera con los datos originales de la serie y no se realiza conversión alguna, como la del dominio del problema.

3.2. Aproximaciones basadas en "firmas"

Definición Una serie temporal puede considerarse como un vector o un punto en un espacio de n dimensiones.

Partiendo de la anterior definición la idea natural para enfrentarse al problema de la comparación de series temporales es la de considerar cada serie como un punto n-dimensional y utilizar cualquiera de los métodos existentes para indexar objetos multidimensionales. De esta forma tendríamos un sistema de localización de las series incluidas en el índice de una forma eficiente.

El primer contratiempo que se presenta viene determinado por los problemas de implementación que tienen los métodos conocidos cuando el número de dimensiones sobrepasa las 15, lo que representa un grave inconveniente puesto que es habitual

hallar series con tamaños superiores a los mil elementos. Este comportamiento se conoce como problema de dimensionalidad o *"dimensionality curse"*.

Para evitar esta limitación se han presentado una variedad de métodos que reemplazan las series originales por puntos en espacios de un reducido número de dimensiones, permitiendo así la aplicación de los sistemas de indexación referidos.

La idea fundamental de estos métodos es la transformación de la serie original en otra que, representando sus características fundamentales, sea de un tamaño mucho menor. Esta transformación puede entenderse como la generación de una firma o vector característico de cada serie. Junto a esto es también necesaria la definición de una distancia entre las series transformadas, que esté relacionada con la distancia entre las series originales de una forma conocida.

Siendo $\vec{x} = \langle x_1, \ldots, x_{n-1} \rangle$ e $\vec{y} = \langle y_1, \ldots, y_{n-1} \rangle$ dos series de longitud n y $f(\vec{x}) = \tilde{x}, f(\vec{y}) = \tilde{y}$ las firmas características respectivas, entonces la relación entre las funciones de distancia d_f y d es

$$d_f(\tilde{x}, \tilde{y}) \leq d(\vec{x}, \vec{y})$$

Esto nos lleva a que en el mejor de los casos las funciones de distancia proporcionarán el mismo valor, y en el resto de casos la distancia entre las transformadas será una cota inferior de la existente entre las series originales.

La mayoría de los trabajos que utilizan esta aproximación la aplican utilizando las características espectrales de las series. Vamos a enumerar de forma detallada estos trabajos que se basan en las características ondulatorias de las series temporales para decidir sobre su similitud, utilizando una descripción de las frecuencias que presentan las series temporales en sustitución de las propias series. Como estas descripciones son de un tamaño reducido se obtiene de forma automática una reducción de la dimensionalidad de los objetos con que se trabaja.

Esta aproximación de las series por una especificación de sus frecuencias se basa en la idea de que las series, entendidas como ondas, se caracterizan por tener una gran concentración de energía en unos pocos armónicos. Al estar hablando de series

de datos sería más correcto referirnos a que se produce una concentración de la información.

Los diferentes trabajos difieren en la forma de hacer esta transformación o en el espacio sobre el que se realiza, pudiéndose agrupar en cuatro bloques fundamentales:

1. Aquellos que utilizan la Transformada Discreta de Fourier,
2. los que usando dicha transformada aplican conceptos geométricos para definir la similitud,
3. trabajos que permiten la localización de subsecuencias dentro de las series
4. y finalmente los que proponen la aplicación de otras transformadas.

Una de las transformadas más usada por los trabajos de este grupo es la Transformada Discreta de Fourier (DFT), debido a ello se ha incluido en el apéndice A una explicación pormenorizada de la misma.

3.2.1. Trabajos con *DFT*.

En este punto vamos a comentar los trabajos que proponen la utilización de una transformada, concretamente la DFT, como único método para abordar el problema de la comparación de series temporales.

La primera propuesta en este sentido se realizó en [AFS93].

Partiendo de la comprobación de que un gran número de series temporales tienen un número pequeño de coeficientes de Fourier con amplitudes grandes se propone su utilización como representación de la serie original. Este hecho indica que en pocos coeficientes se acumula la mayor parte de la energía de la serie y por tanto su utilización en lugar de la serie original genera un error pequeño. Nos referimos a la familia de secuencias que presentan un espectro de amplitud de $O(1/f)$.

Este trabajo define como función de distancia de dos secuencias la raíz de las diferencias al cuadrado, o lo que es lo mismo, la raíz cuadrada de la diferencia de

energía.

$$D(\vec{x},\vec{y}) \equiv \sqrt{\sum_{t=0}^{n-1} |x_t - y_t|^2} \equiv \sqrt{E(\vec{x}-\vec{y})}$$

Si la distancia entre dos secuencias es menor que un valor ε definido por el usuario, entonces las dos secuencias se consideran similares.

La aplicación de esta técnica para un conjunto dado de datos implica la realización de los siguientes pasos:

1. Obtener los primeros f_c coeficientes de la DFT para cada una de las secuencias, donde f_c se denomina *frecuencia de corte.*

2. Construir un índice usando dichos coeficientes. Como cada secuencia se ha convertido en un punto de un espacio de $2f_c$-dimensiones el índice se construye utilizando árboles-$R*$ (ver [BKSS90]) y se denomina *índice-F.*

3. Para una consulta de rango, se obtienen los f_c-coeficientes de la DFT de la secuencia de consulta. Se utiliza el *índice-F* para seleccionar un conjunto de secuencias que están a una distancia menor que ε de la secuencia de consulta.

4. Como sólo se utilizan unos pocos coeficientes en la realización del índice, los errores de aproximación a las secuencias originales provocan falsos positivos en el conjunto obtenido. El conjunto de respuesta correcto es obtenido después de un postprocesado en el que la distancia entre dos series se calcula en el dominio del tiempo y sólo aquellas cuya distancia a la secuencia de consulta sea menor que ε son aceptadas.

La completitud de este método se basa en que el *índice-F*, aunque produzca falsos positivos, no genera ningún falso negativo.

Abundando en la misma dirección que el trabajo [AFS93], en [RM98a] se propone la utilización de los k primeros coeficientes DFT y de los k últimos.

El trabajo realiza varias observaciones importantes sobre las características de las transformadas de las series temporales cuyos elementos sean todos valores reales.

La primera es que los coeficientes de una secuencia de longitud n satisfacen que

$$X_{n-f} = X_f^* \quad para\ f = 1, \ldots, n$$

donde el asterisco denota el complejo conjugado. Esto significa que la transformada de Fourier de toda secuencia es simétrica con respecto su centro.

En las secuencias que presentan un espectro de energía de la forma $O(F^{-2b})$, para $b \geq 0{,}5$, los coeficientes de la transformada son importantes al principio y al final. Esto nos lleva a que si el cálculo de la distancia se basa exclusivamente en los primeros coeficientes, la información almacenada en los últimos coeficientes, aún siendo considerable, es despreciada.

Y la última observación que se presenta en dicho trabajo es que los primeros $(n+1)/2$ coeficientes DFT de una secuencia contienen toda la información de la secuencia. Se presenta la posibilidad de almacenar exclusivamente $n/2$ coeficientes aunque los cálculos se realicen con todos los coeficientes.

La metodología se evalúa con datos reales y sintéticos obteniéndose una aceleración del tiempo de búsqueda en un factor superior a dos.

El trabajo [AFS93] debe considerarse como pionero en la comparación de series temporales presentando un enfoque novedoso que abrió un nuevo camino en la investigación, además de presentar un sistema sencillo y de altas prestaciones.

Pero en su valoración global hay que hacer notar algunos inconvenientes. La suposición de que todas las series en la base de datos y las secuencias de consulta son de igual longitud se antoja como una restricción poco práctica.

Otro problema es que el concepto de similitud está establecido a priori, con la consiguiente pérdida de ajuste al dominio del problema. Además no se considera el concepto de localización temporal.

Aunque muchos de los trabajos que se exponen a continuación siguen haciendo uso de la DFT los englobaremos en otros epígrafes por una mayor claridad en función de su orientación o su idea principal.

3.2.2. Búsqueda de Subsecuencias.

Un aspecto no considerado en los trabajos comentados hasta ahora es la localización de subsecuencias dentro de las series existentes en la base de datos. Para tratar de completar este vacío [FRM94] presenta una ampliación al trabajo de [AFS93] para permitir dicha búsqueda.

El mecanismo presentado consiste en dividir las series en regiones mediante el desplazamiento de una ventana de longitud fija, determinada por la longitud de las consultas soportadas. A cada región se le calculan sus coeficientes DFT que se tratan como puntos de un espacio multidimensional.

Los puntos característicos de desplazamientos cercanos de la ventana son similares, ya que comparten un número elevado de puntos. Así en lugar de almacenar en el índice todos los puntos obtenidos por el desplazamiento de la ventana, se propone agrupar los puntos próximos y representarlos por medio del rectángulo mínimo envolvente o MBR (*Minimum Bounding Rectangle*) que los contiene. Posteriormente, estos MBR son almacenados usando árboles-$R*$.

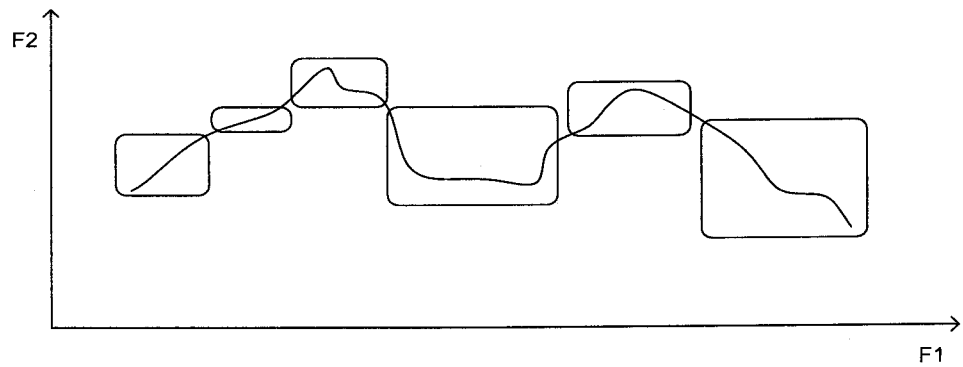


Figura 3.2: Ejemplo de agrupación

En la figura 3.2 se representan los coeficientes DFT en el espacio característico calculados para cada desplazamiento de la ventana sobre una serie determinada. Sólo se han utilizado dos primeros coeficientes $F1$ y $F2$ como representantes de cada subsecuencia. Además se presenta una agrupación en MBR.

Todos los MBR generados de todas las secuencias existentes en la base de datos se indexan por medio de un árbol-$R*$. Para construir dicho árbol los MBR son agrupados recursivamente en otros de mayor tamaño que los incluyen. Este nuevo árbol generado al que se añade información referente a la secuencia de la que se obtuvo el MBR y de las posiciones que representa, conforma el índice propuesto en este trabajo y que se denomina *índice-ST*.

Para la correcta agrupación de los puntos característicos se propone un algoritmo adaptativo basado en la estimación de accesos a disco presentado en [KF93]. El algoritmo va agregando puntos al MBR actual hasta que el coste marginal crece, momento en que se crea un nuevo MBR y se continúa el procedimiento. La aplicación de este método a la generación del *índice-ST* se denomina método *Indexación Adaptativa* (o *I-Adaptative*), teniendo cada nodo hoja del árbol-$R*$ un número variable de puntos en el espacio característico.

Las consultas de rango se implementan por medio de:

1. realizar el cálculo del punto correspondiente a la secuencia dada,
2. la obtención de los MBR que interseccionan con la región indicada por ese punto y un radio ε y,
3. finalmente examinar las subsecuencias a que corresponde cada MBR para descartar falsos positivos.

También se presentan métodos para la localización de prefijos y de búsquedas múltiples que eliminan la restricción de consultas de longitud preestablecida. La búsqueda de prefijos realiza una búsqueda en la base de datos utilizando un prefijo, de longitud fija, de la secuencia de consulta. La búsqueda múltiple divide la secuencia de consulta en subsecuencias no solapadas (de longitud preestablecida) y realiza la búsqueda por cada una de estas subsecuencias. Un inconveniente importante de estos mecanismos es que son muy costosos.

En [FRM94] hay dos inconvenientes prácticos que limitan su utilización:

1. La similitud usada es la distancia euclídea. Siendo esta distancia muy sensible al ruido.

2. Los problemas de escalado en amplitud y en traslación no se consideran.

Además ha de tenerse en cuenta que el método de [FRM94] sólo se probó con series obtenidas de las cotizaciones de valores bursátiles por lo que no hay garantías de que el mecanismo funcione con otros tipos de series.

Para aplicar la idea anterior a la búsqueda de subsecuencias de una manera más eficiente, [KS01b] propone la creación de un nuevo índice denominado $MR-Index$. En este caso la longitud de la ventana no es única sino múltiple.

Siendo s la longitud máxima de todas las secuencias en la base de datos, entonces $\exists b/2^b \leq s < 2^{b+1}$, siendo b entero. Sea 2^a el tamaño mínimo de la cadena, siendo a un entero que cumple que $a \leq b$. Para cada secuencia se construyen todos los árboles-$R*$ formados por el conjunto de MBR creados mediante el uso de una ventana de tamaño 2^i donde $a \leq i \leq b$.

El árbol $A_{i,j}$ representa el conjunto de MBR para la secuencia j-ésima obtenidos con una ventana 2^i. El índice se compone del conjunto de árboles de todas las secuencias, creándose una matriz de árboles.

En la tabla 3.1 se representa la estructura de indexación formada por el conjunto de árboles para el conjunto de secuencias $s_1, s_2, \ldots, s_n$.

Para la realización de una consulta de rango, especificada por una secuencia $\vec{x}$ y un radio ε , se particiona $\vec{x}$ en varios trozos con tamaños iguales a los usados en la ventana deslizante. Posteriormente se realizan una serie de consultas incrementando el tamaño de la partición usada. Los resultados de cada consulta permiten afinar el radio de la siguiente consulta.

Como cada subconsulta se realiza a diferente resolución este método se denomina *Método MR* y de forma análoga el índice comentado como *índice MR*. Ante el posible problema del aumento de los requerimientos de almacenamiento se propone

	s_1	s_2	...	s_n
$w = 2^a$	$A_{a,1}$	$A_{a,2}$	...	$A_{a,n}$
$w = 2^{a+1}$	$A_{a+1,1}$	$A_{a+1,2}$	...	$A_{a+1,n}$
	...	...	...	...
$w = 2^b$	$A_{b,1}$	$A_{b,2}$	...	$A_{b,n}$

Tabla 3.1: Estructura de indexación

un mecanismo de compresión del índice que reduce el almacenamiento sin implicar un aumento excesivo del coste de computación.

Este mismo algoritmo es utilizado en el trabajo [KS01a] para búsquedas de similitud en cadenas de caracteres.

3.2.3. Transformaciones Geométricas.

Los trabajos enmarcados en este epígrafe aplican conocimientos provenientes de la geometría a las series temporales. Si consideramos que las series pueden representarse en el espacio como curvas finitas es inmediata la constatación de que es posible su tratamiento desde un punto de vista geométrico.

Desde una perspectiva novedosa en su momento se presenta en [GK94], que posteriormente será ampliado por otros trabajos, el análisis de funciones de transformación que pueden ser aplicadas a series y que permiten la concreción de lo que el usuario considera como similar.

El trabajo presenta una sintaxis y una semántica para consultas de similitud que permiten la coincidencia aproximada, el escalado y el desplazamiento, presentando

un mecanismo de indexación eficiente.

En el campo de las Transformaciones Geométricas [MP65] se considera que las transformaciones de escalado y desplazamiento preservan la forma. Así desde un punto de vista geométrico dos conjuntos son similares si existe un conjunto de transformaciones que convierta uno en otro.

Una transformación similar $T_{a,b}$ se define como una función descrita por dos números reales, a y b, tal que aplicada sobre una secuencia $\vec{x} = \langle x_1, \ldots, x_n \rangle$ proporciona otra $T_{a,b}(\vec{x}) = \langle x_1^t, \ldots, x_n^t \rangle$ donde $x_i^t = a * x_i + b$, asumiéndose que $a > 0$, lo que implica que la simetría con respecto al eje x no se considera similitud.

De esta forma se establece que una secuencia $\vec{x}$ es similar a una secuencia $\vec{y}$ si existe algún par $(a, b) \in [R^+, R]$ tal que $\vec{x} = T_{a,b}(\vec{y})$, es decir, si existe alguna función de transformación que las haga idénticas.

Las transformaciones tienen un factor de escala, indicado por el parámetro a , y un factor de desplazamiento, representado por el parámetro b , presentando estos comportamientos de forma individual o conjunta.

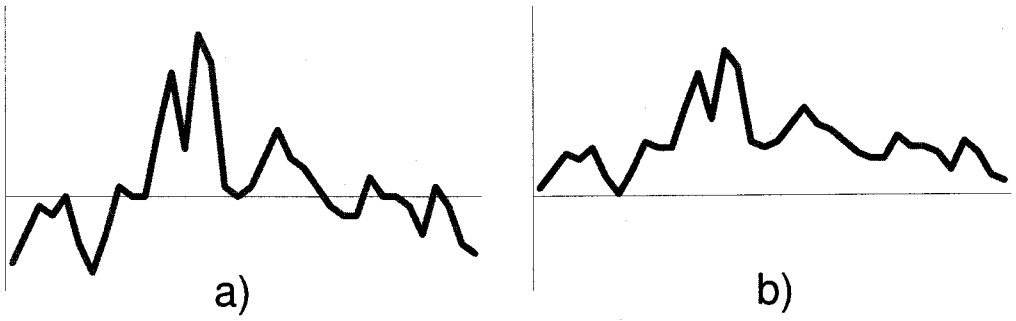


Figura 3.3: Series geométricamente similares

La figura 3.3 muestra dos series geométricamente similares. Los valores de las serie etiquetada como a) son iguales a los de la serie etiquetada como b) si se les aplica un escalado y un desplazamiento positivo en el eje Y.

Otra aportación de este trabajo es definir una secuencia como normal si su media es cero y su desviación de 1. Toda secuencia normal puede considerarse la represen-

tante de la clase de secuencias que pueden obtenerse, a partir de ella, por medio de transformaciones similares y se le denomina forma normal de todos los miembros de la clase. Este proceso de normalización es ampliado a cualquier conjunto de datos en [Gol04].

Finalmente se considera la distancia entre dos secuencias cualquiera como la distancia entre las formas normales de las clases respectivas. Por esto, la distancia entre todos los pares posibles de dos clases de series es la misma.

En este caso se utiliza como métrica la distancia euclídea sobre las series normales, aunque puede utilizarse cualquier otra.

Para permitir un sistema rápido de consulta se especifica un índice que para cada secuencia incluye: una huella de la secuencia formada por los primeros coeficientes DFT, la media y la desviación.

Se permite al usuario la realización de consultas en que especifique su concepto de similitud, por medio de describir el tipo de transformaciones de similitud que diferencian a las secuencias buscadas de la indicada como patrón de búsqueda.

Las transformaciones presentadas se aplican sobre los coeficientes de Fourier que representan cada serie gracias al mantenimiento de la distancia que posee la transformación.

Tras la obtención del conjunto de secuencias que pueden cumplir las condiciones se realiza un filtrado de las falsos positivos.

Una amplia revisión de este trabajo se presenta en [GMK03] incorporando demostraciones, costes de ejecución y una validación experimental.

El trabajo anterior de [GK94] es ampliado en [RM98b] donde se definen una nueva serie de transformaciones lineales como la media móvil, escalado temporal e inversión que permiten al usuario definir el tipo de similitud. Además, se elimina la restricción de reescalados exclusivamente positivos.

En [RM98b] las secuencias originales son mapeadas a un espacio multidimensio-

nal por medio de los primeros coeficientes de Fourier. Con estos puntos multidimensionales se crea un índice basado en árboles-R [Gut84] que se utiliza para acelerar las búsquedas.

Dado un índice, un tipo de transformaciones, un valor de distancia y una serie de consulta el algoritmo se divide en:

1. Transformar al dominio de la frecuencia tanto la serie como la clase de transformaciones usando los primeros coeficientes DFT.

2. Construir un rectángulo de búsqueda para la serie transformada.

3. Para cada nodo del índice aplicarle la trasformación y comprobar si existe solapamiento con el rectángulo de búsqueda.

4. Procesar cada serie candidata para comprobar si la distancia euclídea de sus datos originales a la serie modelo es menor que ε. Si es así agregarla al conjunto de soluciones.

Posteriormente, [Raf99] permite que la similitud sea definida por medio de un conjunto de transformaciones simultáneas dando de esta manera una mayor libertad al usuario. Se considera que las series están normalizadas en el sentido de [GK94], y que se almacena junto con cada serie su media y su desviación típica, presentándose dos nuevas propiedades de las formas normales.

Siendo $\vec{x}$ e $\vec{y}$ las series originales de longitud n, μ y σ representan la media y la desviación típica de una serie respectivamente y $\vec{X}$ e $\vec{Y}$ las series normalizadas, se verifica que:

1. Se minimiza la distancia euclídea con respecto a desplazamientos de escala, i.e. $D(\vec{X} - s_x, \vec{Y} - s_y)$ tiene su mínimo cuando s_x y s_y son respectivamente los valores medios de $\vec{X}$ e $\vec{Y}$.

2. La distancia euclídea entre dos secuencias normalizadas está directamente re-

lacionada con su correlación cruzada[1]

$$D^2(\vec{X},\vec{Y}) = 2(n - 1 - n\rho(\vec{X},\vec{Y}))$$

Este trabajo puede verse como una implementación eficiente de un caso especial del lenguaje presentado por [JMM95].

Junto con esto, se demuestra que es posible la agrupación de transformaciones y la aplicación de un grupo de ellas al índice en un único recorrido, acelerando la búsqueda. Además se analiza el hecho de que la ordenación de las transformaciones según unos determinados criterios puede proporcionar mejoras en la eficiencia de la evaluación.

Aunque los trabajos descritos en el presente apartado proporcionen la gran ventaja de permitir al usuario la definición de su concepto de similitud, presentan graves inconvenientes. Por un lado siguen estando centrados en la comparación de secuencias completas y por otro, y aún más importante, la aplicación de la transformación a todos los nodos es equivalente a la generación de un nuevo índice, lo que representa un gran número de cálculos con gran impacto en el tiempo de ejecución. Esto último hace que su aplicación práctica a bases de datos de gran tamaño sea infructuosa.

3.2.4. Otras transformadas.

Como se comenta al estudiar la Transformada Discreta de Fourier (Apéndice A, son varias las familias de transformaciones que tienen la característica de ortonormalidad y por tanto verifican el teorema de Parseval. Una de estas transformadas es la Transformada de Haar [Haa10], perteneciente a la familia de las Transformadas Wavelet (WT). La Transformada Discretas Wavelet (DWT) o *Discrete Wavelet Transform* son una versión discreta de WT para señales numéricas.

La ventaja de la utilización de DWT es su multiple representación de las señales en diferentes resoluciones. Por otro lado la representación que DWT hace de las

[1] $\rho(\vec{X},\vec{Y}) = \frac{\mu_{\vec{X}*\vec{Y}} - \mu_{\vec{X}}\mu_{\vec{Y}}}{\sigma_{\vec{X}}\sigma_{\vec{Y}}}$

señales incorpora más información que la obtenida con DFT. Mientras DFT extrae los primeros armónicos que representan la forma general de la serie, DWT codifica con un detalle reducido la secuencia original.

Aunque la posibilidad de utilización de DWT para comparar series temporales es apuntada en [KJF97] no es hasta [CWc99] cuando se lleva a la práctica.

En [CWc99] se propone una solución para indexar series basada en la transformada de Haar. Las ventajas del uso de ésta se concretan principalmente en ser más fácilmente calculable, proporcionar mayor información acerca de la serie y mejor escalabilidad frente al *índice-F* que se presentó en [AFS93] . Además se contempla la similitud de series con desplazamiento vertical.

El cálculo de la Transformada de Haar se realiza por medio de la obtención de las series de las medias y las diferencias de cada dos elementos de la serie usada como argumento.

Sea $\vec{x} = \langle x_0, x_1, \ldots, x_{2^n} \rangle$ la serie original, las series de medias $\vec{m}$ y diferencias $\vec{d}$ se definen como

$$\vec{m} = \langle m_1, m_2, \ldots, m_{2^{n-1}} / m_i = (x_{2i-1} + x_{2i})/2 \rangle$$
$$\vec{d} = \langle d_1, d_2, \ldots, d_{2^{n-1}} / d_i = (x_{2i-1} - x_{2i})/2 \rangle$$

Se realiza una aplicación recursiva comenzando por la serie original y usando cada serie de medias como entrada del paso siguiente. Finalmente la transformada está compuesta por el último resultado de las medias, que representa la media global de la serie, y todas las colecciones de diferencias.

Un ejemplo del cálculo de la transformada, $H(\vec{x})$, para la serie $\vec{x} = (6, 4, 3, 5)$ se muestra en la figura 3.4, siendo $H(\vec{x}) = (4, 1, 1, -1)$. Con el valor de resolución 4 la transformada es igual al tamaño de la serie y se toman todos sus valores como entrada del algoritmo.

El método de creación del índice se compone de dos pasos:

1. Seleccionar el modelo de similaridad. El usuario puede elegir entre utilizar la

Resolución	Medias	Diferencias
4	$(6, 4, 2, 4)$	
2	$(5, 3)$	$(1, -1)$
1	(4)	(1)

Figura 3.4: Ejemplo de cálculo de la Transformada de Haar

distancia euclídea o una similitud que soporta el desplazamiento vertical. En el primer caso se aplica la transformada de Haar a las series, mientras en el segundo tras esta aplicación se desprecian los primeros coeficientes ya que no importa la coincidencia de las medias.

2. Se calculan los coeficientes de Haar a las subsecuencias proporcionadas por el desplazamiento de la ventana deslizante. Con los primeros coeficientes se construye un índice sobre un árbol-R.

Un inconveniente de este método es que el número de coeficientes de Haar a considerar se calcula de forma experimental, como un compromiso entre el costo del postprocesado, necesario para el filtrado de las falsos positivos, y la dimensión del índice.

También ha de tenerse presente que en [WAA00] y en [KK02] se demuestra que la utilización de la DWT no reduce el error relativo de coincidencia ni mejora la precisión de las consultas de similitud, como se indica en [CWc99], sino que las ganancias observadas dependen de los datos.

Un caso particular es en el que se concentra el trabajo de [HKT99] que considera las series temporales alineadas; aquellas que tienen el mismo patrón temporal. En este caso la necesidad es localizar características similares que se produzcan en el mismo punto temporal. Para ello propone la utilización de DWT para obtener un conjunto de características.

La aplicación de DWT se presenta en [ATTA04] para obtener información de la estacionalidad y de la tendencia de series de entrenamiento, con el objetivo de

analizar y predecir el comportamiento de series no estacionarias y volátiles.

3.3. Reducción de Datos

Los siguientes trabajos se caracterizan por no manipular de forma conjunta todos los datos originales de cada serie. Al contrario, realizan reducciones seleccionando un subconjunto de los mismos o realizando agrupaciones. Algunos de ellos vuelven a aplicar transformaciones pero ya sobre el subconjunto de agrupaciones resultante.

Uno de los primeros trabajos en utilizar esta aproximación es [KS98]. En él se utiliza una segmentación lineal a trozos de la curva original basada en el algoritmo presentado en [PH7 esta representación.

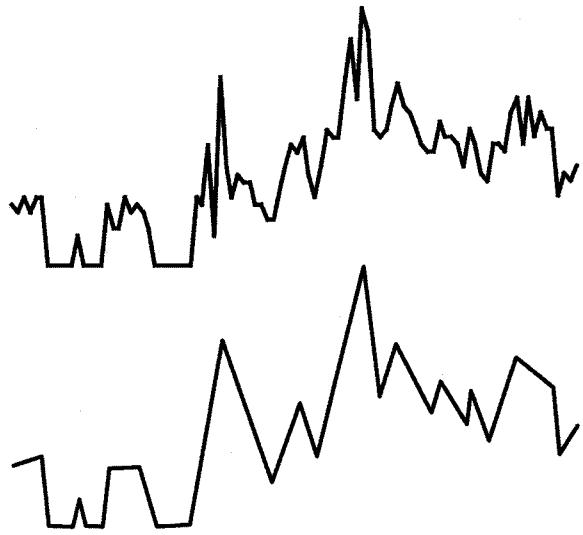


Figura 3.5: Serie temporal y su representación lineal a trozos

Para determinar el número de segmentos con que debe representarse una curva dada, se realiza un algoritmo de abajo a arriba que en cada paso une los dos segmentos con menor aportación al error cuadrático de la aproximación a la curva original. Este algoritmo se detiene cuando la nueva unión provoca un incremento importante de dicho error.

Sobre esa representación lineal se utiliza un modelo probabilístico basado en ca-

racterísticas locales que componen la forma global de la secuencia. Las características locales tienen cierto grado de deformación y la forma global un grado de elasticidad que le permite reducciones a lo largo del tiempo y de la amplitud de la señal. Estos grados de deformación y de elasticidad están relacionados con distribuciones probabilísticas conocidas a priori.

El mecanismo de búsqueda para cada característica se realiza colocando el segmento característico en cada uno de los puntos unión de los diferentes segmentos en que se ha convertido la curva original y calculando la distancia local. Se considera como distancia de deformación media la desviación típica de las distancias de las proyecciones verticales de los puntos de unión. Se obtienen así K distancias, donde K es el numero de segmentos de la secuencia segmentada. El mismo proceso se hace para cada característica y finalmente se busca la mejor coincidencia para la consulta completa.

[KP98] sigue utilizando el mismo método de segmentación pero se modifica la métrica de distancia. Los segmentos obtenidos se identifican por medio de sus puntos inicial y final, además de un valor que sirve de ponderación relativa del segmento con respecto a la serie. Para una secuencia A su versión segmentada $\mathbf{A}$, en k segmentos, es una tupla de 5 vectores de longitud k.

$$\mathbf{A} \equiv \{AXL, AXR, AYL, AYR, AW\}$$

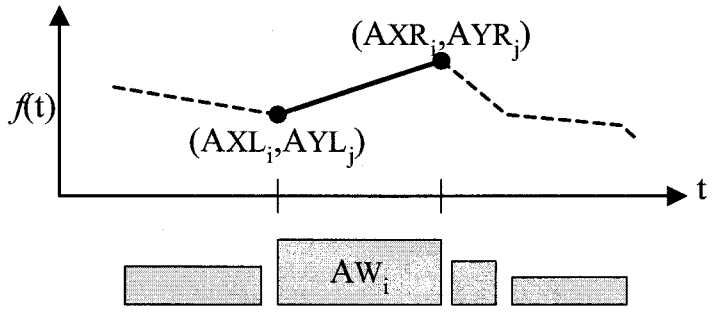


Figura 3.6: Representación de la serie con segmentos lineales y pesos

El segmento i-ésimo está representado por la línea definida entre los puntos

(AXL_i, AYL_i) y (AXR_i, AYR_i) , y AW_i que representa el peso del segmento, como puede apreciarse en la figura 3.6.

Inicialmente todos los pesos son iguales a uno y en caso de que alguno cambie se realiza una renormalización de todos los pesos de forma que la suma de los productos de los pesos por las longitudes de sus respectivos segmentos siempre sea igual a la longitud de la secuencia completa.

Los segmentos son utilizados en un nueva función de distancia en lugar del conjunto completo de valores de la serie lo que reduce el cálculo. Dicha función se define, para las series A y B, como

$$D(A,B) = \sum_{i=1}^{k} AW_i * BW_i * |(AYL_i - BYL_i) - (AYR_i - BYR_i)|$$

Esta distancia es insensible a traslaciones, tendencias lineales y discontinuidades.

En [KP00b] y en [YF00] se realiza una reducción directa de la dimensionalidad de la curva con una aproximación constante a trozos. Aunque ambos autores definen un nombre distinto para este método, lo denominaremos como PAA, Aproximación Lineal a Trozos Agregada o *Piecewise Aggregate Approximation*, así como *índice* PAA al sistema de indexación implementado.

Una serie X de longitud n es representada en un espacio de N-dimensiones por un vector $\overline{X} = \overline{x}_1, \overline{x}_2, \ldots, \overline{x}_n$. El elemento $i-$ésimo de $\overline{X}$ se calcula mediante

$$\overline{x}_i = \frac{N}{n} \sum_{j=\frac{n}{N}(i-1)+1}^{\frac{n}{N}i} x_j$$

La técnica PAA se realiza mediante la división de la curva en una serie de N segmentos abarcando cada uno n/N puntos consecutivos de la curva original y sustituyéndolos por el valor medio en cada segmento, transformándose la serie original en un punto de N dimensiones. Podemos ver un ejemplo en la figura 3.7.

Hay que resaltar dos casos especiales; cuando $N = n$ la transformada es idéntica a la original y cuando $N = 1$ la transformada es la media de los valores de la serie.

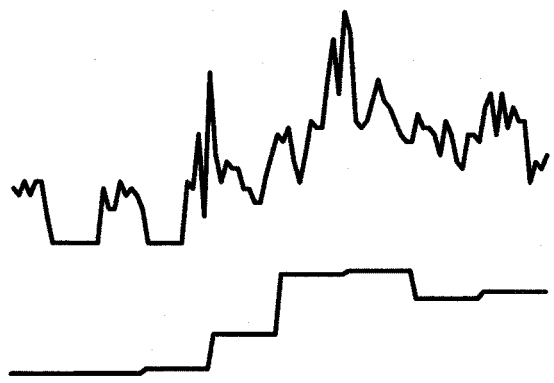


Figura 3.7: Serie temporal y su representación en 8 segmentos constantes

Mientras [KP00b] utiliza la distancia euclídea como base de la nueva distancia que define sobre el espacio de índice, en [YF00] se permite que sea el usuario quién defina el grado de la norma que desee como medida de la similitud.

Aunque el índice está construido para consultas de una longitud igual a las secuencias en la base de datos, se presentan métodos para soportar consultas de cualquier longitud, sin necesidad de modificar el índice.

Se demuestra de forma experimental que la cantidad de series que se manipulan en la fase final de comprobación de falsos positivos es mucho menor que la proporcionada por el índice-F.

Mientras que en estos dos trabajos la división de la serie se realiza en trozos idénticos en [KCMP01] se utiliza un algoritmo adaptativo utilizando segmentos constantes al que se denomina APCA, Aproximación Adaptativa Lineal Constante o *AdaptativePiecewiseConstantApproximation*. En este caso los segmentos siguen siendo constantes pero su longitud es variable para permitir una mejor representación de la serie original. En la figura 3.8 se presentan una serie temporal y una representación adaptativa constante.

La idea de esta aproximación es reducir el error de reconstitución de la serie original mientras que la descripción sigue siendo compacta.

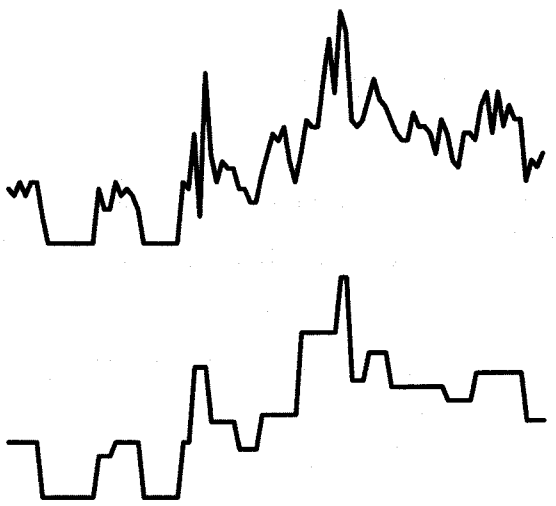


Figura 3.8: Serie temporal y su representación adaptativa constante.

Al ser los segmentos constantes pero de longitud variable, la serie original se convierte en una serie de pares de puntos que especifican el valor de los puntos de ese segmento y el extremo derecho de cada segmento. Considerando esta secuencia de pares la descripción de la serie se convierte en un punto en un espacio multi-dimensional sobre el que se aplican los ya comentados MBR y árboles-$R*$ para realizar un índice, lo que anteriormente había sido calificado como no posible por la irregularidad de la aproximación [YF00].

Sobre este índice el trabajo [KCMP01] define dos medidas de distancia. Una que es cota inferior de la distancia euclídea y que permite búsquedas exactas y la siguiente, que no siendo cota inferior, aproxima en gran medida a la distancia euclídea y que puede encontrar de forma muy rápida vecinos, de forma aproximada. Además el índice permite la utilización de cualquier norma L_p.

En [KCPM01] se realiza una comparación de este método frente a las técnicas que utilizan DFT, SVD y DWT.

Una variación a este trabajo se realiza en [LKLC03] donde una vez que la serie se ha transformado en su versión PAA se realiza una nueva transformación para

obtener una representación simbólica. Esta transformación se realiza utilizando los límites de las regiones equiprobables de la distribución gausiana.

De esta manera se consigue con esto se consigue la reducción del número de valores a manejar y la defición de distancias sobre los valores simbólicos que son cotas inferiores de las distancias sobre los datos originales.

Una mejora realizada a esta idea se presenta en [LLM04] donde el índice agrupa conjuntos de series permitiendo estimar cotas inferiores de la distancia. Esta agrupación se realiza sin crear solapamientos por las regiones que representan a cada serie y se produce así una mejora del índice presentado en $APCA$, que los autores cuantifican en un factor de 3. Como contrapartida este método no es válido para la búsqueda de subsecuencias.

3.4. Máximos y mínimos

Intentando reflejar el comportamiento humano en la búsqueda de similitudes, como han demostrado diversas investigaciones psicológicas y de comportamiento, se propone en [PWZP00] que la detección de patrones similares se realice por medio de marcas (landmarks).

En lugar de trabajar con los datos originales se propone una representación basada en marcas de los mismos. Las marcas se clasifican en órdenes en función de la n-ésima derivada que es cero. Aunque la utilización de marcas de mayores órdenes proporcionan mejores representaciones, también implican mayores árboles de índice por lo que es necesario un compromiso que estará afectado por el dominio del problema. En este trabajo se utilizan únicamente marcas de primer orden, es decir, mínimos y máximos locales.

Como el número de marcas de primer orden puede ser muy elevado en series de alta volatilidad, se presenta también un método de reducción del número de marcas denominado $MDPP$, o Minimal Distance/Percentage Principle.

Implementable en tiempo lineal, el algoritmo elimina aquellos pares de marcas que distan menos de una distancia indicada H para el eje x y un porcentaje señalado P para el eje y. Al contrario que otros filtros, como las medias móviles, este método no suaviza picos ni valles.

En la figura 3.9 se muestra un ejemplo de este método. Dados dos puntos de la serie representados por sus coordenadas (x_1, y_1) y (x_2, y_2) y como se verifica que

$$h < H \quad \text{y} \quad \frac{2v}{y_1 + y_2} < P$$

ambos puntos son eliminados.

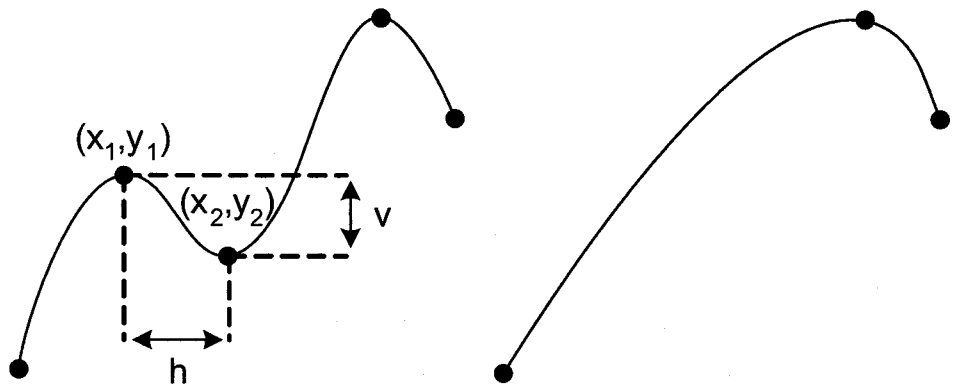


Figura 3.9: Ejemplo de aplicación de $MDPP$ a una curva

Como las marcas son secuenciales se reduce el problema de la creación del índice a un problema de indexación de cadenas. Para la generación del índice se utiliza el árbol-S^2 , presentado en [WP99], que es una combinación de los árboles-X [BKK96] y los árboles de sufijo [Ste94]. Esta nueva estructura permite la localización de subsecuencias.

La gran ventaja de este modelo es que la distancia entre las secuencias es invariante frente a 6 transformaciones: desplazamiento, escalado de amplitud uniforme, escalado temporal uniforme, biescalado uniforme, alineamiento temporal y escalado en amplitud no uniforme.

Aunque se presenta un lenguaje de consulta flexible, el paso de extracción de

marcas requiere la selección de varios parámetros que afectan severamente a la calidad del método. Por otra parte el almacenamiento de picos elimina la noción de tiempo; ello es debido a que los picos pueden localizarse aleatoriamente en las secuencias. Esto nos lleva a que el espacio entre dos picos puede ser diferente y esta información no es accesible a las consultas.

De forma similar [FPG03] se realiza una indexación basada en las líneas que unen los máximos y los mínimos de las series. La selección de los valores extremos de la serie se realiza en función del nivel de compresión que es un concepto análogo al anterior $MDPP$.

3.5. Alineamiento temporal

En esta sección se engloban dos algoritmos que tienen la capacidad de lograr la localización del parecido existente entre dos series, en general, aunque los valores presenten variaciones en la escala temporal.

Veamos cada uno de forma particular.

3.5.1. Alinamiento Temporal Dinámico. DTW

Un algoritmo que ha sido utilizado por multitud de aproximaciones al problema de la similitud es el Alinamiento Temporal Dinámico, *DTW* o *Dynamic Time Warping*. Este algoritmo es bien conocido y utilizado en el campo del reconocimiento de voz desde su proposición por [SC78].

Para poder completar la visión de los trabajos que lo usan, vamos a realizar una revisión en profundidad de este algoritmo en la siguiente sección y posteriormente realizaremos un recorrido por las diferentes propuestas.

Descripción de DTW

El algoritmo *DTW* es usado intensamente en el campo del reconocimiento de voz debido a su capacidad de detectar formas similares de ondas que no estén alineadas en el eje temporal. Esta falta de alineamiento induce catastróficos resultados en una comparación con distancia euclídea.

El fundamento de *DTW* está en buscar un conjunto de mapeos ordenados entre los valores de dos series, de forma que se minimice la distancia global o coste de envoltura (warping cost). La idea básica es intentar descubrir que algunos segmentos de una de las series a comparar son muy similares a los de la otra serie sobre los que se han realizado transformaciones de compresión o expansión; es en realidad una búsqueda de variaciones locales en la frecuencia de las series.

Supongamos dos series Q y C, de longitud n y m respectivamente, donde:

$$Q = q_1, q_2, \ldots, q_i, \ldots, q_n$$
$$C = c_1, c_2, \ldots, c_j, \ldots, c_m$$

Para alinear dos secuencias usando *DTW* se construye una matriz $n * m$ donde el elemento (i^{th}, j^{th}) de la matriz contiene la distancia $d(q_i, c_j)$ entre los puntos q_i y c_j (normalmente se usa la distancia euclídea, por tanto $d(q_i, c_j) = (q_i - c_j)^2$). Cada elemento (i, j) de la matriz corresponde al alineamiento entre los puntos q_i y c_j, como se muestra en la figura 3.10.

Un camino de alineamiento W, es un conjunto continuo (en el sentido que se indica posteriormente) de elementos de la matriz que definen un mapeo entre Q y C. El k-ésimo elemento de W se define como $w_k = (i, j)_k$ teniéndose:

$$W = w_1, w_2, \ldots, w_k, \ldots, w_K \quad max(m, n) \leq K < m + n - 1$$

El camino de alineamiento está sujeto a varias restricciones:

1. Condiciones de los extremos: $w_1 = (1, 1)$ y $w_K = (m, n)$, se obliga a que el camino empiece y termine en esquinas diagonalmente opuestas de la matriz.

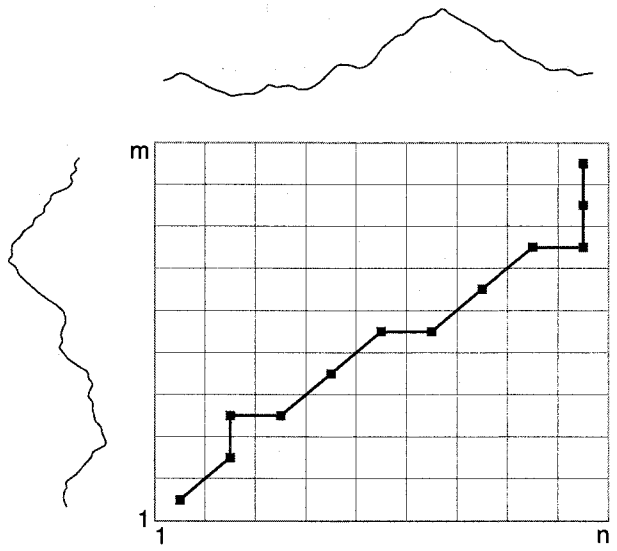


Figura 3.10: Matriz de alineamiento

2. Continuidad: Dado $w_k = (a, b)$ entonces $w_{k-1} = (a', b')$ donde $a - a' \leq 1$ y $b - b' \leq 1$. Esto restringe los saltos permitidos en el camino sólo a celdas adyacentes, incluyendo las adyacentes diagonalmente.

3. Monotonía: Dado $w_k = (a, b)$ entonces $w_k - 1 = (a', b')$ donde $a - a' > 0$ y $b - b' > 0$. Esto fuerza a que los puntos en W estén monótonamente distribuidos en el espacio.

Potencialmente existen muchos caminos de alienamiento que cumplen las condiciones anteriores, pero estamos interesados sólo en el camino que minimiza el coste:

$$DTW(Q, C) = \text{mín} \left\{ \sqrt{\sum_{k=1}^{K} w_k / K} \right.$$

Siendo el parámetro K, presente en el denominador, un mecanismo para compensar el hecho de que los caminos pueden tener diferentes longitudes.

Este camino se puede encontrar de forma eficiente utilizando programación dinámica para evaluar la siguiente fórmula recursiva. Ésta define la distancia $\gamma(i, j)$ como

la distancia $d(q_i, c_j)$, presente en la celda actual, mas el mínimo de la distancia acumulada en los elementos adyacentes:

$$\gamma(i,j) = d(q_i, c_j) + \text{mín}\{\gamma(i-1, j-1), \gamma(i-1, j), \gamma(i, j-1)\}$$

Al camino de alineamiento se le han definido restricciones globales tendentes a acelerar algo el cálculo y, sobre todo, a evitar mapeos patológicos en los que una sección pequeña de una secuencia se mapea en una relativamente grande de la otra.

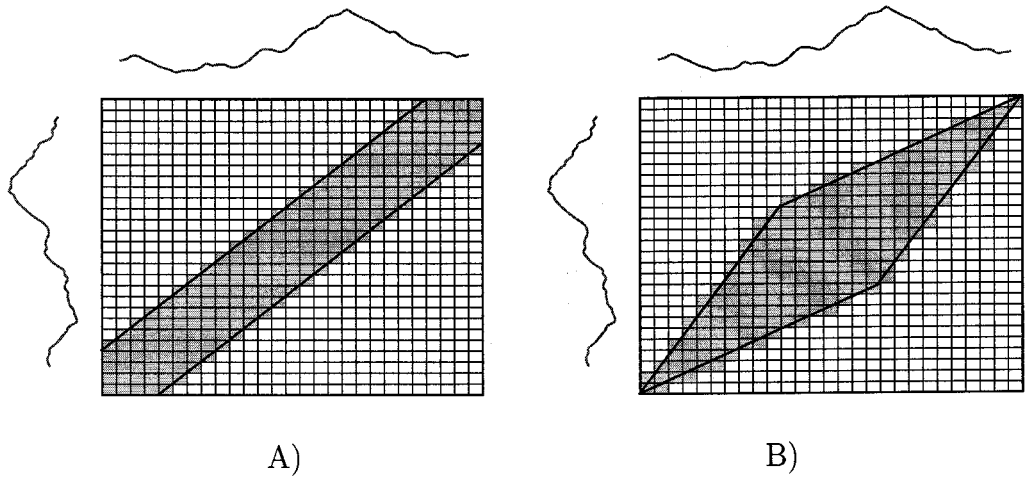


Figura 3.11: Restricciones al posible camino *DTW*: A) Banda Sakoe-Chiba B) Paralelogramo Itakura

Las restricciones globales, o topológicas, más usadas son la banda Sakoe-Chiba, que limita con una banda paralela al camino ideal de alineamiento, y el paralelogramo Itakura, donde se establece un paralelogramo con extremos coincidentes con el camino ideal y con mayor amplitud en la zona central; véanse [SC78] y [RJ93]. Ambas se muestran en la figura 3.11.

Aplicaciones de *DTW*

Aunque la idea de aplicar *DTW* a las series fue introducida en el ámbito de la minería de datos por [BC94], los propios autores reconocían que la complejidad

computacional del algoritmo era un problema que limitaba su aplicación práctica a grandes bases de datos.

Abundando en los incovenientes prácticos de *DTW* en [YJF98] se hacen dos importantes enunciados:

1. Todo método de indexación espacial que asuma, directa o indirectamente, la desigualdad triangular provocará falsos negativos si la función de distancia usada no verifica dicha desigualdad.

2. La distancia proporcionada por *DTW* no verifica la desigualdad triangular.

De esto podemos deducir que el único método para garantizar que no se producen falsos negativos en la aplicación de *DTW* es mediante un escaneado secuencial. Como esto sería impracticable para grandes bases de datos, en este mismo trabajo se propone un método que presenta un compromiso entre una aceleración importante y un número reducido de falsos negativos.

Se propone aplicar el método *FastMap* introducido en [FL95], para generar un índice por medio de realizar un mapeo de las series en puntos de un espacio de menor dimensión. Además, se introduce una función de distancia, D_{lb}, que subestima la distancia del alineamiento temporal.

Siendo $\vec{x} = \langle x_1, \ldots, x_m \rangle$ y $\vec{y} = \langle y_1, \ldots, y_n \rangle$ dos series temporales y representando el recorrido por los intervalos, $R_{\vec{x}} = (min(\vec{x}), man(\vec{x}))$ y $R_{\vec{y}} = (min(\vec{y}), man(\vec{y}))$ la función se define como

$$D_{lb}(\vec{x},\vec{y}) = \begin{cases} \sum_{x_i > max(\vec{y})} |x_i - max(\vec{y})| + \sum_{y_j < min(\vec{x})} |y_j - min(\vec{x})| \\ \qquad si \quad R_{\vec{x}} \quad y \quad R_{\vec{y}} \quad se \quad solapan \\ \\ \sum_{x_i > max(\vec{y})} |x_i - max(\vec{y})| + \sum_{x_i < min(\vec{y})} |x_i - min(\vec{y})| \\ \qquad si \quad R_{\vec{x}} \quad engloba \quad R_{\vec{y}} \\ \\ max\left(\sum_{i=1}^{m} |x_i - max(\vec{y})|, \sum_{j=1}^{n} |y_j - min(\vec{x})|\right) \\ \qquad si \quad R_{\vec{x}} \quad y \quad R_{\vec{y}} \quad son \quad disjuntos \end{cases}$$

Esta distancia está basada en la idea de que todos los puntos de una serie que son mayores que el máximo de la otra contribuyen a la distancia DTW con al menos el cuadrado de la diferencia entre él y el máximo.

Un razonamiento equivalente se aplica a los puntos de una series menores que el mínimo de la otra. La aplicación de ambos mecanismos proporciona un conjunto reducido de series candidatas a las que se les aplica el DTW clásico.

De esta manera se obtiene velocidad y una similitud que soporta variaciones locales de la frecuencia de la señal representada por las series. Por otro lado la aceleración no es muy grande y el método requiere la fijación de una serie de parámetros sin reglas de selección.

Además no es aplicable en multitud de casos ya que no puede evitar falsos negativos.

En los artículos [KP99] y [KP00a] se aplica el algoritmo de alineamiento dinámico del tiempo sobre los datos segmentados. En el primero de estos trabajos se utiliza la segmentación presentada en [KS98]. Se define así un $SDTW$, o $DTW segmentado$ que, manteniendo todas las propiedades de robustez del algoritmo DTW, permite una aceleración muy importante de las comparaciones, al tiempo que posibilita la búsqueda de subsecuencias.

Por el contrario en [KP00a] se utiliza la segmentación PAA enunciada en [KP00b]

y [YF00]. Este método se denomina *PDTW*, *DTW* segmentado lineal o *Piecewise Dynamic Time Warping.*

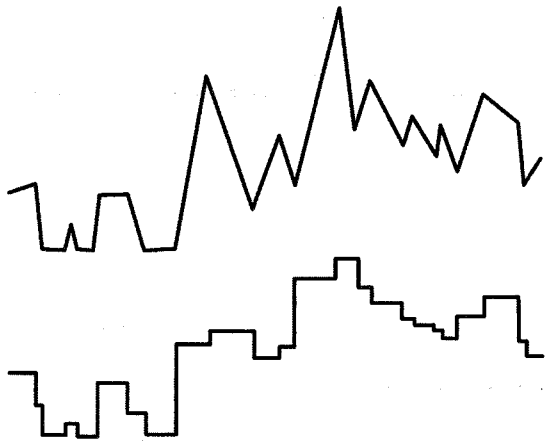


Figura 3.12: Segmentación a trozos y segmentación constante

En la figura 3.12 se muestra la serie segmentada siguiendo el algoritmo presentado en [KS98] y la versión de segmentos constantes de la misma serie.

Aunque ambos mecanismos proporcionan una aceleración sobre la aplicación del algoritmo *DTW* clásico y una mejora de calidad frente a una distancia euclídea, dependen de forma fundamental del grado de segmentación.

La elección del nivel de compresión, o el número de divisiones para representar los datos originales, afecta los resultados obtenidos en [KP00a]. Un nivel de compresión elevado implica una aproximación mas general con lo que el número de falsos negativos aumenta. Por el contrario, si se reduce la compresión el tiempo de computación se acerca al del algoritmo *DTW* clásico.

Ninguno de estos trabajos propone un método sistemático de identificación del valor óptimo de este parámetro.

Para eliminar esta sensibilidad, se presenta el *IDDTW*, *Alineamiento Dinámico de Temporal Iterativa* o *Iterative Deepening Dynamic Time Warping*, en [CKHP02], donde el usuario especifica la consulta y la tolerancia para falsos negativos. La idea es aplicar recursivamente el algoritmo incrementando el número de dimensiones con

que se representan los valores originales, i.e., aumentando los niveles de resolución. En cada nivel sólo se comprueban aquellos objetos que hubiesen sido seleccionados como válidos en el nivel anterior.

Para cada nivel se crea un modelo de distribución de los errores de aproximación por medio de una muestra de la base de datos a la que se le aplica el *PDTW* y el *DTW* original, y se almacena la diferencia. Durante una búsqueda se aplica *PDTW* a la mínima resolución. Esto nos proporciona una distancia estimada pero con el modelo de distribución del error se calcula la probabilidad de que dicha serie sea mejor aproximación que la mejor hasta el momento. Luego se compara esta probabilidad con la tolerancia especificada por el usuario; si la probabilidad está dentro de la tolerancia se continúa en un nivel de resolución mayor y se descarta en caso contrario.

Intentando salvar el escollo del incumplimiento de la desigualdad triangular por parte de *DTW* el trabajo [KPC01] propone una solución. En este caso se define una función de distancia que subestima la distancia de *DTW* y que sí satisface la desigualdad triangular. Esta función toma como parámetro un vector característico de cuatro elementos obtenido de cada serie.

Los componentes del vector característico para una secuencia son: el primer valor, el último, el mayor y el menor. Para dos secuencias S y Q, y tomando la norma máxima como función de distancia, el límite inferior de la distancia del alineamiento temporal queda simplificado a

$$D_{tw-lb}(S,Q) = L_\infty(Feature(S), Feature(Q))$$

definiendo

$$D_{tw-lb}(S,Q) = \text{máx} \begin{cases} |First(S) - First(Q)| \\ |Last(S) - Last(Q)| \\ |Greatest(S) - Greatest(Q)| \\ |Smallest(S) - Smallest(Q)| \end{cases}$$

donde $Feature(S)$ es el vector característico de la secuencia S y $First(S)$,$Last(S)$,$Greatest(S)$ y $Smallest(S)$ proporcionan los elementos del vector característicos ya mencionados.

Aunque es el primer trabajo que logra presentar un índice exacto de las series bajo DTW, tiene la limitación de que, aunque se obtienen cuatro características, sólo se utiliza una, elegida en tiempo de ejecución, en el cálculo la función de límite inferior con lo que se obtiene una comparación muy pobre que genera multitud de falsos positivos.

El documento [PLC99] presenta una técnica diseñada para la detección de subcadenas y define el concepto de coincidencia de subcadenas alienadas.

Partiendo de una base de datos donde existe una representación segmentada de todas las curvas, la forma de localizar las subsecuencias que coinciden con una dada, consiste en segmentar la subsecuencia de consulta y compararla con las porciones de cada curva que tienen igual número de segmentos. Para dicha comparación se utiliza una versión de DTW entre los pares de segmentos.

Para la realización del índice se obtiene de cada segmento un vector característico que está formado por 5 elementos: el número de elementos de la curva original que hay en ese segmento, los valores primero y último del segmento y la maxima desviación negativa y la mínima positiva de la línea que conecta los puntos inicial y final del segmento.

Por medio del algoritmo $MTAH$, *Jerarquía de Tipos de Múltiples Atributos* o *Multiple-attribute Type Abstraction Hierarchy*, se hace una clasificación de los vectores característicos en grupos y se asigna un símbolo a cada grupo. Podemos reescribir cada serie de valores como una serie de símbolos por medio de sustituir cada segmento que la compone por el símbolo representante de la clase a que pertenece.

Con éstas series de símbolos se construye un índice basado en *Árboles de Sufijo Generalizados GST*, o *Generalized Suffix Tree* [Ste94].

Este método tiene dos inconvenientes; el primero es que, en contra de lo indicado

por sus autores, no elimina los falsos negativos, como se reconoce en [PLC01], y el segundo viene producido por los tamaños desmesurados que pueden alcanzar los árboles de sufijo con el crecimiento de la base de datos.

En [PCYH00] se vuelven a utilizar los árboles de sufijo con datos categorizados. Para realizar una categorización sencilla se presentan dos métodos; uno que utiliza categorías de igual longitud y otro que maximiza la entropía.

Posteriormente, en [PLC01], se amplía éste mecanismo utilizando segmentos de crecimiento monotónico. Se añade al vector característico la suma de todas las alturas y la diferencia con el valor mínimo del segmento, de todos elementos del segmento excepto el primero y el último.

En este caso el índice se crea mediante un árbol-R basado en los vectores característicos obtenidos de cada segmento.

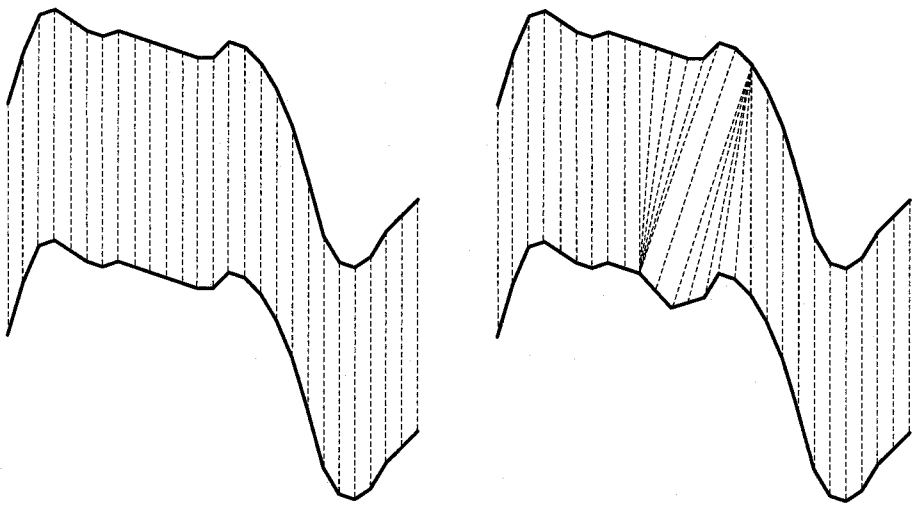
Idénticamente al anterior trabajo se definen una serie de filtros y funciones de distancia que son cota inferior de la distancia DTW entre dos segmentos aprovechando las características monotónicas de los mismos.

Incluso con estos refinamientos los resultados obtenidos son pobres, como se demuestra en [Keo02].

Este método garantiza que no existen falsos negativos puesto que el árbol de sufijo no considera una función de distancia; de esta forma evita el cumplimiento de la desigualdad triangular ya explicado.

De todas formas, el método no proporciona una guía sistemática para realizar una categorización óptima, que es básica para obtener aceleraciones elevadas. Además, sufre una grave reducción de las prestaciones en las búsquedas completas debido al tamaño de los árboles de sufijo.

Volviendo al algoritmo DTW ha de subrayarse que cuando intenta alinear dos secuencias similares excepto por aceleraciones y deceleraciones locales en el eje temporal, el algoritmo tiene éxito. Pero presenta problemas cuando las dos secuencias también difieren en el eje Y.



A) B)

Figura 3.13: A) Alineamiento *DTW* de dos secuencias idénticas. B) Alineamiento *DTW* tras acentuar el valle central

Las diferencias globales, que afectan a toda la secuencia, como diferencia de las medias (traslación vertical) , diferente escala (escalado en amplitud) o tendencias lineales pueden ser eliminadas de forma eficiente. Pero dos series pueden tener diferencias locales en el eje Y, como un valle más profundo en una de ellas que el correspondiente valle de la otra. En ese caso *DTW* intenta explicar la diferencia en términos del eje temporal produciéndose singularidades.

En la figura 3.13A se muestran dos series idénticas y el alineamiento producido al aplicar *DTW*, que lógicamente es uno a uno entre los puntos de ambas series. En la figura 3.13B la serie representada en la posición inferior ha sido modificada por medio de hacer más profundo el valle de la zona central. Al representar análogamente el alineamiento que genera *DTW* se aprecian dos singularidades por la razón expuesta anteriormente.

Este problema de *DTW* fue puesto de manifiesto en [KP01] donde se sugiere una solución. La idea, definida como *DDTW*, *DTW Derivativo* o *Derivative DTW*, consiste en sustituir la distancia euclídea entre los pares de puntos de ambas series por la raíz de las diferencias de las derivadas estimadas para dichos puntos. Para evitar

el uso de métodos sofisticados de estimación de la derivada los autores proponen utilizar la siguiente aproximación

$$D_x[q_i] = \frac{(q_i + q_{i-1}) + ((q_{i+1} - q_{i-1})/2)}{2} \qquad 1 < i < m$$

La estimación de la derivada en un punto q_i es la media entre las pendientes de dicho punto y el anterior y la pendiente entre el punto anterior y el siguiente. Ha de observarse que la estimación no está definida para los puntos primero y último de la secuencia , usándose en su lugar las calculadas para los puntos segundo y penúltimo.

Manteniendo la misma complejidad temporal que el algoritmo original $DDTW$, se mejoran los alineamientos producidos, especialmente en los casos de variaciones locales importantes.

Asumiendo el inconveniente de DTW presentado anteriormente, en [Keo02, KR04] se presenta un índice para DTW que, subestimando la distancia DTW, garantiza la no existencia de falsos negativos.

A cada secuencia Q se le definen unas funciones que la abarcan superior U e inferiormente L obtenidas de la aplicación de una de las limitaciones geométricas de DTW, la banda Sakoe-Chiva y el paralelogramo de Itakura. En la figura 3.14 se muestra un ejemplo utilizando el paralelogramo de Itakura.

Cada una de estas envolturas es representada por medio de una aproximación constante a trozos de reducida dimensión, suficientemente pequeña para poder utilizar estructuras de indexación multidimensional. En la figura 3.15 se representan estas aproximaciones de las funciones límite.

Para localizar secuencias se define una distancia entre una secuencia de consulta y un MBR, que permite ir recorriendo el árbol del índice de forma eficiente. Como en casi todas las técnicas vistas se realiza un paso final de aplicación del algoritmo clásico sobre el conjunto de respuestas candidatas.

Este mecanismo proporciona una alta aceleración por medio de la indexación

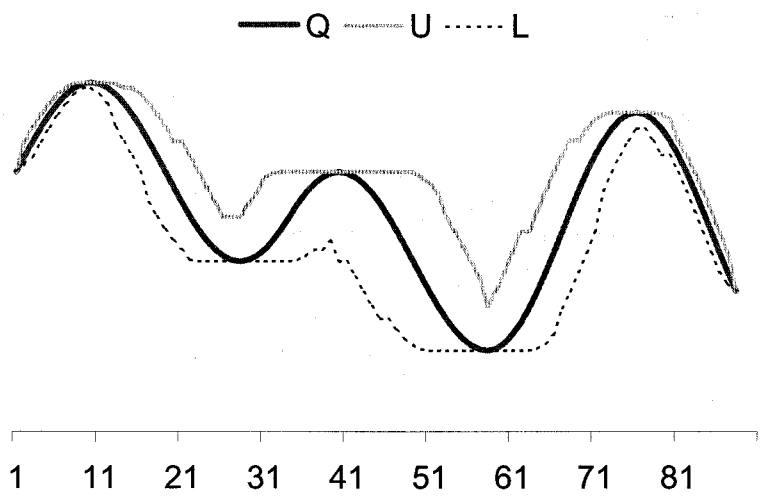


Figura 3.14: Funciones Superior U e Inferior L de la secuencia Q

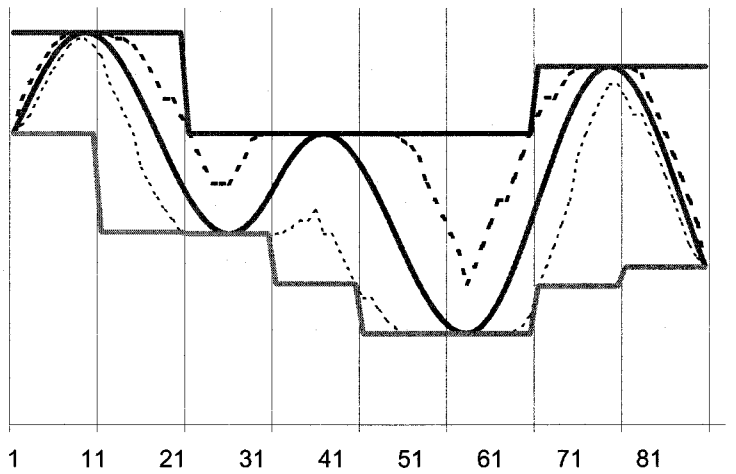


Figura 3.15: Aproximación constante en 8 segmentos de las funciones que acotan la serie.

pero siempre que se tengan presente sus limitaciones: sólo se considera el caso de que las secuencias a comparar tengan la misma longitud y se requiere una restricción topológica a los posibles caminos de alineamiento.

Mientras que la primera sólo requeriría de un paso previo de interpolación para ser salvada, la segunda puede empeorar los resultados presentados en el caso que los

caminos no estén completamente cubiertos por la restricción que se les ha supuesto a priori.

Aunque en los experimentos presentados se ha utilizado una banda Sakoe-Chiba de un tamaño del 10 % del tamaño de la secuencia original, no se proporciona información del efecto que tiene sobre los resultados la variación de este parámetro, ni ninguna regla para su determinación.

Desde un punto de vista novedoso [CMG02] y [LCM+03] definen una variante de *DTW*, denominada *EpDTW*, *DTW sobre Episodios* o *Episodes DTW*, que se aplica sobre el conjunto de episodios en que se descompone cada serie. Los episodios son formas básicas que se localizan en cada serie analizando las derivadas primera y segunda. El mayor problema que presenta está en definir la matriz de distancias entre los diferentes episodios, información fundamental para la obtención de resultados válidos, y para la que no se presenta ningún procedimiento de cálculo.

3.5.2. Subcadena Común Máxima

Trabajando con secuencias de cualquier tipo, desde cadenas de caracteres a secuencias de ADN, una de las medidas de similitud más usadas es la Subsecuencia Común Máxima (LCS) de dos o más secuencias dadas.

Definición de *LCS*: Es la mayor colección de elementos que se encuentran en el mismo orden en dos secuencias distintas.

Sean $S = s_1, s_2, \ldots, s_m$ y $T = t_1, t_2, \ldots, t_n$ sobre un dos secuencias de símbolos definidas sobre alfabeto finito Ω, entonces $LCS(S,T) = U$, $U = u_1, u_2, \ldots, u_r$ tal que existen índices $i_1 < i_2 < \ldots < i_r$ con $1 \leq r \leq m$, y $j_1 < j_2 < \ldots < j_r$, con $1 \leq r \leq m$, que cumplen $s_{i_z} = u_z$ y $t_{j_z} = u_z$.

El algoritmo se divide en dos bloques. Uno que calcula el tamaño de la subsecuencia y otro paso posterior en que se descubren los elementos que la componen.

La obtención de la LCS es un problema muy conocido y que, suponiendo que la longitud de las cadenas a tratar es m y n respectivamente, tiene una complejidad $O(mn)$ en tiempo y espacio utilizando técnicas de programación dinámica. El algoritmo es análogo al definido para DTW salvo que la distancia γ entre cada par de elementos de S y T se define como

$$\gamma(i,j) = \begin{cases} 0 & \text{si} \quad i = 0 \quad o \quad j = 0 \\ \gamma(i-1, j-1) + 1 & \text{si} \quad i \neq 0, j \neq 0 \quad \text{y} \quad s_i = t_j \\ \text{máx}\{\gamma(i, j-1), \gamma(i-1, j)\} & \text{si} \quad i \neq 0, j \neq 0 \quad \text{y} \quad s_i \neq t_j \end{cases}$$

siendo $0 \leq i \leq m$ y $0 \leq j \leq n$.

La distancia entre los diferentes emparejamientos de valores de cada serie será 1 ó 0 en cada comparación con la única consideración de que los símbolos a comparar sean o no iguales.

Nuestro interés en LCS es doble:

1. Como el lenguaje SDL produce una cadena de símbolos desde los valores numéricos de la serie temporal, es posible entonces usar este algoritmo para obtener una distancia entre dos series con abstracción en las formas de las curvas.

2. El algoritmo LCS es un caso especial del DWT. Heredando así todas las características de DWT enunciadas en la sección anterior.

Una simplificación al DTW es que el recorrido del camino óptimo no necesita terminar en la posición $(1, 1)$ puesto que los desplazamientos verticales u horizontales no aportan nuevos elementos a la subsecuencia.

Desde [WF74] hasta [CMP+00] se han realizado diferentes aproximaciones al problema que intentan reducir la complejidad en alguna medida, ya sea considerando alfabetos limitados, requerimiento lineal del espacio, soluciones aproximadas o incluso algoritmos de acotación por aprendizaje.

En [Apo97] se realiza una revisión de un elevado número de estas implementaciones, o variaciones, del algoritmo LCS.

Trabajos que aplican *LCS* No vamos a recoger en este apartado el ingente número de trabajos que utilizan LCS aplicándolo a colecciones de símbolos, nos centraremos en aquellos que han salvado la distancia entre los valores continuo de las series y la aplicación a alfabetos finitos logrando utilizar el método LCS a series temporales.

En [DGM97] se unen la idea de utilizar la subsecuencia común máxima como medida de similitud entre secuencias de objetos, presentada en [YÖ96], con la idea de utilizar funciones de trasformación enunciadas en [GK94].

Dadas dos series temporales $X = \langle x_1, ..., x_n \rangle$ y $Y = \langle y_1, ..., y_m \rangle$, y los parámetros $0 < \gamma, \varepsilon \leq 1$, las series X e Y son (F, γ, ε)-similares si y solo si existe una función $f \in F$ y las subsecuencias $X_f = \langle x_{i_1}, \ldots, x_{i_{\gamma n}} \rangle$ y $Y_f = \langle y_{i_1}, \ldots, y_{i_{\gamma n}} \rangle$, donde $i_k \leq i_{k+1}$ y $j_k \leq j_{k+1}$ para todo $k = 1, \ldots, \gamma n - 1$, tal que $\forall k, 1 \leq k \leq \gamma n$ se verifica que

$$y_{j_k}/(1+\varepsilon) \leq a x_{i_k} + b \leq y_{j_k}(1+\varepsilon)$$

El parámetro ε controla lo próximas que están ambas subsecuencias, mientras γ hace lo propio con la longitud de la subsecuencia de una que se mapea en la otra.

Para una clase de transformaciones y dos secuencias, se presenta un algoritmo exacto que obtiene la transformación que maximiza el tamaño de la LCS y por tanto de la similitud aplicando el hecho de que existen clases de equivalencia en la clase de transformaciones. Todas las transformaciones de una clase de equivalencia proporcionan el mismo resultado. Así se demuestra que el algoritmo se ejecuta en un tiempo $O(n^6)$ en el caso de transformaciones lineales y en $O(n^4)$ en las transformaciones de escala.

Debido a la falta de aplicación práctica de dicho algoritmo se presenta uno aproximado que proporciona los límites inferior y superior de los parámetros de la transformación lineal. Una vez obtenidos esos límites se crea una rejilla de muestreo

sobre el rectángulo resultante y se calcula la LCS para cada punto de la muestra. Finalmente se devuelve el mejor resultado.

En [VGK02] se continúa el trabajo de [DGM97] aportando funciones de traslación y aplicándolo a trayectorias multidimensionales. Además impone una nueva restricción para que dos elementos de ambas series se consideren similares. Esta restricción se descompone en

1. Sus respectivas ubicaciones en cada serie no disten más de d posiciones y
2. la diferencia entre sus valores no sea mayor que un umbral determinado, identificado por ε.

La novedad se refiere al mecanismo para realizar un algoritmo aproximado, frente al requerimiento de tiempo de ejecución del algoritmo exacto. La idea es dividir el conjunto de todas las posibles funciones de traslación en un número predeterminado, probando para cada uno el valor de la LCS y devolviendo el mayor obtenido.

En este trabajo también se aporta un método para la realización de un índice que permita potenciar las consultas. Para ello se realiza una agrupación de las trayectorias en función a su tamaño y posteriormente un clustering de las trayectorias presentes en cada grupo. Además se calcula la trayectoria que tiene una distancia mínima, de todas las demás de su grupo, a una trayectoria determinada.

La búsqueda de la secuencia más aproximada a una dada se realiza por medio de calcular un límite inferior de la distancia de cada nodo, basado en los valores almacenados en cada nodo, procesando sus hijos o descartando todo el subárbol.

Aunque en [VGK02] los resultados aproximados se acercan bastante al exacto y obteniéndose en un tiempo varios órdenes de magnitud menor, se presenta el problema de la selección de los parámetros al igual que ocurre con [DGM97].

En [VGK02] se afirma que empíricamente se ha comprobado que los valores de d deben estar en el entorno $20-30\,\%$ del tamaño de las series, pero no hay regla para determinar el valor correcto de ε, que además parece ser dependiente de los datos a

analizar.

3.6. Otros enfoques

Además de las aproximaciones ya comentadas, existe un grupo de trabajos que no pueden ser incluidos en ninguna de las secciones anteriores. Son trabajos con planteamientos singulares y novedosos dentro del grupo de investigadores dedicados a la comparación de series, la mayoría de los cuales no han tenido continuación en otras propuestas.

Hagamos una revisión de ellos siguiendo un orden cronológico afectado por las posibles relaciones entre los distintos trabajos.

Un trabajo interesante es [CS90], donde se propone el estudio de series con diferentes escalas desde una perspectiva cualitativa. La idea fundamental es reconocer, partiendo de la serie original, los diferentes elementos característicos de la serie realizando un sucesivo aumento de escala hasta reducir la serie a su máxima simplificación. Con este procedimiento se logra un conocimiento de las tendencias existentes en la serie original.

En [ALSS95] se utiliza un concepto de similitud que considera la posibilidad de desplazamientos y reescalado. El problema de determinar la similitud de dos secuencias se divide en:

1. Encontrar los pares de subsecuencias atómicas que son similares, con un tamaño de ventana determinado. Los valores de la secuencia que están dentro de dicha ventana son normalizados para permitir su traslación y escalado. Para que dos subsecuencias se consideren similares una debe envolver a la otra con una distancia máxima dada.

2. Unir las subsecuencias para forma subsecuencias más largas permitiendo espacios de no coincidencia entre ellas.

3. Comprobar el conjunto de subsecuencias que, una vez ordenadas, proporcionan el camino de similitud más largo y valorar si su longitud es suficiente.

Se considera que dos curvas son similares si existe un número suficiente de pares de subsecuencias sin solapamiento y ordenadas que son similares.

Para mejorar la eficiencia de búsqueda de subsecuencias similares se crea un índice con las subsecuencias.

Este trabajo comparte algunos de los inconvenientes que ya se comentaron acerca de [FRM94] como son la sensibilidad al ruido y el haber sido verificados solamente sobre un conjunto de series provenientes de un ámbito muy concreto. En particular el ámbito de las cotizaciones de valores, no pudiéndose extrapolar su idoneidad en series extraidas en otros entornos.

Desde un enfoque más global, en [JMM95], se presenta una arquitectura independiente del dominio del problema que posibilita la manipulación de la similitud. Está compuesto por un lenguaje de patrones P, un lenguaje de reglas de transformación T y un lenguaje de consulta L. El lenguaje de patrones permite la especificación de clases de objetos, mientras el de reglas define transformaciones que preservan la similaridad. Un objeto A es considerado similar a otro objeto B, si B puede ser convertido en A mediante la aplicación de una serie de transformaciones definidas en T.

Partiendo desde un punto totalmente diferente el mecanismo que se adopta en [SZ96] para reducir la cantidad de información a almacenar de cada serie consiste en hacer una división de las series y representar cada subsecuencia por medio de una función. Se sugieren como funciones las curvas Bézier, los polinomios y la interpolación lineal. Con esta forma de sintetizar, toma una gran relevancia la manera de trocear cada secuencia para permitir posteriormente una comparación congruente.

El trabajo de [JB97] se asemeja al trabajo que se presentará en el capítulo 4 en utilizar el SDL de [APWZ95], que se analiza en el apéndice B, para convertir una serie en una secuencias de símbolos. Posteriormente obtiene una firma de cada serie

por medio de una función de hashing.

Esta firma, de menor tamaño que la serie original, se basa en la aplicación de la función de hashing sobre la serie de símbolos que ocupa una ventana deslizante. La firma es usada para determinar la similitud entre las series.

Tanto el tamaño de la ventana como la longitud de la firma afectan significativamente a los resultados obtenidos, no presentándose ningún método para su cálculo ante un conjunto determinado de series.

Desde la aproximación de las distancias de edición en la comparación de cadenas en [CN04] donde se define una nueva métrica que puede entenderse como una variante de una norma L_1 pero que soporta desviaciones temporales o también como una variante de una distancia de edición y de *DTW* pero siendo una métrica.

Se propone una nueva cota inferior que puede indexarse eficientemente por medio de árboles Binarios B+ y se desarrolla un algoritmo que aplica la desigualdad triangular y la cota inferior para obtener los candidatos de una consulta de vecinos cercanos.

Otro enfoque se presenta en [CW99]. Basándose en el sentido de similitud por transformaciones presentado por [RM98b] se considera que dichas transformaciones pueden ser de escalado y de traslación.

[CW99] define la *Transformación libre de desplazamiento* (o shift Eliminated Transformation). Esta transformación mapea las secuencias en el *Plano libre de desplazamiento*, de tal forma que la distancia euclídea se calcula como la mínima tras todos los posibles escalados y desplazamientos de dos secuencias.

Una serie de n elementos puede considerarse un punto en un espacio de n dimensiones y todo punto de un espacio puede describirse con un vector de posición en ese mismo espacio.

Desde un punto de vista geométrico, el reescalado de una serie no es más que el producto escalar de un real por su vector de posición. El conjunto de todas las posibles escalas proporciona una línea en la dirección del vector de posición original.

De forma análoga el desplazamiento es el resultado de la suma al vector de posición de un vector normal multiplicado por un índice de desplazamiento, que también proporciona una línea en la dirección del vector normal.

Con estos antecedentes se definen funciones de similitud que dependen de la distancia entre dos líneas que no permiten la utilización de estructuras de indexación basadas en árboles-R y árboles-X, por estar diseñados para puntos (o vectores estáticos). Así se define una transformación que aplicada a una linea de transformación proporciona un punto en un hiperplano. Esta transformación denominada *Transformación libre de desplazamiento*, proyecta un punto en un plano que pasando por el origen es perpendicular al vector normal del desplazamiento.

La función de distancia no es simétrica lo que puede provocar resultados que contradicen la intuición.

Una nueva vertiente del problema de la comparación de secuencias es el de la similitud en secuencias de múltiples atributos que es abordado en [KSG01] y [KSG02].

Al igual que en [CW99], en éstos trabajos la distancia entre dos vectores se basa en la distancia de sus proyecciones, que en caso de secuencias $\vec{u}$ y $\vec{v}$ de igual longitud se define como

$$d(\vec{u},\vec{v}) = \|TSE(\vec{v})\|_2 \cdot \sqrt{1 - \left(\frac{TSE(\vec{v}) \cdot TSE(\vec{u})}{\|TSE(\vec{v})\|_2 \cdot \|TSE(\vec{u})\|_2}\right)^2}$$

donde $TSE(\vec{u})$ y $TSE(\vec{v})$ son las proyecciones en el plano libre de desplazamiento de las secuencias $\vec{u}$ y $\vec{v}$ respectivamente. Pero como ésta definición no es simétrica se amplía a la mínima entre las dos ordenaciones posibles.

$$dist(\vec{u},\vec{v}) = \text{mín}\left\{d(\vec{u},\vec{v}), d(\vec{v},\vec{u})\right\}$$

Se demuestra que esta definición también es válida para secuencias de múltiples atributos, sean o no dependientes.

Para el caso de secuencias de un atributo, y siendo $\theta_{\vec{u},\vec{v}}$ el ángulo entre los vectores $TSE(\vec{u})$ y $TSE(\vec{v})$ podemos resumir la distancia de dos secuencias como

$$dist(\vec{u},\vec{v}) = \text{mín}\left\{\|TSE(\vec{u})\|_2, \|TSE(\vec{v})\|_2\right\} * \sin\theta_{\vec{u},\vec{v}}$$

por lo que la distancia será proporcional a las longitudes de sus proyecciones en el Plano SE y al ángulo entre dichas proyecciones.

Se propone en ambos trabajos un nuevo índice llamado índice-CS (*Cone Slice*) que permite la comparación de series con desplazamiento y escalado. El índice representa rebanadas"jerárquicas de conos ordenados. La idea es proyectar para cada secuencia las líneas de escalado y de desplazamiento, agrupando las que tengan las líneas de desplazamiento proyectadas más cerca cuyos ángulo entre las líneas de escalado sean menores.

En el caso secuencias de longitud l con k atributos, nos enfrentamos con dos casos distintos:

1. Atributos dependientes: en ese caso se reduce al problema de un único atributo ya que todos están afectados del mismo escalado y desplazamiento. Cada secuencia es un punto en un espacio de kl dimensiones. Esta dimensionalidad puede reducirse y posteriormente se construye el índice CS.

2. Atributos independientes: Cada atributo puede presentar diferentes escalados y desplazamientos. Dividimos cada secuencia en k secuencias de un atributo y longitud l. Como consecuencia a cada serie serie le corresponden k puntos en un espacio de l dimensiones. Se concatenan las proyecciones de dichos puntos y se crea el índice-CS. La distancia se calcula para cada atributo de forma independiente y los resultados son acumulados.

Se han presentado diferentes técnicas para la reducción de la dimensionalidad como SVD, DFT, DWT y PAA, pero todos los enfoques han utilizado cada una de forma individual. En [KCP01] se propone el realizar una indexación utilizando varias técnicas de forma independiente, de forma que cada una de ellas proporcione un índice independiente sobre la base de datos.

Antes de realizar una consulta, se le aplican todas las reducciones y se compara cuál proporciona una representación más fidedigna. Esa infomación permite decidir el orden en que son accedidos los índices para obtener las soluciones.

En lugar de elegir una representación que sea muy buena para una mayoría de objetos pero mala para el resto, se elige la combinación de varios índices cada uno con una cierta especialización.

Intuitivamente se puede imaginar que los tiempos de respuesta para las consultas serán iguales a los que proporciona la técnica seleccionada para cada consulta, pero se demuestra que son mejores.

Un trabajo que se ha centrado en la posibilidad de utilizar algoritmos paralelos es [GGK01]. En él se proponen dos algoritmos paralelos para localizar reglas de asociación secuencial en ordenadores paralelos, uno mediante paralelismo de datos y otro con paralelismo de tareas. Aunque son efectivos con el incremento del número de procesadores, la carga estimada disminuye y la computación se muestra no balanceada.

Continuando el trabajo anterior [GK01a] presenta un rediseño paralelo de un algoritmo secuencial para el descubrimiento de patrones secuenciales que es balanceado y optimizado para ordenadores paralelos de memoria distribuida.

Con un enfoque diferente y muy específicamente diseñado, el método de realizar clustering con secuencias de proteínas de [GK01b], consiste en la localización de patrones (o *"motif"*) que se repiten entre las diferentes secuencias y la caracterización de cada serie por medio de las subsecuencias que presenta.

CAPÍTULO 4

ÍNDICE CUALITATIVO DE SIMILITUD - QSI

Un aspecto novedoso de nuestra aproximación es la inclusión de conocimiento cualitativo dentro de la comparación de series.

Ésa es la idea central de este índice y para ello se propone una medida basada en la coincidencia de etiquetas cualitativas que representan la evolución de los valores de las series. Cada etiqueta representa un rango de valores que pueden considerarse similares desde una perspectiva cualitativa.

La aproximación propuesta presenta una serie de ventajas con respecto a otros métodos aparecidos en la bibliografía. Por un lado, el uso de toda la información contenida en la serie temporal maximiza la exactitud. Por otro, la consideración de grupos de evoluciones como similares prioriza la comparación en la forma general de las curvas y no en sus valores puntuales. En cualquier caso, hay que decir que las series temporales con las que se trabaja se suponen libres de ruido entre muestras, donde la evolución se supone lineal y monotónica.

4.1. Índice Cualitativo de Similitud - QSI

Sea un sistema que evoluciona en el tiempo y del que se registra una serie de estados del sistema x_i para un conjunto de instantes determinados. Se tiene una serie temporal $X = \langle x_0, ..., x_f \rangle$. Cada estado del sistema estará representado por un conjunto de parámetros, pero desde este momento nos centraremos en el caso de una única variable.

La aproximación propuesta en este trabajo se realiza en tres etapas.

1. Primero se realiza una normalización de los valores de la serie temporal, obteniéndose $\overline{X} = \langle \overline{x}_0, ..., \overline{x}_f \rangle$ y a partir de ella se obtiene la serie de diferencias $X_D = \langle d_0, ..., d_{f-1} \rangle$,

2. que se traduce a una cadena de caracteres $S_X = \langle c_0, ..., c_{f-1} \rangle$.

3. La similitud entre dos series temporales se obtiene comparando las dos cadenas obtenidas de la transformación anterior mediante un algoritmo de subsecuencia común máxima, o LCS (véase 3.5.2). El resultado proporcionado por ese algoritmo sirve, en nuestra aproximación, como base para definir una medida de similaridad entre las series.

La principal diferencia de nuestro trabajo respecto a los comentados se centra en que la comparación de las series se hace desde una perspectiva cualitativa de la evolución de las mismas, frente al tradicional análisis cuantitativo de sus valores.

Al mismo tiempo hemos convertido el problema de la comparación de series numéricas en una comparación de cadenas con la simplificación que ello representa. Esta modificación en el dominio del problema proporciona la posibilidad de utilización de metodologías profusamente estudiadas en los últimos años.

Finalmente la utilización de algoritmos de comparación de cadenas también es una novedad, al tiempo que se mantiene una de las características de los algoritmos de alineamiento dinámico temporal muy utilizados en trabajos de reconocimiento

de voz: la detección de series similares aunque existan desplazamientos en la escala temporal.

A continuación vamos a dedicar una sección a cada uno de los pasos comentados para realizar una revisión en profundidad de cada uno de ellos.

4.1.1. Normalización

En primer lugar y con la intención de poder comparar cualitativamente las series se realiza una normalización de sus valores al intervalo [0,1]. Esta normalización permite la comparación de series con diferentes escalas cuantitativas.

La figura 4.1 muestra dos series que presentando escalas distintas puede considerarse idénticas desde un punto de vista cualitativo, ya que presentan la misma evolución.

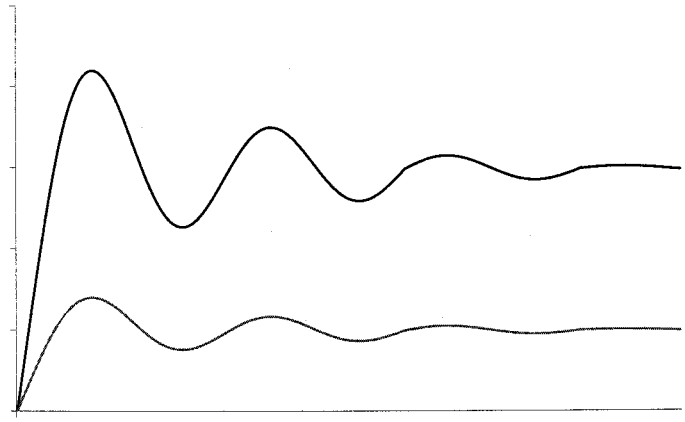


Figura 4.1: Dos series que pueden considerarse cualitativamente idénticas en su forma puesto que su diferencia es un cambio de escala.

Sea $X = \langle x_0, ..., x_f \rangle$ una serie temporal. A partir de ella se obtiene la serie temporal normalizada representada como $\overline{X} = \langle \overline{x}_0, ..., \overline{x}_f \rangle$ donde:

$$\overline{x}_i = \frac{x_i - min(x_0, ..., x_f)}{max(x_0, ..., x_f) - min(x_0, ..., x_f)} \tag{4.1}$$

siendo *min* y *max* operaciones que devuelven los valores mínimo y máximo de una secuencia de números.

Siguiendo los pasos de [APWZ95] se considera que la similitud de las series viene dada por el parecido existente entre sus evoluciones a lo largo del tiempo y no por la proximidad de los valores concretos que tomen. Se desea por tanto obtener una descripción de las series que representen la forma en que se modifican sus valores entre cada dos instantes de tiempo consecutivos; a esta variación le llamaremos *transición.*

La forma más sencilla de obtener esta descripción es utilizando la derivada, que en el caso de las series que tienen un intervalo de muestreo constante en el seguimiento de la magnitud representada, es equivalente a la diferencia de los valores.

Por tanto, a partir de esta serie normalizada, se obtiene la serie de diferencias o transiciones $X_D = \langle d_0, ..., d_{f-1} \rangle$ donde

$$d_i = \overline{x}_i - \overline{x}_{i-1} \tag{4.2}$$

Esta serie de diferencias será utilizada posteriormente en el etiquetado para obtener la cadena de caracteres correspondiente a la serie temporal. Por la definición de la serie de diferencias, y su obtención partiendo de series normalizadas, es evidente que para todo $d_i \in X_D$ su valor pertenece al intervalo [-1,1].

4.1.2. Etiquetado

Una vez que las series han sido preprocesadas y se han convertido en la evolución de las amplitudes resultantes se procede al etiquetado cualitativo. Para este paso se aplica de una forma limitada, una versión simplificada de su sistema de etiquetado, el método presentado en el lenguaje *SDL* en [APWZ95]. Una revisión de este método puede encontrarse en el Apéndice B, donde también se incluye, en la sección B.1, un breve comentario acerca de las diferencias entre su etiquetado y el que se está proponiendo.

La normalización propuesta en el apartado anterior considera la evolución de la pendiente en lugar de los valores de la serie. Con la intención de asignar una etiqueta a cada tipo de pendiente, se divide el intervalo de las posibles pendientes en varios

grupos y se le asigna a cada uno una etiqueta cualitativa. Cada grupo se compone de un rango de valores que se consideran similares.

En QSI se establecen las siguientes restricciones al conjunto de etiquetas seleccionado:

- Resultante del preprocesado previo se conoce que los valores de las series a etiquetar se encontrarán en el intervalo $[-1, 1]$.
- Se establece la obligatoriedad de que los intervalos que definen el conjunto de etiquetas sean disjuntos.

Así se propone una división, proveniente de la experiencia y que ha sido contrastada empíricamente, de los posibles valores de las pendientes que se realiza en siete rangos y de acuerdo con un parámetro δ, $\delta > 0$, proporcionado por los expertos, según el conocimiento que éstos tienen sobre el problema. En este sentido, los expertos deben informar sobre el significado de las diferentes etiquetas cualitativas en el ámbito del problema, es decir, identificar los rangos en los que se mueven cada una de las etiquetas cualitativas.

Con estas premisas se crea un conjunto de etiquetas y sus respectivos rangos que se presentan en la tabla 4.1.

En dicha tabla la primera columna representa la etiqueta cualitativa para cada rango de pendientes, que se muestran en la segunda columna. La última columna contiene el carácter asignado a cada etiqueta.

El alfabeto presentado contiene tres caracteres para el incremento y tres para los decrementos, y uno adicional para el rango constante.

El número de etiquetas es reducido; con un número mayor de etiquetas se tendría una mayor granularidad que haría que la similitud cualitativa se aproximara a la similitud cuantitativa.

Este alfabeto se usa para obtener la cadena de caracteres $S_X = \langle c_0, \ldots, c_{f-1} \rangle$ correspondiente a la serie X , donde cada c_i representa la evolución de la curva

Etiqueta	Intervalo	Símbolo
Fuerte incremento	$(1/\delta, 1]$	H
Incremento moderado	$(1/\delta^2, 1/\delta]$	M
Ligero incremento	$(0, 1/\delta^2]$	L
Sin variación	0	0
Ligero decremento	$[-1/\delta^2, 0)$	l
Decremento moderado	$[-1/\delta, -1/\delta^2)$	m
Fuerte decremento	$[-1, -1/\delta)$	h

Tabla 4.1: Etiquetas cualitativas, conjunto de divisiones de los valores de las series de diferencias en función de δ y símbolo asignado en QSI

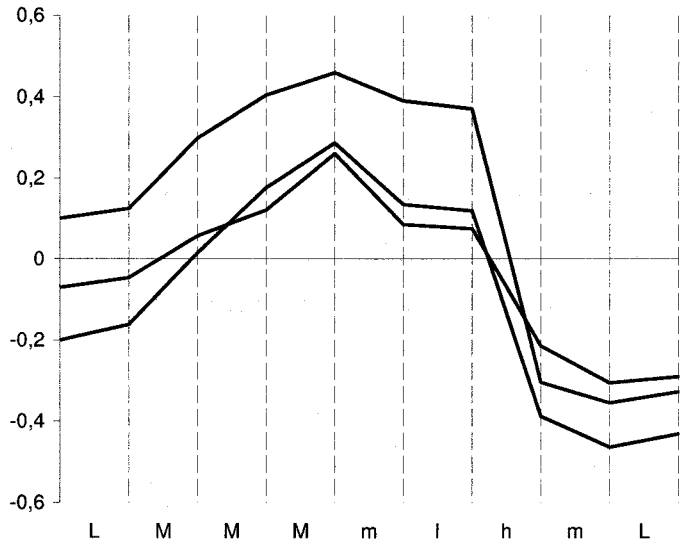


Figura 4.2: Tres series que generan la misma secuencia de símbolos, incluida en la zona inferior.

entre dos puntos adyacentes en el tiempo en X . Se obtiene de $X_D = \langle d_0, \ldots, d_{f-1} \rangle$ asignando a cada d_i su carácter correspondiente con respecto a la tabla anterior.

Esta traducción de la serie temporal a una secuencia de símbolos permite abstraernos de los valores de la serie y centrarnos en su forma. Es evidente que cada cadena de símbolos puede describir un infinito número de series que cumplen con

las restricciones establecidas por cada símbolo a la transición que representa: puede considerarse que cada cadena de símbolos es el representante canónico de una familia de series que son cualitativamente idénticas.

La figura 4.2 muestra tres series diferentes cuya traducción produce la misma secuencia de símbolos. Mientras las tres curvas tienen diferentes puntos de inicio y fin, su evolución es cualitativamente similar, por lo que se puede decir que pertenecen a la misma familia de curvas desde un punto de vista cualitativo de su evolución en el tiempo.

La figura 4.3 resume gráficamente el proceso de obtención de la cadena de símbolos partiendo de una serie y utilizando el alfabeto mostrado en la tabla 4.1 con un valor de $\delta = 5$. En la figura A) se muestra una porción de una serie temporal y en la figura B) la misma parte de la serie una vez normalizada. En la figura C) se han añadido en forma de barras los valores de las diferencias entre cada dos valores consecutivos que forman la serie X_D. Posteriormente se añaden las líneas que marcan los límites de las regiones a las que se les ha asignado una etiqueta cualitativa, como se observa en la gráfica D) y en la E) se han agregado los símbolos que corresponden a cada transición y que conforman la serie de símbolos S_X.

En el siguiente capítulo se realiza una revisión profunda de la calidad de este etiquetado y se realizan comparaciones con otros métodos de etiquetado.

4.1.3. Definición de Similitud QSI

Una vez que tenemos las series originales, $X = \langle x_0, ..., x_f \rangle$ y $Y = \langle y_0, ..., y_f \rangle$, traducidas a cadenas de símbolos cualitativos, S_X, S_Y, podemos definir el *Índice de Cualitativo de Similitud.*

Se define la similitud QSI entre las cadenas como

$$QSI(S_X, S_Y) = \frac{\nabla(LCS(S_X, S_Y))}{m} \tag{4.3}$$

donde ∇S es el cuantificador de conteo aplicado a la cadena S, es decir obtiene

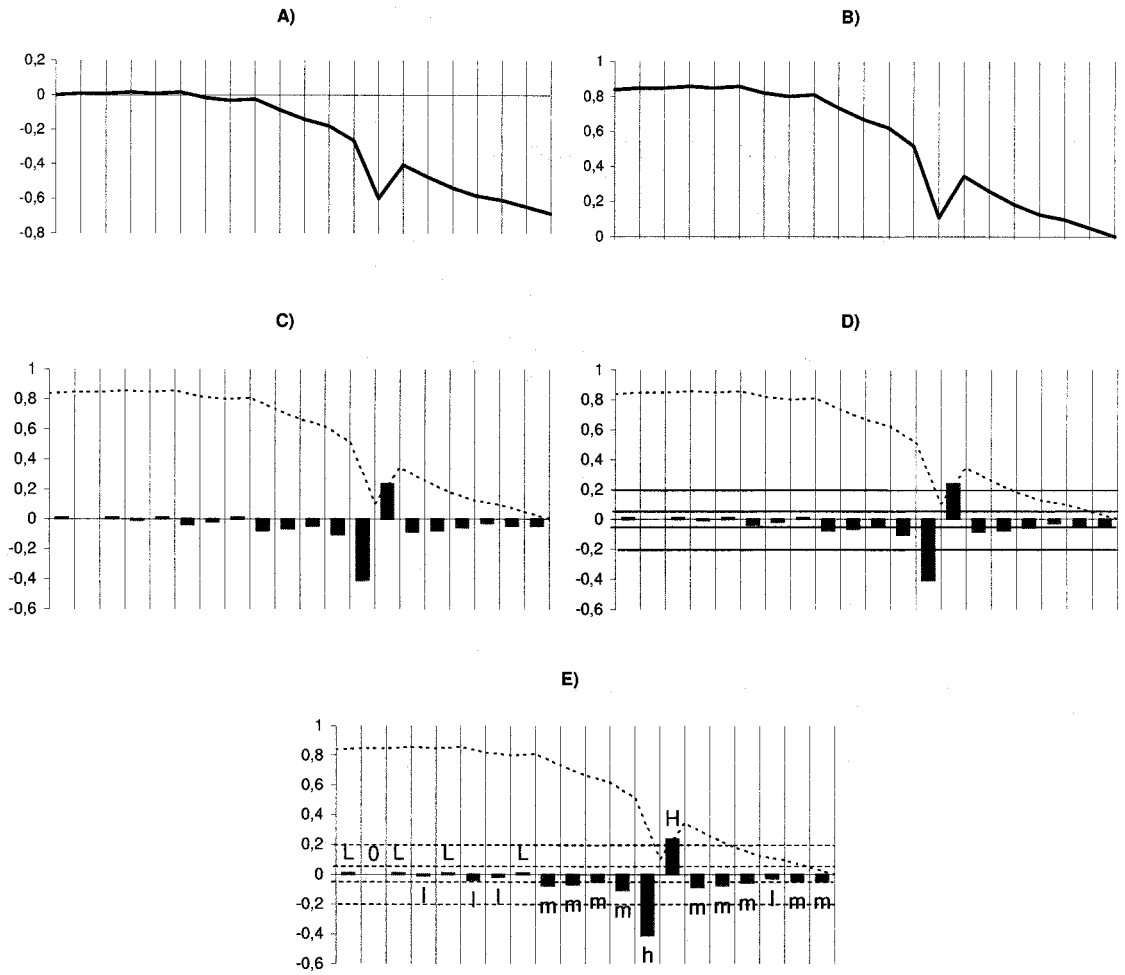


Figura 4.3: Ejemplo de conversión para $\delta = 5$ en la tabla 4.1. Se muestra la serie original en A), la serie normalizada en B), se añaden las barras que representan los valores de las diferencias ($d_i \in X_D$) en C), los límites de las regiones en C) y en D) se completa con los simbolos asignados a cada transición ($c_i \in S_X$).

la longitud de S. Por otro lado, m se define como $m = max(\nabla S_X, \nabla S_Y)$. De esta manera, se puede interpretar la similitud QSI como el número de símbolos que podemos encontrar en el mismo orden en ambas secuencias dividido por la longitud de la secuencia más larga.

Propiedades del QSI

Sean S_X, S_Y, S_Z tres cadenas de caracteres obtenidas de la transformación de tres series temporales. La definición anterior de similitud QSI cumple lo siguiente: 1) $QSI(S_X, S_Y)$ es un número en el intervalo $[0, 1]$. Si X es absolutamente diferente de Y el valor del índice es cero.

$$\begin{gathered} \text{Si } LCS(S_X, S_Y)) = \emptyset \\ \Downarrow \\ \nabla LCS(S_X, S_Y)) = 0 \\ \Downarrow \\ QSI(S_X, S_Y) = 0. \end{gathered} \tag{4.4}$$

El valor de $QSI(S_X, S_Y)$ aumenta de acuerdo al número de caracteres coincidentes. Este valor es 1 si $S_X = S_Y$.

$$\begin{gathered} \text{Si } LCS(S_X, S_Y)) = S_X \\ \Downarrow \\ \nabla LCS(S_X, S_Y)) = S_X \\ \Downarrow \\ QSI(S_X, S_Y) = 1. \end{gathered} \tag{4.5}$$

2) El tamaño de las cadenas también afecta a QSI. En este sentido, dos cadenas de un tamaño aproximadamente igual y que comparten un número determinado de símbolos, son más parecidas que otras dos con el mismo número de símbolos pero de tamaños muy diferentes, es decir,

$$\begin{gathered} \nabla S_X \approx \nabla S_Y, \nabla S_X << \nabla S_Z, \\ \nabla LCS(S_X, S_Y) \approx \nabla LCS(S_X, S_Z) \\ \Downarrow \\ QSI(S_X, S_Y) > QSI(S_X, S_Z) \end{gathered} \tag{4.6}$$

4.2. Características y Limitaciones

Esta definición de QSI tiene unas implicaciones claras sobre el concepto de similitud que se considera, como ya se vio en el capítulo 2. En este caso las caracterísitcas de esta similitud, derivadas de la normalizaión previa y la utilización del algortimo LCS, son:

- Los cambios de escala y los desplazamientos en el eje Y no afectan al índice de similitud.
- Coindencia de subcadenas aunque exista desplazamiento temporal. Un retraso o adelanto de una serie sobre otra no impide la detección de la similitud entre ellas, aunque el índice de similitud se reducirá proporcionalmente al tamaño relativo que presente el retraso o adelanto. Esto también permitirá detectar similitudes entre trozos de las series, siempre en el mismo orden en ambas, y con la limitación anterior.

Los inconvenientes más importantes de esta similitud son:

- No se penalizan los errores. Sólo se tienen en cuenta los aciertos, por lo que no afecta el grado de error en los casos no coincidentes. Se entiende por grado de error la distancia entre símbolos; la comparación del par de símbolos (H, M) (según la tabla 4.1) representando ambos a valores negativos puede considerarse más leve que el caso del par (H, h).
- No hay información sobre el tipo de aciertos. El hecho de que los aciertos estén distribuidos de una u otra forma puede tener implicaciones dependiendo de la aplicación. Con la distribución de los aciertos se hace referencia tanto a la forma en que están situados, su grado de agrupación, como a las zonas de las series en que se encuentran.
- Ajuste del parámetro. Con este método se ha concretado el número de intervalos en que se dividen los posibles valores cada serie pero no se ha proporcionado

ninguna regla para la determinación del parámetro δ. El valor de este parámetro influye en la calidad de los resultados, sin embargo aún no se ha realizado un estudio detallado sobre su importancia en los resultados ni sobre su relación con las series pertenecientes al dominio del problema.

- Tratamiento de valores anormales. La presencia de valores anormales en una serie puede desembocar en resultados inesperados. Este inconveniente se analiza en la siguiente subsección.

4.2.1. Problema de la normalización.

El paso de normalización que se ha incluido en la anterior definición debe ser usado con precaución. Las series a tratar deben estar libres de ruido así como de la inclusión de valores anormales, "outliers", debiendo el usuario garantizar esto por su propio conocimiento del sistema, del método de registro de datos o por un paso previo en que se eliminen estos elementos.

La razón de esta salvedad viene dada por las características de la normalización, que en caso de valores anormalmente grandes provocaría un aplastamiento general de toda la serie normalizada que podría llevar a que se asignaran de forma distinta un conjunto elevado de etiquetas.

En la figura 4.4 se incluyen dos series siendo la *serie*2 idéntica a la *serie*1 salvo los valores finales. Puede observarse en sus versiones normalizadas como la *serie*2 presenta, salvo en el tramo final, una reducción de escala con respecto a los valores de la *serie*1. Este cambio de escala puede hacer que la mayoría de las etiquetas que se asignen a los valores de ambas series sean diferentes, aunque las series presenten un gran parecido inicial.

En aquellas situaciones en que no se pueda asegurar una calidad de los datos, o la depuración datos anómalos no sea aceptable (por su inconveniente de eliminar algunos valores interesantes) puede plantearse la sustitución del procedimiento de normalización por uno de tipificación.

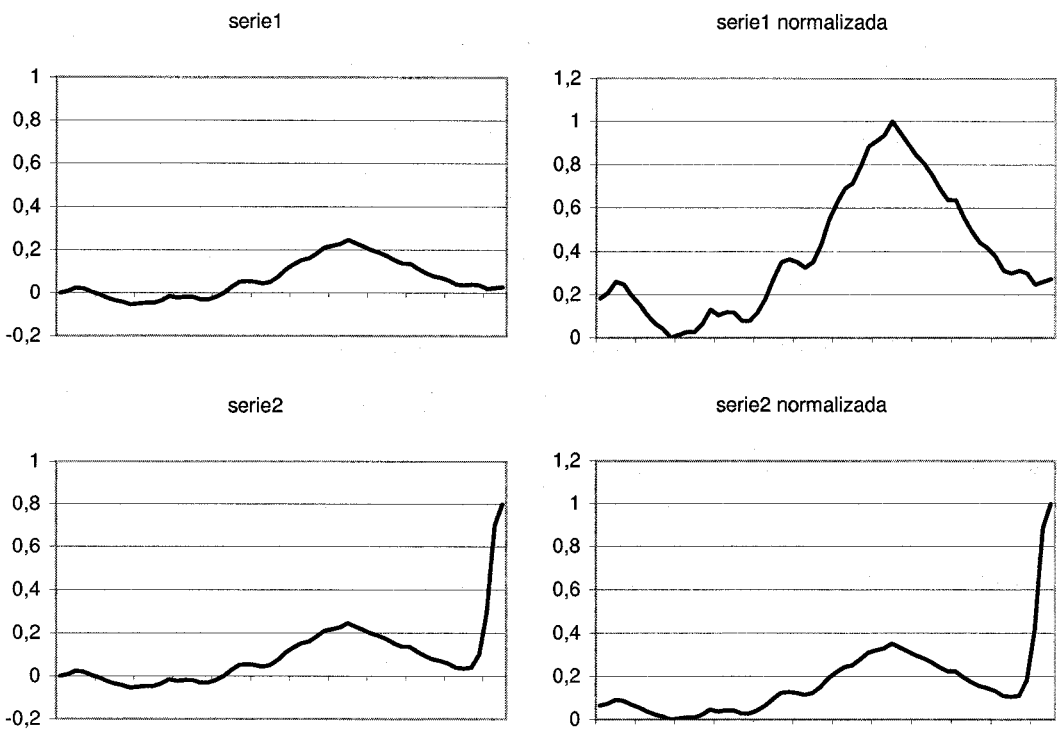


Figura 4.4: Problema de la normalización. Dos series que sólo difieren en los últimos valores presentan grandes diferencias entre sus versiones normalizadas.

En esta tipificación, donde a cada valor de la serie se le resta la media y se divide por la desviación típica, se garantiza que se recogen los desplazamientos en el eje Y al tiempo que las escalas de las diferentes series se aproximan.

Todo el procedimiento se mantendría idéntico salvo que los valores no estarían limitados al intervalo $[-1, 1]$. Así en caso de aplicar la tipificación sería necesario modificar los valores mínimo y máximo de los conjuntos que definen los intervalos de discretización de las series diferenciadas que pasarían a ser $[-\infty, +\infty]$.

CAPÍTULO 5

ETIQUETADOS. EL PROBLEMA DE LA DISCRETIZACIÓN

El procedimiento que en capítulos anteriores se ha identificado como *etiquetado* y consistente en representar con un grupo de etiquetas un rango de valores numéricos es reconocido en el campo de la estadística como *discretización.*

En este capítulo se realiza una revisión del problema de la discretización de variables continuas, las soluciones aportadas en la literatura, y su utilidad. Tras esta visión se incluye nuestra propuesta; el método de discretización *Ameva.* Junto a su definición también puede encontrarse una comparación con otros métodos de discretización.

Finalmente, y volviendo al caso general, se realiza un análisis teórico de la sensibilidad de un etiquetado a la posible presencia de ruido en los datos originales para concluir con una valoración práctica de esta sensibilidad por parte de los principales métodos expuestos.

5.1. Descripción del problema y antecedentes

La extracción de conocimiento y el procesamiento automático de datos son tareas importantes a realizar por los algoritmos de aprendizaje automático. Varios de estos algoritmos han sido específicamente diseñados para la manipulación de datos numéricos o nominales como el algoritmo CN2 [CN89, CB91], los algoritmos CLIP [CK01, CK02] y los algoritmos AQ [KM99, MMHL86b, MMHL86a], resultando que estos algoritmos no pueden aplicarse en conjuntos de datos mixtos (datos continuos y discretos), a menos que las características continuas sean previamente discretizadas.

Por otro lado, otros algoritmos producen mejores resultados con características discretas, independientemente de que puedan aplicarse sobre características continuas [Ker92, Cat91b].

A esta transformación del espacio de datos se le denomina *discretización*. La tarea de discretización de variables numéricas es bien conocida por los estadísticos. El proceso de discretización convierte atributos continuos en discretos, estableciendo intervalos en los cuales el valor del atributo puede residir, en lugar de valores individuales, y asociando a cada intervalo un valor numérico discreto.

Una característica deseable para una discretización práctica es que el atributo discretizado tenga el menor número posible de valores. Así, la discretización debería reducir significativamente el número de posibles valores del atributo continuo ya que un gran número de valores posibles de los atributos contribuye a un procesamiento lento y poco efectivo del aprendizaje automático inductivo [Cat91b].

La aproximación habitual para las tareas de aprendizaje que utilizan el modo mixto (continuo y discreto) es realizar la discretización previamente al proceso de aprendizaje como un paso de preprocesado [DKS95, Cat91b, FI92, Pfa95, ZR00]. Así, el proceso de discretización primero encuentra el número de intervalos discretos, y la anchura, o los límites de los intervalos, dado el rango de valores de un atributo continuo [GG01, DKS95, KC04a, MHBD03]. La discretización debería verse como un descubrimiento de aprendizaje en el que los valores críticos de un dominio continuo

deben ser identificados.

De todas formas las ventajas de la discretización en el periodo de aprendizaje no han sido demostradas aún, aunque se han realizado algunas comparaciones [DKS95]. Por ejemplo, en [MHBD03] se demostró que incluso sobre datos puramente numéricos los resultados de una clasificación textual realizada sobre una representación textual derivada de los datos originales mejoran los resultados de la presentación más sencilla de números como tokens y, más importante todavía, es competitiva con técnicas maduras de clasificación numérica como los métodos C4.5 [Qui93], Ripper [Coh95] y SVM [AG03, CST00, GAVV02, Gon02, SS02].

Ciertamente no existe un método universal que calcule una discretización óptima de un atributo continuo para todos los casos. Hay que considerar que existe una gran multitud de métodos de discretización en la literatura; desde los algoritmos no supervisados (algoritmos "ciegos", donde no hay información disponible de la clase a que pertenece el objeto que está siendo considerado) como los intervalos de igual amplitud, intervalos igual frecuencia [CWC91], el agrupamiento de las k-medias o MCC no supervisado[DKS95]; a los algoritmos supervisados (donde existe la información de la clasificación y puede tenerse en cuenta en la discretización de los datos) como *CAIM* [KC04a, KC04b], $F - CAIM$ [KC03], *ChiSplit* [BB81], *ChiMerge* [Ker92], *Chi2* [LS97, TS02], *Khiops* [Bou01], *CADD* [CWC95], máxima entropía [WC87, KS96], 1*R* [Hol93] o *CUM* [Coc92, GG01, COGV04]. Una larga lista puede encontrarse en [DKS95].

Nosotros definimos un nuevo método de discretización denominado *Ameva*, un método supervisado de discretización autónomo basado en la medida estadística Chi cuadrado (χ^2).

Existen otras aproximaciones basadas en Chi cuadrado, *ChiMerge* [Ker92], *ChiSplit* [BB81] and *Chi2* [LS97, TS02]. El algoritmo *ChiMerge* ha sido ampliamente usado y referenciado en trabajos de aprendizaje automático. Este algoritmo ha sido refinado en una nueva versión denominada *Chi2*.

Estos algoritmos, al igual que le ocurre al etiquetado incluido en el índice QSI y

presentado en la sección 4.1.2, necesitan que un experto proporcione algunos parámetros para su mejor uso.

Por un lado la especificación de los valores de estos parámetros puede ser difícil por múltiples razones: la existencia de un ruido importante en los datos, errores en las propias medidas, fallos en los datos como valores anormales, incluso la no disponibilidad de un experto, por ser un problema sobre el que no existe un conocimiento previo suficiente, o que, existiendo el experto, no sea capaz de plasmar de forma numérica su propio conocimiento.

Por otro lado, como se indica en [KLR04], estos parámetros que normalmente se presentan como una posibilidad de mejorar los resultados suelen, en la mayoría de los casos, ser determinantes y por tanto obligatorios de tener en consideración, requiriendo una optimización. Es más, se indica que los métodos automáticos para el ajuste de éstos parámetros tienen, a su vez, parámetros iniciales.

Otro inconveniente de la existencia de parámetros es la dificultad que éstos pueden representar en el intento de otros investigadores de repetir los experimentos publicados para su verificación y contraste.

Por otra parte, en los últimos algoritmos supervisados, el algoritmo debería maximizar la interdependencia entre los valores del atributo discreto y las etiquetas de clase, minimizando la pérdida de información debida a la discretización [KC04a]. Es más, la discretización en si misma puede verse como un proceso de descubrimiento de conocimiento en el que los valores críticos dentro de un dominio continuo deben ser identificados.

El algoritmo propuesto *Ameva* se ha comparado con el algoritmo $CAIM$ (del que puede encontrarse su descripción en el Apéndice C), uno de los algoritmos de discretización más relevantes, porque ambos comparten características similares y ninguno necesita que el usuario proporcione ningún parámetro.

5.2. Definición del problema

Ahora, se introducen algunas definiciones básicas.

Sea $X = \{x_1, \cdots, x_N\}$ un conjunto de datos de entrenamiento de un atributo continuos $\mathcal{X}$ proveniente de datos mixtos, tal que cada ejemplo x_i pertenezca a sólo una de las ℓ clases de las variables de clase denotadas por

$$\mathcal{C} = \{C_1, \cdots, C_\ell\}$$

La discretización de una variable continua es una función $\mathcal{D} : \mathcal{X} \rightarrow \mathcal{C}$ que asigna una clase $c \in \mathcal{C}$ a cada valor $x \in X$ en el dominio del atributo que está siendo discretizado.

Siempre es posible encontrar un esquema de discretización $\mathcal{D}$ en X que discretiza el dominio continuo del atributo $\mathcal{X}$ en k intervalos discretos:

$$\mathcal{L}(k; X; \mathcal{C}) = \{[d_0, d_1], (d_1, d_2], \cdots, (d_{k-1}, d_k]\}$$

tal que d_k es el máximo valor y d_0 es el mínimo valor del atributo X, y los valores d_i están en orden ascendente.

Sea L_1 el intervalo $[d_0, d_1]$ y L_j el intervalo $(d_{j-1}, d_j]$, $j = 2, \cdots, k$. Así, se puede definir una variable de discretización

$$\mathcal{L}(k; X; \mathcal{C}) = \{L_0, L_1, \cdots, L_k\}$$

y es verdad que para todo $x_i \in X$ existe un único L_j tal que $x_i \in L_j$ donde $i = 1, \cdots, N$ y $j = 1, \cdots, k$. La variable de discretización $\mathcal{L}(k; X; \mathcal{C})$ del atributo $\mathcal{X}$ y la variable de clase $\mathcal{C}$ son tratadas desde un punto de vista descriptivo, esto es son dos atributos discretos[1], de forma que puede construirse una tabla de frecuencias bidimensional (denominada tabla de contingencia) como se muestra en la tabla 5.1.

Por simplicidad, se denotará $\mathcal{L}(k; X, \mathcal{C})$ como $\mathcal{L}(k)$

[1]Es similar considerar ambos atributos como variables aleatorias. En éste caso, se utilizan probabilidades en lugar de frecuencias.

$C_i \mid L_j$	L_1	$\cdots$	L_j	$\cdots$	L_k	$n_{i\cdot}$
C_1	n_{11}	$\cdots$	n_{1j}	$\cdots$	n_{1k}	$n_{1\cdot}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
C_i	n_{i1}	$\cdots$	n_{ij}	$\cdots$	n_{ik}	$n_{i\cdot}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
C_ℓ	$n_{\ell 1}$	$\cdots$	$n_{\ell j}$	$\cdots$	$n_{\ell k}$	$n_{\ell\cdot}$
$n_{\cdot j}$	$n_{\cdot 1}$	$\cdots$	$n_{\cdot j}$	$\cdots$	$n_{\cdot k}$	N

Tabla 5.1: Tabla de contingencia: Tabla de frecuencias bidimensional para el atributo X y el esquema de discretización $\mathcal{L}(k)$

En la tabla 5.1, n_{ij} denotan el número total de valores continuos que pertenecen a la clase C_i que están dentro del intervalo $L_j = (d_{j-1}, d_j]$. $n_{i\cdot}$ es el número total de muestras que pertenecen a la clase C_i, y $n_{\cdot j}$ es el número total de muestras que pertenecen al intervalo L_j, para $i = 1, \cdots, \ell$ y $j = 1 \cdots, k$. De forma que:

$$n_{i\cdot} = \sum_{j=1}^{\ell} n_{ij}\,, \quad n_{\cdot j} = \sum_{i=1}^{k} n_{ij}\,, \quad N = \sum_{i=1}^{\ell}\sum_{j=1}^{k} n_{ij} \tag{5.1}$$

5.2.1. La discretización Ameva

Dada la tabla 5.1, definiremos el criterio de discretización basado en la coeficiente de contingencia (χ^2) que mide la independencia entre las variables de clase $\mathcal{C}$ y la variable de discretización $\mathcal{L}(k)$.

Es bien conocido en estadística [CH74, AL96], que dados dos atributos discretos $\mathcal{C}$ y $\mathcal{L}(k)$ son (estadísticamente) independientes si para todo $C_i \in \mathcal{C}$ y $L_j \in \mathcal{L}(k)$ se cumple que

$$n_{ij} = \frac{n_{i\cdot}\, n_{\cdot j}}{N}, \qquad i = 1, \cdots, \ell \quad j = 1, \cdots, k$$

En otro caso, existe una asociación entre ambos atributos.

Por tanto, una vía de medir la asociación (o interdependencia) entre la variable de clase $\mathcal{C}$ y la variable descripción $\mathcal{L}(k)$ es el estudio del valor

$$\sum_{i=1}^{\ell}\sum_{j=1}^{k}\left(n_{ij}-\frac{n_{i\cdot}\,n_{\cdot j}}{N}\right)^2$$

Pero, es mejor considerar una medida relativa, denotada por $\chi^2(k) \stackrel{def}{=} \chi^2(\mathcal{L}(k),\mathcal{C}|X)$:

$$\chi^2=\sum_{i=1}^{\ell}\sum_{j=1}^{k}\frac{\left(n_{ij}-\frac{n_{i\cdot}\,n_{\cdot j}}{N}\right)^2}{\frac{n_{i\cdot}\,n_{\cdot j}}{N}}$$

No es difícil probar, usando (5.1), que:

$$\chi^2(k)=N\left(-1+\sum_{i=1}^{\ell}\sum_{j=1}^{k}\frac{n_{ij}^2}{n_{i\cdot}\,n_{\cdot j}}\right) \tag{5.2}$$

y

$$\max_{X,\mathcal{L}(k),\mathcal{C}}\chi^2(k)=N\,(\min\{\ell,k\}-1) \tag{5.3}$$

Definiremos el coeficiente *Ameva* y lo denotaremos por $Ameva(k) \stackrel{def}{=} Ameva(\mathcal{L}(k),\mathcal{C}|X)$ como sigue:

$$Ameva(k)=\frac{\chi^2(k)}{k(\ell-1)} \tag{5.4}$$

para $k,\ell \geq 2$. El factor $(\ell-1)$ se analizará posteriormente.

Teniendo en cuenta [CK01], el valor del criterio *Ameva* se calcula sobre la tabla de contingencia. El coeficiente *Ameva* tiene las siguientes propiedades:

- El valor mínimo de $Ameva(k)$ es 0 y si este valor es alcanzado significa que los atributos $\mathcal{C}$ y $\mathcal{L}(k)$ son (estadísticamente) independientes.
- El valor máximo de $Ameva(k)$ significará la mejor correlación entre las etiquetas de clases y los intervalos discretos. Si $k \geq \ell$ entonces esto significa que para todo $x \in C_i$ existe un único j_0 tal que $x \in L_{j_0}$ (el resto de los intervalos $(k-\ell)$ no tiene ningún elemento), esto es, el valor máximo del coeficiente *Ameva* se alcanza cuando cada intervalo tiene todos valores agrupados en una sola etiqueta de clase.

- El valor acumulado es dividido por el número de intervalos k de forma que el criterio favorece esquemas con el menor número posible de intervalos.

El objetivo del algoritmo *Ameva* es maximizar la relación de dependencia entre las etiquetas de clase $\mathcal{C}$ y los valores continuos del atributo $\mathcal{L}(k; X; \mathcal{C})$, al tiempo que se minimiza el número de intervalos discretos.

Puede observase que el coeficiente *Ameva* depende de X,$\mathcal{L}(k)$ y $\mathcal{C}$. Es sencillo demostrar siguiendo (5.3) que $\max\limits_{X,\mathcal{L}(k),\mathcal{C}} Ameva(k) = Ameva_{max}(k)$ es

$$Ameva_{max}(k) = \begin{cases} \dfrac{N(k-1)}{k(\ell-1)} & \text{if } k < \ell \\ \\ \dfrac{N}{k} & \text{if } k \geq \ell \end{cases}$$

Así se tiene que $Ameva_{max}(k)$ es una función creciente de k si $k \leq \ell$, y una función decreciente de k si $k > \ell$. Además, se verifica que

$$\max_{k \geq 2} Ameva_{max}(k) = Ameva_{max}(\ell) = \frac{N}{\ell}$$

es decir, el máximo valor del coeficiente *Ameva* es alcanzado en la situación óptima (todos los valores de C_i están en único intervalo L_j y viceversa). Cuando $k > \ell$ se observa que se realiza una mayor penalización de la función $Ameva(k)$. También puede observarse que este máximo depende del número de clases ℓ. En la figura 5.1 se representa un ejemplo de $Ameva_{max}(k)$ para $k = 2, \cdots, 15$, $N = 200$ y $\ell = 5$.

5.2.2. El algoritmo *Ameva*

Se han desarrollado dos aproximaciones a la discretización basada en el coeficiente *Ameva*.

Primera aproximación: Habitualmente el problema de encontrar un esquema de discretización $\mathcal{L}(k)$ con un valor óptimo global del criterio *Ameva* es altamente combinatorio. La primera aproximación realiza la tarea de discretización con un

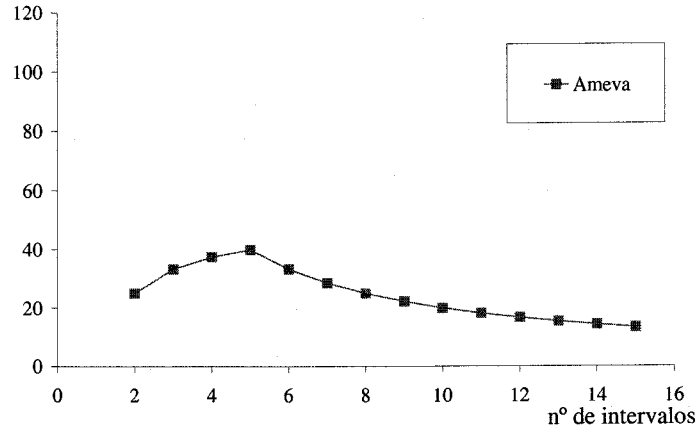


Figura 5.1: Valores de $Ameva_{max}$ en función de k, para $\ell = 5$ y $N = 200$.

costo computacional razonable, de forma que pueda aplicarse a atributos continuos con gran número de valores diferentes; y calcula un valor máximo local del criterio *Ameva*.

El algoritmo funciona con un esquema *top-down* dividiendo uno de los intervalos existentes en dos nuevos utilizando como criterio los resultados obtenidos en la búsqueda del valor óptimo de (5.4) después de la división, comenzando como único intervalo. El algoritmo tiene dos partes y se detallan en la figura 5.2.

Este algoritmo encuentra el menor número de intervalos que proporcionan la mejor relación "número de intervalos-coeficiente de asociación" basado en el valor de *Ameva*.

Segunda Aproximación: normalmente el número de etiquetas de clase es pequeño en algunos problemas y como el criterio *Ameva* proporciona un número reducido de intervalos es posible definir para estos problemas un algoritmo genético que encuentre el valor global máximo de los valores del criterio *Ameva* con un coste computacional que no sea excesivo.

Input: M ejemplos, S clases, F_i atributos continuos
Para cada F_i hacer:
Paso1.
1.1 buscar el valor máximo (d_n) y mínimo (d_o) de F_i
1.2 formar un conjunto con todos los valores diferentes de F_i en orden ascendente, inicializar los límites de los intervalos B con el mínimo, el máximo y los puntos intermedios de todos los pares adjacentes del conjunto.
1.3 establecer el esquema de discretización inicial como $D : [d_o, d_n]$, y hacer $GlobalAmeva = 0$
Paso2.
2.1 inicializar $k = 1$;
2.2 probar cada elemento de B, que no este en D, y calcular el correspondiente valor $Ameva$
2.3 después de probar todas las posibilidades elegir la que resulta el valor más alto de $Ameva$
2.4 si $Ameva > GlobalAmeva$ entonces actualizar D con elemento elegido en el paso 2.3 y hacer $GlobalAmeva = Ameva$, sino terminar
2.5 hacer $k = k + 1$ y pasar al punto 2.2
Output: Esquema de discretización D

Figura 5.2: Pseudocódigo del algoritmo *Ameva*.

5.3. El método *Ameva* frente a otros criterios

Es en esta sección donde se realizará la comparación del método *Ameva* con otros métodos desde un punto de vista teórico.

5.3.1. *Ameva* Versus *ChiSplit*, *ChiMerge* y *Chi2*

Los métodos *ChiSplit* y *Ameva* siguen un esquema *top-down*, empiezan con un intervalo inicial y recursivamente lo dividen en dos nuevos intervalos. Por su parte los métodos *ChiMerge* y *Chi2* utilizan un sistema *bottom-up* comenzando con un conjunto de intervalos con un único valor y continúan uniendo intervalos adyacentes iterativamente. Todos estos métodos están basados en el coeficiente estadístico Chi cuadrado y para todos es muy importante el siguiente lema.

Lema 5.3.1 *Sean $\mathcal{L}(k)$ y $\mathcal{L}(k+1)$ dos esquemas de discretización donde $\mathcal{L}(k+1)$ se obtiene por la división del intervalo $\mathcal{L}(k)$ en dos, entonces:*

$$\chi^2(k+1) \geq \chi^2(k) \tag{5.5}$$

Prueba: por simplicidad, tenemos k intervalos y suponemos que se divide el intervalo final en dos nuevos, así la tabla de contingencia de $\mathcal{L}(k)$ y $\mathcal{L}(k+1)$ son iguales a excepción de las columnas finales (véase la tabla 5.2 donde L_k^* es el intervalo k-ésimo de $\mathcal{L}(k)$, L_k y L_{k+1} son los intervalos k-ésimo y $k+1$-ésimo de $\mathcal{L}(k+1)$) y cumple algunas desigualdades:

$$\begin{aligned} n^*_{ik} &= n_{ik} + n_{ik+1}, \qquad i = 1, \ldots, \ell \\ n^*_{\cdot k} &= n_{\cdot k} + n_{\cdot k+1} \\ n^*_{i\cdot} &= n_{i\cdot}, \qquad i = 1, \cdots, \ell \end{aligned}$$

Siguiendo estas notaciones y (5.2) se tiene que $\chi^2(k+1) - \chi^2(k)$ es igual a

$$N \sum_{i=1}^{\ell} \left(\frac{n_{ik}^2}{n_{i\cdot} n_{\cdot k}} + \frac{n_{ik+1}^2}{n_{i\cdot} n_{\cdot k+1}} - \frac{(n^*_{ik})^2}{n_{i\cdot} n^*_{\cdot k}} \right)$$

y simplificando se tiene que

$$\chi^2(k+1) - \chi^2(k) = N \sum_{i=1}^{\ell} \frac{1}{n_{i\cdot}} \frac{(n_{ik} n_{\cdot k+1} - n_{ik+1} n_{\cdot k})^2}{n_{\cdot k} n_{\cdot k+1} (n_{\cdot k} + n_{\cdot k+1})}$$

y el lema queda probado.

	L_k^*		L_k		L_{k+1}
C_1	n_{1k}^*	$=$	n_{1k}	$+$	n_{1k+1}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
C_i	n_{ik}^*	$=$	n_{ik}	$+$	n_{ik+1}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
C_ℓ	$n_{\ell k}^*$	$=$	$n_{\ell k}$	$+$	$n_{\ell k+1}$
	$n_{\cdot k}^*$	$=$	$n_{\cdot k}$	$+$	$n_{\cdot k+1}$

Tabla 5.2: Comparación de la columna previa en la tabla de contingencia de X y las nuevas que la sustituyen $\mathcal{L}(k)$ and $\mathcal{L}(k+1)$ por la aplicación del esquema de discretización

Nota 5.3.2 *No es difícil probar que el lema 5.3.1 es verdad si el método une intervalos en lugar de dividirlos, y por simetría si se usa la unión o división de etiquetas en lugar de intervalos.*

Según (5.5) el coeficiente de contingencia crece cuando se dividen intervalos y disminuye cuando los intervalos son fusionados. ▲

Por otro lado, los algoritmos *ChiSplit*, *ChiMerge* y *Chi2* necesitan un criterio de parada que se utiliza para medir si la diferencia entre sucesivas interacciones es o no relevante. En estos algoritmos, el criterio de parada es elegido por un experto de entre un conjunto de parámetros con los problemas ya comentados. Opuestamente, vamos a mostrar que en el algoritmo *Ameva* el criterio de parada es seleccionado automáticamente.

De (5.5), se tiene que:

$$Ameva(k+1) = \frac{k}{k+1} Ameva(k) + \frac{AA(k+1)}{k+1} \tag{5.6}$$

donde

$$AA(k+1) = \frac{N}{(\ell-1)} \sum_{i=1}^{\ell} \frac{1}{n_{\cdot i}} \frac{(n_{ik} n_{\cdot k+1} - n_{ik+1} n_{\cdot k})^2}{n_{\cdot k} n_{\cdot k+1} (n_{\cdot k} + n_{\cdot k+1})}$$

Esto es, $Ameva(k+1)$ es una media aritmética ponderada de $Ameva(k)$ con un peso $\frac{k}{k+1}$ y $AA(k+1)$ como un peso $\frac{1}{k+1}$. Así, el criterio de palabra es automáticamente elegido utilizando (5.6), el paso 2.4 del algoritmo *Ameva* puede verse así

$$\chi^2(k) < (\ell-1)\, k\, AA(k+1)$$

y $AA(k+1)$ depende de X, $\mathcal{C}$, $\mathcal{L}(k)$ y $\mathcal{L}(k+1)$ es decir, depende del conjunto de datos, las clases y el esquema de discretización.

Por otro lado, es importante señalar que si $\mathcal{C}$ y $\mathcal{L}(k)$ son consideradas variables aleatorias entonces, puede probarse que la variable aleatoria $\chi^2(k)$ posee una distribución que es aproximadamente una Chi cuadrado para un N suficientemente grande. Esta aproximación es utilizada en [Bou01, Bou04] donde se define un nuevo método basado en Chi cuadrado denominado *Khiops*. Es un método *bottom-up* y el criterio de parada se basa en un nivel de confianza con Chi cuadrado.

5.3.2. *Ameva* Versus *CAIM*

Ahora, se realizará una comparación teórica entre *Ameva* y el reciente algoritmo *CAIM*.

El algoritmo *CAIM* es similar al algoritmo *Ameva* excepto que en el punto 2.4 presenta una segunda condición. La cláusula condicional completa pasa a ser ($CAIM > GlobalCAIM$ or $k < \ell$). Esta restricción hace que el algoritmo supere el hecho de que normalmente $CAIM(k) > CAIM(\ell+1)$ para todo $k = 2, 3, \cdots, \ell$, y garantiza que el esquema de discretización finalmente obtenido $\mathcal{L}(k)$ tiene, al menos, tantos intervalos como clases.

Esta restricción tiene un fundamento empírico, demostrando la experiencia que un número de intervalos igual al número de clases preserva mejor la información en

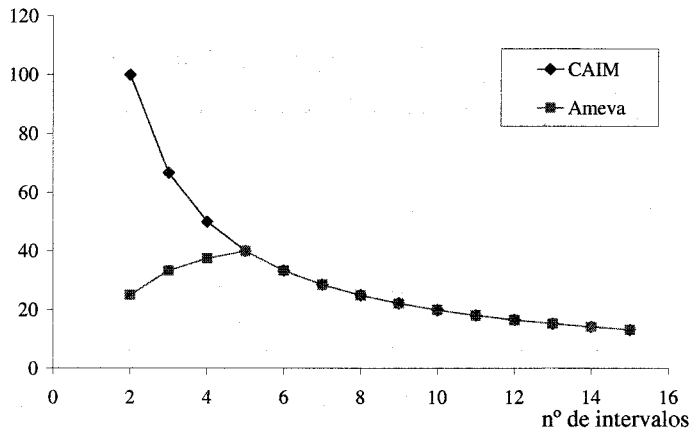


Figura 5.3: Valores de $Ameva_{max}$ y $CAIM_{max}$ en función de k, para $\ell = 5$ y $N = 200$.

tarea de clasificación [Kur04]. De todas formas, esto es un importante inconveniente en la implementación del algoritmo $CAIM$.

Un problema que presenta la definición del criterio $CAIM$ es que su máximo valor posible depende de k y se demuestra que

$$\max_{X,\mathcal{L}(k),\mathcal{C}} CAIM(k) = CAIM_{max}(k) = \frac{N}{k}$$

y

$$\max_{k=2,\cdots} CAIM_{max}(k) = \frac{N}{2} \tag{5.7}$$

Esta dependencia implica que el valor $CAIM_{max}(k)$ decrece hiperbólicamente, haciéndose más difícil localizar esquemas de discretización con un valor $CAIM(k) > CAIM(k-1)$ en cada incremento de k. Puede observarse en la figura 5.3 los valores máximos de $CAIM$ en función de k.

Por ejemplo, sea X_1 y X_2 dos muestras aleatorias con un tamaño $N = 500$ con una distribución uniforme y un número de clases $\ell = 10$ entonces la probabilidad de que el coeficiente $CAIM(10)$ para la muestra X_1 sea mayor que el coeficiente $CAIM(2)$ para la muestra X_2 es menor que 0,1 y esta probabilidad es tan pequeña por la construcción del algoritmo.

Por la misma razón, no es normalmente necesario continuar el proceso para $k > \ell$

porque casi siempre se tiene que $CAIM(\ell) > CAIM(\ell+1)$. Esto puede observarse en la figura 5.3, al igual que el hecho de que si $k \geq \ell$ los valores máximos de ambos coeficientes *Ameva* y $CAIM$ son iguales.

Por otro lado, este coeficiente es menos interpretable que el coeficiente *Ameva* porque la mínima asociación debe darse cuando los dos atributos son independientes. Así, si $\mathcal{L}(k)$ y $\mathcal{C}$ son independientes entonces $Ameva(k) = 0$ pero puede demostrarse que en ese caso

$$CAIM_{min\,asociacion}(k) = \frac{1}{kN} \max_{i=1,\cdots,\ell} n_{i\cdot}^2$$

El valor mínimo de $CAIM(k)$ es

$$CAIM_{min}(k) = \frac{N}{k\ell^2}$$

y este valor es alcanzado cuando $n_{ij} = \frac{n_{\cdot j}}{\ell}$ para todo $i = 1, \cdots \ell$. Así, si fuese un problema de clasificación este valor proporciona la peor situación posible (máxima entropía).

Al igual que con el coeficiente *Ameva*, se ha probado que

$$CAIM(k+1) = \frac{k}{k+1} CAIM(k) + \frac{1}{k+1} A(k+1)$$

donde

$$A(k+1) = \frac{\max_i n_{ik}^2}{n_{\cdot k}} + \frac{\max_i n_{ik+1}^2}{n_{\cdot k+1}} - \frac{\max_i (n_{ik} + n_{ik+1})^2}{n_{\cdot k} + n_{\cdot k+1}}$$

i.e. $CAIM(k+1)$ es una media aritmética ponderada de $CAIM(k)$ con un peso $\frac{k}{k+1}$ y $A(k+1)$ con peso $\frac{1}{k+1}$.

Otro problema que se deriva de la definición del criterio $CAIM$ es su falta de distinción entre situaciones diferentes. Sea un problema con tres clases diferentes C_1, C_2 y C_3 donde $n_{1\cdot} = 50$, $n_{2\cdot} = 250$ y $n_{3\cdot} = 50$ respectivamente. Entonces para las siguientes tablas de contingencia

	L_1	L_2	L_3
C_1	25	0	25
C_2	50	150	50
C_3	25	0	25

	L_1	L_2	L_3
C_1	50	0	0
C_2	50	150	50
C_3	0	0	50

el valor de $CAIM$ es el mismo para ambas tablas de contingencia y se puede ver que la segunda tabla es mejor que la primera para un problema de clasificación [(2)]. Además, es fácil calcular el valor del coeficiente *Ameva* para ambas tablas (17,5 y 46,667, respectivamente).

En la siguiente sección se realizan algunas comparativas entre ambas aproximaciones.

5.4. Comparación Práctica

Con la idea de clarificar las mejoras de *Ameva* frente a $CAIM$ se han aplicado ambos métodos a varios conjuntos de datos mixtos y de datos únicamente continuos. El algoritmo *C4.5* [Qui93] se utilizó para generar reglas de clasificación.

En ésta sección se presentan los resultados de la aplicación del algoritmo *Ameva* y el algoritmo $CAIM$ a doce conjuntos de datos continuos y de tipo mixto muy utilizados en la literatura.

En este trabajo no se ha incluido ninguna comparativa entre los resultados obtenidos por el algoritmo *Ameva* frente al algoritmo *ChiMerge* en funciones de clasificación. Pueden consultarse los resultados presentados en [Ris99] donde se realizó una comparativa entre cuatro algoritmos discretización que puede ser utilizada como contraste. Esta comparación puede hacerse al haberse utilizado en ambos trabajos los mismos conjuntos de datos. También se utilizan estos conjuntos en [Bou04], donde se comparan los resultados obtenidos con los algoritmos *Chi2*, *ChiSplit* y *Khiops* en tareas de clasificación.

[(2)]Debe reseñarse que $CAIM$ es sensible a los tamaños de cada clase.

Primero se realiza una comparación de *Ameva* y *CAIM* con los conjuntos de datos utilizados en [KC04a]. Para el segundo experimento se seleccionaron cuatro conjuntos con un número de clases superior al existente en los conjuntos utilizados en el primer experimento. El último experimento es una comparativa entre las dos aproximaciones del algoritmo *Ameva* que se comentarán más adelante.

En todos los experimento los algoritmos son aplicados a cada conjunto de datos obteniendo un nuevo conjunto discretizado siguiendo el esquema de validación cruzada de 10 pliegues. Los conjuntos discretizados son usados posteriormente en una tarea de aprendizaje realizada por el algoritmo *C4.5*.

La medida de la calidad de la discretización de ambos algoritmos vendrá dada por el número total de intervalos obtenidos, el tiempo invertido en el proceso de discretización, el número de nodos que presenta el árbol de decisión (generado por el algoritmo *C4.5*) y la precisión obtenida en la identificación.

5.4.1. Primer experimento

Los nombres de los conjuntos de datos utilizados y sus respectivas abreviaturas identificativas son:

- Iris Plants (IRI)
- John Hopkins Univ. Ionosphere (ION)
- Statlog Project Heart Disease data set (HEA)
- Pima Indians Diabetes (PID)
- Statlog Project Satellite Image (SAT)
- Thyroid Disease (THY)
- Waveform (WAV)
- Attitudes Towards Workplace Smoking Restrictions (SMO)

Las razones para la selección de este conjunto de datos es la existencia, en [KC04a], de una comparación previa, usando estos mismos conjuntos, del algoritmo *CAIM* frente a otros seis algoritmos de discretización muy extendidos, comprobándose su mayor capacidad.

Todos los conjuntos datos se obtuvieron del repositorio UC Irvine Machine Learning [BM98] excepto el último que fue obtenido del archivo de conjuntos de datos Statlib [Vla00]. La tabla 5.3 presenta las características de estos conjuntos.

	Conjuntos							
Propiedades	IRIS	SAT	THY	WAV	ION	SMO	HEA	PID
N° de clases	3	6	3	3	2	3	2	2
N° de ejemplos	150	6435	7200	5000	351	2855	270	768
N° de atributos	4	36	21	21	34	13	13	8
N° de atributos continuos	4	36	6	21	34	2	6	8

Tabla 5.3: Características de los conjuntos de datos usados en el primer experimento. En negrita el mejor resultado para cada conjunto y criterio.

La figura 5.4 muestran los valores de los criterios de calidad mencionados obtenidos por la aplicación de los algoritmos *CAIM* y *Ameva* para estos conjuntos de datos. Al realizarse el experimento por medio de un sistema de validación cruzada, los resultados incluyen el valor medio y de desviación estándar de cada valor.

En todos los conjuntos, debido a la definición de *CAIM*, el número de intervalos a obtener puede ser previamente estimado como resultado de multiplicar el número de atributos (s) por el número de clases ($s \cdot \ell$). En el conjunto de datos ION se produce una excepción a esta regla obteniéndose 67 intervalos frente al número teórico de 68 ($34 \cdot 2$); la razón de esto se encuentra en que uno de los atributos sólo presenta un valor numérico, que queda incluido dentro de un único intervalo.

Comparando el número de intervalos generados por los métodos *CAIM* y *Ameva*, podemos ver que ambos métodos igualan al producir cada uno el menor número

		Dataset															
Criterio	Método Disc.	IRIS		SAT		THY		WAV		ION		SMO		HEA		PID	
		Media	Desv.	Media	Desv.	Media	Desv.	Media	Desv.	Media	Desv.	Media	Desv.	Media	Desv.	Media	Desv.
N° total de intervalos	CAIM	12,0	,000	216,0	,000	18,0	,000	63,0	,000	**67,0**	,000	**6,0**	,000	**12,0**	,000	**16,0**	,000
	Ameva	**10,0**	,000	**100,4**	,699	**15,6**	,699	**47,1**	3,81	107,1	2,13	7,1	1,91	13,6	1,9	18,8	,919
T. ejecución (segundos)	CAIM	,012	,004	3,69	,011	,924	,007	9	,021	**0,76**	,006	**,115**	,006	**,031**	,003	**,192**	,004
	Ameva	**,010**	,000	**3,30**	,015	**,896**	,010	**7,44**	,363	1,49	,038	0,134	0,02	,042	,008	,226	,005
Tamaño árbol C4.5	CAIM	4,3	,949	884	67,8	29,3	8,29	493	43,2	21,2	4,37	1,9	1,45	31,8	4,71	**16,0**	4,74
	Ameva	4,3	,949	**528**	48,9	**17,6**	1,26	**319**	27,4	**18,7**	3,02	**1,00**	0	**31,2**	5,29	23,2	7,48
% precision C4.5	CAIM	93,3	5,4	**85,8**	1,9	98,7	0,52	76,2	2,6	**88,9**	5,5	69,5	0,2	80,8	4,9	72,9	3,3
	Ameva	93,3	5,4	82,8	1,4	**99,1**	0,34	**76,8**	1,3	88	4,6	**69,6**	0,1	80,8	6,7	**76,0**	3,7

Tabla 5.4: Comparación de resultados obtenidos por los algoritmos *CAIM* y *Ameva* en el primer experimento.

de intervalos en cuatro conjuntos, pero es interesante comprobar que el algoritmo *CAIM* lo hace en tres conjuntos de datos que sólo poseen dos clases y en uno que presenta tres. Los métodos tienen un comportamiento similar con una clara ventaja de *Ameva* en los conjuntos SAT y WAV, y un menor número de intervalos de *CAIM* en el conjunto ION.

Una situación análoga se encuentra en el tiempo de ejecución al implicar un menor número de intervalos un menor número de iteraciones y de evaluaciones.

Sólo desde la perspectiva del número de nodos incluidos en el árbol de decisión puede encontrarse un importante dominio de *Ameva*, siendo el mejor en seis, igualando en uno y perdiendo en otro. Esto es importante porque un menor número de nodos lleva a una implementación más rápida de la tarea de identificación. Evidentemente esto es más importante cuanto mayores sean los árboles obtenidos; en los conjuntos de datos SAT y WAV el algoritmo de aprendizaje *C4.5* aplicado sobre los conjuntos discretizados por *CAIM* genera árboles de 884 y 492 nodos mientras que en la aplicación sobre los conjuntos obtenidos con *Ameva* los tamaños de los árboles se reducen a 528 y 319 respectivamente.

Ha de señalarse que en el algoritmo *C4.5*, en general, un menor número de intervalos implica un tamaño menor en el árbol de decisión, pero esto no es una condición suficiente como puede observarse en los conjuntos ION, SMO y HEA.

Finalmente, la precisión es casi idéntica en seis conjuntos de datos con diferencias menores al 1 %. En los dos conjuntos de datos restantes existe una ventaja del 3 % para el método que produce un mayor número de intervalos, *CAIM* para SAT y *Ameva* para PID.

Los resultados muestran una posible relación, para estos ejempos, entre el número de clases y el número de intervalos obtenidos con ambos métodos: cuando el número de clases es bajo (dos en este caso) *CAIM* produce un menor número de intervalos, y cuando el número de clases aumenta el método que genera menos intervalos es *Ameva*. Este comportamiento se hace patente en la figura 5.4 donde se muestra, para cada conjunto de datos, el número de clases que presenta y el número de intervalos producidos por ambos métodos. El número de intervalos es un porcentaje relativo alcanzando el 100 % el método que produce el mayor valor.

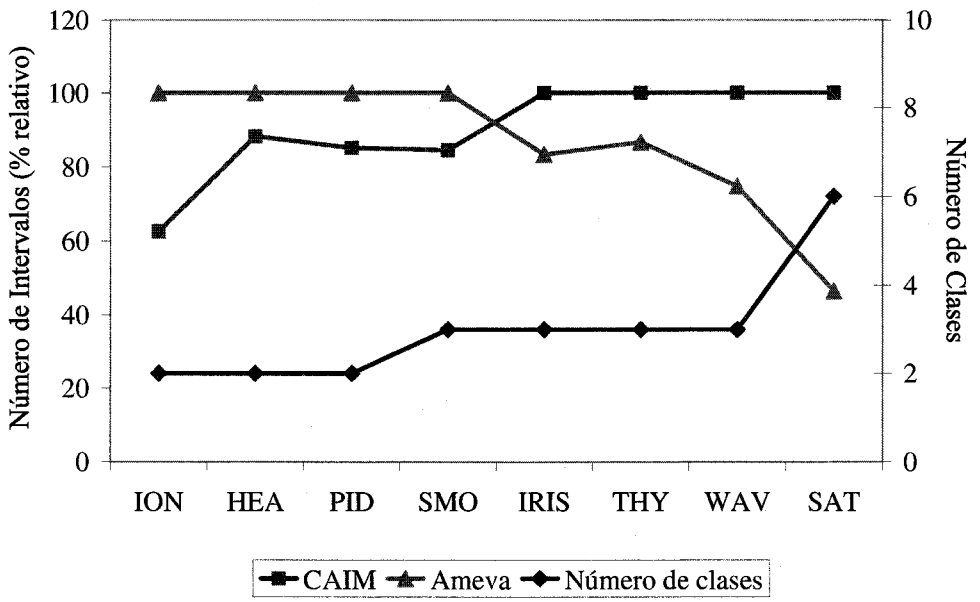


Figura 5.4: Relación entre el número de intervalos y el número de clases. Los intervalos se representan como porcentajes relativos.

Unas pequeñas diferencias se aprecian en términos del número de intervalos, el tiempo de ejecución y la precisión de clasificación de ambos métodos. La conclusión de este experimento debe ser que existe un comportamiento equivalente entre ambos.

5.4.2. Segundo experimento

Como fue establecido anteriormente, el algoritmo *CAIM* representa la restricción de producir, al menos, el mismo número de intervalos que el número de clases presente el conjunto de datos. La mayoría de los conjuntos utilizados en [KC04a] tienen dos o tres clases, con el conjunto SAT presentando seis. En estos casos (bajo número de clases), puede observarse en la tabla 5.4 y la figura 5.4 que el número de intervalos en el algoritmo *Ameva* es aproximado al proporcionado por el algoritmo *CAIM* (excepto para ION). Por otro lado, cuando el número de clases es mayor, el algoritmo *Ameva* produce un menor número de intervalos. Así, en esta segunda comparación se desea comprobar la influencia de la restricción del algoritmo *CAIM* cuando el número de clases aumenta.

Se han seleccionado cuatro nuevos conjuntos de datos del repositorio UCI ML. Estos conjuntos (y la abreviatura con la que serán identificados a partir de aquí) son los siguientes:

- Glass Identification Database (GLA)
- Image Segmentation database(SEG)
- Vehicle Silhouettes(VEH)
- Vowel Recognition (VOW)

Las características de los conjuntos de datos se incluyen en la figura 5.5 y la repetición del experimento anterior con estos nuevos conjuntos produce los resultados mostrados en la figura 5.6.

Los resultados muestran que con estos cuatro conjuntos de datos el algoritmo *Ameva* produce el menor número de intervalos porque no está influido por una restricción como la de $CAIM$ de generar un número intervalos igual o mayor el número de clases. Como el número de intervalos obtenido es menor, el número de operaciones del algoritmo también se reduce. Así, la implementación de *Ameva* presenta un tiempo de ejecución menor, con una reducción que oscila desde un 20 % en VEW a un 66 % en el conjunto VOW.

Al mismo tiempo un número reducido intervalos proporciona un árbol de menor tamaño en el algoritmo *C4.5*. El número de nodos obtenido con los conjuntos discretizados por *Ameva* son como mínimo la mitad de los obtenidos por $CAIM$ (alcanzándose una tercera parte en el conjunto VOW). El número de intervalos generados por $CAIM$ para el conjunto de datos SEG es menor que el valor teórico por la misma razón comentada previamente en el caso del conjunto ION.

Finalmente, la precisión es ligeramente mejor en los conjuntos discretizados por $CAIM$, mejorando los resultados de *Ameva* en tres y empeorando en uno. Las diferencias de exactitud son relevantes en el conjunto SEG (cercana al 4 %) y en el conjunto VOW (aproximadamente un 6 %).

Pero si la pérdida de exactitud, en estos conjuntos, es contrastada por la importante reducción del número de intervalos y del tamaño del árbol generado ($\pm$50 % para SEG y $\pm$70 % para VOW en ambos criterios) se puede concluir que, en algunas aplicaciones, esta reducción en la área de identificación es un inconveniente asumible

	Conjuntos			
Propiedades	GLA	SEG	VEH	VOW
Nº de clases	6	7	4	11
Nº de ejemplos	214	2310	846	990
Nº de atributos	9	19	18	10
Nº de atributos continuos	9	19	18	10

Tabla 5.5: Conjuntos de datos utilizados en el segundo experimento.

Criterio	Método Disc.	Dataset GLA Media	GLA Desv.	SEG Media	SEG Desv.	VEH Media	VEH Desv.	VOW Media	VOW Desv.
N° total de intervalos	CAIM	54	0	119,9	,316	72	0	110	0
	Ameva	**25,5**	1,84	**50,8**	,422	**44**	1,63	**30,2**	,919
T. Ejecución (segundos)	CAIM	,415	,005	64,7	,371	,538	,007	54,1	,192
	Ameva	**,280**	,036	**30,2**	,596	**,444**	,037	**16,2**	,543
Tamaño árbol C4.5	CAIM	58	11,7	187,2	20,1	164,2	27,0	635,7	47,5
	Ameva	**31,7**	4,0	**88,8**	12,7	**100,5**	10,4	**225,7**	30,6
% precisión C4.5	CAIM	67,3	6,63	**94,8**	1,69	**67,9**	2,22	**68,5**	3,92
	Ameva	**68,3**	6,89	90,5	2,26	67,4	3,89	62,5	5,07

Tabla 5.6: Comparación de los resultados obtenidos por los algoritmos CAIM y Ameva en conjuntos datos con un elevado número de clases. En negrita el mejor resultado para cada conjunto y criterio.

por su pequeña magnitud.

Esta reducción de la exactitud puede explicarse por medio de la pérdida de información que se produce al utilizar un menor número de intervalos. En la tabla 5.7 se muestran los resultados obtenidos cuando se incluye en la implementación de *Ameva* la restricción existente en *CAIM* de producir tantos intervalos como clases. A esta versión de algoritmo se le denomina $Ameva^R$.

Los resultados de $Ameva^R$ son más similares a los obtenidos por *CAIM*, presentando menores diferencias en los cuatro criterios seleccionados, que el algoritmo original *Ameva*.

En el número de intervalos la única diferencia es un 5 % de incremento para el conjunto de datos SEG. El tiempo de ejecución es el criterio donde se producen mayores variaciones con $Ameva^R$ empleando desde un 8 % menos a un 25 % más que *CAIM* en los conjuntos de datos SEG y GLA respectivamente.

Una mayor similitud se produce en el número de nodos de los árboles generados con tamaños que varían entre un 7 % menor a un 8 % mayor. El análisis de la

exactitud muestra unas diferencias muy pequeñas con *Ameva* siendo ligeramente mejor que *CAIM* en tres de los cuatro conjuntos.

		Dataset							
Criterio	Método Disc.	GLA		SEG		VEH		VOW	
		Media	Desv.	Media	Desv.	Media	Desv.	Media	Desv.
total # de intervalos	CAIM	**54**	0	**119,9**	,316	**72**	0	**110**	0
	$Ameva^R$	54,9	,738	127	0	72,1	,316	110,1	,316
T, ejecución (segundos)	CAIM	**,415**	,005	64,7	,371	**,538**	,007	54,1	,192
	$Ameva^R$	,554	,019	**59,0**	,316	,582	,007	**53,9**	,368
Tamaño árbol C4,5	CAIM	58	11,7	**187,2**	20,1	**164,2**	27,0	635,7	47,5
	$Ameva^R$	**54,6**	8,59	204,7	16,3	170,4	8,54	**601,9**	36,0
%precisión C4,5	CAIM	**67,3**	6,63	94,8	1,69	67,9	2,22	68,5	3,92
	$Ameva^R$	65,4	9,3	**95,0**	1,8	**68,1**	3,0	**70,8**	4,5

Tabla 5.7: Resultados obtenidos por $Ameva^R$ en conjuntos de datos con elevado número de clases.

Esta comparativa evidencia que la pérdida de información debida al menor número de intervalos seleccionados es la razón de la reducción en la capacidad de identificación por parte de *Ameva* en algunos conjuntos de datos.

Una cuestión abierta es si los límites de los intervalos seleccionados por *Ameva* son un subconjunto de los seleccionados por *CAIM*. Se realiza para ello una comparación entre los conjuntos de límites.

Como ambos algoritmos comparten un esquema top-bottom los límites exteriores son iguales. Del conjunto de límites interiores obtenidos por *Ameva* se calcula el porcentaje de los que aparecen en el conjunto proporcionado por *CAIM*. Los resultados, incluyendo media y desviación estándar, se muestran en la tabla 5.8.

De estos datos puede verse que, en la mayoría de los casos, al menos la mitad de los obtenidos por *Ameva* no aparecen en el conjunto de *CAIM*.

Así las conclusiones de este experimento son que con un número elevado de clases el método *Ameva* reduce drásticamente el número de intervalos, el tiempo de

Conjunto	Media	Desv.
IRIS	81,7	12,3
SAT	54,8	6,51
THY	52,4	8,88
WAV	39,5	7,98
ION	42,0	1,20
SMO	22	18,4
HAE	65,8	15,4
PID	27,8	8,48

Tabla 5.8: Porcentaje de límites interiores presentes en ambos esquemas de discretización del total existente en *Ameva*.

ejecución y el tamaño del árbol de clasificación. Por contra se produce una ligera reducción en la precisión de la clasificación, debida al menor número de intervalos utilizados. Finalmente se comprueba que los límites seleccionados por *Ameva* son distintos de los obtenidos con $CAIM$.

5.4.3. Tercer experimento

El problema de encontrar un esquema de discretización $\mathcal{L}(k)$ con un valor globalmente óptimo de *Ameva* es altamente combinatorio. De todas formas, se ha diseñado un algoritmo genético que encuentra esos valores máximos globales del criterio *Ameva*, para $k = 2, \ldots, 2\ell$, y ha sido aplicado a los conjuntos de datos del primer experimento. Una comparativa entre ambas aproximaciones se ha realizado obteniéndose los resultados presentados en la tabla 5.9.

Existen muy pocas diferencias entre ambas aproximaciones, la iterativa y la combinatoria, excepto en el tiempo de ejecución. El número de intervalos obtenidos por ambos algoritmos en los conjuntos discretizados es casi idéntico.

El tiempo de ejecución presenta un crecimiento exponencial. Como la aproxima-

Criterio	Método Disc.	Dataset															
		IRIS		SAT		THY		WAV		ION		SMO		HEA		PID	
		Mean	StdD.	Mean	StdD.	Mean	StdD.	Mean	StdD.	Mean	StdD.	Mean	StdD.	Mean	StdD.	Mean	StdD.
N° total de intervalos	Ameva	10,0	,000	100,4	,699	15,6	,699	**47,1**	3,81	107,1	2,13	**7,10**	1,91	**13,6**	1,90	18,8	,919
	A.Gen.	10,0	,000	100,4	,843	**15,5**	,972	47,2	1,55	**104,6**	2,07	8,10	,568	13,8	1,03	**18,6**	,699
T. ejecución (segundos)	Ameva	**,010**	,000	**3,30**	,015	**,896**	,010	**7,44**	,363	**1,49**	,038	**,134**	,020	**,042**	,008	**,226**	,005
	A.Gen.	2,74	,218	234,0	5,07	9,17	,406	54,7	15,9	13,6	,404	1,92	,203	1,34	,107	2,78	,275
Tamaño árbol C4.5	Ameva	**4,30**	,949	**527,9**	48,9	17,6	1,26	**318,7**	27,4	**18,7**	3,02	1,00	,000	31,2	5,29	**23,2**	7,48
	A.Gen.	4,50	1,58	565,2	43,7	**16,9**	3,70	337,0	18,8	20,4	4,35	1,00	,000	**26,9**	4,01	24,3	6,16
% precisión C4.5	Ameva	93,3	,056	**82,8**	,056	99,1	,056	76,8	,056	88,0	,056	69,6	,056	80,8	,056	**76,0**	,056
	A.Gen.	**96,0**	,056	81,7	,056	99,1	,056	**77,1**	,056	**89,4**	,056	69,6	,056	**81,9**	,056	75,6	,056

Tabla 5.9: Comparación entre el algoritmo incremental y el genético. En negrita el mejor resultado para cada conjunto y criterio.

ción combinatoria debe buscar el mejor valor para todos los posibles números de intervalos, y no se detiene cuando un valor es menor que lo obtenido en el paso anterior, el tiempo de ejecución crece dramáticamente, desde un factor 7x en el conjunto WAV a un factor 274x para el conjunto IRIS.

Los árboles generados por *C4.5* son mayores que los obtenidos por el *Ameva* original en cuatro conjuntos, iguales en uno y menores en tres si se aplica la aproximación combinatoria, pero las diferencias no son mayores del 10 %.

Las mejoras del algoritmo genético, en términos de precisión, son muy pequeñas. Sólo se encuentra mejoría en cuatro conjuntos de datos pero nunca superior al 3 % y normalmente menor que un 1 %.

Esto implica que el algoritmo iterativo, diseñado en [KC04b], encuentra una solución cuasi-óptima de los valores máximos de *Ameva*.

Con tan pequeña mejoría, si la hay, en la precisión y unos tiempos de ejecución tan dilatados no hay ninguna razón que sugiera una implementación práctica del algoritmo genético.

5.5. Estudio del ruido en la discretización

Aunque la hipótesis inicial, para la aplicación de las técnicas presentadas, es que las series no tienen ruido, analicemos lo que ocurriría en caso de que exista.

Evidentemente, la sensibilidad al ruido de QSI depende del proceso de etiquetado pero podemos analizar la sensibilidad común a cualquier división.

Sea $\overline{X} = \langle \overline{x}_1, ..., \overline{x}_T \rangle$ una serie temporal normalizada y un conjunto de valores que determinan los extremos de los intervalos de clases $L_0 < L_1, \cdots < L_k$ (k intervalos de clase, los extremos pueden ser valores infinitos). Primero, las diferencias entre dos valores consecutivos de la serie temporal:

$$p(t) = \Delta\overline{x}(t+1) = \overline{x}_{t+1} - \overline{x}_t, \qquad t = 1, \cdots, T-1$$

De esta nueva serie y de los extremos de los intervalos de clase, se calcula una nueva serie:

$$\epsilon(t) = \max_{L_i} \{|p(t) - L_i|, \, i = 1 \cdots, k\}, \qquad t = 1, \cdots, T-1$$

Esta serie verifica:

1. $\epsilon(t)$ está bien definido y existe para todo t.
2. $\epsilon(t) \geq 0$ para todo t.
3. Es verdad que:

$$\epsilon(t) \leq L = \frac{1}{2} \min \{L_i - L_{i-1}, \; i = 1, \cdots, k\}. \tag{5.8}$$

Esta serie temporal puede tratarse como una colección atemporal de valores. Se puede elegir un valor $0 \leq \alpha < 1$ y calcular el percentil de orden α del conjunto $\{\epsilon(t)\}_{t=1}^{T-1}$ y se indica como ϵ_α (ver la figura 5.5).

Con el estudio de la sensibilidad al ruido de la serie temporal $\overline{x}_t$ como nuestro objetivo, se considera una nueva serie normalizada, con la forma:

$$\widehat{x}_t = \overline{x}_t + u(t) \cdot p_\alpha$$

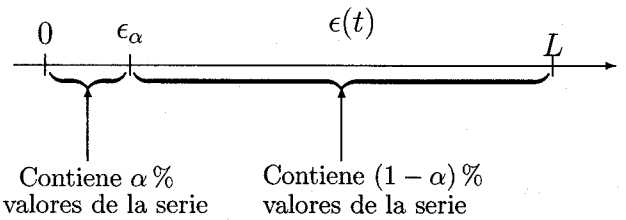


Figura 5.5: Percentil de la serie $\epsilon(t)$

donde

$$-1 \leq u(t) \leq 1, \forall t$$

y se calcula el correspondiente etiquetado de esta serie. Las diferencias son:

$$\begin{aligned} \widehat{p}(t) &= \Delta\widehat{x}(t+1) = \widehat{x}_{t+1} - \widehat{x}_t \\ &= \overline{x}_{t+1} - \overline{x}_t + (u(t+1) - u(t)) \cdot p_\alpha \end{aligned}$$

para $t = 1, \cdots, T-1$. Entonces tenemos:

$$\begin{aligned} |\widehat{p}(t) - L_i| &= |p(t) + (u(t+1) - u(t)) \cdot p_\alpha - L_i| \\ &\geq |p(t) - L_i| - |(u(t+1) - u(t)) \cdot p_\alpha| \\ &\geq |p(t) - L_i| - 2 \cdot p_\alpha \\ &= |p(t) - L_i| - \epsilon_\alpha = \epsilon(t) - \epsilon_\alpha, \end{aligned}$$

i.e.,

$$|\widehat{p}(t) - L_i| \geq \epsilon(t) - \epsilon_\alpha, \qquad \text{para } i = 1, \cdots, k$$

y por la definición de ϵ_α al menos el $(1-\alpha)$ % de las etiquetas asignadas a las series $\widehat{p}(t)$ coinciden en el mismo orden con las etiquetas de la serie $p(t)$.

De este razonamiento podemos concluir:

- Sea K la constante de normalización factor de escala utilizada en la normalización de la serie original $x(t)$, entonces en lugar de p_α se define

$$p_{1\alpha} = \frac{\epsilon_\alpha}{2\,K}.$$

- De esta forma podemos asignar a cada etiquetado un valor p_α del **nivel de ruido** que soporta con un grado de confianza $(1 - \alpha)$. Si el valor de p_α es relativamente alto, entonces tendremos un gran confianza en el etiquetado QSI proporcionado a la serie.

- Como en Estadística, podemos fijar, por defecto en el programa informático, el grado de significación α en un 5 %, y de esta forma el etiquetado tendría un grado de confianza para un nivel de error $p_{0,05}$ del 95 %.

- Si para valores altos de α, el valor de p_α tomase el valor cero, entonces se concluiría que el etiquetado QSI de la serie objeto de estudio es muy sensible al nivel de ruido.

Este estudio es válido para analizar el nivel de ruido soportado por una serie temporal para un esquema de etiquetado.

Ahora realizaremos una aplicación de este estudio a varios etiquetados en QSI.

5.5.1. Posibles etiquetados

Además de valorar la influencia del ruido en el etiquetado de QSI existe interés en conocer si es posible mejorar los resultados obtenidos utilizando etiquetados alternativos.

Con esta idea se han seleccionado los siguientes métodos:

- Método $CAIM$. Ya utilizado para contrastar la capacidad del método de discretización *Ameva*.

- *Amplitud Fija.* La experiencia muestra que en la división de un grupo de valores en rangos, o intervalos, con la misma amplitud es la división menos sensible al ruido. Seleccionando este método queremos verificar esta hipótesis en el etiquetado. Este método, como señala [Cat91a], es muy vulnerable a los valores anormales, *outliers*, que distorsionan el rango de valores.

- *Percentiles* o *Intervalos de igual frecuencia.* Este método está relacionado con el anterior y su aproximación es la de buscar intervalos que representen al mismo número de valores. Así cada símbolo tiene el mismo poder de representación en el conjunto de series. Los extremos de los intervalos son seleccionados por los correspondientes percentiles. Tanto este método como el anterior no utilizan las etiquetas de las instancias en la localización de los límites de las particiones, perdiéndose la información de la clasificación.

- *Método CUM.* Este método fue desarrollado en [Coc77] y se realizó su implementación en [GG01]. La división de los valores en intervalos se realiza minimizando la media de las desviaciones, con la condición de que todas las marcas de clase sean igualmente representativas. Este proceso se define basándose en técnicas estadísticas de muestreo y un estudio completo puede encontrarse en [Coc77] y [GG01].

A estos cuatro métodos se unirán los sistemas de etiquetado ya presentados en este proyecto de tesis; la definición original *QSI* y el método de discretización *Ameva*. Con todos ellos se realizará un análisis comparativo.

5.5.2. Un ejemplo de la afectación del ruido

La sensibilidad del ruido depende de las series originales y de los intervalos que abarcan los rangos que se definen para las etiquetas.

En este caso se utilizará un conjunto de series que representan la audiencia televisiva, cuota de pantalla, de las siete principales televisiones que operan en Andalucía durante el año 2003. Los datos han sido proporcionados por Canal Sur Televisión, una compañía perteneciente al Grupo Radio Televisión de Andalucía, y generados por [Sof].

Las series representan la cuota de pantalla media para intervalos de quince minutos, por lo que cada serie tiene una longitud de 96 elementos.

Se han seleccionado los datos correspondientes a los primeros 10 miércoles del año 2003 para realizar este análisis. Las series están etiquetadas con el nombre de la correspondiente cadena de televisión.

Este subconjunto de 70 series se divide en dos grupos de 35 conformándose uno como conjunto de aprendizaje, sobre el que realizar el procedimiento de localización de los límites de las etiquetas, y el segundo como conjunto de test o prueba, debiendo identificarse la clase a que pertenece cada una de sus series por la similitud que presenten con las series del conjunto de aprendizaje.

Para comprobar la influencia del etiquetado se utilizarán las técnicas de discretización comentadas. Aplicando estas técnicas al conjunto de datos se obtienen los siguientes límites de intervalos.

De la definición original de QSI con un $\delta = 5$ se obtiene el primer conjunto de límites de intervalos. Como este conjunto presenta siete etiquetas se utilizará este valor como parámetro de los algoritmos de amplitudes fijas y percentiles, de forma que ambos produzcan también siete divisiones.

La aplicación del método de discretización $CAIM$ también proporciona siete etiquetas al coincidir con el número de clases, cadenas de televisión, presentes en el conjunto de datos.

Finalmente la discretización realizada por el algoritmo *Ameva* sólo produce tres etiquetas y, debido a su definición, no hay mecanismos para imponer la obtención de un número mayor. De esta manera aunque los otros cuatro métodos podrán compararse con el mismo número de etiquetas, la aplicación de *Ameva* sólo se realizará con tres símbolos, lo que permitirá también ver la influencia de la reducción de símbolos en los resultados.

Los valores concretos de los límites de los rangos para cada etiquetado pueden verse en la columna central de la tabla 5.10.

El análisis se inicia presentando los valores medios de p_α, explicado en la sección anterior, del conjunto de series seleccionados para cada uno de los esquemas de

Método	Límites de intervalos	Éxitos Ident.
QSI	$-1, -0,2, 0,04, 0, 0, 0,04, 0,2, 1$	20
Ameva	-1,-0,0068,0,0068,1	21
CAIM	-1,-0,0238,-0,0161,-0,0068,0,0068,0,0161,0,0238,1	19
CUM	-1,-0,2165,-0,1046,-0,0366,0,0340,0,1006,0,2005,1	19
Percentiles	-1,-0,1212,-0,0435,0,0,0,0488,0,125,1	21
Amplitudes	-1,-0,7143,-0,4286,-0,1429,0,1429,0,4286,0,7143,1	15

Tabla 5.10: Diferentes etiquetados; límites de las divisiones y número de identificaciones correctas.

Método	p_α									
	,5	,45	,4	,35	,3	,25	,2	,15	,1	,05
QSI	,0500	,0417	,0375	,0333	,0300	,0265	,0219	,0175	,0156	,0133
Ameva	,1395	,1019	,0828	,0716	,0578	,0493	,0421	,0357	,0279	,0244
CAIM	,1310	,0934	,0743	,0631	,0493	,0407	,0335	,0272	,0193	,0159
CUM	,0404	,0221	,0170	,0170	,0170	,0170	,0170	,0166	,0144	,0123
Amplitudes	,0714	,0714	,0714	,0714	,0665	,0595	,0571	,0558	,0519	,0464
Percentiles	,0831	,0447	,0256	,0184	,0156	,0125	,0109	,0098	,0088	,0077

Tabla 5.11: Valores de p_α para cada método

etiquetado comentados, en la tabla 5.11.

Con el conjunto de procesos de etiquetados podemos verificar la calidad de cada uno. Esta calidad vendrá dada por el número de identificaciones correctas de la clase a la que pertenecen las series del conjunto de prueba.

En la columna final de la tabla 5.10 se presenta el número de identificaciones correctas del conjunto de prueba, sobre las 35 posibles, para cada etiquetado . Aplicando el índice QSI como medida de similitud con las regiones obtenidas con *Ameva* y con 7 percentiles se obtienen los mejores resultados, 21 aciertos. Valores ligeramente inferiores se producen con los etiquetados de QSI y con el proporcionado por

CAIM, quedando el método de amplitudes fijas muy distanciado.

Método	Porcentaje de símbolos						
	S1	S2	S3	S4	S5	S6	S7
QSI	5,8 %	31,2 %	0 %	24,0 %	9,4 %	24,0 %	5,6 %
Ameva	37,0 %	24,0 %	39,1 %	-	-	-	-
CAIM	35,0 %	1,4 %	0,5 %	24,0 %	0,4 %	1,5 %	37,1 %
CUM07	5,0 %	10,2 %	13,5 %	39,6 %	15,3 %	10,8 %	5,5 %
Percentiles7	12,8 %	14,9 %	9,2 %	24,0 %	11,0 %	16,1 %	12,0 %
Amplitudes7	0,2 %	0,5 %	10,3 %	78,9 %	9,4 %	0,6 %	0 %

Tabla 5.12: Distribución de símbolos

Una información importante es la distribución de símbolos producido por cada método de etiquetado. Estos se muestran en la tabla 5.12. Se observa que el método de amplitudes fijas casi no presenta apariciones en cuatro de los siete símbolos.

La primera evaluación de la influencia del ruido en las etiquetas puede obtenerse calculando el número etiquetas que difieren entre la serie original y la serie con ruido. Pero no sólo la cantidad sino también la magnitud de este cambio es importante. Así, definimos como niveles de salto de una etiqueta ruidosa"el número de posiciones que se distancia de la etiqueta original. Así todas las etiquetas que permanezcan invariables no presentarán ningún salto, o podremos hablar de un salto de nivel 0.

En la tabla 5.13 se incluye el porcentaje de etiquetas para cada nivel de salto y para cada nivel del ruido introducido. Se han utilizado niveles de ruido desde el 1 % al 10 %. El nivel de ruido se ha generado siguiendo una distribución normal y de desviación estándar igual al nivel de ruido. El carácter aleatorio que presenta esta generación ha sido contrarrestado con la repetición 10 veces del experimento.

Ha de reseñarse que, salvo el caso de división por percentiles, todos los métodos presentan un porcentaje superior al 55 % para un nivel de ruido del 10 %.

Como era de esperar el método de etiquetado menos influenciado por el ruido es el de las amplitudes fijas, seguido de *Ameva* que se ve favorecido por el hecho de presentar sólo tres etiquetas.

Nivel Ruido	Saltos 0	1	2	3	4	5	6
QSI							
1%	74,0	14,1	11,9	0,02	0	0	0
2%	71,9	15,7	12,0	0,36	0,02	0	0
3%	69,7	17,2	12,0	0,97	0,09	0	0
4%	67,4	18,1	12,5	1,72	0,23	0	0
5%	65,4	19,2	12,6	2,29	0,54	0	0
6%	63,3	20,1	12,8	2,92	0,92	0	0
7%	61,3	21,0	13,0	3,22	1,42	0,01	0
8%	59,6	21,6	13,1	3,66	2,04	0,02	0
9%	58,2	22,0	13,5	3,87	2,46	0,04	0
10%	56,1	23,1	13,5	4,47	2,85	0,07	0
CAIM							
1%	94,9	4,49	,52	0,09	0	0	0
2%	90,4	6,66	1,87	0,94	0,09	0,03	0,02
3%	87,7	6,96	2,72	2,12	0,26	0,15	0,11
4%	85,3	7,18	3,31	3,17	0,41	0,35	0,32
5%	83,4	7,20	3,51	4,12	0,60	0,47	0,75
6%	81,8	7,01	3,84	4,82	0,80	0,62	1,09
7%	80,0	7,19	3,77	5,56	0,93	0,84	1,76
8%	78,9	6,72	3,73	6,29	1,09	0,98	2,25
9%	77,7	6,61	4,02	6,59	1,23	1,09	2,78
10%	76,6	6,51	3,85	7,09	1,23	1,28	3,48
Percentiles7							
1%	73,5	26,5	0,02	0	0	0	0
2%	70,0	29,5	0,42	0,03	0	0	0
3%	66,0	32,6	1,24	0,11	0	0	0
4%	62,3	35,2	2,16	0,38	0,02	0	0
5%	59,2	36,8	3,07	0,83	0,11	0	0
6%	56,2	38,3	3,95	1,31	0,23	0,02	0
7%	53,8	39,5	4,49	1,68	0,42	0,04	0
8%	52,0	39,6	5,52	2,20	0,59	0,11	0,01
9%	49,4	40,4	6,11	2,91	0,96	0,18	0,01
10%	47,7	40,8	6,63	3,29	1,20	0,33	0,02

Nivel Ruido	Saltos 0	1	2	3	4	5	6
Ameva							
1%	98,2	1,79	0	-	-	-	-
2%	95,4	4,45	0,20	-	-	-	-
3%	93,1	6,30	0,62	-	-	-	-
4%	91,3	7,38	1,29	-	-	-	-
5%	89,6	8,43	1,97	-	-	-	-
6%	87,7	09,3	2,97	-	-	-	-
7%	86,4	09,7	3,88	-	-	-	-
8%	85,3	10,2	4,54	-	-	-	-
9%	84,1	10,4	5,49	-	-	-	-
10%	83,1	10,7	6,23	-	-	-	-
CUM07							
0,01	95,4	4,59	0	0	0	0	0
0,02	90,8	09,2	0	0	0	0	0
0,03	86,6	13,4	0,07	0	0	0	0
0,04	82,4	17,3	0,28	0,01	0	0	0
0,05	78,6	20,6	0,71	0,02	0	0	0
0,06	75,4	23,3	1,28	0,03	0	0	0
0,07	71,6	26,2	2,06	0,15	0,02	0	0
0,08	69,3	27,4	2,92	0,30	0,04	0	0
0,09	66,7	28,7	3,94	0,52	0,08	0,02	0
0,10	65,1	29,4	4,63	0,72	0,11	0,02	0
Amplitudes7							
1%	98,8	1,21	0	0	0	0	0
2%	97,8	2,21	0	0	0	0	0
3%	96,7	3,27	0	0	0	0	0
4%	95,8	4,21	0	0	0	0	0
5%	94,8	5,17	0	0	0	0	0
6%	93,6	6,44	0	0	0	0	0
7%	92,3	7,75	0	0	0	0	0
8%	91,2	8,82	0	0	0	0	0
9%	90,1	9,90	0,01	0	0	0	0
10%	89,1	10,9	0,02	0	0	0	0

Tabla 5.13: Porcentaje de saltos de etiquetas en presencia de ruido

Pero como ya se indicó anteriormente la capacidad de identificación de las series del conjunto de test es más importante que las posibles modificaciones de símbolos en las series con ruido. De manera que se realiza un experimento de identificación para cada etiquetado y nivel de ruido.

Las figuras de la 5.6a a 5.6f muestran el número de identificaciones correctas de cada método para los diferentes niveles de ruido. Como el ruido es introducido de una forma aleatoria se presentan los valores medio, máximo y mínimos obtenido para cada etiquetado.

Como era de esperar, el esquema de etiquetado que concentra un alto número de etiquetas en pocos intervalos no es influenciado por la presencia del ruido. Así en los resultados se puede ver que el método de amplitudes fijas mantiene los resultados estables, si analizamos el valor medio de identificaciones, ya que los valores mínimo y máximo pueden tener una componente aleatoria debida al ruido.

Los demás métodos presentan una mayor variabilidad de los resultados en función del nivel de ruido introducido y, salvo el método de etiquetado original de QSI que proporciona valores mínimos muy reducidos, presentan un comportamiento similar. De entre todos estos métodos los valores obtenidos con el etiquetado *Ameva* son ligeramente mejores que el resto de métodos.

El elegir tres etiquetas en *Ameva* puede favorecer su robustez, como en el caso de amplitudes fijas, pero la correcta selección de los intervalos proporciona uno valores de identificación muy superiores a los que se obtienen con dicho método.

Tendremos que concluir diciendo que la presencia de ruido en el proceso de identificación tiene influencia, en porcentaje de reducción, similar a nivel del ruido. Ha de identificarse esta influencia como reducida, no existiendo un nivel a partir del cual los resultados se reduzcan drásticamente. Los experimentos han sido realizados, aunque no se incluyen aquí los datos, con niveles de ruido de hasta un 30 % y el comportamiento se mantiene en una forma lineal.

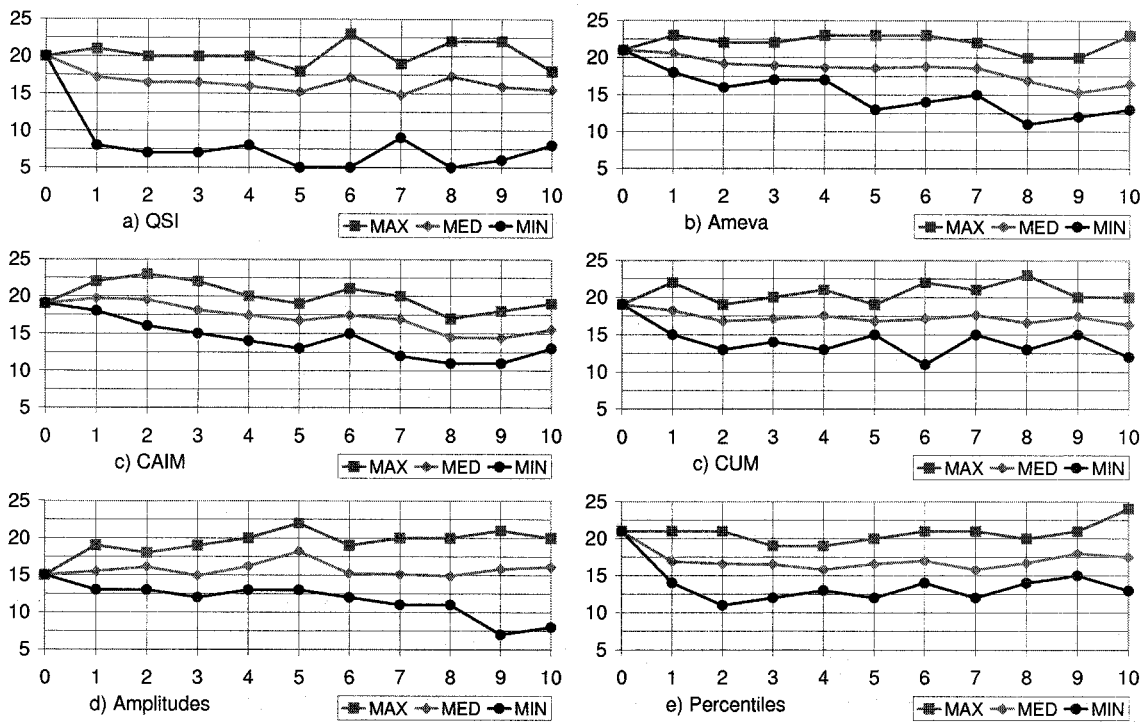


Figura 5.6: Identificaciones correctas vs. nivel de ruido.

5.6. Conclusiones

Se ha presentado un nuevo algoritmo para la discretización de variables continuas caracterizado por la selección de un reducido número de intervalos y sus límites.

El algoritmo *Ameva* utiliza la información de interdependencia clase-atributo (una medida estadística que cuantifica la interdependencia entre las clases y los atributos discretizados) como el criterio para una óptima discretización, la cual tiene el menor número de intervalos discretos y la mínima pérdida de información de la interdependencia. El algoritmo incluye un criterio de parada que depende de los datos sin intervención del usuario.

El algoritmo *Ameva* utiliza una aproximación incremental, que localiza un valor máximo local del criterio *Ameva*, con una fundamentación teórica en la función χ^2 y un criterio de parada automático que favorece números reducidos de intervalos.

Comparado con un algoritmo de buenos resultados, conocido y verificado, como *CAIM*, la aproximación presentada obtiene resultados muy similares en tareas de clasificación, de conjuntos de datos discretizados, basadas en el algoritmo *C4.5*.

Cuando el número de clases en los conjuntos de datos aumenta, *Ameva* obtiene conjuntos de intervalos muy reducidos que favorecen tiempos de ejecución y árboles de clasificación menores que los generados por *CAIM*. Por contra, es posible que la exactitud en la identificación se vea ligeramente reducida en algunas situaciones, pero siempre se garantiza una reducción drástica en los otros tres criterios manejados.

Los conjuntos de límites seleccionados por *Ameva* son distintos de los obtenidos con *CAIM*.

La implementación genera una solución cuasi óptima en una fracción del tiempo de ejecución utilizado por la aproximación combinatoria realizada mediante un algoritmo genético.

Con respecto al número de intervalos discretos, ambos algoritmos proporcionan un número pequeño cuando el número de clases es reducido (dos o tres), pero *Ameva* siempre proporciona el menor número de intervalos discretos si el número de clases es superior (mayor 3).

En conclusión, se ha presentado un algoritmo mejorado para tareas de clasificación con un alto número de clases que necesiten una respuesta rápida, aceptando incluso una mínima reducción de la precisión. En otras situaciones nuestro algoritmo es, al menos, de prestaciones similares a las de *CAIM*.

Por otra parte el análisis de la sensibilidad a la presencia de ruido que presenta el método de discretización *Ameva*, cuando se utiliza como mecanismo de etiquetado de la similitud *QSI*, determina que tiene un comportamiento robusto y obtiene los mejores resultados de los métodos comparados.

CAPÍTULO 6

KERNEL ENTRE LITERALES

Una vez que se han etiquetado las series y se han convertido en cadenas de símbolos, por medio de un mecanismo de discretización, se plantea la decisión de elegir un algoritmo para comparar dichas cadenas.

Dentro de todos los algoritmos existentes para la comparación de cadenas se tienen: los basados en la subsecuencia común máxima, las innumerables versiones basadas en la distancia de edición y últimamente las técnicas basadas en las funciones núcleo (FALTAN REFERENCIAS) aplicadas a la comparación de textos. Algunos de estos algoritmos ya han sido comentados en el capítulo 3.

En esta tesis se propone una nueva aproximación al problema de comparar cadenas de caracteres basada en las funciones núcleos, que está específicamente diseñada para llevar a cabo la comparación entre cadenas procedentes de un proceso discretización de una característica continua, en nuestro caso de series temporales. De ésta forma, esta aproximación tiene en cuenta que las letras del alfabeto (conjunto de símbolos) siguen una determinada escala ordinal (orden de magnitud) y por tanto puede aplicarse a un problema más general.

6.1. Nucleos

El término núcleo (kernel) proviene del uso de estos tipos de funciones, en el campo de los operadores integrales, por Hilbert y otros investigadores en la primera decena del siglo XX. Así una función $K(\cdot,\cdot)$ la cual viene de un operador T_K a través de

$$(T_K f)(x) = \int_{\mathcal{X}} K(x,x')f(x')dx'$$

donde $\mathcal{X}$ es el dominio de trabajo, se le denomina núcleo (kernel) T_K.

Los núcleos pueden ser considerados como una generalización de los productos escalares. Así, se tiene que cualquier producto escalar es un núcleo, sin embargo la linealidad en los argumentos, la cual es una propiedad intrínseca del producto escalar, no se cumple necesariamente con los núcleos [SS02].

Las funciones núcleos son utilizadas en diferentes campos de investigación (Teoría de Operadores Integrables, Aproximaciones Espacio de Hilbert, etc) enmarcándose el enfoque que se sigue en este trabajo dentro de la teoría del Aprendizaje Estadístico [Vap98]. En este contexto, el objetivo de la construcción de funciones núcleo es asegurar la existencia de una aplicación ϕ definida del conjunto de trabajo, $\mathcal{X}$ (el cual no necesariamente está provisto de ninguna estructura matemática a priori) a un espacio vectorial dotado de un producto escalar denominado espacio de características, $\mathcal{F}$, donde el producto escalar será denotado por $\langle\cdot,\cdot\rangle_{\mathcal{F}}$.

A partir de esta función ϕ, en general no lineal, se define una función núcleo, que se denota $k(\cdot,\cdot)$ sobre pares de elementos del conjunto de trabajo como el producto escalar de sus transformados dentro del espacio de características,

$$k(\cdot,\cdot) = \langle\phi(\cdot),\phi(\cdot)\rangle_{\mathcal{F}} \tag{6.1}$$

Para diferenciar las distintas definiciones de funciones núcleos que aparecen en distintas áreas de investigación a este tipo de núcleos se les conocen como núcleos de Mercer.

De esta manera, la función núcleo $k(\cdot,\cdot)$ permite establecer similitudes entre los elementos originales a partir de sus transformados, lo que conlleva en ocasiones poder definir una distancia entre los puntos origen. La aplicación ϕ, por tanto, debe ser capaz de resaltar las características fundamentales de los elementos del conjunto inicial que deben ser consideradas a la hora de elaborar una medida de similitud y distancia; es por ello que al espacio imagen de la aplicación ϕ se le denomina espacio característico o espacio de características [Gon02, GVG05].

Un estudio detallado en castellano de las funciones núcleos puede encontrarse en [Gon02] y [Ang01]. Existen diferentes textos en inglés entre ellos [Vap98, CST00] sobresaliendo entre ellos [SS02].

Veamos ahora como puede hacerse compatible esta definición de núcleos con la utilización de literales provenientes de una discretización numérica.

La discretización de una variable continua, asigna una etiqueta a un intervalo de valor, de ésta forma si queremos establecer una medida entre etiquetas debemos saber cuantificar medidas sobre intervalos y esto se analiza en la siguiente sección.

6.1.1. Nucleo entre intervalos

Esta sección sigue principalmente los trabajos sobre distancia núcleo intervalar dados en [GAVV02] y [GVA+04]. Sea $\mathcal{I}$ la familia formada por todos los intervalos (a,b)[1] contenidos en la recta real de dimensión finita

$$\mathcal{I} = \{(a,b) \subset, \mathbb{R} : a < b, a \neq -\infty, b \neq +\infty\} \tag{6.2}$$

Aunque ésta es la forma natural de expresar conjuntos de intervalos desde este punto se denotarán los intervalos de la forma $I = (c-r, c+r)$ (forma de Borel) donde c es el centro del intervalo y r es el radio puesto que las características que se incluyen a continuación se expresan de una forma más compacta. Se define entonces

[1] Aunque inicialmente se trabajará con intervalos abiertos el estudio puede trasladarse de forma natural a intervalos cerrados.

la función ϕ :

$$\phi : \mathcal{I} \rightarrow \mathbb{R}^2$$

$$\phi(I) = A \begin{pmatrix} c \\ r \end{pmatrix} \tag{6.3}$$

donde A es una matriz 2×2 simétrica con determinante no nulo.

Teniendo en cuenta la definición de núcleo, ver (6.1), se sigue que

$$k(I_1, I_2) = \begin{pmatrix} c_1 & r_1 \end{pmatrix} S \begin{pmatrix} c_2 \\ r_2 \end{pmatrix} \tag{6.4}$$

y se puede definir una medida de distancia entre intervalos como:

$$d_I^2(I_1, I_2) = \begin{pmatrix} \Delta c & \Delta r \end{pmatrix} S \begin{pmatrix} \Delta c \\ \Delta r \end{pmatrix} \tag{6.5}$$

siempre que $I_2 = (c_1 - r_1, c_1 + r_1)$ e $I_2 = (c_2 - r_2, c_2 + r_2)$, $\Delta c = c_2 - c_1$ y $\Delta r = r_2 - r_1$. La matriz A debe ser una matriz 2×2 no singular con objeto de que ϕ sea una aplicación inyectiva y $S = A^t A$ sea una matriz simétrica y definida positiva; con lo cual se tiene garantizado que la función k es un núcleo y d una distancia.

El significado de la matriz S es poder controlar el peso que corresponde dar a la posición de los intervalos, c, y a su tamaño, r, según él investigador.

La utilidad de este núcleo viene motivada por el hecho de que la conversión de una característica continua en etiquetas a partir de la construcción de diferentes intervalos de clase nos posibilita tomar como distancia entre etiquetas, aquella existente entre los intervalos a que se asignan las etiquetas.

Como ya se indicó, en [GVA+04] se encuentra más detalladamente esta sección, todo ésto nos llevará a la formulación de núcleo entre literales procedentes de un proceso de discretización [GVC+04].

6.1.2. Construcción de un núcleo entre literales

Sea un alfabeto compuesto por l símbolos que denotamos:

$$\mathcal{A} = \{A_1, A_2, \ldots, A_l\} \tag{6.6}$$

y sea $\mathcal{P}$ el conjunto de todas las palabras posibles que se pueden formar con ese alfabeto. Consideramos dos palabras P_1 y P_2 de $\mathcal{P}$ las cuales serán denotadas como:

$$P_1 = P1_1 P1_2 \ldots P1_n \qquad P_2 = P2_1 P2_2 \ldots P2_m \tag{6.7}$$

con $n \geq m$, y $P1_i, P2_j \in \mathcal{P}$.

A partir de aquí, consideramos que la discretización se ha llevado a cabo utilizando letras del alfabeto en lugar de símbolos, ya que esta forma se hace más patente la existencia de una escala de orden entre las etiquetas.

Dentro del proceso de etiquetado de una serie temporal es muy importante saber que en el lenguaje que se obtiene del proceso de etiquetado cada palabra tiene un significado concreto, ya que representa a una familia de series temporales (clase de equivalencia), y es por ello por lo que nos debemos preguntar cuales son las características que deseamos tener en consideración en cada palabra del lenguaje con objeto de saber interpretar el significado de éstas. Nosotros consideramos las siguientes:

- **El orden de las letras en cada palabra:** Esto es fundamental en el análisis de una serie temporal y basta observar en la figura 6.1 que la palabra $ACDBEAE$ no es la misma que $EABEDCA$ a pesar de que tienen las misma letras en diferente orden y evidentemente cada una de ellas reflejan distintas familias de series temporales. Por tanto el orden debe ser tenido en consideración en la elaboración del índice.
- **El tamaño de las palabras:** Claramente no es lo mismo una palabra con tres que con seis letras ya que, en general y refiriéndolo al caso particular de las series temporales, a mayor tamaño más información proporciona la serie.

$d(\cdot,\cdot)$	A	B	C	D	E
A	0	1	2	3	4
B	1	0	1	2	3
C	2	1	0	1	2
D	3	2	1	0	1
E	4	3	2	1	0

Tabla 6.1: Escala de distancias entre letras. Cuantificación de saltos

- **Comparativa letra a letra:** Es de sentido común que no debe ser lo mismo la cuantificación de la distinción entre las letras del alfabeto ya que provienen de un orden de magnitud, es decir, entre las letras existe una escala ordinal: $A < B < C < D < E$ y por tanto existe una mayor proximidad entre A y B que entre A y E.

De esta forma, siguiendo las características anteriormente apuntadas se va a construir un índice que cuantifique parecidos (similitudes) entre series. En primer lugar, señalamos que para cuantificar inicialmente la diferencia entre letras seguimos la escala más simple y natural posible:

Donde se refleja el salto que dentro de la escala ordinal se necesita para pasar de una letra a otra. Las escalas consideradas son simétricas, esto trae como consecuencia la natural simetría de la distancia que se define posteriormente. Evidentemente se podría modificar esta premisa, lo que se analizará en un trabajo futuro.

6.1.3. Palabras de igual tamaño

Siguiendo este esquema de trabajo veamos un ejemplo concreto que nos servirá de referencia. Calculamos el número de saltos necesarios para pasar de la palabra $ACDBEAE$ a la palabra $EABEDCA$ (las representadas en la figura 6.1) a

partir de la escala dada en la tabla 6.1.

A	C	D	B	E	A	E	
4	2	2	3	1	2	4	$\longrightarrow$ 16 saltos
E	A	B	E	D	C	A	

Si fijamos la primera palabra (evidentemente es indiferente fijar una u otra palabra, es decir, el planteamiento es simétrico) y reducimos en una unidad las diferencias entre cada una de las letras de ésta con las letras de la segunda palabra se obtiene una nueva palabra $DBCDEBB$: la cual evidentemente esta más próxima a la primera[(2)]

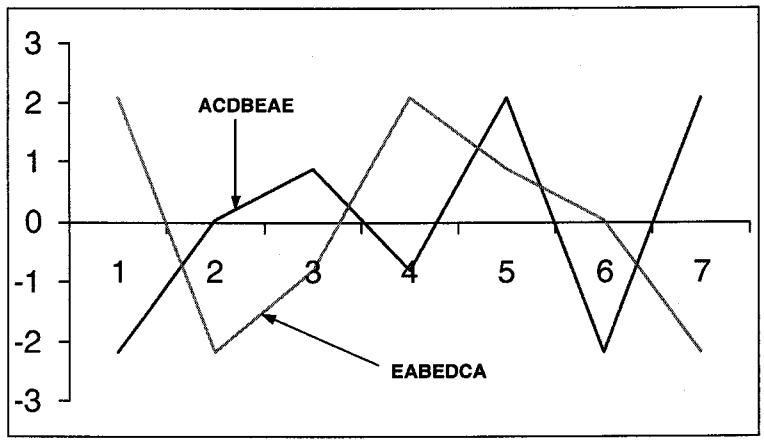


Figura 6.1: Representación de dos palabras con las mismas letras desordenadas. Los valores asignados a las etiquetas coinciden con los centros de los intervalos de clase proporcionados por el método CUM (ver en la tabla 6.1.3).

que la otra ya que tiene 10 saltos.

A	C	D	B	E	A	E	
3	1	1	2	0	1	3	$\longrightarrow$ 10 saltos
D	B	C	D	E	B	B	

Sin embargo, debemos tener presente que la escala ordinal $A < B < C < D < E$ proviene de dar un etiquetado a diferentes intervalos de clase, por ello sería

[(2)]En términos de series temporales estamos acercando los valores de una serie a la otra.

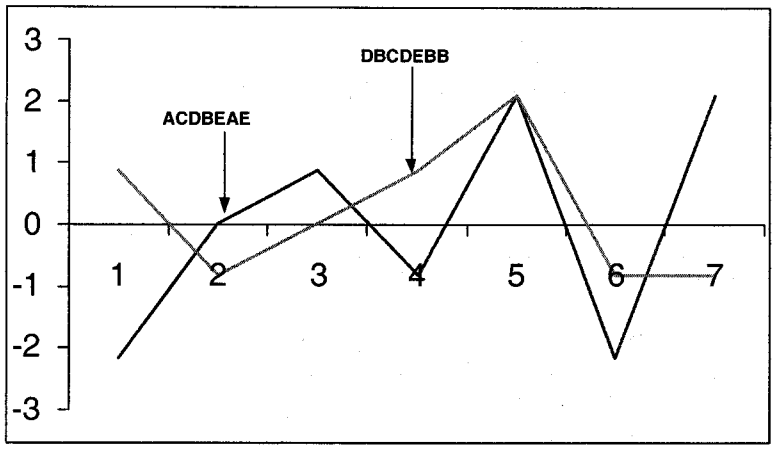


Figura 6.2: Representación de las palabras *ACDBEAE* y *DBCDEBB*.

mucho más exacto establecer una separación entre las etiquetas a partir de una distancia entre los centros de los intervalos, o mejor aún considerar una distancia entre intervalos (ver sección anterior). Siguiendo este planteamiento, hemos de saber que el método *CUM* (ver apartado 5.5.1) aplicado a las dos series que estamos utilizando como referencia proporciona los siguientes intervalos:

Extremo inferior	Extremo superior	Etiqueta	Marca de clase	Radio
-3,1844	-1,1911	A	-2,1877	0,9966
-1,1911	-0,4619	B	-0,8265	0,3646
-0,4619	0,4982	C	0,0182	0,4801
0,4982	1,2639	D	0,8811	0,3828
1,2639	2,9047	E	2,0843	0,8204

Tabla 6.2: Límites de cada etiqueta obtenidos mediante la aplicación del método de discretización *CUM*

De esta forma, tomando como referencia la sección anterior y eligiendo como matriz A a la matriz identidad se tiene la siguiente la escala entre las 5 letras del alfabeto de nuestro ejemplo[3] :

[3]La tabla se ha simplificado teniendo en cuenta la simetría y que la distancia entre dos letras

$d(\cdot,\cdot)$	A	B	C	D
B	1,5008			
C	2,2656	0,8525		
D	3,1296	1,7077	0,8684	
E	4,2757	2,9463	2,0940	1,2803

Tabla 6.3: Nueva matriz de distancias entre las letras del alfabeto.

De manera que si calculamos la distancia entre las palabras $ACDBEAE$ y $EABEDCA$, que anteriormente era de 16 (saltos), como la suma de las distancias entre cada una de las letras que ocupan la misma posición dentro de la palabra, queda:

$$\begin{array}{ccccccc} A & C & D & B & E & A & E \\ 4{,}2757 & 2{,}2656 & 1{,}7077 & 2{,}9463 & 1{,}2803 & 2{,}2656 & 4{,}2757 \\ E & A & B & E & D & C & A \end{array} \longrightarrow 19{,}017$$

y entre las palabras $ACDBEAE$ a $DBCDEBB$:

$$\begin{array}{ccccccc} A & C & D & B & E & A & E \\ 3{,}1296 & 0{,}8525 & 0{,}8684 & 1{,}7077 & 0 & 1{,}5008 & 2{,}9463 \\ D & B & C & D & E & B & B \end{array} \longrightarrow 11{,}005$$

cuando anteriormente señalaba 10 saltos. Destaca en este caso que ambas tablas 6.1 y 6.1.3 derivarían a conclusiones muy similares ya que los valores están próximos.

De esta forma, si denotamos por $d(P1_i, P2_i)$ la diferencia ("distancia") entre las letras que ocupan el lugar i-ésimo en las palabras $P1$ y $P2$, entonces se tiene como una medida de separación, $d(P1, P2)$ a:

$$d_1(P1, P2) = \sum_{i=1}^{n} d(P1_i, P2_i) \tag{6.8}$$

donde n es el tamaño de cada una de las palabras.

Nota 6.1.1 *Es claro que esta "distancia" entre palabras incorpora en su construcción las características que deseamos destacar de cada palabra:*

iguales es cero

- *El orden de las letras al comparar cada una de ellas en una palabra con la correspondiente en la otra.*
- *El tamaño de la palabra al considerar una suma desde* 1 *hasta el tamaño* n.
- *La diferencias entre las letras al utilizar el correspondiente orden de magnitud que cuantifica las separaciones o distancias entre las letras.*

▲

Nota 6.1.2 *A veces puede no interesar tener en cuenta el tamaño de la palabra, por ejemplo para cuantificar la separación media entre las letras de cada palabra. En estos casos basta con tomar la misma definición anterior promediando por el número de letras, es decir,*

$$\frac{1}{n}\sum_{i=1}^{n} d(P1_i, P2_i)$$

■ ▲

Una distancia en un espacio métrico, nos define una medida de separación entre elementos, así nos cabe preguntar si $d_1(P1, P2)$ es ciertamente una distancia. Posteriormente veremos que lo es utilizando los desarrollos de las funciones núcleos.

De esta manera podemos dar una medida del parecido entre dos palabras con el mismo número de letras, pero está pendiente cómo hacerlo entre palabras de distintos tamaños.

6.1.4. Palabras de distinto tamaño

Como en el apartado anterior, planteamos el problema a partir de un ejemplo. Sean las palabras $EBCDACDE$ y $ABBECD$ y lo que haremos será fijar la palabra

más larga y comparar la palabra más corta de la siguiente forma[4]:

$$\begin{array}{cccccccc} A & B & B & E & C & D & & \\ 4 & 0 & 1 & 1 & 2 & 1 & & \longrightarrow 9 \text{ saltos} \\ E & B & C & D & A & C & D & E \end{array}$$

$$\begin{array}{cccccccc} & A & B & B & E & C & D & \\ & 1 & 1 & 2 & 4 & 0 & 0 & \longrightarrow 8 \text{ saltos} \\ E & B & C & D & A & C & D & E \end{array}$$

$$\begin{array}{cccccccc} & & A & B & B & E & C & D \\ & & 2 & 2 & 1 & 2 & 1 & 1 \quad \longrightarrow 9 \text{ saltos} \\ E & B & C & D & A & C & D & E \end{array}$$

es decir, se calculan todas las separaciones posibles de las subpalabras del mismo tamaño de la letra más larga con la palabra más corta, sin modificar el orden de ninguna de las palabras utilizadas.

Así, sean $P1$ la palabra de mayor tamaño n y $P2$ la otra palabra de tamaño m $(m \leq n)$. Entonces vamos a definir como medida de similitud entre estas dos palabras la que se sigue del siguiente razonamiento: Si denotamos por $d(P1_i, P2_j)$ la diferencia entre la letra que ocupa el lugar i-ésimo en la palabra $P1$ y la letra que ocupa el lugar j-ésimo en la palabra $P2$, se tiene como una primera medida de separación, $d_1(P1, P2)$:

$$d_1(P1, P2) = \min_k \left\{ \sum_{i=1}^{m} d(P1_{i+k}, P2_i),\ k = 0, \cdots, n-m \right\} \tag{6.9}$$

De esta manera, la similitud entre las palabras $EBCDACDE$ y $ABBECD$ es

$$d_1(EBCDACDE,\ ABBECD) = \min\{10, 8, 9\} = 8$$

Si utilizamos distancia intervalar entonces se sigue:

$$d_1(EBCDACDE,\ ABBECD) = \min\{9{,}5424,\ 8'3366, 9'7166\} = 8'3366$$

y ambas coinciden con la misma ordenación de la palabra más corta.

[4] Análogamente se haría si se considera la distancia intervalar.

Nota 6.1.3 *Si las palabras son del mismo tamaño se tiene la definición dada en la subsección 6.1.3. Por otro lado se elige el mínimo (en lugar del máximo, la media o cualquier otra característica) ya que se quiere estudiar la similitud entre la palabra más corta y la subpalabra de la palabra más larga que más se parece a ella.* ■ ▲

Nota 6.1.4 *Otra posibilidad sería promediar las* $\sum_{i=1}^{m} d(P1_{i+k}, P2_i) \quad k = 0, \cdots, n-m$ *pero esto nos señalaría el parecido medio entre la palabra más corta y la más larga, pero nuestro objetivo es detectar patrones parecidos entre ambas palabras y por ello es mucho mejor seleccionar como criterio el mínimo ya que si, por ejemplo, este valor mínimo es nulo entonces significa que hay una coincidencia exacta de la palabra corta dentro de la larga.* ■ ▲

La "distancia" d_1, presenta el inconveniente de no reflejar bien la diferencia entre los tamaños de las palabras. Así por ejemplo se tiene que:

$$d_1(EBCDACDE\,,\,A) = \text{mín}\,\{4,1,2,3,0,2,3,4\} = 0$$

Las diferentes formulaciones para resolver este problema chocan con el hecho de que si la "distancia" definida es nula, si elegimos cualquier factor que penalice el diferente tamaño de las palabras, la multiplicación también será nula y no podremos discriminar adecuadamente la similitud en función del tamaño. Si esta distancia es nula, significa que existe una cadena en la palabra mayor que coincide con la palabra más corta.

Para evitar el anterior problema tomamos la idea del núcleo entre literales que aparece en [CST00] y [SS02]. Así, sea $0 < \lambda < 1$ y como $\lambda^0 = 1$ se evita la situación con distancia nula si se utiliza d_1 como exponente. Además en lugar de medir disimilitudes (con la "distancia") vamos directamente a buscar una función que mida similitudes entre dos palabras:

Una primera aproximación sería usar directamente la función[5]

$$K(P1, P2) = \lambda^{d_1(P1,P2)}$$

pero esta tiene el problema que los valores de las distancias se van compensando y no se tiene en consideración el tamaño final de la palabra y tampoco proporciona una solución adecuada al mínimo. Para evitar este inconveniente se define la función núcleo entre dos palabras como:

$$K_\lambda(P1, P2) = \text{máx}\left\{\sum_{i=1}^{m} \lambda^{d(P1_{i+k}, P2_i)},\ k = 0, \cdots, n-m\right\} \tag{6.10}$$

Nótese que se pide el máximo, en lugar del mínimo ya que se busca establecer similitudes, parecidos, cercanía.

Nota 6.1.5 *Si las palabras son del mismo tamaño, $n = m$, entonces:*

$$K_\lambda(P1, P2) = \sum_{i=1}^{n} \lambda^{d^2(P1_i, P2_i)} \tag{6.11}$$

Nota 6.1.6 *Este kernel es una función de base radial (R.B.F.) porque se define como una función de la distancia $f(d(P1, P2))$.*

Veamos a continuación algunas de las principales propiedades del núcleo.

Propiedad 1: *Si $0 < \lambda_1 < \lambda_2 < 1$ entonces:*

$$K_{\lambda_1}(P1, P2) \leq K_{\lambda_2}(P1, P2) \quad \textit{para todas} \quad P1,\ P2 \in \mathcal{P}$$

[5]Se tendría una función de base radial (R.B.F.) ya que se define como una función de una distancia $f(d(P1, P2)$.

Esta propiedad se sigue directamente del hecho que $\lambda_1^x \leq \lambda_2^x$ para cualquier $x \geq 0$ si $\lambda_1 < \lambda_2 < 1$. Señalar que lo realmente interesante de esta propiedad es que nos dice que cuanto más próximo este λ de la unidad menor importancia se da a la coincidencia entre letras, como puede verse gráficamente en la figura 6.3, ya que para cualquier λ la coincidencia entre letras es la unidad, pero si la distancia entre letras es d entonces cuando mayor sea λ más próxima a la unidad se encuentra λ^d y menor será la diferencia con las coincidencias entre letras, es decir, para cualquier $d \geq 0$ se sigue $0 \leq 1 - \lambda_1^d \leq 1 - \lambda_2^d$.

Propiedad 2: *$K_\lambda(P1, P2) \leq mP$ para toda $P1$, $P2 \in \mathcal{P}$ y para cualquier $0 < \lambda < 1$ se tiene:*

$$K_\lambda(P_1, P_2) \leq m$$

y además la cota se alcanza.

Demostración: Si $0 < \lambda < 1$ entonces $\forall x : x \geq 0$ se tiene que $\lambda^x \leq 1$ entonces:

$$K_\lambda(P1, P2) \leq \sum_{i=1}^{m} 1 = m$$

Si consideramos la palabra $P2 = P1_1 P1_2 \cdots P1_m$ entonces es fácil comprobar que $K_\lambda(P1, P2) = m$ y por tanto la cota se alcanza. ■

Propiedad 3: *Sea $r = \text{máx}_{ij}\, d(A_i, A_j)$ con $A_i, A_j \in \mathcal{A}$. Entonces*

$$m\lambda^r \leq K_\lambda(P1, P2)$$

para toda $P1$, $P2 \in \mathcal{P}$ y para cualquier $0 < \lambda < 1$ y además la cota se alcanza.

Demostración: Sea $0 < \lambda < 1$ entonces si $x \leq y$ se tiene que $\lambda^x \geq \lambda^y$ y por tanto para cualquier distancia entre las letras del alfabeto se sigue que:

$$K_\lambda(P1, P2) \geq \sum_{i=1}^{m} \lambda^r = m\lambda^r$$

Veamos que la cota inferior se alcanza: Sean A y $B \in \mathcal{A}$ tales que $d(A,B) = r$ entonces si consideramos $P1 = AA\cdots A$ y $P2 = BB\cdots B$ con tamaño de $P1$ y de $P2$, n y m respectivamente. Entonces se sigue que $K_\lambda(P1,P2) = m\lambda^r$ de forma inmediata. ■

Consideramos una modificación en la definición de la función núcleo K_λ, que en la propiedad 4 se justifica. Así, será la función K_λ^* definida de $\mathcal{P} \times \mathcal{P}$ en $\mathbb{R}$ como sigue:

$$K_\lambda^*(P1,P2) = \text{máx}\left\{\sum_{i=1}^{m} \lambda^{d^2(P1_{i+k},P2_i)},\ k = 0,\cdots,n-m\right\}$$

Nótese que la modificación sobre el anterior K_λ es que se ha sustituido la distancia d en el exponente del parámetro λ por la distancia al cuadrado. Por tanto, y siguiendo un razonamiento idéntico al aplicado sobre K_λ se sigue que para cualquier $0 < \lambda < 1$ se verifica:

$$m\,\lambda^{r^2} \leq K_\lambda^*(P1,P2) \leq m, \quad \forall P1,\ P2 \in \mathcal{P}$$

■

La siguiente propiedad es la que realmente da valor a la construcción realizada ya que anteriormente se hablaba de saltos y similitudes desde un punto de vista intuitivo, con este resultado se justifica plenamente la validez de los resultados previos.

Propiedad 4 *Sea $\mathcal{A}$ un alfabeto y $\mathcal{P} = \{P_1P_2\cdots P_n,\ P_i \in \mathcal{A}\}$ (el conjunto de todas las palabras de tamaño n). Entonces*

$$K_\lambda^*(P1,P2) = \sum_{i=1}^{n} \lambda^{d^2(P1_i,P2_i)}$$

es un núcleo de Mercer.

Demostración: Dado un alfabeto $\mathcal{A}$ procedente del etiquetado de una serie temporal se define una aplicación ϕ de $\mathcal{A}$ a $\mathbb{R}^2$ de la siguiente forma:

Una letra es la marca que asignamos a los valores de la serie que se encuentran dentro de un determinado intervalo $(c-r\,,\,c+r)$ con c el centro del intervalo y

r la amplitud, i.e., se tiene una aplicación que asigna un intervalo a una etiqueta. Se considera la aplicación ϕ_1 del conjunto de todos los intervalos $\mathcal{I}$ en $\mathbb{R}^2$ dada en la sección 6.1.1. Considerando la composición de las dos, se tiene una aplicación $\phi : \mathcal{A} \longrightarrow \mathbb{R}^2$ tal que

$$\phi(A) = P \begin{pmatrix} c \\ r \end{pmatrix}$$

donde P es una matriz cuadrada de orden 2 no singular. A partir de esta aplicación se define una aplicación

$$k_1 : \mathcal{A} \times \mathcal{A} \longrightarrow \mathbb{R}$$

de la siguiente forma:

$$k_1(A, B) = \langle \phi(A), \phi(B) \rangle$$

la cual por construcción se tiene garantizado que es un núcleo, y siguiendo los resultados de 6.1.1, la distancia intervalar coincide con:

$$d^2(A, B) = \langle \phi(A) - \phi(B), \phi(A) - \phi(B) \rangle = ||\phi(A) - \phi(B)||^2$$

donde $||\cdot||$ denota la norma euclídea en $\mathbb{R}^2$.

Aplicando el corolario 3.13 (descrito en la página 43 de [CST00]), en lugar de a la función exponencial e^{-x} a la función $\left(\frac{1}{\lambda}\right)^{-x}$ con $0 < \lambda < 1$ se tiene garantizado que la aplicación

$$k_2 : \mathcal{A} \times \mathcal{A} \longrightarrow \mathbb{R}$$

definida de la siguiente forma:

$$k_2(A, B) = \lambda^{||\phi(A) - \phi(B)||^2} = \lambda^{d^2(A,B)}$$

es un núcleo. Por tanto si se considera la aplicación

$$K_\lambda^* : \mathcal{P} \times \mathcal{P} \longrightarrow \mathbb{R}$$

como

$$K_\lambda^*(P1, P2) = \sum_{i=1}^{n} \lambda^{d^2(P1_i, P2_i)}$$

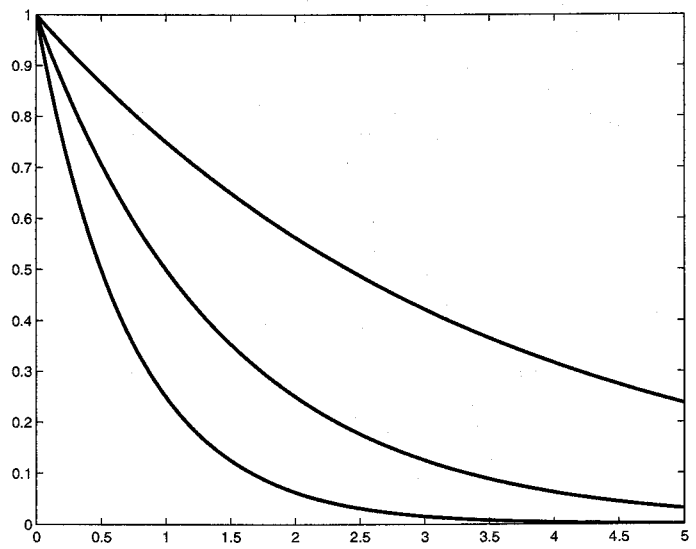


Figura 6.3: Representación gráfica de la función $f(x) = \lambda^x$ para $\lambda = 0'75, 0,5$ y 0,25 que permite observar la cuantificación dada a la separación entre letras.

se tiene garantizado por la proposición 3.12 de [CST00](página 42) que ciertamente es un núcleo de Mercer. ■

Ejemplos:

Veamos cómo sería la similitud entre las palabras de igual tamaño $ACDBEAE$ y $EABEDCA$ cuya distancia anteriormente era de 19,017 unidades, considerando la función de similitud núcleo K^*_λ con $\lambda = 0{,}5$. De esta forma se sigue que:

$$K_{0,5}(ACDBEAE\,,\; EABEDCA) = 0,513$$

y si se aplica a la otra palabra que se obtiene disminuyendo el número de saltos por letra en una unidad, se tiene:

$$K_{0,5}(ACDBEAE\,,\; DBCDEBB) = 2,5431$$

que en cuanto a distancia esta más cerca pero en términos de similitud debe ser mayor que la anterior similitud, como así es.

Veamos como sería en el ejemplo de palabras de distinto tamaño. Sean las palabras $EBCDACDE$ y $ABBECD$ y calculemos, para distintos valores de λ, la similitud entre ellas.

EBCDACDE	λ		
	0.5	0.75	0.90
ABAECD–	2,5467	3,4739	4,4193
-ABAECD-	2,9466	3,7718	4,5962
–ABAECD	1,3327	2,896	4,5016
Similitud	2,9466	3,7718	4,5962

De la tabla se observa que la cuantificación de las similitudes son distintas pero se mantiene el mismo orden de magnitud entre ellas. Puede verse como en las dos últimas ordenaciones la diferencia se reduce al aumentar λ puesto que este parámetro determina la importancia relativa de la coincidencia exacta.

6.2. Generalización de la similitud

Sean $P1$ y $P2$ dos palabras de tamaño n del conjunto $\mathcal{P}$. En la definición de similitud entre palabras $K^*_\lambda(P1, P2) = \sum_{i=1}^{n} \lambda^{d^2(P1_i, P2_i)}$ se da el mismo peso al orden en la palabra de cada una de las letras, por ello una generalización de la similitud K^*_λ es asignar diferentes pesos, de tal manera que la suma de todos estos coincidan con el tamaño de la palabra[6] tan Así, sean $w_1, w_2, \cdots, w_n \in \mathbb{R}$ y positivos tales que $\sum_{i=1}^{n} w_i = n$ se puede definir la similitud generalizada inicialmente de dos maneras diferentes:

$$K_\lambda 1(P1, P2) = \sum_{i=1}^{n} \lambda^{w_i \cdot d2(P1_i, P2_i)} \qquad K_\lambda 2(P1, P2) = \sum_{i=1}^{n} w_i \cdot \lambda^{d2(P1_i, P2_i)}$$

[6]Si se promedia, la suma de los pesos es la unidad. Esto se hace de esta forma buscando una normalización de los pesos.

No es difícil demostrar que ambas definiciones proporcionan núcleos de Mercer[7], pero indudablemente la segunda proporciona una interpretación más intuitiva del significado de los pesos en la construcción de la similitud final. Además de las propiedades de la función exponencial se sigue que:

$$K_\lambda 1(P1, P2) = \sum_{i=1}^{n} \lambda^{w_i \cdot d2(P1_i, P2_i)} = \sum_{i=1}^{n} w_i' \cdot \lambda^{d2(P1_i, P2_i)}$$

donde $w_i' = \lambda^{(w_i - 1)\, d2(P1_i, P2_i)}$. Ambas definiciones tienen la misma forma pero en el caso segundo la suma de los pesos es la unidad, lo que no es verdad necesariamente en el primero, y depende de la distancia.

La elección de los pesos es muy general y subjetiva. Así se tienen diferentes elecciones de ellos en la figura 6.4 que reflejan distintas formas de entender el significado de similitud. En la figura superior izquierda se da el mismo peso (peso unitario) a

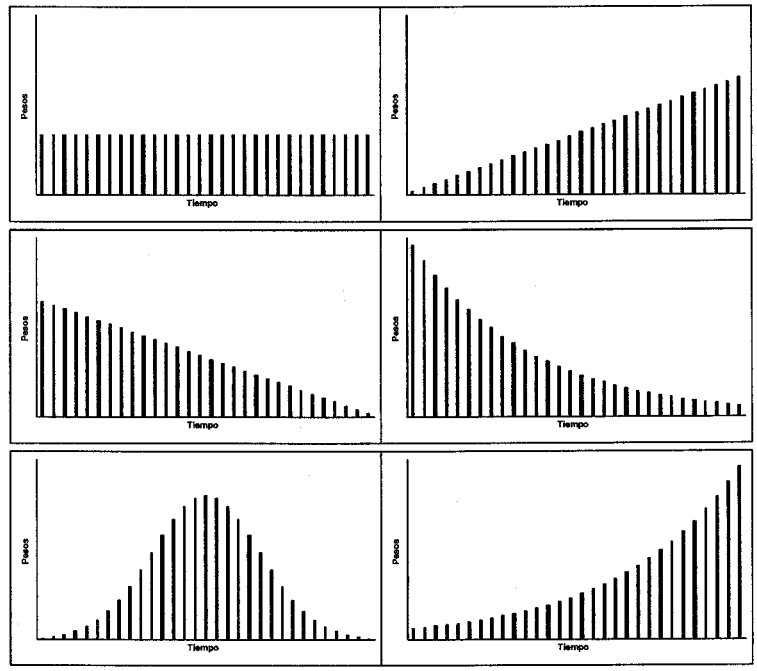


Figura 6.4: Diferentes pesos

todas y cada una de las letras que componen la palabra (sería la definición origi-

[7]La suma de núcleos y el producto de un escalar por un núcleo, son núcleos.[CST00]

nal). En la figura superior derecha se da una mayor importancia a las letras finales de las palabras que a las iniciales utilizando unos pesos siguiendo una progresión aritmética creciente, esta situación podría ser la adecuada para entender si los índices de dos cotizaciones bursátiles pueden considerarse tanto más parecidas ya que en estos casos lo que se puede buscar es el conocimiento de una para inferir sobre la otra. Si estamos estudiando un sistema dinámico asociado al comportamiento de un motor, el sentido de similitud puede ser diferente dando una mayor importancia al transitorio que al estacionario, ello se puede conseguir con los pesos de la figura del centro izquierda (asignando pesos siguiendo una progresión aritmética decreciente). Análogamente se puede conseguir lo mismo pero utilizando unos pesos siguiendo una progresión geométrica decreciente (figura 6.4 centro-derecha) o creciente (inferior-derecha). Otra posibilidad que aparece en la figura 6.4 en la parte inferior izquierda consiste en considerar que los pesos se distribuyen según un modelo normal discretizado, donde se da una mayor importancia a las letras del centro de la palabra que a aquellas que se encuentran en los extremos.

6.3. Algunos comentarios

Esta construcción de similitud entre palabras hay que entenderla tal cual se ha definido, es decir, se tiene en cuenta el tamaño de las palabras, el orden de las letras y se cuantifica la diferencia entre letras que ocupan la misma posición. Esto significa que si se desea considerar otras características de las palabras o no se desea algunas de las consideradas anteriormente, necesariamente la medida de similitud por nosotros considerada no sería la adecuada y se debería construir una nueva adecuada al problema que se quiere resolver.

Por ejemplo, sean las palabras $AEAEAEAE$ y $AEAEAEAE$, la similitud, dada por K_λ^*, entre ellas es máxima e igual a 8. Sin embargo, si sobre la segunda realizamos un pequeño cambio[8] $EAEAEAEA$ entonces la similitud entre ellas es mínima, pero

[8]La primera letra, la colocamos al final de la palabra.

puede ser útil en determinados estudios considerar que el parecido entre ellas es alto. Sin embargo, a pesar de que la similitud K_λ^* no esta especialmente diseñada para ello, esta situación puede explotarse también ya que esto significa que asociada a una letra de una palabra, la letra de la otra palabra que ocupa el mismo orden es la más alejada de ella, es decir, si la letra es A la letra de la otra palabra es E y al revés. Pero, ¿admite nuestra similitud esta interpretación de forma general? Veamos que si:

Sean las distancias entre letras dada anteriormente:

$d(\cdot,\cdot)$	A	B	C	D
B	1,5008			
C	2,2656	0,8525		
D	3,1296	1,7077	0,8684	
E	4,2757	2,9463	2,0940	1,2803

En esta caso se tiene que $r = 4,2757$ y si tomamos $\lambda = 0,90$, y los pesos unitarios entre cada letra ($w_i = 1$), se tiene que para cualquier tamaño n de dos palabras $P1$ y $P2$

$$\lambda^{r^2} = 0,1457066 \leq \frac{1}{n} K_\lambda^*(P1,P2) \doteq \overline{K_\lambda^*}(P1,P2) \leq 1$$

obteniéndose en este caso con $\lambda^{d^2} = \frac{1}{n} K_\lambda^*(P1,P2)$ una similitud media entre cada dos letras de sendas palabras.

Por otro lado, de la tabla de distancia entre palabras se tiene que cuando se produce un salto entre cualesquiera dos letras, la distancia es inferior que cuando se produce dos saltos; y de igual forma cuando se produce dos saltos entre cualesquiera dos letras, la distancia es inferior que cuando se produce tres saltos y así sucesivamente (esta interpretación resulta la más natural y da una mayor consistencia a la definición de distancia entre intervalos).

De esta forma, la interpretación sobre la similitud media entre las palabras $AEAEAEAE$ y $AEAEAEAE$, nos dice que $\lambda^{d^2} = 1 = \lambda^0$ y esto significa que el salto medio, entre letras que ocupan la misma posición, es nulo y como debe ser siempre positivo significa que la distancia media entre cada palabra es nula y por

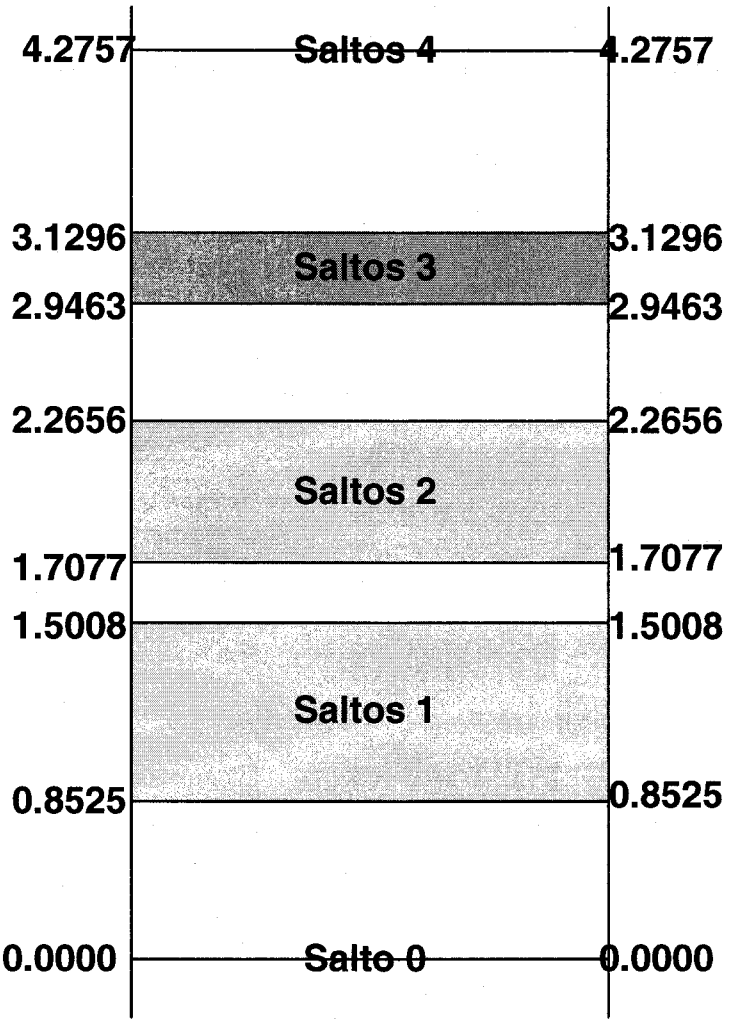


Figura 6.5: Interpretación de la *similitud media*.

tanto coinciden en significado y orden todas las letras de cada una de las dos palabras. Por otro lado, la interpretación sobre la similitud media entre las palabras $AEAEAEAE$ y $EAEAEAEA$, nos dice que $\lambda^{d^2} = 0,1457066 = \lambda^{4,2757^2}$ y esto significa que el salto medio es cuatro ya que coincide con la distancia máxima entre letras de cada una de las dos palabras.

La figura 6.5 nos permite interpretar el valor de la similitud media entre palabras de igual tamaño como sigue: Si $0{,}8525 \leq d \leq 1,5008$ el número medio de saltos entre palabras es uno, si $1{,}7007 \leq d \leq 2{,}2656$ el número medio de saltos entre palabras

es dos, etc. Incluso esta interpretación nos permite realizar una tosca, pero fácil predicción de una palabra a partir de otra como se verá en el capítulo siguiente.

CAPÍTULO 7

METODOLOGIA

Se incluye en este capítulo una metodología sistemática para la identificación de series que unifica todas nuestras aportaciones a la comparación de series como la aproximación cualitativa al problema, los algoritmos de etiquetado desarrollados y el kernel de similitud para cadenas de símbolos.

Dado un conjunto de aprendizaje se realiza una evaluación de la capacidad de varias alternativas para proporcionar identificaciones correctas. Para ello, se convierten las series en cadenas de símbolos por medio de varios métodos de discretización. Posteriormente se utiliza una distancia basada en un núcleo entre literales para calcular la similitud entre las series, y se utiliza un algoritmo de vecinos próximos para identificar la clase a que pertenece.

Las ventajas de método son unos porcentajes de identificación muy altos, la necesidad de un único vecino para obtener estas propiedades y la simplicidad de utilización del sistema por parte del usuario.

7.1. Metodología propuesta

El sistema off-line que implemente la metodología que presentamos será capaz, después de analizar el conjunto de aprendizaje, de identificar los orígenes de las series de un conjunto de trabajo. Un esquema global se presenta en la figura 7.1, donde se han omitido algunas operaciones para favorecer la claridad, y ahora realizaremos una somera revisión de cada apartado.

Sea B una base de datos que contenga series temporales pertenecientes a K clases distintas obtenidas por medio del registro de alguna o algunas magnitudes físicas, o por medio de simulación de un dinámico.

Cada serie incluirá una etiqueta que indicará la clase a la que pertenece, por razón de su origen o de un procedimiento de etiquetado previo por parte de un experto.

Sobre el conjunto de series se realiza un procedimiento de normalización para posibilitar la comparación valida de series en escalas distintas. De entre los múltiples métodos de normalización que pueden elegirse se practicará un procedimiento de tipificación.

Con el nuevo conjunto de series se deben seleccionar los componentes que formarán tanto el subconjunto de aprendizaje como el de verificación. Esta división se realizará aleatoriamente de acuerdo a la estratificación que presente la base de datos de series.

El paso posterior consiste en transformar las series en cadenas de símbolos, para realizar una aproximación cualitativa a la comparación de series. Como no conocemos ningún método que garantice las mejores prestaciones para cualquier conjunto de datos se realiza una comparación entre varios de ellos.

Inicialmente trataremos con un aprendizaje supervisado aunque el objetivo siguiente será extender al caso de no supervisado, por ello la elección de los distintos métodos de discretización.

Los métodos utilizados serán el método *Ameva*, el método *CAIM*, el método CUM, el método de igual amplitud en todos los intervalos y el de intervalos de igual frecuencia. Tenemos por tanto métodos supervisado y no supervisados, en el sentido expuesto en el punto 5.1 de tener o no en consideración la información del origen de cada serie.

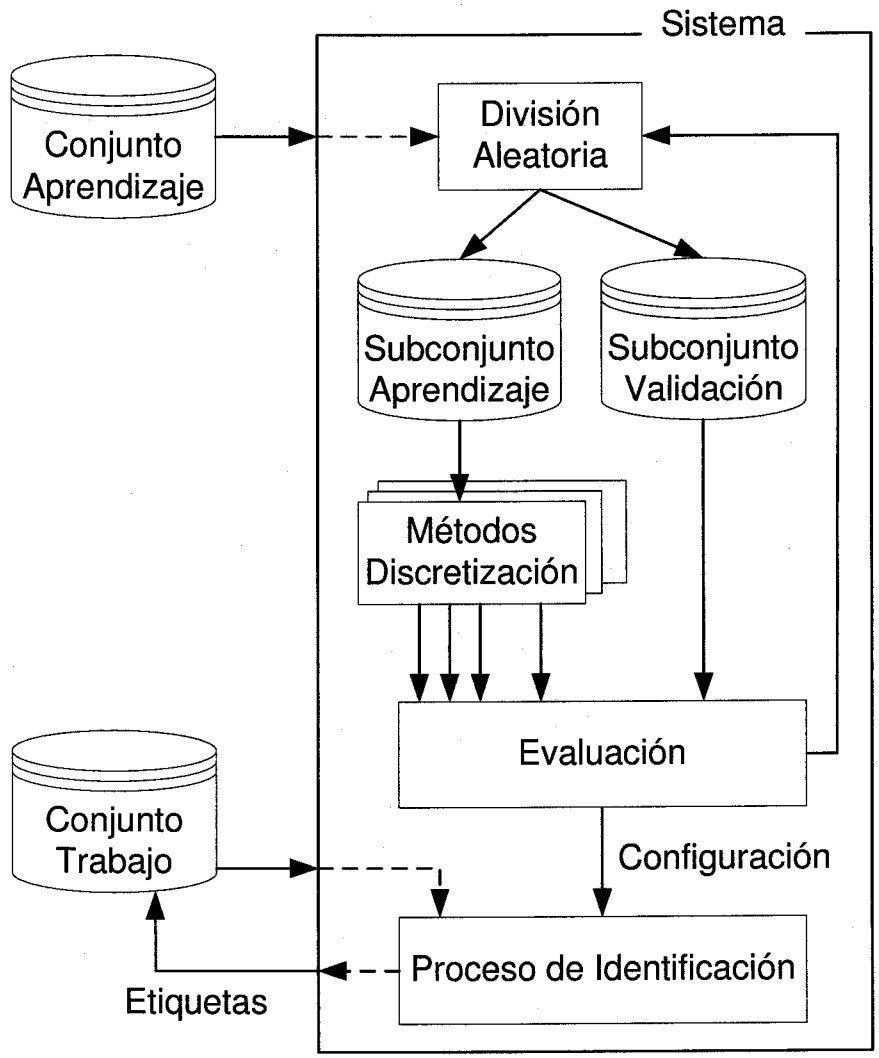


Figura 7.1: Metodología Propuesta

Tras la aplicación de los diferentes métodos utilizaremos los conjuntos de marcas proporcionadas para convertir las series en un conjunto de series de símbolos, que proporcionan una información cualitativa de la evolución de la serie.

El sistema de valoración de la calidad de los etiquetados consistirá en contabilizar las identificaciones correctas del conjunto de validación por medio de la aplicación de una distancia intervalar expuesta en el capítulo 6 sobre las series de símbolos.

Como la división inicial de la base de datos en los conjuntos de aprendizaje y verificación se realiza por medio de un muestreo estratificado simple, es necesario repetir todo el procedimiento varias veces para poder promediar los resultados con respecto a las posibles divisiones.

Una vez finalizado, el sistema determinará el mejor método para identificar las series que componen la base de datos original, quedando preparado para realizar la identificación de las nuevas series que se le vayan proponiendo.

En las siguientes secciones se describirán todos los pasos de esta metodología con mayor detalle.

7.1.1. Tipificación y Diferenciación

Antes de trabajar con las series se realiza un paso de tipificación. Este proceso no se ha incluido en la figura 7.1.

La tipificación produce un nuevo conjunto de series.

Sea $X = \{x_0, ..., x_n\}$ una serie temporal, y sea $X_T = \{\tilde{x}_0, ..., \tilde{x}_n\}$ la serie temporal tipificada obtenida de X.

Los elementos de la serie tipificada se calculan como sigue:

$$\tilde{x}_i = \frac{x_i - \overline{X}}{S_X} \tag{7.1}$$

donde $\overline{X}$ y S_X son una media y la desviación estándar de S, respectivamente.

Las series proporcionadas por la tipificación se caracterizan por:

- No tener unidades.
- La media de la serie es cero.

- La desviación estándar es 1.
- Son invariantes ante cambios de origen y de escala, siempre que el factor de escala sea mayor que cero, como se define en [GK94] donde ($a > 0$).

Este paso se realiza por las mejores características matemáticas de la tipificación frente a un proceso de normalización ya comentadas en la sección 4.2.1. Aunque existen dichas ventajas con el proceso de tipificación se realizará una implementación que permita al usuario elegir entre este y el de normalización, cuando tenga la certeza de la no existencia de valores anormales.

Una vez que las series ha sido procesadas, y con la intención de centrar la comparación es la evolución de las series y no en sus valores concretos, se realiza el paso de diferenciación. En este paso se tienen las denominadas *series de diferencias* por medio de tener el incremento de cada valor de la serie con respecto a su adyacente anterior.

Así la serie de diferencias o transiciones $X_D = \langle d_0, ..., d_{n-1} \rangle$ se obtiene calculando cada d_i según

$$d_i = \tilde{x}_i - \tilde{x}_{i-1} \tag{7.2}$$

7.1.2. Creación de conjuntos

En este punto se procederá a crear los subconjuntos de aprendizaje y prueba haciendo una selección de las series existentes en la base de datos. Esta selección se realizará de forma aleatoria considerando la estratificación del conjunto de series.

No existe ninguna regla que establezca los tamaños óptimos de los conjuntos de aprendizaje y verificación con respecto al tamaño total de la base de datos. Ese valor óptimo depende de la naturaleza de los datos, la cual no es conocida a priori.

Es una conclusión aceptada universalmente que tamaños muy pequeños, con respecto al total de datos, del conjunto de aprendizaje proporcionan resultados difícilmente extrapolables. Igualmente tamaños muy grandes del conjunto de aprendizaje

produce un sobreajuste que hay que evitar por su falta de generalización.

Es por todo ello que en este trabajo se ha optado por utilizar un esquema de validación cruzada.

Una vez obtenidos los dos conjuntos podemos proceder a realizar el aprendizaje.

7.1.3. Aplicación de métodos. Conversión a cadenas

No existe un método universal que proporcione una división óptima en intervalos de un valor continuo. Nuestra aproximación consistirá en la aplicación simultánea de varios de estos métodos.

Evidentemente hay que considerar que existe una gran multitud de métodos de discretización, como ya se analizó en el punto 5.1. De entre ellos, los métodos seleccionados para aplicarse serán los siguientes :

- Nuestra aproximación, el *algoritmo Ameva.*
- *Intervalos de Amplitudes Fijas.*
- *Percentiles* o *Intervalos de Igual Frecuencia.*
- *El método CUM.*
- *Algoritmo CAIM.*

Puede encontrarse un comentario más detallado sobre éstos métodos puede encontrarse en el punto 5.5.1

Los métodos de intervalos de igual amplitud, intervalos de igual frecuencia y CUM requieren que se les indique el número de intervalos a calcular. Como no podemos conocer a priori el número óptimo de intervalos, estos métodos se realizan desde 2 al número de clases existentes. Sería posible el aplicar la regla empírica presentada en [WC87] para estimar el número de intervalos:

$$NI = \frac{NE}{3 \cdot NC}$$

donde NI es el número de intervalos a localizar, NE el número de ejemplos en la base de datos y NC el número de clases. Nuestros experimentos nos muestran que números reducidos de etiquetas proporcionan unos buenos resultados, sobretodo cuando el número de ejemplos es elevado donde al elevarse el número de etiquetas se incrementa la granularidad de los intervalos y se hace una aproximación mayor al caso cuantitativo.

Una vez que se dispone de las series convertidas a cadenas de símbolos gracias a la división de los valores posibles intervalos, entonces podemos aplicar una medida de similitud entre las series de símbolos.

7.1.4. Evaluación

En este punto se evaluará la calidad de cada uno de los métodos propuestos. Definimos la calidad de un método de discretización en función de su capacidad de detectar convenientemente el tipo a que corresponden las series que se le presenten.

La prueba consistirá en tratar de identificar cada serie de verificación por medio de algoritmo del vecino más cercano. La etiqueta de la serie de aprendizaje a que más se parece se confrontará con la etiqueta que posee la serie a identificar, comprobándose si el sistema elige la correcta.

La similitud entre las series que se están comparando se podrá obtener por medio de aplicar:

- El índice QSI basado en las subcadenas comunes.
- La distancia intervalar basada en un kernel y que ha sido presentada en el capítulo 6. Ésta distancia está reservada al caso en que todas las series de la base de datos sean de la misma longitud.

Una vez obtenidos los resultados de cada uno de los métodos para los diferentes intentos, que se realizan cambiando el conjunto de aprendizaje y verificación, se decide el método más apropiado para el actual conjunto de datos.

Aunque inicialmente proponemos la utilización de uno de los métodos para los sucesivos procesos de identificación de nuevas series, también podría utilizarse una mixtura de todos los métodos presentados. Esta opción será analizada en un futuro trabajo.

7.1.5. Presentación de Salida

La metodología sigue el modelo de caja negra.

Una vez terminado el proceso descrito con anterioridad, el sistema estará preparado para recibir un nuevo conjunto de series que intentará identificar. En este caso la salida será la lista de clases en que se ha identificado cada elemento del conjunto.

Opcionalmente se puede presentar al usuario, para tareas de depuración o contraste con otros métodos, el mejor método encontrado con toda su parametrización de intervalos y sus valores de acierto en la tarea de identificación.

7.2. Implementación

La implementación práctica de esta metodología se ha realizado por medio del desarrollo de un operador para el entorno de aprendizaje YALE, del que se incluye información detallada en el apéndice D.

El operador ParameterOptimization realiza un producto cartesiano con todos los parámetros y valores que se indiquen, como los métodos *Ameva* y *CAIM* sólo requieren una única ejecución al no tener parámetros, se hace necesaria su separación del ejecución de los métodos restantes. Esto imposibilita una estructura más simple del experimento que debe, por esta razón, realizar dos optimizaciones independientes.

Además, esta falta de unificación impide la selección automática del mejor método por parte del experimento definido YALE. Debiéndose por tanto realizar dos fases

distintas:

- Fase 1: en ella se leerá el conjunto de aprendizaje y realizará ambas optimizaciones, en dos experimentos diferentes, generando una salida con el mejor método de cada uno de los grupos, véase la figura 7.2. El usuario deberá comprobar ambos resultados para obtener el mejor método para este conjunto de datos y configurarlo en el experimento de la segunda fase.

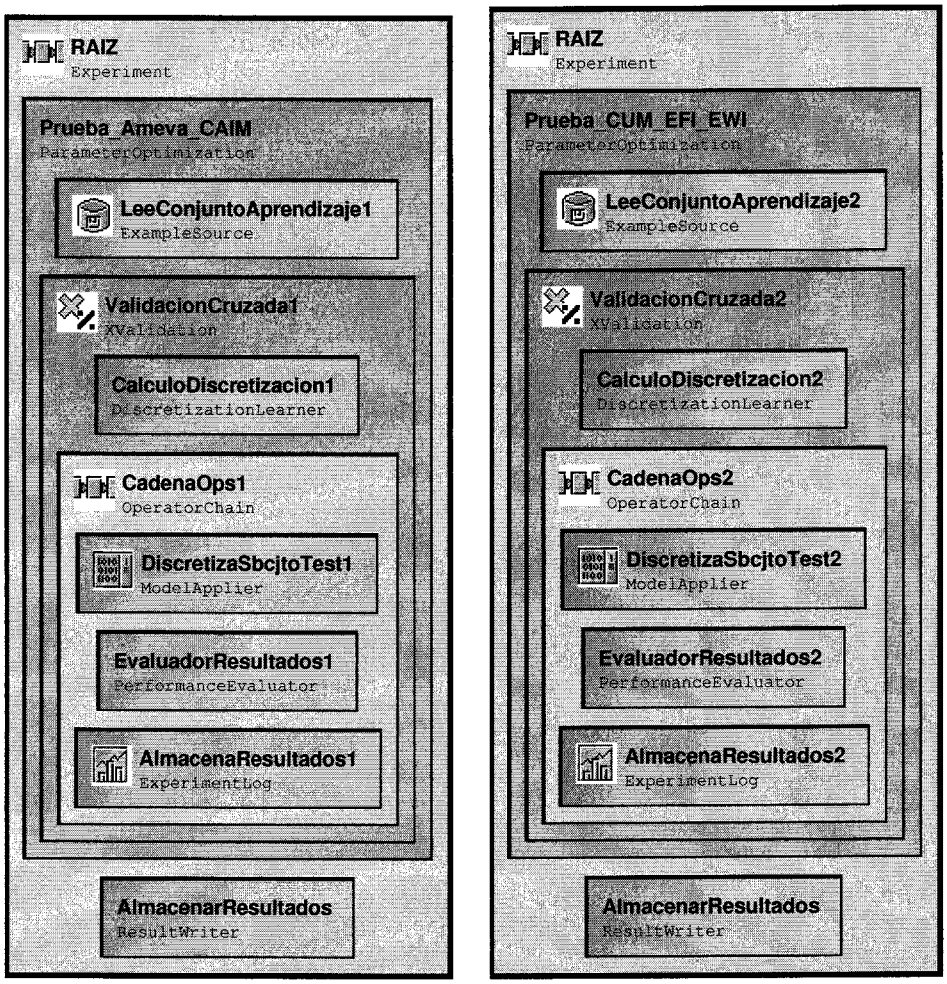


Figura 7.2: Fase 1. Experimentos de confrontación de los métodos de discretización

- Fase 2: en este caso leerá los conjuntos de aprendizaje y trabajo, proporcionando la precisión obtenida al aplicar el método que haya indicado el usuario.

Esta fase se describe gráficamente en la figura 7.3.

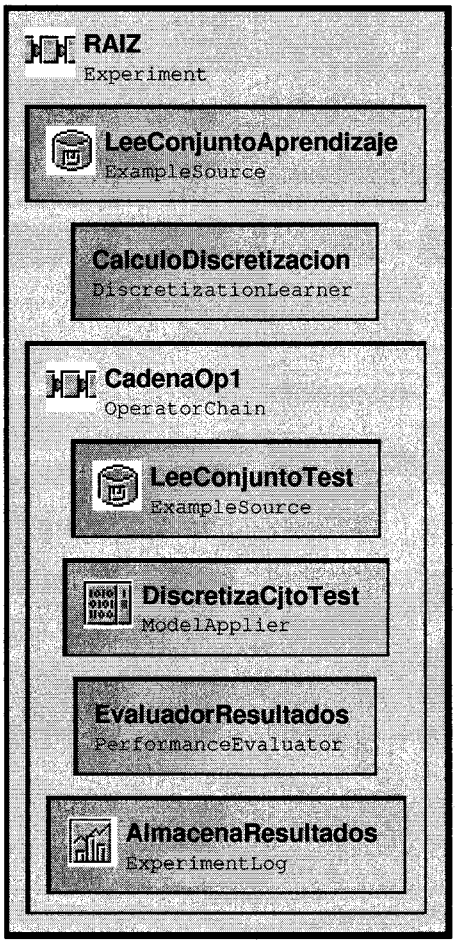


Figura 7.3: Fase 2. Experimento de Identificación del conjunto de Trabajo

Este pequeño inconveniente se espera que pueda evitarse en futuras versiones de la plataforma de aprendizaje YALE.

CAPÍTULO 8

APLICACIONES

En este capítulo vamos a presentar los resultados de aplicar la metodología propuesta a diferentes conjuntos de datos.

Para cada uno de ellos se realizará una descripción del conjunto y los resultados del último paso de la metodología. En el primer conjunto utilizado (Datos de Audiencias) se añadirán también los resultados intermedios de la metodología y se completarán algunas validaciones de ideas ya esbozadas anteriormente.

8.1. Datos de Audiencias

Este conjunto de datos ha sido proporcionado por el Departamento de Audiencia de Canal Sur Televisión, una compañía perteneciente al Grupo Radio y Televisión de Andalucía, y generado por [Sof]. Los datos representan las audiencias estadísticamente estimadas a partir de un conjunto de medidores instalados en hogares de toda Andalucía a lo largo del año 2003.

De entre la multitud de medidas presentes en este conjunto se ha utilizado la denominada cuota de pantalla, "share", que representa el porcentaje de televidentes

están sintonizando una cadena en un momento determinado. Cada serie representa esta cuota de pantalla para un período de 24 horas, entre las 02:30 de un día y la misma hora del día siguiente. Los valores están promediados para bloques de 15 minutos, resultando series de 96 elementos de longitud.

Se han seleccionado los datos relativos a las siete cadenas más importantes y con cobertura en todo el territorio de Andalucía, y de los días de la semana se ha optado por restringir la comprobación a uno en concreto para todas las cadenas, en éste caso el miércoles. Se compone pues el conjunto de 52 series para cada una de las siete cadenas.

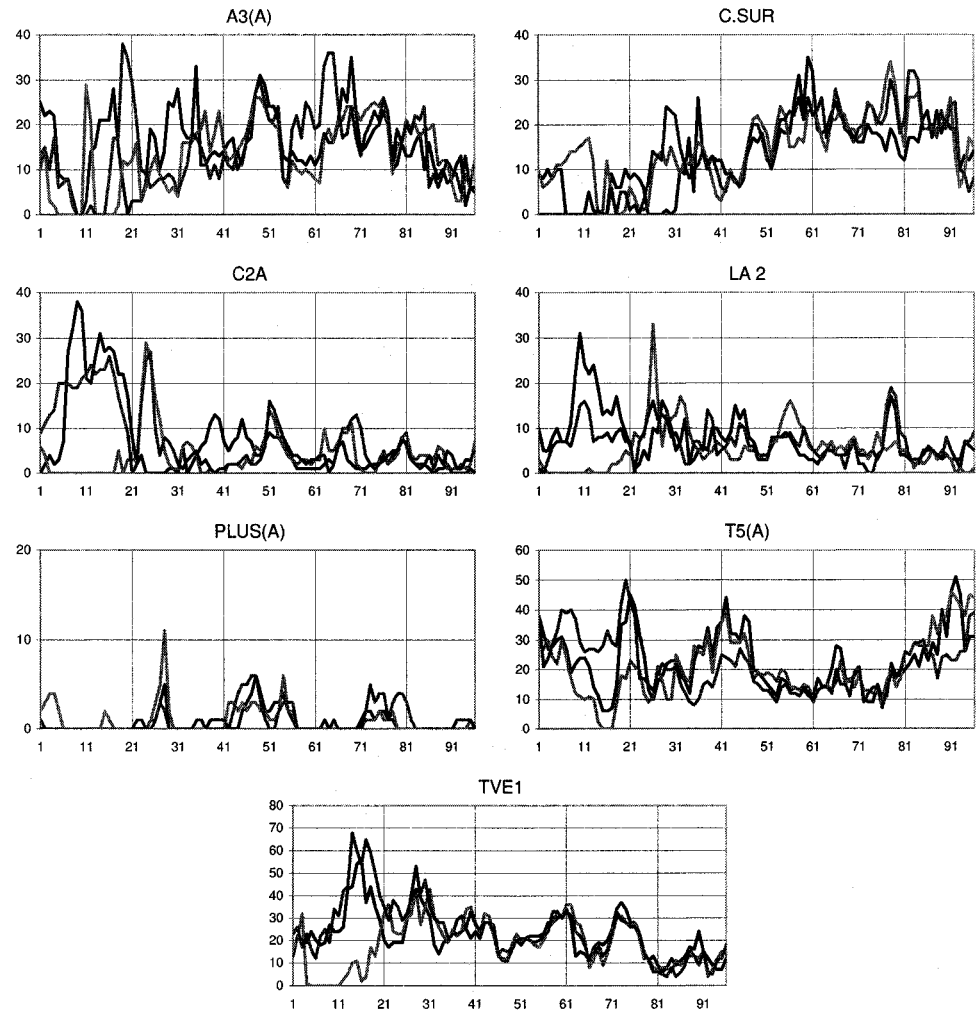


Figura 8.1: Ejemplos de las series de Audiencias

8.1.1. Test

Hemos seleccionado los primeros 32 miércoles del año 2003 como el conjunto de aprendizaje, quedando los 20 restantes como conjunto de trabajo. Las series están etiquetadas con el nombre correspondiente a su cadena de televisión.

Sobre las 224 series existentes en el conjunto de aprendizaje (32∗7) se realizan los experimentos correspondientes a la fase 1 de la metodología aplicando los métodos de discretización siguiendo un esquema de validación cruzada de 10 conjuntos.

Mejor Método de Discretización :	Ameva
Número de Intervalos:	3
Medida de similitud:	Kernel Intervalar
Precisión media	93,75 %

Matriz de Confusión	Real						
Pred.	TVE1	LA	T5	A3	PLUS	C.SUR	C2A
TVE1	32	4	0	0	0	0	0
LA	0	21	0	0	0	0	2
T5	0	1	32	0	0	0	0
A3	0	1	0	32	0	0	0
PLUS	0	0	0	0	32	0	1
C.SUR	0	4	0	0	0	32	0
C2A	0	1	0	0	0	0	29
Por Cadena	100,00 %	65,62 %	100,00 %	100,00 %	100,00 %	100,00 %	90,62 %

Tabla 8.1: Resultado comparación Ameva-CAIM

De esta aplicación se tienen los siguientes resultados parciales como los mejores métodos de cada experimento que se presentan en las tablas 8.1 y 8.2.

Resulta por tanto el método de intervalos de frecuencia fija con 3 intervalos el que presenta la mejor capacidad de identificación. Se parametriza convenientemente el segundo experimento y se tienen los siguientes resultados de identificación sobre las 140 series del conjunto de trabajo.

Mejor Método de Discretización :	EFI
Número de Intervalos:	3
Medida de similitud:	Kernel Intervalar
Precisión media	96,43 %

Matriz de Confusión	Real						
Pred.	TVE1	LA	T5	A3	PLUS	C.SUR	C2A
TVE1	31	0	1	1	0	0	0
LA	1	30	0	0	0	0	1
T5	0	0	31	0	0	0	0
A3	0	0	0	30	0	0	0
PLUS	0	1	0	0	32	0	1
C.SUR	0	0	0	0	0	32	0
C2A	0	1	0	1	0	0	30
Por Cadena	96,88 %	93,75 %	96,88 %	93,75 %	100,00 %	100,00 %	93,75 %

Tabla 8.2: Resultado comparación $CUM - EFI - EWI$

Precisión media aplicando EFI con 3 intervalos	95,00 %
Medida de similitud:	Kernel Intervalar

Matriz de Confusión	Real						
Pred.	TVE1	LA	T5	A3	PLUS	C.SUR	C2A
TVE1	20	1	0	0	0	0	1
LA	0	17	0	0	0	0	0
T5	0	0	19	0	0	0	0
A3	0	1	0	18	0	0	0
PLUS	0	1	0	0	20	0	0
C.SUR	0	0	0	0	0	20	0
C2A	0	0	1	2	0	0	19
Por Cadena	100,00 %	85,00 %	95,00 %	90,00 %	100,00 %	100,00 %	95,00 %

Tabla 8.3: Resultado de identificación del conjunto de trabajo

La aplicación de la metodología presentada alcanza un 95 % de identificaciones correctas sobre las series del conjunto de trabajo, 133 sobre 140, como se observa en la tabla 8.3.

8.1.2. Validaciones

Una vez que se ha aplicado la metodología propuesta sobre un conjunto de datos reales pueden plantearse la influencia que sobre el resultado final tienen tanto el tamaño del conjunto aprendizaje como el valor del parámetro λ definido en el kernel presentado.

Con respecto al primero de estos interrogantes se realiza la aplicación de la metodología a diferentes divisiones del conjunto de datos. Los resultados presentados en la tabla 8.4 incluyen en este orden: el tamaño de los conjuntos de aprendizaje y verificación, el método con mejor comportamiento en su aplicación al conjunto de aprendizaje con el número de intervalos a obtener y la precisión obtenida, así como los intervalos realmente obtenidos y la precisión en la aplicación al conjunto de verificación.

Estos datos muestran dos características interesantes. Por un lado se observa como cuando el subconjunto de aprendizaje tiene los tamaños más pequeños el método discretización que mejor se comporta en el proceso de aprendizaje es *Ameva*, cambiando a intervalos de igual frecuencia (EFI) para tamaños superiores. En ambos casos el número de intervalos obtenido en el procedimiento de discretización es de 3.

Por otro, y siguiendo el comportamiento habitual, el valor de la precisión obtenida crece para hacerlo el tamaño del subconjunto de aprendizaje aún cuando se utilicen dos métodos diferentes. Éste comportamiento que se verifica en líneas generales tienen una mínima excepción en la aplicación a un conjunto de datos de aprendizaje de 140 elementos y puede observarse gráficamente en la figura 8.2.

La segunda interrogante planteada concierne a la influencia del valor λ sobre la

Tamaño Conj.		Mejor método en Apren.			Aplicación Test	
Apren.	Test	Método	Intervalos	Precisión	Intervalos	Precisión
112	252	Ameva	-	95,54 %	3	90,47 %
140	224	Ameva	-	97,14 %	3	89,29 %
168	196	EFI	3	97,02 %	3	90,82 %
196	168	EFI	3	96,94 %	3	92,26 %
224	140	EFI	3	96,43 %	3	95,00 %
252	112	EFI	3	94,84 %	3	97,32 %

Tabla 8.4: Método y precisión obtenida sobre los conjuntos de aprendizaje y test con diferentes tamaños de los mismos.

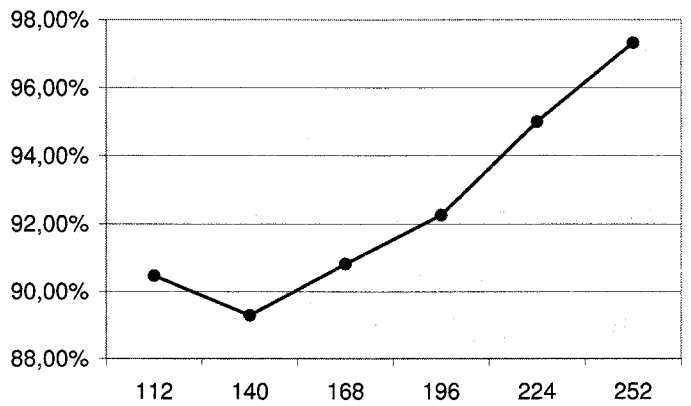


Figura 8.2: Porcentaje de identificación(%) en subconjunto de Test vs. Tamaño del conjunto de aprendizaje

precisión obtenida cuando se utiliza el kernel intervalar como medida de similitud. Ya se avanzó en la presentación del propio kernel la poca influencia de éste parámetros sobre los resultados y ahora se verificará realizando el experimento de aplicar la metodología sobre los conjuntos de aprendizaje y verificación utilizados en la sección anterior (112/252) variando el valor de λ.

Los resultados de esta aplicación, para valores del parámetro entre 0,1 y 0,9, se incluyen en la tabla 8.5 donde se ratifica el comentario anterior de la casi nula influencia del parámetro al no obtenerse diferencias superiores a medio punto porcentual.

λ	Precisión
0.1	90,08 %
0.2	90,48 %
0.3	90,48 %
0.4	90,48 %
0.5	90,48 %
0.6	90,48 %
0.7	90,48 %
0.8	90,08 %
0.9	90,08 %

Tabla 8.5: Precisión obtenida para diferentes valores de *lambda* sobre los conjuntos de aprendizaje y verificación (112-252)

8.2. Vocales

En las siguientes secciones se realiza la aplicación de la metodología a un conjunto de dato obtenido del repositorio de la Universidad de California en Irvine [Bay99].

El conjunto de vocales incluye el registro de 11 sonidos vocálicos (en inglés) pronunciados por 15 personas. Los registros de cuatro hombres y cuatro mujeres fueron utilizados como datos de entrenamiento y los de cuatro hombres y tres mujeres para la verificación de los resultados. Cada participante realizó seis pronunciaciones de cada una de las vocales.

Las señales registradas fueron filtradas mediante un paso bajo a $4{,}7kHz$ y digitalizadas a 12 bits con una frecuencia de muestreo de $10kHz$. Se aplicó un análisis predictivo lineal de orden 12 sobre seis segmentos obtenidos utilizando ventanas de Hamming con 512 muestras a partir de la parte estacionaría de cada vocal. Se utilizaron coeficientes de reflexión para calcular 10 parámetros de área logarítmica, obteniendo como resultado un espacio de 10 dimensiones.

En las figuras 8.3 y 8.4 se muestran los dos ejemplos de cada una de las 11 clases existentes en el conjunto de datos.

Del anterior procedimiento de filtrado y obtención de coeficientes se deduce que no estamos manipulando una serie temporal sino un conjunto de valores que representan determinadas características particulares de dicha serie. Estamos por tanto ante un conjunto de datos que presenta 10 atributos para cada una de las instancias incluidas y que por tanto es susceptible de aplicarse la metodología desde el punto de vista de los diferentes parámetros. Al hacerlo así cada uno de los atributos será discretizado de forma independiente.

La aplicación de la metodología a este conjunto de datos proporciona como mejor método sobre el conjunto de aprendizaje el método $CAIM$ utilizando como medida de similitud el Kernel Intervalar. Al realizar la aplicación de este método sobre el conjunto de test se obtienen los valores incluidos en la tabla 8.6.

Esta precisión está en línea con los niveles de precisión presentados por otros trabajos que oscilan entre el $62,55\,\%$ obtenida por [Die04] y el $56,0\,\%$ presentado en [Kub98]. En éste último trabajo también se presentan los resultados obtenidos

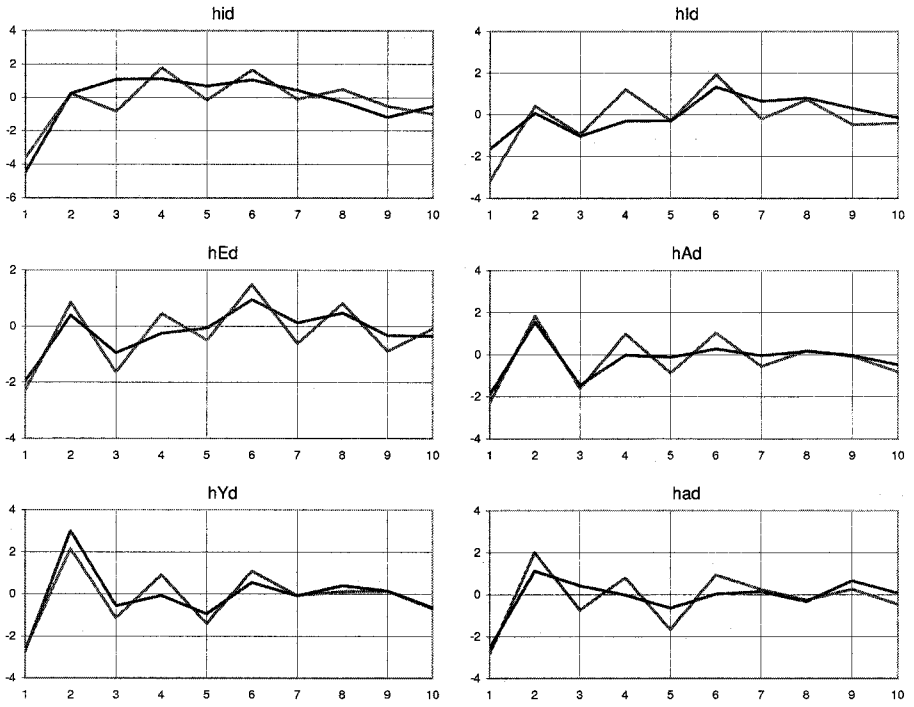


Figura 8.3: Ejemplos de las clases de VOW (1 de 2)

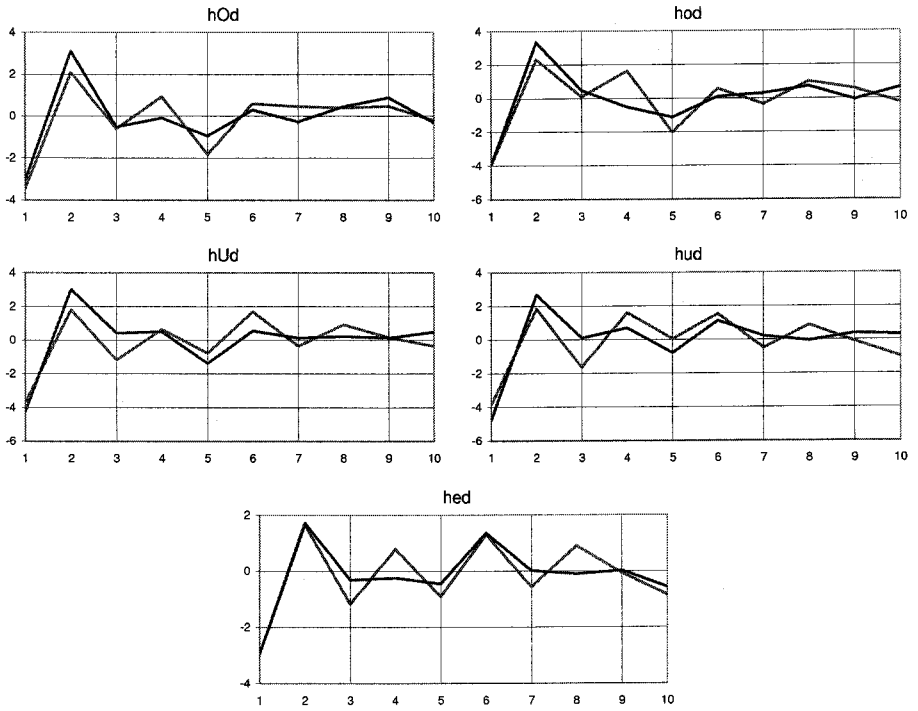


Figura 8.4: Ejemplos de las clases de VOW (2 de 2)

al aplicar una variedad importante de métodos basados en redes de neuronas pero todos obtienen precisiones inferiores.

Estos niveles de precisión relativamente bajos tienen su razón en la propia división del conjunto. Los datos del conjunto de verificación han sido obtenidos de hablantes distintos a los que generan el conjunto de entrenamiento. Si, como han realizado varios autores, se unen ambos subconjuntos y se utiliza una validación cruzada sobre el resultante, entonces los porcentajes de precisión superan el 80 %.

8.3. Disparos

Este conjunto de datos proviene del entorno de la vigilancia por vídeo y está disponible en el repositorio de series temporales para minería de datos de la Universidad de California Riverside [KF02] . El conjunto tiene dos clases, cada una de 100 instancias. Todas las instancias se crearon grabando las acciones de un actor y una

Mejor Método de Discretización :	CAIM
Número de Intervalos:	11(x10)
Medida de similitud:	Kernel Intervalar
Precisión media	55,63 %

Matriz de Confusión	Real										
Pred.	hid	hId	hEd	hAd	hYd	had	hOd	hod	hUd	hud	hed
hid	19	0	0	0	0	0	0	0	0	2	0
hId	23	29	3	0	0	0	0	0	0	7	0
hEd	0	6	28	2	0	1	0	0	0	3	0
hAd	0	0	0	30	0	0	0	0	0	0	3
hYd	0	0	0	0	15	9	7	0	0	0	2
had	0	2	9	10	22	24	0	0	0	0	3
hOd	0	0	0	0	5	0	21	1	8	2	0
had	0	0	0	0	0	0	14	38	5	1	0
hOd	0	3	0	0	0	0	0	3	14	18	4
hod	0	0	0	0	0	0	0	0	9	9	0
hUd	0	2	2	0	0	8	0	0	6	0	30
Por Vocal	45,2 %	69,1 %	66,7 %	71,4 %	35,7 %	57,1 %	50,0 %	90,5 %	33,3 %	21,4 %	71,4 %

Tabla 8.6: Conjunto Vocales. Resultado de identificación del conjunto de trabajo

actriz en una única sesión y las clases representan:

- Disparar: Los actores tienen sus manos a los lados. Cogen una réplica de pistola de la cartuchera situada en su cadera, apuntan al blanco durante un segundo, devuelven el arma a la cartuchera y sus manos a los lados.

- Apuntar: Los actores tienen sus manos a los lados. Apuntan al blanco durante un segundo con sus dedos índice, y colocan sus manos a los lados.

Para ambas clases se registró el centroide de la mano derecha en los ejes X e Y aunque el conjunto de datos sólo contiene la información del eje X.

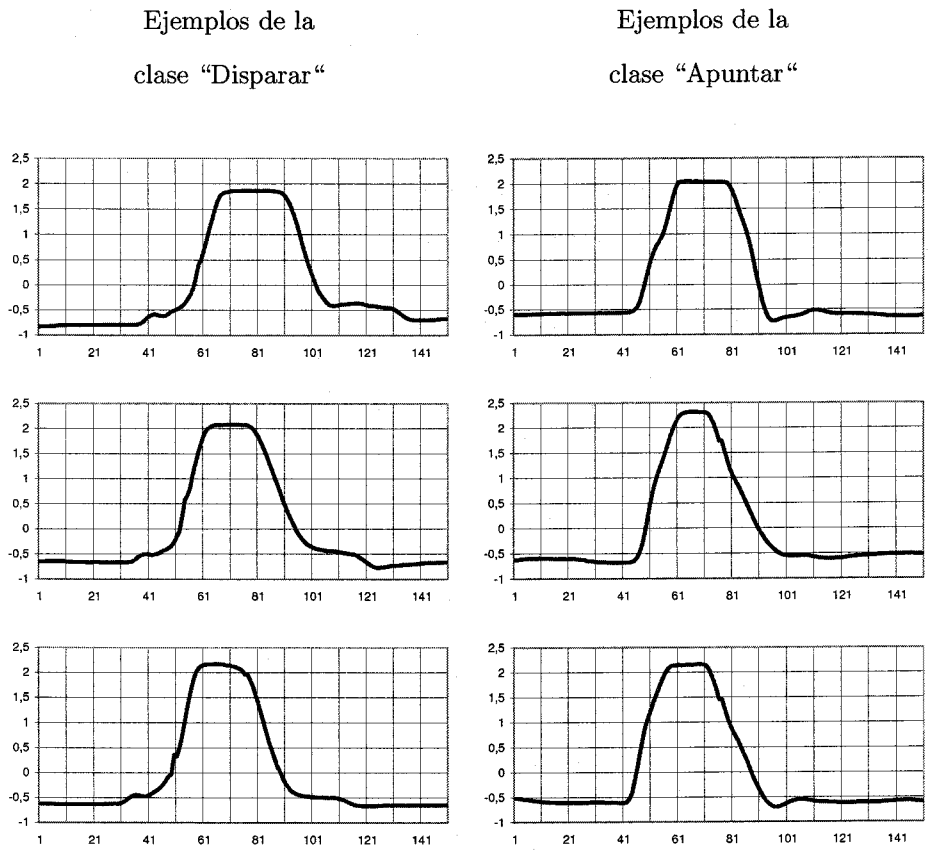


Figura 8.5: Ejemplos del conjunto Disparos

Cada instancia tiene la misma longitud de 150 puntos y está normalizada. El conjunto de datos se divide por la mitad teniendo cada subconjunto 100 instancias.

Varios ejemplos de cada una de las clases se pueden encontrar en la figura 8.5.

La primera fase de aplicación de la metodología establece el método de Intervalos de igual frecuencia (EFI) con 5 intervalos como el más adecuado para este conjunto de datos. Los resultados de aplicar la segunda fase con esta parametrización proporciona los siguientes valores:

Una precisión del $99,5\,\%$ que iguala a la mejor publicada en [Die04] utilizando tecnología de boosting sobre características extraídas de las series originales y que mejora las publicadas por [RK04] que obtiene un $5,5\,\%$ aplicando la distancia euclídea y del $4,5\,\%$ si se utiliza DTW en un algoritmo de vecino más cercano.

Mejor Método de Discretización :	EFI
Número de Intervalos:	5
Medida de similitud:	Kernel Intervalar
Precisión media	99,5 %

Matriz de Confusión	Real	
Pred.	Disparar	Apuntar
Disparar	99	0
Apuntar	1	100
Por Clase	99,0 %	100,0 %

Tabla 8.7: Conjunto Disparos. Resultado de identificación del conjunto de trabajo

CAPÍTULO 9

CONCLUSIONES Y TRABAJO FUTURO

Finalmente, en este capítulo se realizará una enumeración de las conclusiones obtenidas en la realización de esta tesis y de las líneas que han quedado abiertas para el trabajo futuro.

9.1. Conclusiones

Se ha presentado una aproximación cualitativa al problema de la comparación de series temporales. Esta nueva visión permite la aplicación de algoritmos contrastados diseñados en otros campos y posibilita una mayor comprensión de los resultados por parte de los usuarios.

Todos nuestros trabajos han sido unificados en una metodología *off-line* para la identificación de series temporales, por similitud con conjunto de entrenamiento dado. Dicha metodología presenta las siguientes características:

- Plantea un mecanismo automático.
- Presenta una gran simplicidad de uso estando exenta de parametrizaciones.

- Es aplicable a series de longitud fija y variable.

- Obtiene unos resultados de identificación en línea con los mejores resultados publicados hasta la fecha con cualquier otro método.

Esta metodología se ha implementado en una herramienta de distribución libre en la forma de una extensión al entorno *YALE*.

Inicialmente se ha presentado un índice basado en la comparación de cadenas que permite la detección de la similitud entre cadenas de distinta longitud aplicando la técnica de alineamiento temporal.

Ante la dificultad de definición de parámetros se ha optado por el desarrollo de un nuevo método de discretización cuyos objetivos son: el maximizar la interdependencia entre los intervalos en que se dividen los valores de un atributo y las clases a que pertenecen, proporcionando al mismo tiempo el mínimo número posible de intervalos.

Este método de discretización ha sido contrastado como uno de los algoritmos más prometedores resultando más rápido en su ejecución y proporcionando menores números de intervalos. Este comportamiento es sobresaliente cuando el conjunto de datos presentan un número elevado de clases, aunque se presenta una ligera reducción en la capacidad de identificación.

Es posible utilizar el método de discretización como paso previo para la transformación de un conjunto de datos antes de la aplicación de cualquier otro método de aprendizaje.

Finalmente se ha presentado una nueva distancia entre series temporales resultado de un proceso de discretización. Esta distancia, basada en los conjuntos de intervalos obtenidos en la discretización y en el concepto de las funciones núcleo, permite la evaluación de la similitud entre series de longitud fija.

9.2. Trabajo futuro

Nuestro principal objetivo en el futuro será el de desarrollar el índice y la distancia propuestos de forma que puedan aplicarse a conjuntos de datos con objetos que presenten múltiples atributos, incluso mezclando atributos simples y series completas. Quedaría en un segundo paso la generalización de la metodología para poder manipular conjuntos de datos mixtos, numéricos y nominales.

Aunque inicialmente proponemos la utilización, en la metodología, de uno de los métodos para los sucesivos procesos de identificación de nuevas series, también podría utilizarse una mixtura de los métodos presentados.

Resta, en todo caso, la aplicación de la metodología a un número mayor de conjuntos de datos con la intención de contrastar sus capacidades y de perfilar las características ideales de los conjuntos a que debe ser aplicada para obtener los óptimos resultados.

La distancia intervalar definida debe ser analizada y verificada en profundidad para el caso de series de longitud variable.

Una posibilidad abierta por la demostración de las características de distancia del núcleo intervalar permite la implementación de un sistema de indexación que explote la característica de la desigualdad triangular. Un camino sería el seleccionar un conjunto de series como patrones principales que permitirían realizar una poda rápida en las tareas de búsqueda de similitud.

Si esto se completa con la posibilidad de disponer de una cota inferior de la similitud entre dos series, lo que no parece complejo debido a la definición de la distancia al menos en el caso de longitudes fijas, se tendría un sistema totalmente eficiente.

Un último refinamiento a realizar en el futuro en el método de discretización *Ameva* es acelerar su cálculo. Se presentan dos opciones:

- Utilizar un método de cálculo similar al aplicado en $F - CAIM$: no se consideran como candidatos válidos al conjunto de cortes aquellos que provengan de la misma clase que el candidato adyacente. Se reduce así el número de candidatos y se acelera la ejecución.

- Permitir sólo como pares de cortes candidatos adyacentes aquellos que engloben, al menos, un porcentaje determinado de instancias en el conjunto de datos. Se parte de la idea que variaciones de muy pocas instancias (sobre el total) en los intervalos de la matriz de contingencia provocan variaciones muy pequeñas de los valores de *Ameva*.

APÉNDICE A

DFT

La Transformada Discreta de Fourier de n-puntos de una señal $\vec{x} = [x_i], i = 0, \ldots, n-1$ se define como la secuencia $\vec{X}$ de n números complejos $X_F, F = 0, \ldots, n-1$, tal que

$$X_F = 1/\sqrt{n} \sum_{i=0}^{n-1} x_i \exp(-j2\pi Fi/n) \quad con \quad F = 0, 1, \ldots, n-1$$

donde j es la unidad imaginaria, $j = \sqrt{-1}$. La señal $\vec{x}$ puede reconstruirse por medio de la transformada inversa:

$$x_i = 1/\sqrt{n} \sum_{F=0}^{n-1} X_F \exp(2\pi Fi/n) \quad con \quad i = 0, 1, \ldots, n-1$$

X_F es un número complejo (con la excepción de X_0 , que es real, si la señal $\vec{x}$ lo es). La energía $E(\vec{x})$ de una secuencia $\vec{x}$ se define com la suma de las energías (cuadrado de su amplitud $|x_i|$) para todo punto de la secuencia:

$$E(\vec{x}) \equiv \|\vec{x}\|^2 \equiv \sum_{i=0}^{n-1} |x_i|^2$$

Dicho de otra forma la DFT realiza una transformación del dominio del tiempo al dominio de la frecuencia.

Por otro lado, el teorema de Parseval [OS75] muestra que la distancia euclídea entre dos secuencias $\vec{x}$ e $\vec{y}$ en el dominio del tiempo es la misma que la distancia euclídea en el dominio de la frecuencia

$$\|\vec{x} - \vec{y}\| \equiv \|\vec{X} - \vec{Y}\|$$

Esta es la pieza fundamental para la aplicación de cualquiera de los métodos que aparecerán a continuación, ya que este mantenimiento de la distancia nos permite operar en cualquiera de los dos dominios, del tiempo o de la frecuencia, de forma indistinta y obtener resultados que son válidos en ambos.

Generalizando, podemos decir que toda transformación ortonormal posee esta característica de preservar la distancia. Las transformaciones ortonormales se dividen en dos clases:

1. las dependientes de los datos, que necesitan todas las secuencias para determinar la matriz de transformación, y
2. las que son independientes de los datos, como DFT , la Transformada Discreta del Coseno o DCT , o las trasformadas de paquetes de ondas, donde la matriz de transformación está determinada a-priori.

Las transformadas dependientes de los datos presentan la ventaja de poder ser optimizadas para cada conjunto de datos, pero si éstos evolucionan en el tiempo entonces es imprescindible un recálculo de la matriz de transformación para evitar la degradación.

La importancia de DFT radica en la existencia de un algoritmo de rápida computación, denominado Transformada Rápida de Fourier o FFT, que permite el cálculo de los coeficientes de DFT en un tiempo de $O(n \lg n)$, haciéndola computacionalmente implementable.

APÉNDICE B

LENGUAJE DE DEFINICIÓN DE FORMAS (SDL)

Este lenguaje de definición de formas (Shape Definition Language) propuesto en [APWZ95] es muy apropiado para realizar consultas sobre la forma de evolución de valores o magnitudes a lo largo del tiempo.

Dado un conjunto de valores registrados durante un periodo, la primera idea en *SDL* es dividir el intervalo de los posibles valores de las variaciones entre sucesivos elementos de la serie en rangos disjuntos y asignarle a cada uno una etiqueta.

La Figura B.1 presenta un ejemplo de división en tres zonas de la parte positiva de los posibles valores de variación y las etiquetas que se le han asignado. El comportamiento de una serie se describe considerando las transiciones entre las sucesivas muestras. Se construye una derivada con respecto al tiempo de la serie, calculando la diferencia de amplitud entre las muestras adyacentes. El valor de esas diferencias se encuadra en uno de los diferentes rangos disjuntos definidos proporcionando una etiqueta del alfabeto.

Así la traducción produce una secuencia de transiciones basada en un alfabeto cuyos símbolos describen la magnitud de los incrementos de los valores de la serie

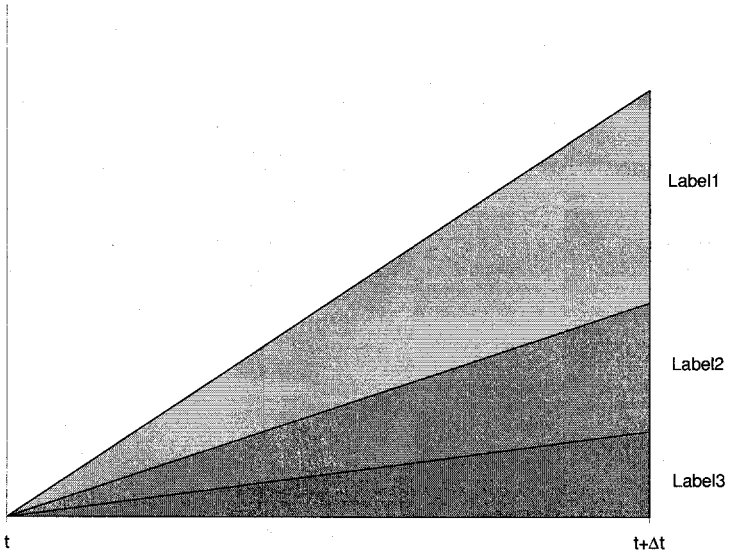


Figura B.1: Posible asignación de etiquetas

temporal.

Símbolo	Descripción	V.I.	V.S.	Rest.Inicial	Rest.Final
up	Transición levemente creciente	.05	.19	cualquiera	cualquiera
Up	Transición fuertemente creciente	.20	1.0	cualquiera	cualquiera
down	Transición levemente decreciente	-.19	-.05	cualquiera	cualquiera
Down	Transición fuertemente decreciente	-1.0	-.19	cualquiera	cualquiera
appears	Transición desde cero a un valor distinto de cero	0	1.0	cero	no cero
disappears	Transición desde un valor distinto cero a un valor cero	-1.0	0	no cero	cero
stable	Los valores inicial y final son aproximados	-.04	.04	cualquiera	cualquiera
zero	Los valores inicial y final son cero	0	0	cero	cero

Figura B.2: Ejemplo de alfabeto

Puede observarse en la figura B.2 un alfabeto donde cada símbolo es definido por cuatro descriptores. Los dos primeros, indicados en la figura por las abreviaturas "V.I.z "V.S.", son el límite inferior y superior permitidos a la variación entre los puntos inicial y final de la transición. Los dos últimos especifican las restricciones sobre los valores inicial y final de la transición, identificados por Rest.Inicialz - est.Finalrespectivamente.

Otra característica del alfabeto es que los conjuntos de transiciones posibles de los diferentes símbolos no están obligados a ser disjuntos; se permite que exista un grado de ambigüedad que permite definir con cadenas diferentes una misma serie.

Plasmando esta idea sobre el alfabeto mostrado en la figura B.2 se evidencia que una transición entre los valores 0 y 0,01 puede describirse con los símbolos *stable* o *appears* de forma equivalente.

El alfabeto propuesto en dicho trabajo [APWZ95] propone la utilización de ocho símbolos diferentes. El tamaño del alfabeto es muy pequeño comparado con los diferentes valores reales que la curva toma a lo largo del tiempo. Una traducción de este tipo permite dar prioridad a la forma sobre los valores originales.

En la Figura B.3 se muestra un ejemplo de traducción utilizando el alfabeto anteriormente expuesto (figura B.2).

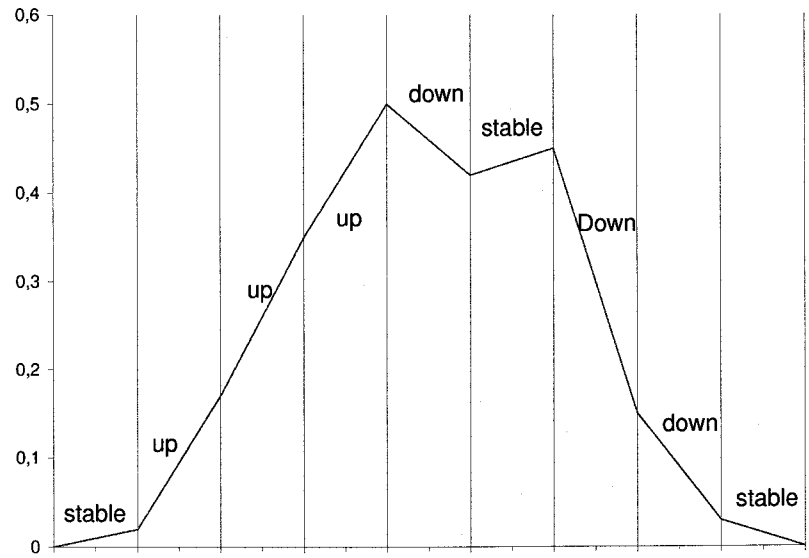


Figura B.3: Ejemplo de traducción

Cada cadena de símbolos puede describir un infinito número de curvas que cumplan con las restricciones impuestas por los símbolos a las transiciones que representan. La Figura B.4 muestra tres curvas diferentes cuya traducción produce la misma secuencia de símbolos; las curvas tienen diferentes puntos iniciales y finales aunque

debe reconocerse un grado elevado de similitud en su evolución.

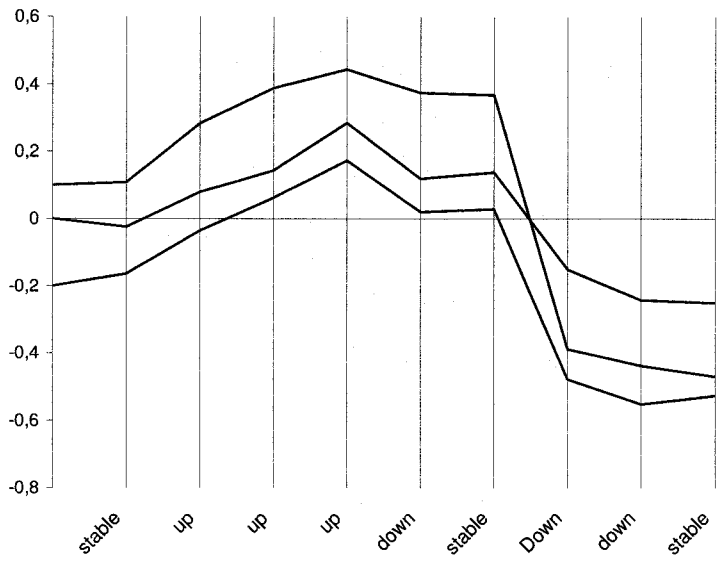


Figura B.4: Traducción con idéntica secuencia

El lenguaje *SDL* considera que cada uno de los símbolos del alfabeto describe una forma elemental y proporciona operadores para poder definir, partiendo de ellas, formas complejas derivadas. Se pueden generar nuevas formas por medio de las operaciones de concatenación, repetición, multiselección y parametrización de formas elementales u otras derivadas.

Finalmente en este trabajo también se propone un tipo de indexación y su mecanismo de almacenamiento para permitir la implementación de consultas que utilizan las operaciones de composición de formas.

B.1. Diferencias entre los etiquetados SDL y QSI

Como ya se indicó la utilización de este lenguaje en el presente trabajo a su método de etiquetado, quedando sin aplicar la composición de formas complejas, el mecanismo de indexación y la implementación de consultas.

Dentro del sistema de etiquetado la implementación que se realiza en *QSI* es

una versión simplificada y ligeramente modificada.

La variación fundamental entre el etiquetado QSI y SDL consiste en utilizar un único carácter para representar las etiquetas cualitativas en lugar de símbolos. Esta pequeña modificación posibilita la representación de las series de símbolos cualitativos como secuencias de caracteres en lugar de utilizar construcciones más complejas. Esta representación es la que abre la puerta a poder aplicar los algoritmos existentes para comparación de cadenas de caracteres, habiéndose completado la transformación del dominio del problema de la comparación de series temporales desde su perspectiva numérica a una comparación de cadenas cualitativas.

Asimismo la definición del alfabeto de conversión se ha simplificado en los siguientes aspectos:

- Los intervalos de valores deben ser disjuntos eliminándose la posible ambigüedad que se incluía en SDL.
- En el etiquetado QSI no se ha aplicado la posibilidad de definir restricciones a los valores inicial y final de cada intervalo.
- Por la definición de QSI se conoce que los valores de las series a etiquetar se encontrarán en el intervalo $[-1, 1]$.
- La definición del alfabeto se realiza en un número fijo de símbolos, siete, y su descripción se realiza en función de un único parámetro, λ, que es determinado por los expertos.

APÉNDICE C

CAIM

C.1. Algoritmo de Discretización *CAIM*

Partiendo de definición del problema de la discretización que se realizó en la sección 5.2, se tiene la matriz de contingencia

Tabla C.1: Tabla de contingencia.

$C_i \mid L_j$	L_1	$\ldots$	L_j	$\ldots$	L_k	$n_{i\cdot}$
C_1	n_{11}	$\ldots$	n_{1j}	$\ldots$	n_{1k}	$n_{1\cdot}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
C_i	n_{i1}	$\ldots$	n_{ij}	$\ldots$	n_{ik}	$n_{i\cdot}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
C_ℓ	$n_{\ell 1}$	$\ldots$	$n_{\ell j}$	$\ldots$	$n_{\ell k}$	$n_{\ell\cdot}$
$n_{\cdot j}$	$n_{\cdot 1}$	$\ldots$	$n_{\cdot j}$	$\ldots$	$n_{\cdot k}$	N

El criterio *CAIM* [KC04a, KC04b, KC03] utiliza la información de interdepen-

dencia clase-atributo como criterio para una discretización óptima. Se denota como $CAIM(k) \stackrel{def}{=} CAIM(\mathcal{L}(k), \mathcal{C}|X)$ y se define como

$$CAIM(k) = \frac{1}{k} \sum_{j=1}^{k} \frac{max_{i=1,\cdots,\ell} \{n_{ij}^2\}}{n_{\cdot j}}$$

El objetivo del método $CAIM$ es conseguir el menor número posible de intervalos y minimizar la pérdida de interdependencia entre $\mathcal{C}$ y $\mathcal{L}(k)$ [KC04a, KC04b, KC03] al tiempo que se evitan las desventajas de los algoritmos $CADD$ [CWC95] y $CAIUR$ [Hua96] [KC04a].

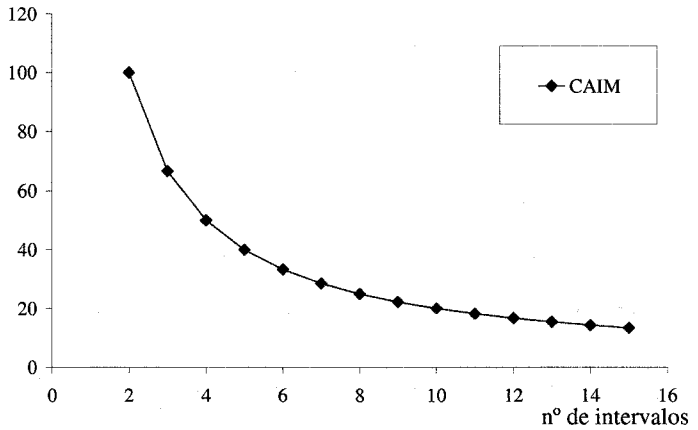


Figura C.1: $CAIM_{max}$ en función de k, para $\ell = 5$ y $N = 200$.

El valor de $CAIM(k)$ tiene las siguientes propiedades:

- El algoritmo favorece esquemas de discretización donde cada intervalo tiene todos sus valores agrupados en una única etiqueta de clase.
- Puede probarse que el valor mínimo de $CAIM(k)$ es

$$\min_{X,\mathcal{L}(k),\mathcal{C}} CAIM(k) = \frac{N}{k\ell^2}$$

  y este valor se alcanza cuando $n_{ij} = \frac{n_{\cdot j}}{\ell}$ para todo $i = 1, \cdots \ell$. Así, para una cierta tarea de clasificación este valor indica la peor situación posible.
- El valor máximo de $CAIM(k)$ significa la mejor correlación posible entre las etiquetas de clase y los intervalos discretos.

- El valor obtenido en el sumatorio es divido por k, el número de intervalos, lo que favorece esquemas con el menor número posible de intervalos.
- Se comprueba en [KC04a] que el valor máximo de $CAIM$ depende de k y

$$\max_{X,\mathcal{L}(k),C} CAIM(k) \stackrel{def}{=} CAIM_{max}(k) = \frac{N}{k}$$

esto es, $CAIM_{max}(k)$ decrece de manera hiperbólica, (véase la Fig. C.1), y además se tiene que

$$\max_{k\geq 2} CAIM_{max}(k) = \frac{N}{2}$$

Los inconvenientes que presenta este método ya han sido presentados en la sección 5.3.2.

APÉNDICE D

SOFTWARE DESARROLLADO

La implementación del software necesario para la ejecución de los algoritmos analizados y desarrollados en el proceso de elaboración de esta tesis ha dado lugar a la creación de un módulo de extensión sobre la plataforma YALE[RKF+01].

En las siguientes secciones se realiza una breve referencia a las características de YALE y posteriormente una revisión de la ampliación diseñada.

D.1. YALE

YALE es una plataforma de aprendizaje para experimentos de aprendizaje automático y minería de datos. Los experimentos pueden construirse por medio de un número arbitrario de operadores anidables y su configuración se describe con ficheros XML que pueden gestionarse fácilmente desde una interfaz gráfica.

Habiendo sido desarrollado por Unidad de Inteligencia Artificial de la Universidad de Dortmund desde el año 2001, en el 2004 el proyecto ha pasado a hospedarse en SourceForge ?? como un proyecto más de la comunidad de software libre, estando accesible bajo Licencia Pública General (o GPL) con lo que puede ser distribuida,

modificada y usada de forma libre y gratuita.

YALE se basa en la utilización de la unidad básica definida como operador, que realizan una tarea determinada sobre los datos de entrada. Los operadores pueden representar procesos de validación, de pensamiento, ampliar algo más.

La unidad básica dentro de YALE, como ya se ha indicado, es el operador que realizan tareas determinadas sobre los datos de entrada. Los operadores pueden representar procesos de acceso a datos, de procesamiento, aprendizaje y clasificación de los datos, así como la validación, análisis, almacenamiento y presentación de resultados.

Los operadores definen las entradas esperadas y las salidas que generan, junto con sus parámetros opcionales y obligatorios, lo que permiten al sistema comprobar la validez de la definición de los experimentos o procesos de aprendizaje.

La plataforma de aprendizaje YALE está implementada completamente en Java y de entre la enorme lista de características que incorpora podemos relacionar las siguientes:

- Algoritmos de Aprendizaje Automático: máquinas de soporte vectorial para regresión y clasificación, clasificadores mediante árboles de decisión ($C4{,}5$ y otros), algoritmos de agrupamiento, y un "wrapper.ª todos los clasificadores, métodos de agrupamiento, minería mediante reglas de asociación y múltiples esquemas de meta clasificadores incluidos en la herramienta $WEKA$.

- Operadores de características: algoritmos de selección como selección hacia adelante ("forward selection"), eliminación hacia atrás ("backward elimination") y varios algoritmos genéticos, operadores para la extracción de características de series temporales, ponderación de características, relevancia de características y generación de nuevos atributos.

- Preprocesado de Datos: discretización, filtrado de ejemplos y de características, reemplazo de valores infinitos e indeterminados, normalización, eliminación de características innecesarias, muestreado y reducción de la dimensionalidad.

- Evaluación de las prestaciones: validación cruzada y otros esquemas de validación, múltiples criterios de capacidad para clasificación y regresión, operadores para optimización de parámetros en operadores interiores o cadenas de operadores.

- Visualización: operadores para el almacenamiento y presentación de los resultados. Creación inmediata de gráficos en dos y tres dimensiones de los datos, los modelos aprendidos y los resultados de los experimentos.

- Entrada y salida: Operadores flexibles para gestión de entrada y salida, soporte de múltiples formatos de ficheros incluyendo *arff*, *C*4,5, *csv*, *bibtex*, *dBase* y acceso directo a gestores de bases de datos, con soporte de reordenaciones flexibles y utilización de la meta información existente.

Hay que reseñar especialmente la facilidad de gestión de experimentos que proporciona el entorno gráfico implementado. En la figura D.1 se puede observar el fichero descriptor de un experimento en que se realiza una validación cruzada sobre un conjunto de datos al que se aplica un aprendizaje por SVM y se evalúa en función de tres criterios. Opuestamente en la figura D.2 se presenta su estructura visual dentro del entorno de YALE presentando el bloque central izquierdo la estructura de operadores y el derecho los parámetros del operador seleccionado, en este caso el clasificador SVM.

Una característica fundamental de YALE es el permitir a los usuarios la implementación de nuevos operadores. Para implementar un operador el usuario necesita conocer los tipos de entrada, las salidas que generará, sus parámetros y la funcionalidad interna del operador. Este operador será implementado en Java y se agregará de forma dinámica, en tiempo de ejecución, al entorno sin necesidad de recopilación ni de alterar la plataforma completa.

Esta característica permite que, manteniendo un núcleo desarrollado de forma centralizada, pueden diseñarse ampliaciones por parte de la comunidad científica que amplíen su capacidad sin necesidad de generar nuevas versiones.

```
<operator name="Root" class="Experiment">
  <operator name="Input" class="ExampleSource">
    <parameter key="attributes" value="data/polynomial.xml"/>
  </operator>
  <operator name="XVal" class="XValidation">
    <operator name="Training" class="LibSVMLearner">
      <parameter key="kernel_type"  value="poly"/>
      <parameter key="C"    value="1000.0"/>
      <parameter key="svm_type" value="epsilon-SVR"/>
    </operator>
    <operator name="ApplierChain" class="OperatorChain">
      <operator name="Test" class="ModelApplier">
      </operator>
      <operator name="Evaluation" class="PerformanceEvaluator">
        <parameter key="squared"    value="true"/>
        <parameter key="absolute"   value="true"/>
        <parameter key="main_criterion" value="scaled"/>
      </operator>
    </operator>
  </operator>
</operator>
```

Figura D.1: Código XML correspondiente a un experimento de validación cruzada.

D.2. Detalle de implementación

Esta implementación permite realizar funciones de discretización y comparación de similitudes en series de longitud fija o variable.

Para la definición de series de longitud variable se ha realizado una ampliación a las especificaciones dadas en YALE, donde todo atributo debe tener una longitud conocida y exacta, no estando siquiera contemplado por la ampliación de gestión de series temporales.

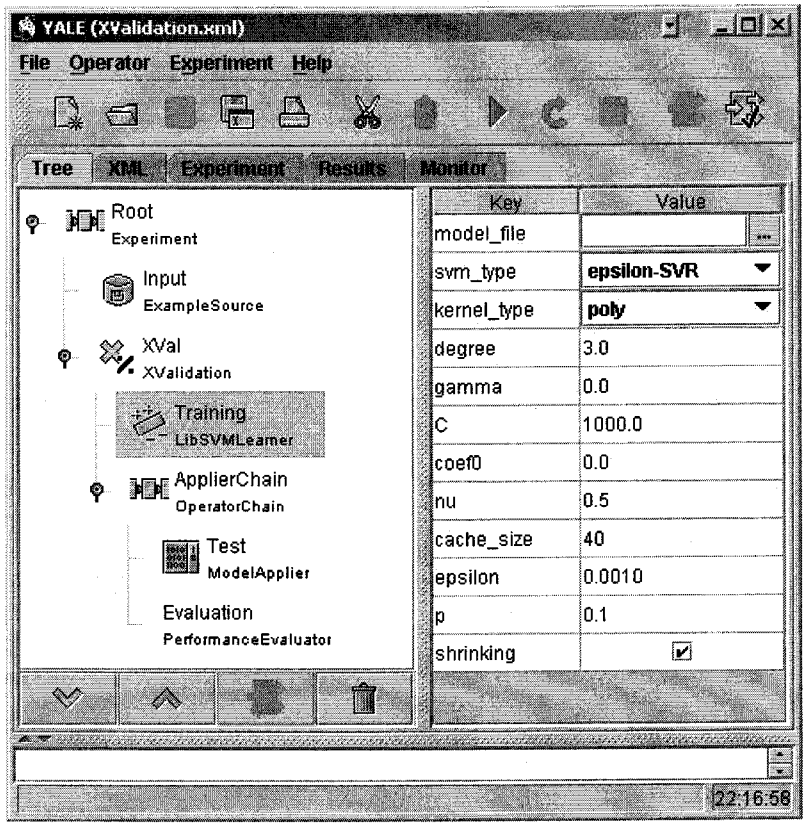


Figura D.2: Configuración visual de experimento con validación cruzada.

Para definir series de longitud variable se incluirá un atributo en el conjunto de datos cuyo nombre será exactamente VLS_SIZE. Su valor indicará, para cada uno de los ejemplos, el número total de valores que deben considerarse de cada una de las series que lo componen; aparezcan en el conjunto de datos; el resto de los valores, hasta completar el tamaño máximo que se ha definido para cada atributo, puede presentar cualquier valor y no será considerado por el operador (se sugiere que los atributos desconocidos presentan un valor que genere un Double.NaN en su procesado).

Las razones que ha llevado a esta implementación de un nuevo atributo en lugar de utilizar los conjuntos dispersos ya incluidos en YALE son:

- En los conjuntos dispersos todos los valores que no sean especificados serán completados como 0 lo que no permite al operador reconocer el final de la

serie, ya que entre los valores válidos de la serie pueden existir valores nulos.

- Aunque los valores no indicados apareciesen con un valor especial, como Double.NaN, sería necesario encontrar del tamaño de la serie y almacenarlo como un nuevo atributo.

Los métodos de discretización implementados son:

- Ameva [GCVO04]
- CAIM [KC04a]
- CUM [Coc92]
- EFI o Intervalos de igual frecuencia
- EWI o Intervalos de igual tamaño

Mientras que la similitud entre series se tiene por medio de los algoritmos

- QSI [COGT02a]
- IntervalKernel [GVC+04]
- DTW [SC78]

Los métodos de discretización pueden aplicarse a conjuntos de datos con atributos independientes. En este caso se produce la discretización de los valores existentes para cada atributo y su conversión en un carácter. No tiene sentido (por su propia definición) la aplicación de los métodos de similitud a conjuntos de atributos independientes.

En la siguiente subsección se presenta una revisión del principal operador desarrollado para la aplicación de los métodos y algoritmos relacionados. La información del operador ha sido reestructurada siguiendo la convención del "Tutorial de Yale", Referencia de Operadores"(capítulo 5), mientras que en la figura D.3 se muestra el aspecto visual del operador y sus parámetros.

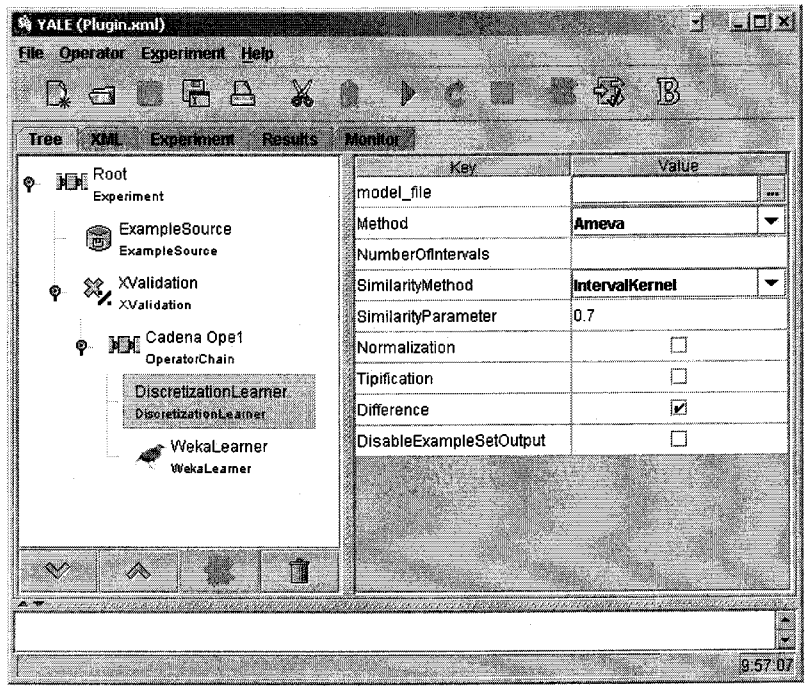


Figura D.3: Ejemplo de uso del operador creado con sus parámetros.

D.2.1. DiscretizationLearner

Entradas:

- ExampleSet

Salidas generadas:

- DiscretizationModel
  - ExampleSet

Parámetros:

- **model_file**:

- **method**: Método de discretización. (Ameva, CAIM, CUM, EFI, EWI; por omisión: Ameva).

- **number_of_intervals**: Número de intervalos de discretización. NOTA: sólo para los métodos EWI y EFI. entero 2-+∞; por omisión : 2) - **similarity_method**:

Método de similitud para clasificación de series. IntervalKernel sólo está definido para series de longitud fija. (QSI, IntervalKernel; por omisión: IntervalKernel)

- **similarity_parameter**: Parámetro del método de similitud. Sin aplicación en QSI, el valor de λ en IntervalKernel. (real; $-\infty$-$+\infty$; por omisión:0.0)

- **normalization**: Preprocesado de las series. Normalización [0,1]. (binario; por omisión: falso)

- **tipification**: Preprocesado de las series. Tipification. (binario; por omisión: falso)

- **difference**: Preprocesado de las series. Diferencia de valores adyacentes. (binario; por omisión: falso)

- **disable_expampleset_output**: Deshabilitar la generación de una versión discretizada del conjunto de ejemplos de entrada. (binario; por omisión: true)

Resultados:

- **num_intervals**: Número de intervalos de discretización obtenidos en el proceso.

- **applycount**: Número de aplicaciones del operador.

- **looptime**: Tiempo transcurrido desde el inicio de la iteración actual.

- **time**: Tiempo transcurrido desde el inicio de la ejecución del operador.

Descripción:

Este operador permite la aplicación de procedimientos de discretización a series temporales. Para ello:

- Calcula un conjunto de intervalos de discretización para cada una de las series

incluidas en el conjunto de datos de entrada (*LearningSet*).

- Este esquema de discretización se aplica sobre el *LearningSet* para obtener una versión discretizada del mismo, o *DiscLearningSet*.

El esquema de discretización y el *DiscLearningSet* componen el modelo generado por este operador.

La aplicación del operador sobre un nuevo conjunto de entrada, o *TestSet*, realiza la predicción de la clase de cada uno de los ejemplos. Para la clasificación se utiliza un algoritmo del vecino más cercano de los ejemplos de *TestSet* discretizados, por el esquema calculado inicialmente, con los ejemplos de *DiscLearningSet*. Esta clasificación se basa en la similitud entre las series.

Para la tarea de discretización se han implementado métodos supervisados y los supervisados. Los métodos no supervisados de igual frecuencia (EFI) e igual amplitud (EWI) de los intervalos de discretización requieren como parámetro (**number_of_interv** el número de intervalos deseados para cada atributo.

La similitud entre dos ejemplos puede calcularse por medio de los métodos QSI, IntervalKernel o DTW. El método de similitud IntervalKernel permite la especificación de un parámetro λ que puede modificarse por el usuario **similarity_parameter**. El valor de λ debe estar en el rango $[0, 1]$.

NOTA: Aplicación a conjuntos de ejemplos con atributos normales. En el caso de que el conjunto de ejemplos de entrada no presente ningún atributo de tipo sería la aplicación del operador se realizará sobre cada uno de los atributos discretizándose los de tipo numérico. en este caso se requiere que un método de aprendizaje para la clasificación de nuevos conjuntos de entrada ya que no tiene sentido utilizar QSI o IntervalKernel sobre conjuntos de ejemplos que no sean de tipo serie.

D.2.2. Ejemplos de uso

Aquí se muestran algunos ejemplos de uso del operador presentado.

```
<attributeset default_source=".\cuota.dat">
  <attribute
    name       ="share"
    sourcecol  ="1"
    sourcecol_end = "96"
    valuetype  ="real"
    blocktype  ="value_series"
    blocknumber = "10"
  />
  <id
    name       ="id"
    sourcecol  ="97"
    valuetype  ="integer"
    blocktype  ="single_value"
  />
  <label
    name       ="tv_station"
    sourcecol  ="98"
    valuetype  ="nominal"
    blocktype  ="single_value"
  />
</attributeset>
```

Figura D.4: Fichero descriptores datos para series de longitud fija.

Primero un ejemplo de un conjunto de datos con series de longitud fija. Los datos representan la cuota de pantalla de 7 cadenas de televisión y una descripción detallada puede encontrarse en [COVG03b]. El fichero de atributos presenta una serie de 96 elementos, una identificación entera y una etiqueta. Éste fichero se incluye en la figura D.4.

Como ya se comentó la utilización de series de longitud variable obliga a la inclusión de un atributo de nombre **"VLS_SIZE"**. En la figura D.5 se presenta un fichero descriptor de un conjunto con series de longitud variable. El fichero es un subconjunto (sólo el valor x de la mano izquierda) del conjunto de signos australianos [BM98]. La primera columna indica la longitud de la serie. Las columnas de la 2 a la 151 acogen los valores de la serie, habiéndose completado los valores desconocidos

con un signo "?". Los datos incluidos en las columnas 152 a 451 son ignorados.

```
<attributeset default_source=".\ASL.dat">
    <attribute
    name       ="VLS_SIZE"
    sourcecol  ="1"
    valuetype  ="integer"
    blocktype  ="single_value"
  />
  <attribute
    name       ="LX"
    sourcecol  ="2"
    sourcecol_end = "151"
    valuetype  ="real"
    blocktype  ="value_series"
  />
  <id
    name       ="id"
    sourcecol  ="452"
    valuetype  ="integer"
    blocktype  ="single_value"
  />
  <label
    name       ="palabra"
    sourcecol  ="453"
    valuetype  ="nominal"
    blocktype  ="single_value"
  />
</attributeset>
```

Figura D.5: Fichero de descripción de atributos con series de longitud variable.

Finalmente se presenta un ejemplo de experimento utilizando el operador. Las series son normalizadas y diferenciadas, siendo calculada la descripción por medio del algoritmo *Ameva*. La similitud de la serie de test se calculan por el método IntervalKernel. El código de este experimento se presenta en la figura D.6.

```
<operator name="Root" class="Experiment">
  <parameter key="logfile"  value=".\logX_cuota"/>
  <parameter key="logverbosity" value="operator"/>
  <operator name="ExampleSource" class="ExampleSource">
    <parameter key="attributes" value=".\cuota.att"/>
  </operator>
  <operator name="XValidation" class="XValidation">
    <operator name="DiscretizationLearner" class="DiscretizationLearner">
      <parameter key="method"   value="Ameva"/>
      <parameter key="similarity_method"    value="IntervalKernel"/>
      <parameter key="normalization"    value="true"/>
      <parameter key="difference"   value="true"/>
    </operator>
    <operator name="OperatorChain" class="OperatorChain">
      <operator name="ModelApplier" class="ModelApplier"> </operator>
      <operator name="PerformanceEvaluator" class="PerformanceEvaluator">
        <parameter key="absolute"   value="true"/>
        <list key="additional_performance_criteria"> </list>
        <parameter key="classification_error"   value="true"/>
        <parameter key="accuracy"   value="true"/>
      </operator>
      <operator name="ExperimentLog" class="ExperimentLog">
        <list key="log">
          <parameter key="Paso" value="operator.XValidation.value.iteration"/>
          <parameter key="Absoluto"
             value="operator.PerformanceEvaluator.value.absolute"/>
          <parameter key="Classif.Error"
             value="operator.PerformanceEvaluator.value.classification_error"/>
          <parameter key="Accuracy"
             value="operator.PerformanceEvaluator.value.accuracy"/>
        </list>
        <parameter key="filename" value=".\logExperimento"/>
      </operator>
    </operator>
  </operator>
</operator>
```

Figura D.6: Validación cruzada de un proceso de aprendizaje con discretización *Ameva*. Preprocesado de las series con normalización y diferenciación. Similitud basada en IntervalKernel.

APÉNDICE E

PUBLICACIONES

E.1. Publicaciones realizadas

En este apéndice se incluyen las publicaciones y colaboraciones que se han generado en la tarea de investigación de la presente tesis. La tabla E.1 incluye la lista completa de éstas agrupadas por tipos.

Publicaciones en Revistas	
	[LCM+03],[COGT02c], [COGdlR01a]
Capítulos de Libros	
	[JRAJ04]
Contribuciones a Congresos	
	[GVC+04],[COGV04],[Cub04], [COVG03a][COVG03b],[Cub03], [COGT02a],[COGT02b], [COGP02], [COGdlR01b]

Tabla E.1: Listado de Publicaciones

BIBLIOGRAFÍA DEL TRABAJO

[AFS93] R. Agrawal, C. Faloutsos, and A. Swami. Efficient similarity search in sequence databases. In *Proc. of the Fourth Intl. Conf. on Foundations of Data Organization and Algorithms (FODO '93)*, 1993.

[AG03] C. Angulo and L. González. 1-v-1 Tri-Class SV Machine. In *Proc. 11th European Symposium on Artificial Neural Networks, ESANN*, 2003.

[AL96] M. Ato and J.J. López. *Análisis Estadístico para datos categóricos*, volume 1 of *Metodología de las ciencias del comportamiento*. Sintesis, S. A., 1996. In Spanish.

[ALSS95] R. Agrawal, K.I. Lin, H.S. Sawhney, and K. Shim. Fast similarity search in the presence of noise,scaling, and translation in time series databases. In *The 21st VLDB Conference*, 1995.

[Ang01] C. Angulo. *Learning with Kernel Machines into a Multi-Class Environment*. Doctoral thesis, Technical University of Catalonia, April 2001. In Spanish.

[Apo97] A. Apostolico. *Handbook of Formal Languages*, volume 2 Linear Modeling: Background and Application, chapter String Editing and Longest Common Subsequences, pages 361–398. 1997.

[APWZ95] R. Agrawal, G. Psaila, E. L. Wimmers, and M. Zaït. Querying shapes of histories. In *The 21st VLDB Conference*, pages 502–514., 1995.

[ATTA04] S. Ahmad, T. Taskaya-Temezil, and K. Ahmad. Summarizing time series: Learning patterns in 'volatile' series. In Z.R. Yang R. Everson and H. Yin, editors, *Proceedings of the 5th International Conference on Intelligent Data Engineering and Automated Learning (IDEAL 2004)*, number 3177 in Lecture Notes in Computer Science, pages 523–532, Heidelberg, 2004. Springer-Verlag.

[Bay99] Stephen D. Bay. The uci kdd archive, 1999.

[BB81] P. Bertier and J.M. Bouroche. *Analyse des données multidimensionnelles.* Presses Universitaires de France, 1981. In French.

[BC94] D.J. Berndt and J. Clifford. Finding patterns in time series: A dynamic programming approach. In *AAAI-94 Workshop on Knowledge Discovery in Databases (KDD-94.)*, 1994.

[BKK96] S. Berchtold, D.A. Keim, and H.-P. Kriegel. The x-tree : An index structure for high-dimensional data. In *Proc. of the* 22^{th} *VLDB Conference*, pages 28–39, 1996.

[BKSS90] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The r*-tree: an eficient and robust access method for points and rectangles. In *ACM SIGMOD*, pages 322–331, 1990.

[BM98] C.L. Blake and C.J. Merz. Uci repository of machine learning databases. http://www.ics.uci.edu/ mlearn/MLRepository.html, UC Irvine, Depto Information and Computer Science, 1998.

[Bou01] M. Boullé. Khiops: discretization des attributs numériques pour le data mining. Note technique NT/FTR&D/7339, France Telecom R& D, 2001. In French.

[Bou04] M. Boullé. Khiops. a statistical discretization methods of continuous attributes. *Machine Learning*, (55):53–69, 2004.

[Cat91a] J. Catlett. *Megainduction: machine learning on very large databases.* PhD thesis, University of Sydney, 1991.

[Cat91b] J. Catlett. On changing continuous attributes into ordered discrete attributes. In *Proceeding European working session on learning*, pages 164–178, 1991.

[CB91] P. Clark and R. Boswell. Rule induction with cn2: Some recent improvements. In Springer-Verlag, editor, *Proceeding of the European Working Session on Learning*, Lecture Notes in Artificial Intelligence:, 1991.

[CH74] D.R. Cox and D.V. Hinkley. *Theoretical Statistics.* Chapman and Hall, 1974.

[CK01] K.J. Cios and L. Kurgan. Hybrid inductive machine learning: An overview of clip algorithms. In Jain C. and Kacprzyk J., editors, *New Learnning Paradigms in Soft Computing*, pages 276–322. Physica-Verlag (Springer), 2001.

[CK02] K.J. Cios and L. Kurgan. Hybrid inductive machine learning algorithm that generates inequalities rules. *Information Science*, page acepted, 2002. Special Issue of Soft Computing Data Mining.

[CKHP02] S. Chu, E.J. Keogh, D. Hart, and M.J. Pazzani. Iterative deepening dynamic time warping for time series. In *To appear in the Second SIAM International Conference on Data Mining (SDM-02)*, 2002.

[CMG02] J. Colomer, J. Meléndez, and F. Gamero. Pattern recognition based on episodes and dtw. application to diagnosis of a level control system. In *16th International Workshop on Qualitative Reasoning - QR02*, pages 37–44, Sitges, Barcelona (Spain), 2002.

[CMP+00] G. Cormode, S. Muthukrishnan, M. Paterson, S.C. Sahinalp, and U. Vishkin. Techniques and applications for approximating string distances. 2000.

[CN89] P. Clark and Y.Ñiblett. The cn2 algorithm. *Machine Learning*, (3):261–283, 1989.

[CN04] Lei Chen and Raymond Ng. On the marriage of lp-norms and edit distances. In *Proc. of the* 30^{th} *International Conference on Very Large Data Base*, pages 792–803, 2004.

[Coc77] W.G. Cochran. *Técnicas de muestreo.* Editorial Continental, Mexico, 6 edition, 1977. in Spanish.

[Coc92] W.G. Cochran. *Técnicas de muestreo.* Editorial Continental, Mexico, 1992. in Spanish.

[COGdlR01a] F.J. Cuberos, J.A. Ortega, R.M. Gasca, and F. de la Rosa. Comparación cualitativa de series temporales. Índice cualitativo de similitud - qsi. *Computación y Sistemas*, 5(2):96–108, October 2001. In Spanish.

[COGdlR01b] F.J. Cuberos, J.A. Ortega, R.M. Gasca, and F. de la Rosa. Qsi - indice cualitativo de similitud. aplicación a un modelo semicualitativo de crecimiento logístico con retraso. In *Primeras Jornadas de Trabajo sobre Diagnosis.* ARCA, 2001. In spanish.

[COGP02] Francisco J. Cuberos, Juan A. Ortega, Marcela Genero, and Mario Piattini. Aplicación del Índice cualitativo de similitud (qsi). In *III Jornadas de Trabajo DOLMEN (2002)*, 2002.

[COGT02a] F.J. Cuberos, J.A. Ortega, R.M. Gasca, and M. Toro. Qsi - qualitative similarity index. *Butlleti de l'Associació Catalana d'Intelligència Artificial*, (28):121–128, 2002.

[COGT02b] F.J. Cuberos, J.A. Ortega, R.M. Gasca, and M. Toro. Qsi - qualitative similarity index. In *16º International Workshop on Qualitative Reasoning - QR02*, pages 45–51, Sitges, Barcelona (Spain), 2002.

[COGT02c] F.J. Cuberos, J.A. Ortega, R.M. Gasca, and M. Toro. Qualitative comparison of temporal series. qsi. *Lecture Notes on Artificial Intelligence*, 2504:75–78, October 2002.

[COGV04] F.J. Cuberos, J.A. Ortega, L. González, and F. Velasco. A methodology for qualitative learning in time series. In *18º International Workshop on Qualitative Reasoning*, 2004.

[Coh95] W.W. Cohen. Fast effective rule induction. In *Proceeding of the Twelfth International Conference on Machine Learning*, pages 115–123, 1995.

[COVG03a] F.J. Cuberos, J.A. Ortega, F. Velasco, and L. González. Qsi - alternative labelling and noise sensitivity. In *17º International Workshop on Qualitative Reasoning*, pages 999–999, Brasilia, (Brasil), 2003.

[COVG03b] F.J. Cuberos, J.A. Ortega, F. Velasco, and L. González. Qsi - labelling and noise sensitivity. In *Congreso CAEPIA*, volume Actas, pages 445–448, 2003. ISBN 84-8383-564-4.

[CS90] J.T. Cheung and G. Stephanopoulos. Representation of process trend - part ii. the problem of scale and qualitative scaling. *Computers and Chemical Engineering*, pages 511–539, 1990.

[CST00] N. Cristianini and J. Shawe-Taylor. *An introduction to Support Vector Machines and other kernel-based learning methods.* Cambridge University Press, 2000.

[Cub03] F.J. Cuberos. Series temporales: mecanismos de indexación y comparación. In *V Jornadas ARCA*, 2003. In spanish.

[Cub04] F.J. Cuberos. Una metodología para el aprendizaje cualitativo en series temporales. In *VI Jornadas ARCA*, 2004. In spanish.

[CW99] K.K.W. Chu and M.H. Wong. Fast time-series searching with scaling and shifting. In *Proceedings of the 18th ACM SIGACT-SIGMOD-*

SIGART Symposium on Principles of Database Systems (PODS'99), pages 237–248, 1999.

[CWC91] D. Chiu, A. Wong, and B. Cheung. *Information Discovery through Hierachical Maximum Entropya Discretization and Synthesis.* Knowledge Discovery in Database. MIT Press, 1991.

[CWC95] J.Y. Ching, A.K.C. Wong, and K.C.C. Chan. Class-dependent discretization for inductive learning for continuous and mixed-mode data. *IEEE Transactions on Patern Analysis and Machine Intelligence*, 17(7):641–651, 1995.

[CWc99] K. Chan and F.A. Wai-chee. Efficient time series matching by wavelets. In *Proc. 15th International Conference on Data Engineering*, 1999.

[DGM97] G. Das, D. Gunopulos, and H. Mannila. Finding similar time series. In *In proceedings of Principles of Data Mining and Knowledge Discovery,* 1^{st} *European Symposium*, pages 88–100, 1997.

[Die04] Juan José Rodríguez Diez. *Técnicas de Aprendizaje Automático para la Clasificación de Series.* PhD thesis, Universidad de Valladolid, 2004.

[DKS95] J. Dougherty, R. Kohavi, and M. Sahami. Supervised and unsupervised discretization of continuous features. In A. Preditis and S. Russell, editors, *Machine learning: Proceeding of the twelfth international conference*, San Francisco, 1995. Morgan Kaufmann Publishers.

[FI92] U.M. Fayyad and K.B. Irani. On the handling of continuous valued attributes in decision tree generation. *Machine Learning*, 8:87–102, 1992.

[FL95] C. Faloutsos and K.I. Lin. Fastmap: A fast algorithm for indexing, data-minig and visualization of traditional and multimedia datasets. In *The ACM SIGMOD Conference on Management of Data*, 1995.

[FPG03] Eugene Fink, Kevin B. Pratt, and Harith Suman Gandhi. Indexing of time series by major minima and maxima. In *Proceedings of th IEEE Internationa Conference on Systems, Man and Cybernetics*, pages 2332–2335, 2003.

[FRM94] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series databases. In *The ACM SIGMOD Conference on Management of Data*, pages 419–429, 1994.

[GAVO04] L. Gonzalez, C. Angulo, F. Velasco, and J.A. Ortega. Núcleos, distancias y similitudes entre intervalos e hipercubos. *Intelegencia Artificial*, pages 999–999, 2004. in spanish.

[GAVV02] L. González, C. Angulo, F. Velasco, and M. Vilchez. Máquina ℓ-SVCR con salidas probabilísticas (ℓ-SVCR machine with probabilistic outputs). *Inteligencia Artificial. Revista Iberoamericana de IA*, (17):72–82, 2002. In Spanish.

[GCVO04] L. González, F.J. Cuberos, F. Velasco, and J.A. Ortega. Núcleos, distancias y similitudes entre intervalos e hipercubos. Technical Report TR-005, Dpt. of Applied Economy I. University of Seville (Spain), 2004. in spanish.

[GG01] L. González and J.M. Gavilán. Una metodología para la construcción de histogramas. Aplicación a los ingresos de los hogares andaluces. *XIV Reunión ASEPELT España*, 2001. In Spanish.

[GGK01] V. Guralnik, N. Garg, and V. Kumar. Parallel tree projection algorithm for sequence mining. In *In Proc. of EuroPar2001*, 2001.

[GK94] D.Q. Goldin and P.C. Kanellakis. On similarity queries for time-series data: constraint specification and implementation. In *In 1st Intl. Conf. on the Principles and Practive of Constraint Prog*, pages 419–429., 1994.

[GK01a] V. Guralnik and G. Karypis. Dynamic load balancing algorithms for sequence mining. Technical Report 00-056, Department of Computer Science, University of Minnesota, 2001.

[GK01b] V. Guralnik and G. Karypis. A scalable algorithm for clustering protein sequences. In *Workshop on Bioinformatics BIOKDD*, pages 73–80, 2001.

[GMK03] Dina Q. Goldin, Todd D. Milstein, and Ayferi Kutlu. Flexible and efficient similarity queying for time-series data. Technical Report TR-03-4, UConn BECAT/CSE, University of Connecticut, 2003.

[Gol04] Dina Q. Goldin. Normalization of life science data for shape-based similarity querying. Technical Report BECAT/CSE TR-04-1, University of Connecticut, January 2004.

[Gon02] L. González. *Análisis discriminante utilizando máquinas núúcleos de vectores soporte. Función núcleo similitud (Discriminative analysis using vector kernel machines support. The similitary kernel function).* Doctoral thesis, University of Seville, 2002. In Spanish.

[Gut84] A. Guttman. R-trees: a dynamic index structure for spatial searching. In *ACM SIGMOD Conf. on the Management of Data*, pages 47–57. ACM, 1984.

[GVA+04] L. González, F. Velasco, C. Angulo, J.A. Ortega, and F. Ruiz. Sobre núcleos, distancias y similitudes entre intervalos. *Inteligencia Artificial*, 8(23):113–119, june 2004.

[GVC+04] L. González, F. Velasco, F.J. Cuberos, J.A. Ortega, and C. Angulo. A kernel to use with a discretization of continuous features. In *Proceeding of the Learning'04. International Conference*, pages 45–50, Elche, Spain, 2004. Universidad Carlos III.

[GVG05] L. González, F. Velasco, and R.M. Gasca. A study of the similarities between topics. *Computational Statistics*, 20(3), 2005. In press.

[GW02] L. Gao and X. Wang. Improving the performance of continuous queries on fast data streams: Time series case. In *SIGMOD/DMKD workshop 2002*, June 2002.

[Haa10] A. Haar. Theorie der orthogonalen funktionen-systeme. *Mathematische Annalen*, pages 331–371, 1910.

[HKT99] Y. Huhtala, J. Kärkkäinen, and H. Toivonen. Mining for similarities in aligned time series using wavelets. In *Data Mining and Knowledge Discovery: Theory, Tools, and Technology. SPIE Proc*, page Vol. 3695, 1999.

[Hol93] R. C. Holte. Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11:63–90, 1993.

[Hua96] W. Huang. *Discretization of Continuous Attributes for Inductive Machine Learning.* PhD thesis, Dept. Computer Science, Univerity of Toledo, Ohio, 1996.

[JB97] H.A. Jönsson and D.Z. Badal. Retrieval of one-dimensional data. In *First European Symposium on Principles of Data Mining and Knowledge Discovery*, 1997.

[JMM95] H. Jagadish, A.O. Mendelzon, and T. Milo. Similarity-based queries. In *In Proc. of the 14th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'95)*, pages 36–45., 1995.

[JRAJ04] Mo Jamshidi, Matthias Reuter, Diego Andina, and Jila S. Jamshidi, editors. *Soft Computing With Industrial Applications*, volume 17, pages 229–239. TSI Press, Alburquerque, NM, USA, 2004.

[KC03] L. Kurgan and K.J. Cios. Fast class-attribute interdependence maximization (CAIM) discretization algorithm. In M. Arif Wani, Krzysztof J. Cios, and Khalid Hafeez, editors, *Proceedings of the 2003 In-*

ternational Conference on Machine Learning and Applications (ICMLA), pages 30–36. CSREA Press, 2003.

[KC04a] L. Kurgan and K.J. Cios. CAIM discretization algorithm. *IEEE Transactions on Knowledge and Data Engineering*, 16(2):145–153, 2004.

[KC04b] L. Kurgan and K.J. Cios. Falta falta falta falta falta falta falta. *FALTA FALTA FALTA*, 16(2):145–153, 2004.

[KCMP01] E.J. Keogh, K. Chakrabarti, S. Mehrotra, and M.J. Pazzani. Locally adaptative dimensionality reduction for indexing large time series databases. In *In Proc. of ACMSIGMOD*, pages 151–162, 2001.

[KCP01] E.J. Keogh, S. Chu, and M.J. Pazzani. Ensemble-index: A new approach to indexing large databases. In *Proc. of the 7th ACM SIGKDD International Conf. On Knowledge Discovery and Data Mining*, pages 117–125, 2001.

[KCPM01] E.J. Keogh, K. Chakrabarti, M.J. Pazzani, and S. Mehrotra. Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and Information Systems*, pages 263–286, 2001.

[Keo02] E. J. Keogh. Exact indexing of dynamic time warping. In *Proc. of the* 28^{th} *International Conf. On Very Large Databases*, 2002.

[Ker92] R. Kerber. Chimerge: Discretization of numerical attributes. In AAAI Press. the MIT Press, editor, *Proc. AAAI-9, 9th International conference on Artificial Intelligence*, pages 123–128, 1992.

[KF93] Ibraim Kamel and Christos Faloutsos. On packing r-trees. In *Second Intl. Conf. on Information and Knowledge Management (CIKM)*, 1993.

[KF02] Eamonn Keogh and T. Folias. The UCR time series data minig archive, 2002. http://www.cs.ucr.edu/ eamonn/TSDMA/index.html.

[KJF97] F. Korn, V. Jagadish, and C. Faloutsos. Efficiently porting ad hoc queries in large datasets of time sequences. In *In Proc. of the ACM SIGMOD Conf. of Management of Data*, 1997.

[KK02] E. Keogh and S. Kasetty. On the need for time series data mining benchmarks: A survey and empirical demonstration. In *Proc. of th* 8^{th} *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 23–26, 2002.

[KLR04] E.J. Keogh, S. Lonardi, and C. Ratanamahatana. Towards parameter-free data mining. In Won Kim, Ron Kohavi, Johannes Gehrke, and William DuMouchel, editors, *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, Washington, USA, August 22-25, 2004*, pages 206–215, 2004.

[KM99] K. A. Kaufman and R. S. Michalski. Learning from inconsistent and noisy data: The AQ18 approach. In *Proceedings of 11th International Symposium on Methodologies for Intelligence Systems*, pages 411–419, 1999.

[KP98] E.J. Keogh and M.J. Pazzani. An enhaced representation of time series wich allows fast and accurate classification, clustering and relevance feedback. In *Proc. 4th International Conference of Knowledge Discovery and Data Mining*, pages 239–241. AAAI Press, 1998.

[KP99] E.J. Keogh and M.J. Pazzani. Scaling up dynamic time warping to massive datasets. In *Proc. Principles and Practice of Knowledge Discovery in Databases*, 1999.

[KP00a] E.J. Keogh and M.J. Pazzani. Scaling up dynamic time warping for datamining applicactions. In *Proc. of the 6th ACM SIGKDD International Conf. On Knowledge Discovery and Data Mining*, pages 285–289, 2000.

[KP00b] E.J. Keogh and M.J. Pazzani. A simple dimensionality reduction technique for fast similarity search in large time series databases. In *In Proc. of the Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 2000.

[KP01] E.J. Keogh and M.J. Pazzani. Derivative dynamic time warping. In *In First SIAM International Conference on Data Mining (SDM'2001)*, 2001.

[KPC01] S.-W. Kim, S. Park, and W.W. Chu. An index-based approach for similarity search supporting time warping in large sequence databases. In *In Proc. 17th IEEE Int'l Conf. on Data Engineering*, 2001.

[KR04] Eamonn J. Keogh and Chotirat Ann Ratanamahatana. Exact indexing of dynamic time warping. *Knowledge and Information Systems*, 7(3):358–386, 2004.

[KS96] R. Kohavi and M. Sahami. Error-based and entropy-based discretization of continuous features. In J. Han E. Simoudis and U. Fayyad, editors, *Proc. of the 2nd International Conference on Knowledge Discovery and Data Mining*, pages 114–119. AAAI Press, 1996.

[KS98] E.J. Keogh and P. Smyth. A probabilistic approach to fast pattern matching in time series databases. In *Proceedings of the 9th International Conference on Tools with Artifial Inteligence*, pages 578–584. IEEE Press, 1998.

[KS01a] T. Kahveci and A. Singh. An efficient index structure for string databases. In *In VLDB*, pages 351–360, 2001.

[KS01b] T. Kahveci and A. Singh. Variable length queries for time series data. In *In Proceedings of the 17th Intl. Conf. on Data Engineering*, 2001.

[KSG01] T. Kahveci, A. Singh, and A. Gurel. Shift and scale invariant search of multi-attribute time sequence. 2001.

[KSG02] T. Kahveci, A. Singh, and A. Gurel. Similarity searching for multi-attribute sequences. In *SSDBM 2002*, 2002.

[Kub98] Miroslav Kubat. Desision trees can initialize radial-basis function networks. *IEEE Transactions on Neural Networks*, 9:813–821, 1998.

[Kur04] L. Kurgan. Personal communication, 2004.

[LCM+03] D. Llanos, F.J. Cuberos, J. Meléndez, Fco. I. Gamero, J. Colomer, and J.A. Ortega. Recognition of system behaviours based on temporal series similarity. *Computación y Sistemas*, 7:1–16, July 2003.

[LKLC03] J. Lin, E. Keogh, S. Lonardi, and B. Chiu. A symbolic representation of time series, with implication for streaming algorithms. In *Proc. of the 8th ACM SIGMOD Workshop on Reaserch Issues in Data Mining and Knowledge Discovery*, 2003.

[LLM04] Quanzhong Li, Inés Fernando Vega López, and Bongki Moon. Skyline index for time series. *IEEE Transactions on Knowledge and Data Engineering*, 16(6):669–684, June 2004.

[LS97] Huan Liu and Rudy Setiono. Feature selection via discretization. *IEEE Transactions on Knowledge and Data Engineering*, 9(4):642–645, 1997.

[MHBD03] A. Macskassy, H. Hirsh, A. Banerjee, and A. Dayanik. Converting numerical classification into text classification. *Artificial Inteligence*, (143):51–77, 2003.

[MMHL86a] R. Michalski, I. Mozetic, J. Hong, and N. Lavrač. The AQ15 inductive learning system: an overview and experiments. In *Proceedings of IMAL 1986*, Orsay, 1986. Université de Paris-Sud.

[MMHL86b] R. Michalski, I. Mozetic, J. Hong, and N. Lavrač. The multipurpose incremental learning system AQ15 and its testing application to three

medical domains. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 1041–1045. Morgan-Kaufmann, 1986.

[MP65] Modenov and Pakhomenko. Geometric transformations. *Academic Press*, 1965.

[OGT99] J.A. Ortega, R.M. Gasca, and M. Toro. A semiqualitative methodology for reasoning about dynamic systems. In *In the 13th International Workshop on Qualitative Reasoning*, pages 169–177., 1999.

[OS75] V. Oppenheim and R.W. Schafer. *Digital Signal Processing.* Prentice Hall, Englewood Cliffs, N.J., 1975.

[PCYH00] S. Park, W.W. Chu, J. Yoon, and C. Hsu. Efficient searches for similar subsequences of different lengths in sequence databases. In *Proc. IEEE ICDE,*, pages 23–32, 2000.

[Pfa95] B. Pfahringer. Compression-based discretization of continuous attributes. In A. Preditis and S. Russell, editors, *Machine learning: Proceeding of the twelfth international conference*, pages 456–463, San Francisco, 1995. Morgan Kaufmann Publishers.

[PH74] T. Pavlidis and S. Horowitz. Segmentation of plane curves. *IEEE Transactions on Computers*, 1974.

[PLC99] S. Park, D. Lee, and W. W. Chu. Fast retrieval of similar subsequences in long sequence databases. In *In Proc. 3rd IEEE Knowledge and Data Engineering Exchange Workshop (KDEX)*, pages 60–67, 1999.

[PLC01] S. Park, D. Lee, and W. W. Chu. Segment-based approach for subsequence searches in sequence databases. In *Proceedings of the 2001 ACM Symposium on Applied Computing (SAC)*, 2001.

[PWZP00] C.S. Perng, H. Wang, S.R. Zhang, and D.S. Parker. Landmarks: a new model for similarity-based pattern querying in time series databases. In *Proc. IEEE ICDE*, pages 33–42, 2000.

[Qui93] J.R. Quinlan. *C 4.5 Programs for Machine Learning.* Morgan Kaufmann, 1993.

[Raf99] D. Rafiei. On similarity-based queries for time series data. In *In Proc. of the 15th International Conference on Data Engineering*, 1999.

[Ris99] K.M. Risvik. Discretization of numerical attribute - preprocessing for machine learning. Project 45073, Knowledge Systems Group. Dpto of computer and Information Science, Norwegian University of Science and Technology. N-7034 Trondheim, Norway, 199?

[RJ93] L. Rabiner and B. Juang. *Fundamentals of speech recognition.* Prentice Hall, Englewood Cliffs,N.J., 1993.

[RK04] C. Ratanamahatana and E.J. Keogh. Making time-series more accurate using kernels learned constraints. In *Proceedings of the SIAM International Conference on Data Mining (SDM'04)*, 2004.

[RKF+01] Oliver Ritthoff, Ralf Klinkenberg, Simon Fischer, Ingo Mierswa, and Sven Felske. *LLWA 01 - Tagungsband der GI-Workshop-Woche Lernen - Lehren - Wissen - Adaptivität*, chapter Yale: Yet Another Machine Learning Environment, pages 84–92. Number 763. Dortmund, Germany, 2001.

[RM98a] D. Rafiei and A. Mendelzon. Efficient retreival of similar time sequences using dft. In *In Proc. of the 5th Intl. Conf. on Foundations of Data Organization and Algorithms (FODO '98)*, 1998.

[RM98b] D. Rafiei and A. Mendelzon. Similarity-based queries for time data series. In *In Proc. of the ACM SIGMOD Intl. Conf. of Management of Data(SIGMOD '97)*, pages 13–24, 1998.

[SC78] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Trans. Acoustics Speech and Signal Proc.*, 1978.

[Sof] Sofres. Sofres - TNS Audiencia de Medios. A Service of Sofres AM company. http://www.sofresam.com.

[SS02] B. Schölkopf and A. J. Smola. *Learning with Kernel.* MIT Press, 2002.

[Ste94] G.A. Stephen. *String Searching Algorithms.* World Scientific Publishing, 1994.

[SZ96] H. Shatkay and S. Zdonic. Approximate queries and representation for large data sequences. In *In Proc. of the 12th International Conference on Data Engineering*, pages 546–553, 1996.

[TS02] F.E.H. Tay and L. Shen. A modified chi2 algorithm for discretization. *IEEE Transactions on Knowledge and Data Engineering*, 14(3):666–670, 2002.

[Vap98] V. Vapnik. *Statistical Learning Theory.* John Wiley & Sons, Inc., nada edition, 1998.

[VGK02] M. Vlachos, D. Gunopoulos, and G. Kollios. Discovering similar multidimensional trajectories. In *18th International Conference on Data Engineering (ICDE'02).*, pages 673–684, 2002.

[Vla00] P. Vlachos. Statlib project repository. http://lib.stat.cmu.edu/ datasets/csb/, 2000.

[WAA00] D. Wu, D. Agrawal, and A. Abbadi. A comparison of dft and dwt based similarity search in time-series databases. In *Proc. of the 9th International Conference on Information and Knowledge Management*, 2000.

[WC87] A.K.C. Wong and D.K.Y. Chiu. Synthesizing statistical knowledge from incomplete mixed-mode data. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 9:796–805, 1987.

[WF74] R.A. Wagner and M.J. Fisher. The string-to-string correction problem. *Journal of the ACM*, pages 168–173., 1974.

[WP99] H. Wang and C.S. Perng. The s^2-tree: An index structure for subsequence matching of spatial objects. Technical Report Technical Report 990050, Computer Science Department, University of California, Los Angeles, 1999.

[YF00] B.K. Yi and C. Faloutsos. Fast time sequence indexing for arbitrary l_p norms. In *Proceedings of the 26th Intl. Conf. on Very Large Databases*, 2000.

[YJF98] B.K. Yi, H. Jagadish, and C. Faloutsos. Efficient retrieval of similar time sequences under time warping. In *IEEE Intl. Conf. on Data Engineering*, pages 201–208., 1998.

[YÖ96] N. Yazdani and M. Z. Özsoyoglu. Sequence matching of images. In *Proc. of the Eighth Intl. Conference on Scientific and Statistical Database Management*, pages 53–62, 1996.

[ZR00] D.A. Zighed and R. Rakotomalala. *Graphes d´induction.* Hermes Science Publications, 2000.

FRANCISCO JAVIER CUBEROS GARCIA-BAQUERO

INDICE PARA LA COMPARACIÓN CUALITATIVA DE SERIES TEMPORALES

SOBRESALIENTE CUM LAUDE

17 JUNIO 2005

El Presidente,