Construcción de un Clasificador Borroso mediante Programación Genética basada en Registros

Luis Junco Navascués  Luciano Sánchez Ramos
Departamento de Informática. Universidad de Oviedo
E.S.M.C. Campus de Viesques
Gijón. Asturias
junco,luciano@lsi.uniovi.es

Resumen

Un sistema de clasificación borroso puede inducirse automáticamente a partir de una muestra de ejemplos clasificados, mediante la combinación de un algoritmo genético y de un lenguaje basado en reglas borrosas. En este trabajo definiremos una metodología en la que se realiza una estimación no paramétrica de las funciones de pertenencia que definen la semántica del lenguaje mencionado, mediante una estrategia de tipo Pittsburg combinada con técnicas propias de la programación genética. La programación genética se emplea para inducir las expresiones de las funciones de pertenencia de los conjuntos borrosos que definen la semántica del clasificador. Estas expresiones consisten en una secuencia de instrucciones interpretables en una máquina de registros.

Palabras clave: Programación Genética Basada en una Máquina de Registros, Sistemas Basados en Reglas Borrosas Evolutivos, Algoritmos GA-P

1 Introducción

Los sistemas basados en reglas borrosas son una de las aplicaciones más importantes de la teoría de conjuntos borrosos de Zadeh. Estos sistemas extienden los sistemas clásicos basados en reglas y se han aplicado a una gran cantidad de problemas, entre los cuales está la generalización borrosa del problema del análisis discriminante. Llamaremos a este problema clasificación borrosa.

La clasificación borrosa admite que la frontera entre dos clases vecinas no es una superficie sino una región borrosa, de forma que un objeto puede tener pertenencia parcial a varias clases. Este enfoque no sólo refleja la realidad de que muchas categorías reales tienen fronteras difusas, sino que también proporciona una representación simple de una partición potencialmente complicada del espacio de características. Usando este esquema podemos aplicar reglas si-entonces para describir un clasificador [12][13]. Una regla típica de clasificación podría ser

si \( X_1 \) es \( A_1 \) y \( X_2 \) es \( A_2 \) y \( \ldots \) y \( X_n \) es \( A_n \)
entonces \( Z \) es \( C \)

donde \( X_1, X_2, \ldots, X_n \) son las variables de entrada y \( A_1, A_2, \ldots, A_n \) son variables lingüísticas, caracterizadas por funciones de pertenencia apropiadas, que describen los valores de las propiedades del objeto \( Z \). La fuerza con que dispara una regla, o el grado de validez de ésta, es el grado con el que el objeto pertenece a la clase \( C \). Llamaremos descriptivo a un banco de reglas definido de este modo.

Cuando cada regla define una región borrosa arbitraria (es decir, que no necesariamente sea el resultado de componer etiquetas lingüísticas mediante conectivas lógicas) la interpretabilidad del sistema es menor, pero cada regla puede cubrir regiones de forma más compleja y en general el banco será más sencillo. Los sistemas aproximativos constan de reglas

si \( (X_1, X_2, \ldots, X_n) \) es \( A \) entonces \( Z \) es \( C \)

donde \( A \) es un conjunto borroso no necesariamente asociado a un término lingüístico.

El objetivo de este trabajo es encontrar expresiones adecuadas para las funciones de pertenencia de las variables \( A_i \) en los sistemas descriptivos o para los conjuntos \( A \) en los sistemas aproximativos. Estas pertenencias han de dar lugar a reglas compatibles con una muestra de objetos clasificados previamente. Este estudio ya ha sido realizado por otros autores cuando las funciones de pertenencia se definen paramétricamente. Entre otros métodos, se pueden buscar los valores de los parámetros desconocidos mediante un algoritmo genético [9][1]. También hay sistemas que ajustan modificadores más o menos complejos [7], lo que permite mayor riqueza en la definición. Nosotros no definiremos paramétricamente los conjuntos borrosos, sino
que buscaremos un algoritmo que computará la forma de esa función. Las únicas limitaciones impuestas al algoritmo son su longitud máxima (definida como el número de instrucciones que emplea) y la ausencia de composiciones iterativas y llamadas recursivas en su definición. Llamaremos a esta representación funcional. En la representación funcional, cada uno de los conjuntos borrosos se asociará a una secuencia de instrucciones en una máquina de registros, cuya ejecución (tomando como dato el valor de las variables de entrada) producirá como salida el valor de la función de pertenencia. Estas secuencias de instrucciones se inducirán mediante Programación Genética.

2 Derivación de una base de reglas borrosas mediante Fuzzy GP y Fuzzy GA

La "programación genética borrosa" (Fuzzy GP) es un nuevo método de aprendizaje de máquina que se basa en la combinación de un algoritmo genético con un lenguaje de reglas borrosas. En contraste con los algoritmos genéticos borrosos (Fuzzy GA) [1] la programación genética borrosa no optimiza los parámetros de funciones de pertenencia trapecoidales o con forma de campana de una base de reglas con estructura fija, sino que explora diferentes estructuras de clasificadores definidos mediante un lenguaje borroso a partir de un diccionario en el que se definen las variables lingüísticas y varios modificadores [3]. La programación genética borrosa no cambia la semántica del lenguaje de reglas borrosas, pero en cambio es capaz de derivar la estructura de la base de reglas, que no necesita definirse por el programador. De este modo, las variables que no contribuyen a producir buenas soluciones son eliminadas durante el proceso de evolución. Se hace notar que ésta no es la única aplicación de la programación genética a los sistemas de clasificación. Por ejemplo, un clasificador basado en una idea similar, pero empleando funciones discriminantes y un lenguaje no borroso, fue aplicada también a problemas de visión artificial, en una metodología orientada a reducir al mínimo el tiempo de clasificación en problemas con gran número de entradas [11] y pueden encontrarse otras aplicaciones en [6].

Por otra parte, es bien sabido que los algoritmos genéticos se combinan con los sistemas borrosos de tres formas fundamentales, conocidas como enfoques Michigan, Pittsburg e Iterativo [1] y que con estos métodos se pueden inducir tanto la estructura del banco como su semántica. En este estudio emplearemos programación genética en combinación con el segundo de estos métodos (y por tanto nuestro método, aun siendo un algoritmo de programación genética aplicado al aprendizaje de reglas borrosas, no está relacionado directamente con la programación genética borrosa tal y como ha sido definida en [3]). Plantearnos el empleo de la programación genética en la determinación de la semántica del lenguaje y no en la determinación de la estructura del banco, lo que es un problema abierto, según este mismo autor:

However, a lot of work remains to be done.
The most important problems to solve are to increase the speed of convergence while maintaining stability, [...], including the definition of the membership function in the genetic coding, ...

La solución que proponemos supone una mejoría respecto a los métodos basados solamente en algoritmos genéticos ya que la codificación de las funciones de pertenencia no es paramétrica. Cada función de pertenencia se codifica mediante un fragmento de programa con lo que la gama de funciones representables es muy amplia. Nuestra propuesta está basada en la combinación de dos metodologías: la Programación Genética Basada en una Máquina de Registros y los Algoritmos GA-P.

La primera de las técnicas logra una codificación muy eficiente de un programa en una cadena de símbolos similares a las empleadas en algoritmos genéticos. Esta representación tiene dos ventajas importantes con respecto a la programación genética canónica [6], en la que el código se representa mediante árboles: la evaluación de los individuos requiere menos cálculos, lo que permite simular un número mayor de generaciones, y la evolución de la población hacia soluciones adecuadas requiere menor número de generaciones, debido a la forma en que se definen los operadores de cruce y mutación. Por otro lado, los algoritmos GA-P mejoran a la programación genética clásica en la forma en que se manejan las constantes, ya que el valor de éstas se hace evolucionar, al contrario que en el algoritmo de Kosa, y permiten el empleo de técnicas de optimización local [2]. La programación genética basada en registros y los algoritmos GA-P se describen, respectivamente, en [10] y [5].

3 Metodología propuesta

El método de diseño de clasificadores que proponemos es una variante del enfoque Pittsburg [9]. Hemos asociado las pertenencias de los conjuntos borrosos que definen las etiquetas lingüísticas con secuencias de instrucciones de longitud fija. El lenguaje que hemos empleado se interpreta en una máquina elemental basada en registros, que describiremos en el siguiente apartado.
Al igual que en el algoritmo GA-P de Howard [5] cada individuo consta de dos partes: la parte GA y la parte GP. En la parte GA se almacenan valores de constantes, con codificación real, y en nuestra variante del GA-P en la parte GP se concatenan tantas cadenas de instrucciones como valores pueden tomar las variables lingüísticas de las que hace uso el banco de reglas borrosas.

En los bancos de reglas descriptivas se completarán dos fases: en primer lugar se define una sintaxis del problema (se asignan funciones de pertenencia trapezoidales a las variables $A_1, \ldots, A_n$) y se determina la estructura del banco de reglas que optimice un índice de funcionamiento. En esta etapa se puede incorporar el conocimiento lingüístico del que se disponga acerca del sistema de clasificación (se mostrará un ejemplo práctico de esta incorporación de conocimiento en una sección posterior). En la segunda fase se optimizan las funciones de pertenencia reemplazando sus definiciones por secuencias de instrucciones que se hacen evolucionando mediante el esquema de la programación genética basada en registros. La definición de los sistemas aproximativos es aun más sencilla; la representación funcional que proponemos permite que cada clase sea discriminada por una única regla sin que se precise abordar la primera de las fases.

Las partes GA y GP pueden tratarse de forma homogénea, al contrario que en el GA-P de Howard. Emplearemos las operaciones de cruce y mutación definidas en [10] sobre ambas partes.

3.1 La máquina virtual de registros

Como hemos visto, cada función de pertenencia se define mediante una función, implementada en el lenguaje de una máquina de registros que definimos a continuación.

En nuestra máquina todas las instrucciones realizan operaciones aritméticas sobre los registros. Cada programa lleva asociado un vector de entrada, un número determinado de constantes, una serie de registros y diversos procesos de inicialización y término de la ejecución. La longitud máxima de un programa está limitada a un número dado de instrucciones aunque, de hecho, su longitud efectiva va a ser siempre menor, ya que gran parte del código no interviene en el resultado final. Estos fragmentos de código no utilizado o introns son necesarios para el proceso de búsqueda.

El proceso de inicialización asigna el valor cero a cada uno de los registros. El proceso terminal calcula la media aritmética de una serie de registros seleccionados (normalmente el registro $R_0$) y aplica al resultado una función sigmoide con un rango de salida entre 0.0 y 1.0 para normalizar la salida del programa.

3.1.1 Conjunto de instrucciones

El conjunto de instrucciones de la máquina virtual de registros define la base del comportamiento de las funciones de pertenencia que vamos a codificar con los mismos.

Para los casos más generales se han implementado las siguientes operaciones (llamamos $R_i$ al registro $i$-ésimo, $K_j$ a la constante $j$-ésima y $X_k$ a la entrada $k$-ésima). Todas las operaciones están protegidas para evitar desbordamientos.

<table>
<thead>
<tr>
<th>Instrucción</th>
<th>$R_i \leftarrow 0$</th>
<th>$R_i \leftarrow R_i$</th>
<th>$R_i \leftarrow K_j$</th>
<th>$R_i \leftarrow X_k$</th>
<th>$R_i \leftarrow R_i + K_j$</th>
<th>$R_i \leftarrow R_i - K_j$</th>
<th>$R_i \leftarrow R_i * K_j$</th>
<th>$R_i \leftarrow R_i \div K_j$</th>
<th>$R_i \leftarrow \sin(R_i)$</th>
<th>$R_i \leftarrow \cos(R_i)$</th>
<th>$R_i \leftarrow \cos(X_k)$</th>
</tr>
</thead>
</table>

4 Resultados Experimentales

En orden a analizar y comparar las características de nuestro clasificador lo hemos aplicado a dos problemas clásicos, bien conocidos: Iris y Pima. Hemos empleado bancos de reglas aproximativos, con una única regla por clase (o lo que es lo mismo, los conjuntos borrosos obtenidos son una partición del espacio de características).

En la próxima sección aplicaremos la metodología expuesta anteriormente para solucionar un problema práctico de clasificación con un banco de reglas descriptivo.
4.1 Iris

El problema inicialmente plantead por Fisher [14] cuenta con 150 ejemplos en los que se han medido cuatro atributos con los que tratamos de diferenciar tres tipos diferentes de flores (setosa, versicolor y virgínica).

Hemos aplicado el método de validación cruzada con diez particiones aleatorias para medir la capacidad de aprendizaje en cada uno de los conjuntos de entrenamiento y calcular una estimación del error a través de los subsiguientes tests. En la función de fitness, que mede la adecuación del clasificador durante la etapa de entrenamiento, intervienen dos factores: el porcentaje de aciertos por un lado, y la longitud del programa generado por otro. Este último está relacionado con la capacidad de generalización del clasificador. A menos instrucciones se corresponden funciones más sencillas, menos tendentes a sobreentrenar el sistema.

Para cada una de las series hemos lanzado 10 veces el programa de generación del clasificador, eligiendo aquel con mejor ajuste para realizar el test. Los resultados para cada una de las series se muestran a continuación:

<table>
<thead>
<tr>
<th>Serie</th>
<th>Entrenamiento</th>
<th>N. Instr.</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100.00%</td>
<td>48</td>
<td>93.34%</td>
</tr>
<tr>
<td>2</td>
<td>100.00%</td>
<td>24</td>
<td>100.00%</td>
</tr>
<tr>
<td>3</td>
<td>100.00%</td>
<td>29</td>
<td>100.00%</td>
</tr>
<tr>
<td>4</td>
<td>100.00%</td>
<td>43</td>
<td>93.34%</td>
</tr>
<tr>
<td>5</td>
<td>100.00%</td>
<td>42</td>
<td>100.00%</td>
</tr>
<tr>
<td>6</td>
<td>100.00%</td>
<td>23</td>
<td>100.00%</td>
</tr>
<tr>
<td>7</td>
<td>100.00%</td>
<td>29</td>
<td>93.34%</td>
</tr>
<tr>
<td>8</td>
<td>100.00%</td>
<td>34</td>
<td>100.00%</td>
</tr>
<tr>
<td>9</td>
<td>99.26%</td>
<td>54</td>
<td>93.34%</td>
</tr>
<tr>
<td>10</td>
<td>99.26%</td>
<td>30</td>
<td>93.34%</td>
</tr>
<tr>
<td>Global</td>
<td>99.85%</td>
<td></td>
<td>96.67%</td>
</tr>
</tbody>
</table>

Los resultados obtenidos con varios algoritmos clásicos (C4.5 [15], CN2 [16] y LVQ [17]) pueden apreciarse en la siguiente tabla.

<table>
<thead>
<tr>
<th>Algoritmo</th>
<th>Entrenamiento</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>C4.5</td>
<td>98.38%</td>
<td>92.77%</td>
</tr>
<tr>
<td>CN2</td>
<td>98.92%</td>
<td>94.16%</td>
</tr>
<tr>
<td>LVQ</td>
<td>98.55%</td>
<td>95.72%</td>
</tr>
</tbody>
</table>

Observando los resultados podemos notar que nuestro sistema posee una gran capacidad de generalización y un gran porcentaje de aciertos en Test, al mismo tiempo. Los programas generados poseen una media de 36 instrucciones, lo que da una idea de la sencillez de las soluciones encontradas.

4.2 Pima

Esta base de datos contiene 768 casos de diagnóstico de diabetes donde se toman 8 variables para diferenciar los dos posibles estados: enfermo o sano.

A continuación se muestran los resultados globales para todas y cada una de las series así como los porcentajes de clasificación de otros sistemas.

<table>
<thead>
<tr>
<th>Serie</th>
<th>N. Instr.</th>
<th>Entrenamiento</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>62</td>
<td>80.26%</td>
<td>76.65%</td>
</tr>
<tr>
<td>2</td>
<td>82</td>
<td>81.08%</td>
<td>77.08%</td>
</tr>
<tr>
<td>3</td>
<td>34</td>
<td>79.65%</td>
<td>75.13%</td>
</tr>
<tr>
<td>4</td>
<td>36</td>
<td>79.65%</td>
<td>79.29%</td>
</tr>
<tr>
<td>5</td>
<td>68</td>
<td>79.90%</td>
<td>74.35%</td>
</tr>
<tr>
<td>Global</td>
<td>57</td>
<td>80.10%</td>
<td>76.50%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Algoritmo</th>
<th>Entrenamiento</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>C4.5</td>
<td>96.06%</td>
<td>71.4%</td>
</tr>
<tr>
<td>CN2</td>
<td>85.4%</td>
<td>74.5%</td>
</tr>
<tr>
<td>LVQ</td>
<td>83.68%</td>
<td>67.71%</td>
</tr>
</tbody>
</table>

Es interesante reseñar que el clasificador obtenido para la serie 4 únicamente posee 36 instrucciones y depende de cuatro de las 8 variables de entrada. En otro artículo [11], como ya hemos mencionado, mostrábamos la capacidad de un método de similares características (basado en GA-P tradicional y no en registros) para actuar como selector de características en problemas de alta dimensionalidad.

5 Aplicación práctica: Clasificación de defectos de fabricación en planchas de vidrio

Este es el problema original que nos impulsó a desarrollar el método de clasificación. En este caso vamos a abordar el entrenamiento de las funciones de pertenencia correspondientes a un juego de etiquetas lingüísticas que forman parte de la base de conocimiento. Contamos con un conjunto de 360 ejemplos, cada uno de los cuales posee 5 atributos tomados de medidas realizadas sobre las imágenes de los defectos. Cada tipo de defecto viene caracterizado por una regla definida a priori por uno de los operarios que anteriormente trabajaba en la identificación visual de los mismos. No entraremos a detallar la base del funcionamiento del sistema automático de localización y extracción de características de cada defecto. La descripción completa del problema está en [8].

Para el entrenamiento del clasificador hemos codificado cada una de las 5 funciones de pertenencia a cada etiqueta conjuntamente en cada cromosoma de la población. Cada programa posee una sola variable de entrada (la correspondiente a su etiqueta) y cuatro constantes. Se ha impuesto un número máximo de instrucciones reducido (20) con el fin de forzar al sistema la búsqueda de funciones sencillas. Los parámetros que definen el algoritmo de entrenamiento son:
Figura 2: Relaciones entre los descriptores “forma de anillo” y “contraste”

<table>
<thead>
<tr>
<th></th>
<th>300 individuos</th>
</tr>
</thead>
<tbody>
<tr>
<td>Población</td>
<td></td>
</tr>
<tr>
<td>N. Subpoblaciones</td>
<td>3</td>
</tr>
<tr>
<td>N. funciones de pertenencia</td>
<td>5</td>
</tr>
<tr>
<td>Long. programa por cada f. de pert.</td>
<td>20 instrucciones</td>
</tr>
<tr>
<td>N. constantes por cada f. de pert.</td>
<td>4</td>
</tr>
<tr>
<td>N. registros</td>
<td>16</td>
</tr>
<tr>
<td>Prob. cruce</td>
<td>80%</td>
</tr>
<tr>
<td>Prob. mutación</td>
<td>20%</td>
</tr>
<tr>
<td>Prob. intercambio subpoblaciones</td>
<td>0.1%</td>
</tr>
<tr>
<td>Generaciones</td>
<td>350</td>
</tr>
</tbody>
</table>

En la tabla que sigue, se comparan los resultados obtenidos mediante este método, un clasificador estadístico y un clasificador borroso diseñado mediante el método Pittsburg pero con funciones de pertenencia paramétricas trapezoidales.

<table>
<thead>
<tr>
<th>Algoritmo</th>
<th>Entr.</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>K-Vecinos (K=3)</td>
<td>87.40%</td>
<td></td>
</tr>
<tr>
<td>Parametr. trapezoidal</td>
<td>94.2%</td>
<td>92.2%</td>
</tr>
<tr>
<td>Codificación funcional</td>
<td>97.38%</td>
<td>98.83%</td>
</tr>
</tbody>
</table>

También se ha utilizado el método aproximativo para resolver el mismo problema. Se aprecia una nueva mejora en los resultados debido a que el sistema es capaz de relacionar aspectos de los descriptores que el operador habría pasado por alto o no considerando relaciones demasiado complejas entre las variables.

En la gráfica 2 se relacionan los descriptores “Forma de anillo” y “Contraste”. El clasificador logró encontrar una función que haciendo únicamente uso de estos dos parámetros discriminaba en un 99.3% todos los ejemplos del entrenamiento. Nótese que haciendo uso de un clasificador descriptivo no es posible llegar a discriminar del mismo modo.

6 Conclusiones

En este trabajo se ha definido un método que permite inducir bancos de reglas borrosas aproximativos y descriptivos en problemas de clasificación. Los conjuntos borrosos que definen la semántica de los bancos de reglas se han obtenido mediante programación genética y no se basan en una parametrización de las funciones de pertenencia de los mismos, por lo que pueden definirse pertenencias de forma compleja que reducen el número de reglas necesario para describir el clasificador. Los resultados experimentales muestran que el método es capaz de competir con ventaja con otros clasificadores clásicos y borrosos.

Agradecimientos

Agradecemos a O. Cordón, M. José del Jesús y a F. Herrera la colaboración prestada así como sus valiosos comentarios acerca de este trabajo.

A Apéndice

A.1 Resultados obtenidos en la aplicación práctica

A continuación se incluye como ejemplo la decodificación de uno de los términos del individuo ganador en el problema de la clasificación de defectos de fabricación en placas de vidrio, así como la representación gráfica de la función de pertenencia correspondiente (figura 3).

[Funció 5 (término: Forma de Anillo)]

K00 = 63.498774; K02 = 10.411939; K01 = 30.524333; K03 = -88.616320;

319
Referencias


SELECCIÓN DE VARIABLES CON TÉCNICAS DIFUSO-EVOLUTIVAS EN MINERÍA DE DATOS

Antonio F. Gómez-Skarmeta Fernando Jiménez Jesús Ibáñez
Departamento de Informática y Sistemas,
Facultad de Informática, Universidad de Murcia,
30071-Espinardo, Murcia
skarmeta@dif.um.es, fernan@dif.um.es, jesus@dif.um.es

Resumen

Se pueden encontrar muchas similitudes entre los procesos de Minería de Datos y de Modelado Difuso, especialmente cuando tenemos que tratar con grandes conjuntos de datos con ruido e imprecisión. Una de ellas corresponde al preprocesamiento de los datos para seleccionar las variables más adecuadas a usar en el proceso de descubrimiento de conocimiento por medio de técnicas de la minería de datos o del modelado difuso. En este documento se investiga un sistema de aprendizaje híbrido, que, como un paso de preprocesamiento, combina técnicas difuso-evolutivas para encontrar características o variables útiles para representar los datos.

Palabras Clave: algoritmos evolutivos, preprocesamiento de datos, sistemas híbridos.

1 INTRODUCCIÓN

La propuesta de Zadeh de modelar el mecanismo del pensamiento humano con valores lingüísticos difusos en vez de con números, condujo a la introducción de la borrosidad dentro de la teoría de sistemas y al desarrollo de una nueva clase de sistemas llamados sistemas difusos. En el modelado difuso, el problema más importante es la identificación de un modelo difuso usando datos del tipo entrada-salida. De un modo similar, la Minería de Datos tiene como objetivo la criba de datos para revelar información útil por medio de una representación adecuada al usuario, comprimiendo enormes registros de datos. En este contexto, un problema común a ambas técnicas es cómo tratar con mecanismos de selección de variables a partir de la posible colección de variables que pueden ser consideradas de los datos disponibles.

Aunque hemos usado el término Minería de Datos, usamos, como en [6], el término Descubrimiento de Conocimiento en Bases de Datos (DCBD) o sólo Descubrimiento de Conocimiento (DC) para denotar el proceso completo de extraer conocimiento de alto nivel a partir de datos de bajo nivel, y el término Minería de Datos como el acto concreto de extraer patrones o modelos de los datos. Por tanto, muchos pasos preceden al de minería de datos, y uno de ellos es el preprocesamiento de los datos para seleccionar las variables adecuadas para ser usadas en la construcción o extracción del modelo. Este paso es similar a la identificación de variables dentro del paso de identificación de la estructura, en el proceso de modelado difuso.

Dado un conjunto de datos para el que presumimos alguna dependencia funcional, surge la cuestión de si hay alguna metodología adecuada para derivar reglas (difusas) a partir de los datos que caracterizan la función desconocida, de forma tan precisa como sea posible. Recientemente se han propuesto numerosas aproximaciones para generar automáticamente reglas if-then a partir de datos numéricos sin expertos en el dominio [9]. Este tipo de problema es similar al que podemos encontrar en el área de Descubrimiento de Conocimiento, y en este sentido presentamos aquí una técnica Soft Computing que intenta tener en cuenta uno de los problemas fundamentales como es el preprocesamiento de los datos para seleccionar las características o variables adecuadas para que el proceso de Minería de Datos pueda realizarse con más información.

A medida que intentamos resolver problemas del mundo real, nos damos cuenta de que son normalmente sistemas mal definidos, difíciles de modelar y con espacios de solución de gran escala. En estos casos, los modelos precisos son poco prácticos, demasiado caros o inexistentes. La información relevante disponible está normalmente en la forma de conocimiento empírico previo y datos del tipo entrada-salida representando instancias del comportamiento...
del sistema. Así pues, necesitamos sistemas de razonamiento aproximado capaces de manejar esa información imperfecta. Soft Computing es un término acuñado recientemente que describe el uso simbiótrico de muchas disciplinas emergentes de computación que intentan manejar la información imperfecta. De acuerdo con Zadeh (1994): "... en contraste con lo tradicional, hard computing, soft computing es tolerante a la imprecision, a la incertidumbre y a la verdad parcial". Dentro de estas técnicas tenemos la Lógica Difusa, el Razonamiento Probabilístico, las Redes Neuronales y los Algoritmos Evolutivos. Durante los últimos años hemos visto un número creciente de algoritmos híbridos, en los que dos o más tecnologías Soft Computing se han integrado para mejorar el rendimiento global del algoritmo [2].

Como sucede en la minería de datos, si intentamos revelar información útil de los datos, un paso fundamental es el preprocesamiento de los datos para seleccionar las variables más adecuadas. En el contexto del modelado y más concretamente del modelado difuso, el objetivo es encontrar un conjunto de relaciones que describan el comportamiento presente en los datos por medio de una colección de patrones o reglas if-then [11]. En este proceso, la identificación de la estructura de un sistema tiene que encontrar las variables que representan los datos de un modo más preciso, de entre una colección de posibles variables. En este contexto tenemos que seleccionar un número finito de variables entre una colección finita de posibles candidatos. Ésto es un problema combinatorio. Una posible aproximación a este problema es intentar asignar cierto grado a cada variable dependiendo de su importancia en la consecución del objetivo final. Ésto es similar a una fusión multisensor, que consiste en una combinación de diferentes fuentes de información en un formato de representación. Cuando cada variable representa características diferentes, tratamos con información complementaria. Su fusión nos permite resolver ambigüedad en la información [3].

En este proceso de fusión de las diferentes variables de entrada posibles, tenemos dos objetivos. Primero queremos determinar la importancia de las entradas a fusionar; segundo queremos determinar los parámetros exactos de las funciones de agregación usadas en la fuunión. Encontrar los mejores parámetros de la función de agregación es un proceso de optimización. Los algoritmos evolutivos han demostrado ser muy útiles para este proceso porque aportan una búsqueda robusta en espacios de búsqueda complejos y no quedan atrapados en mínimos locales como les sucede a las técnicas de gradiente descendente.

Una vez que las variables son identificadas, el siguiente paso en la identificación de la estructura tiene que ver con la identificación de relaciones o, en otras palabras, con el descubrimiento de los patrones o el modelo a partir de los datos, y su representación por medio de diferentes alternativas como por ejemplo reglas difusas. En este paso se pueden usar diferentes técnicas soft computing en el contexto del DC, como las propuestas en [3][8].

2 UN ALGORITMO EVOLUTIVO PARA IDENTIFICACIÓN DE VARIABLES

Los Algoritmos Evolutivos (AE) [1] son procedimientos adaptativos de optimización y búsqueda que encuentran soluciones a problemas, inspirados en los mecanismos de la evolución natural. Imitan, a un nivel abstracto, en una aproximación basada en poblaciones, principios biológicos tales como la herencia de información, la variación de información mediante cruce/mutación, y la selección de individuos en base a la aptitud. Los AE comienzan con un conjunto inicial (población) de soluciones alternativas (individuos) para el problema dado, que son evaluadas en términos de la calidad de la solución (aptitud). Entonces, se aplican los operadores de selección, replicación y variación para obtener nuevos individuos (descendencia) que constituyen una nueva población. La interacción de selección, replicación y variación de los más aptos lleva a soluciones cuya calidad va en aumento a lo largo de muchas iteraciones. Cuando finalmente se alcanza un criterio de terminación, tal como un número máximo de generaciones, el proceso de búsqueda termina, y la solución final se muestra como salida.

Básicamente, podemos decir que un AE se basa en los siguientes componentes, para cualquier tipo de aplicación: una representación de soluciones del problema, un modo de crear una población inicial de soluciones; una función de evaluación para medir la aptitud de cualquier solución, que juega además el papel de "entorno", en el que las mejores soluciones (es decir, aquellas que tienen un mejor valor objetivo) pueden tener mayor probabilidad de supervivencia; operadores de variación (reglas de transición probabilística) que realizan la composición de los hijos durante la reproducción; y valores para los parámetros que el algoritmo utiliza para guiar su evolución: tamaño de la población, número de generaciones, probabilidades de cruce y mutación, etc.

La clase de AE mejor conocida es la de los Algoritmos Genéticos (AG) [7], que han recibido mucha atención en los últimos años. Otras variantes de AE, tales como la Programación Genética, Estrategias de Evolución, o Programación Evolutiva son menos populares, aunque muy poderosas también. Los AG clásicos usan cadenas
binarias de longitud fija para representar individuos y dos operadores genéticos básicos (mutación binaria y cruce binario). No obstante, el campo de los AG se caracteriza por un gran dinamismo, y continuamente se desarrollan modificaciones y extensiones de la tecnología.

En este documento describimos un AE para identificación de variables en Minería de Datos y DC. Expresado en términos de programación matemática, el problema puede ser formulado como sigue:

\[
\begin{align*}
\min \quad & E = \sqrt{\frac{1}{n} \sum_{i=1}^{n} \left( y_i - \left( \sum_{j=1}^{p} w_j x_j^i \right)^{1/f} \right)^2} \\
\text{s. a.:} \quad & \sum_{j=1}^{p} w_j = 1, \quad 0 \leq w_j \leq 1, \quad j = 1, \ldots, p
\end{align*}
\]

(1)

donde \( n \) es el número de datos, \( p \) es el número de variables, \( y_i \) es el valor esperado de salida para el vector de variables de entrada \( X_i = \{x_1^i, \ldots, x_p^i\} \), y 1/f es el valor del modelo media generalizada (Dyckhoff and Pedrycz [5]) con grado de borrosidad \( f \). Este tipo de operador de agregación es similar al operador OWA (Ordered Weighted Averaging) introducido por Yager [13]. Los pesos \( w_j, j = 1, \ldots, p, \) han de ser averiguados.

Aunque pueden usarse numerosas conectivas de conjuntos difusos con la finalidad de agregación [14], el operador media generalizada satisface las propiedades deseables. Por lo tanto, el operador puede usarse como unión o como intersección en los casos extremos, y el ratio de compensación puede controlarse variando el grado de borrosidad \( f \).

Buczak y Uhrig [3] describen un AG para resolver el mismo problema con aplicación a la fusión de datos, siendo la representación de soluciones al problema y el mecanismo de satisfacción de restricciones las principales diferencias con respecto al AE que aquí describimos.

Más adelante se describen las principales características del AE propuesto. Estas características son una representación de soluciones al problema, satisfacción de restricciones, mecanismos para crear una población inicial de soluciones, la función de evaluación, operadores de variación y parámetros usados. Además, se hacen algunos comentarios al diseño.

Para diseñar el AE hemos tenido en cuenta la siguiente idea: parece (Michalevich [10]) que una representación natural de una solución potencial para un problema dado junto con una familia de operadores de variación aplicables podría ser bastante útil en la aproximación de soluciones a muchos problemas. Además, las restricciones no triviales pueden implementarse incorporando conocimiento del problema específico en el AE.

2.1 REPRESENTACIÓN

Un individuo \( W \) de la población se representa como una colección de \( p \) componentes, es decir, \( W = \{w_1, \ldots, w_p\} \), donde \( w_j \in [0,1], j = 1, \ldots, p \), es el peso asociado con la variable de entrada \( x_j \).

2.2 SATISFACCIÓN DE RESTRICCIONES

Todos los mecanismos para crear un nuevo individuo \( W = \{w_1, \ldots, w_p\} \) en el proceso evolutivo, es decir los procedimientos de inicialización y los operadores de variación, aseguran que se satisfacen las restricciones

\[
\sum_{j=1}^{p} w_j = 1, \quad 0 \leq w_j \leq 1, \quad j = 1, \ldots, p.
\]

2.3 POBLACIÓN INICIAL

El siguiente procedimiento \textit{población inicial} obtiene una población completa de \textit{tampob} individuos que satisfacen las restricciones impuestas. Consideramos dos procedimientos de inicialización. El procedimiento \textit{pesos1} genera una colección de pesos \( W = \{w_1, \ldots, w_q\} \) tales que \( w_l \in [0,1], l = 1, \ldots, q, \) y \( \sum_{l=1}^{q} w_l = val \), con 0 \leq val \leq 1. El procedimiento \textit{pesos2} es una modificación del procedimiento \textit{pesos1} para fijar un ratio aleatorio de pesos iguales a cero. Nótese que para \( q = p \) y \( val = 1 \), ambos procedimientos \textit{pesos1} y \textit{pesos2} generan una solución factible para el problema. Estos procedimientos se usan con igual probabilidad para obtener la población inicial.

\textbf{procedimiento población inicial;}

\textbf{entrada:} tamaño población \textit{tampob}; número de variables \textit{p};

\textbf{salida:} población \textit{POB} = \{\textit{W}_1 = \{w_1, \ldots, w_p\}, \ldots, \textit{W}_{\textit{tampob}} = \{w_1, \ldots, w_p\}\}

de \textit{tampob} individuos tales que \textit{w}_j \in [0,1], j = 1, \ldots, p, \textit{w}_1 = 1, \ldots, \textit{tampob};

1. \textit{i} \leftarrow 1;
2. \( \text{aleat} \leftarrow \text{valor real aleatorio} \in [0, 1]; \)

3. Si \( \text{aleat} \leq 0.5 \) entonces \( \text{pesos1(entrada : } p, 1; \text{salida : } W) \)
   sino \( \text{pesos2(entrada : } p, 1; \text{salida : } W) \)

4. \( W_i \leftarrow W; \)

5. Si \( i < \text{tampob} \) entonces
   \( i \leftarrow i + 1; \)
   Ir al paso 2;

procedimiento pesos1;

entrada: número entero \( q \), con \( 1 \leq q \leq p; \) valor real \( \text{val} \), con \( 0 \leq \text{val} \leq 1; \)

salida: colección \( W = \{w_1, \ldots, w_q\} \) tal que \( w_l \in [0, 1], l = 1, \ldots, q, \) y \( \sum_{l=1}^{q} w_l = \text{val}; \)

1. \( w_l \leftarrow \text{valor real aleatorio} \in [0, 1], \) para \( l = 1, \ldots, q; \)

2. \( w_l \leftarrow \text{val} \cdot w_l / \sum_{i=1}^{q} w_i, \) para \( l = 1, \ldots, q; \)

procedimiento pesos2;

entrada: número entero \( q \), con \( 1 \leq q \leq p; \) valor real \( \text{val} \), con \( 0 \leq \text{val} \leq 1; \)

salida: colección \( W = \{w_1, \ldots, w_q\} \) tal que \( w_l \in [0, 1], l = 1, \ldots, q, \) y \( \sum_{l=1}^{q} w_l = \text{val}; \)

1. Seleccionar aleatoriamente \( K = \{k_1, \ldots, k_r\} \subseteq \{1, \ldots, q\} \) tal que \( 1 \leq r \leq q; \)

2. \( M \leftarrow \{1, \ldots, q\} - K; \)

3. \( w_l \leftarrow 0, l \in M; \)

4. \( \text{pesos1(entrada : } r, \text{salida : } V); \)

5. \( w_{k_l} \leftarrow v_l, l = 1, \ldots, r; \)

2.4 Función de evaluación

La función de evaluación de los individuos \( W_i = \{w_{1i}, \ldots, w_{pi}\}, i = 1, \ldots, \text{tampob}, \) es claramente determinada por la función objetivo del problema:

\[
\begin{align*}
\text{eval}(W_i) &= \sqrt{\frac{\sum_{i=1}^{n} \left( y_l - \left( \frac{\sum_{j=1}^{p} w_j x_{lj}}{n} \right)^{1/2} \right)^2}{\sum_{i=1}^{n} \left( \frac{\sum_{j=1}^{p} w_j x_{lj}}{n} \right)^{1/2}}} \\
&= \left( \sum_{i=1}^{n} \left( y_l - \left( \frac{\sum_{j=1}^{p} w_j x_{lj}}{n} \right)^{1/2} \right)^2 \right)^{1/2}
\end{align*}
\]

2.5 Mecanismo de selección y reemplazo generacional

Proponemos usar la selección por torneo [7]. En este método, se prueban un grupo de \( n_{torneo} \) individuos de la población y se elige para la reproducción al individuo con mejor aptitud. Se aplican operadores de variación a los individuos seleccionados y la descendencia se copia a la siguiente población. Este proceso se repite hasta que se genera completamente la nueva población (reemplazo generacional completo). Además se usa la estrategia elitista, que siempre siempre copia los mejores miembros de una población a la siguiente.

2.6 Operadores de variación

Hemos considerado cinco operadores de variación (tres mutaciones, cruces e inversión). El operador mutación1 realiza un cambio mínimo, intercambiando dos elementos aleatorios de los padres, mientras que los operadores mutación2 y mutación3 usan los procedimientos pesos1 y pesos2 respectivamente para producir un cambio en una parte arbitraria de los padres. El operador cruce aritmético produce dos descendientes mediante la combinación lineal convexa de los padres. Con el operador inversión, un individuo \( W = \{w_1, \ldots, w_p\} \) es invertido produciendo otro \( W' = \{w_p, \ldots, w_1\} \).

procedimiento mutación1;

entrada: Padre \( W = \{w_1, \ldots, w_p\} \);

salida: Descendencia \( W' = \{w_1', \ldots, w_p'\} \);

1. Seleccionar aleatoriamente \( K = \{k_1, k_2\} \subseteq \{1, \ldots, p\} \);

2. \( M \leftarrow \{1, \ldots, p\} - K; \)

3. \( w_{k_1} \leftarrow 0, l \in M; \)

4. \( \text{pesos1(entrada : } r, \text{salida : } V); \)

5. \( w_{k_1} \leftarrow v_l, l = 1, \ldots, r; \)

procedimiento mutación2;

entrada: Padre \( W = \{w_1, \ldots, w_p\} \);

salida: Descendencia \( W' = \{w_1', \ldots, w_p'\} \);
1. Seleccionar aleatoriamente \( K = \{k_1, \ldots, k_r\} \subseteq \{1, \ldots, p\} \) tal que \( 1 < r \leq p \);
2. \( M \leftarrow \{1, \ldots, p\} - K \);
3. \( w'_i \leftarrow w_i, i \in M \);
4. \( \text{val} \leftarrow \sum_{k=1}^{r} w_{k_i} \);
5. \( \text{pesos1}(\text{entrada} : r, \text{val}; \text{salida} : V) \);
6. \( w'_{k_i} \leftarrow v_i, l = 1, \ldots, r \);

procedimiento mutación3;

entrada: Padre \( W = \{w_1, \ldots, w_p\} \);
salida: Descendencia \( W' = \{w'_{1}, \ldots, w'_p\} \);

1. Seleccionar aleatoriamente \( K = \{k_1, \ldots, k_r\} \subseteq \{1, \ldots, p\} \) tal que \( 1 < r \leq p \);
2. Set \( M = \{1, \ldots, p\} - K \);
3. \( w'_i \leftarrow w_i, i \in M \);
4. \( \text{val} \leftarrow \sum_{k=1}^{r} w_{k_i} \);
5. \( \text{pesos2}(\text{entrada} : r, \text{val}; \text{salida} : V) \);
6. \( w'_{k_i} \leftarrow v_i, l = 1, \ldots, r \);

procedimiento cruce.aritmético;

entrada: Padres \( W_1 = \{w'_1, \ldots, w'_p\} \) y \( W_2 = \{w^2_1, \ldots, w^2_p\} \);

salida: Descendencia \( W'_1 = \{w^{1}_1, \ldots, w^{1}_p\} \) y \( W'_2 = \{w^{2}_1, \ldots, w^{2}_p\} \);

1. \( c_1 \leftarrow \text{valor real aleatorio} \in [0, 1] \);
   \( c_2 \leftarrow 1 - c_1 \);
2. \( w^j_1 \leftarrow c_1 \cdot w^j_1 + c_2 \cdot w^2_j, j = 1, \ldots, p \);
   \( w^j_2 \leftarrow c_2 \cdot w^j_1 + c_1 \cdot w^2_j, j = 1, \ldots, p \);

procedimiento inversión;

entrada: Padre \( W = \{w_1, \ldots, w_p\} \);

salida: Descendencia \( W' = \{w'_1, \ldots, w'_p\} \);

1. \( w'_{p-i+1} \leftarrow w_i, l = 1, \ldots, p \);

2.7 PARÁMETROS

Los parámetros a considerar por parte del AE son el número máximo de generaciones (mazgen), el tamaño de la población (tampob), el número de miembros usados en la selección por torneo (n_torneo), la probabilidad de cruce (p_c), la probabilidad de mutación (p_m), y la probabilidad de inversión (p_i). Nótese que la probabilidad p_m corresponde a la suma de las probabilidades de todos los operadores de mutación descritos. La tabla 1 muestra los valores de los parámetros para los cuales se han obtenido buenos resultados en el proceso de experimentación.

<table>
<thead>
<tr>
<th>PARAMETRO</th>
<th>VALOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>mazgen</td>
<td>1000</td>
</tr>
<tr>
<td>tampob</td>
<td>20</td>
</tr>
<tr>
<td>n_torneo</td>
<td>6</td>
</tr>
<tr>
<td>p_c</td>
<td>0.6</td>
</tr>
<tr>
<td>p_m</td>
<td>0.6</td>
</tr>
<tr>
<td>p_i</td>
<td>0.1</td>
</tr>
</tbody>
</table>

3 EJEMPLOS NUMÉRICOS

Para validar el AE hemos considerado un test estándar propuesto en Dyckhoff y Pedrycz (1984) y utilizado también en Buczak y Uhrig (1996). El conjunto de datos consiste en una colección de 2 variables de entrada \( z_1 \) y \( z_2 \) y una variable de salida. Para comprobar la bondad del AE, hemos añadido a los datos de entrada 6 nuevas variables \( z_3, z_4, z_5, z_6, z_7 \) y \( z_8 \) formadas a partir de las variables \( z_1 \) y \( z_2 \) añadiéndoles ruido uniforme en un 10% (a las variables \( z_3 \) y \( z_4 \)), en un 40% (a las variables \( z_5 \) y \( z_6 \)), y en un 100% (a las variables \( z_7 \) y \( z_8 \)). La tabla 2 muestra los pesos y fitness obtenidos considerando solamente las 2 variables de entrada del problema, para 3 valores diferentes de \( f \) (con el objetivo de poder comparar con los ejemplos de la literatura). La tabla 3 muestra los pesos y fitness obtenidos teniendo en cuenta las 2 variables de entrada originales más las 6 variables ficticias de ruido. Los resultados muestran como el AE propuesto detecta las variables que realmente influyen en la variable de salida, excluyendo aquellas ficticias introducidas artificialmente.

4 CONCLUSIONES

Las técnicas Soft Computing para Análisis de Datos como selección o identificación de variables y descubrimiento de modelo, son una poderosa alternativa a las técnicas clásicas como las usadas hasta hoy en
la comunidad de DC. En este documento intentamos mostrar cómo pueden ser usadas estas técnicas en el preprocesamiento de datos, aunque son relevantes a los diversos pasos del proceso de DC, como la minería de datos o la selección de datos [12].

aplicarse con éxito en el contexto del DC y la Minería de Datos, dado que tienen dos características que los hacen muy adecuados:

- la primera es la imprecisión subyacente de los modelos o patrones que queremos encontrar, teniendo en cuenta que hemos asumido tratar con datos incompletos y con incertidumbre.

- la segunda es el poder computacional que aportan los algoritmos difusos y evolutivos, que es especialmente importante en el contexto de grandes bases de datos.

Los resultados obtenidos hasta ahora con la aplicación de este algoritmo, aunque no definitivos, muestran un buen rendimiento y su uso puede ser de gran interés para los encargados de la toma de decisiones en un entorno de aprendizaje supervisado.

**Agradecimientos**

Los autores agradecen a la Comisión Interministerial de Ciencia y Tecnología (CICYT) por el apoyo dado a la realización de este trabajo a través del proyecto con referencia TIC97-1343-C02-02, así como al Instituto de Fomento de la Región de Murcia por su financiación a través del Programa Séneca.

**Referencias**


Circuito Experimental de Inferencia de Lukasiewicz

Luis de Salvador, Pedro E. Bernad, Julio Gutiérrez
Lab. de Hw. y Control
Instituto Nacional de Técnica Aeroespacial
Ctra. Ajalvir km. 4 - Torrejón de Ardoz
28850 - Madrid - España

Resumen

El operador de Lukasiewicz es uno de los más interesantes para implementar en un procesador borroso de propósito general. Su diseño hardware ofrece ciertas dificultades cuando se quiere obtener un circuito de elevado rendimiento y reducidas dimensiones. En esta comunicación se ofrece una solución basada en arquitecturas dígito-serie y notación signo-dígito.

Palabras Clave: Hardware, fuzzy, FPGA, Lukasiewicz, signo-dígito, dígito-serie.

1 INTRODUCCIÓN

Esta trabajo se enmarca dentro de un proyecto de la CICYT que tiene por objeto el desarrollo de un procesador de lógica borrosa lo más general posible. Para que un procesador borroso pueda ser de propósito general es necesario contemplar las operaciones Max-Min, sino también otras que sean de uso común en la inferencia sobre reglas borrosas [4] [6] [9].

En este marco de trabajo, se ha desarrollado un circuito para evaluar la inferencia de Lukasiewicz de forma eficiente. La implementación de este prototipo tiene como propósito investigar la adecuación de este tipo de inferencia para un sistema de procesamiento de elevado rendimiento de información borrosa. Por lo tanto, la arquitectura diseñada está orientada a conseguir un procesamiento muy rápido de la información.

Adicionalmente, se persigue construir un circuito de pequeñas dimensiones y flexible en alguno de sus parámetros, adecuado para su implementación en FPGA. La flexibilidad de este circuito se va a concretar en no fijar a priori el número de dígitos empleados en la codificación de los valores de pertenencia.

1 OPERADOR A IMPLEMENTAR

El algoritmo de Lukasiewicz para la realización de inferencias borrosas se basa en la aplicación de los operadores del mismo nombre:

La función T o t-norma:

\[ w(x, y) = \max(0, x + y - 1) \]  

La función S o t-conorma:

\[ w^*(x, y) = \min(1, x + y) \]

El par (T, S) de Lukasiewicz es un conjunto de operaciones dual:

\[ s(x, y) = 1 - T(1 - x, 1 - y) \]

(cumpliendo las leyes de Morgan)

1.1 EVALUACIÓN DE LA INFERENCIA

El sistema de reglas se considera formado por reglas del tipo MISO:

\[ r_j: \text{Si } x_1 \in B_{j1} \land x_2 \in B_{j2} \land ... \land x_n \in B_{jn} \quad \text{ENTonces } y \in D_j \]

Cada regla se evalúa para producir un conjunto borroso, cuya función de pertenencia es:

\[ D_j(y) = r(B_{j1}(x_1), B_{j2}(x_2), ..., B_{jn}(x_n)) \]

Cada uno de los valores \( B_{j}(x_i) \) se representa por un valor de verdad sobre un conjunto borroso, no por un conjunto borroso.

El operador de agregación que combina todas las salidas es el operador de unión, implementado con la t-norma:

\[ D'(y) = s(D_j(y)); \text{ para } j = (1,m) \]

1.2 APLICACIÓN DE LA T-NORMA DE LUKASIEWICZ PARA UN CONJUNTO DE VALORES DE PERTENENCIA

Para la aplicación de la t-norma sobre un conjunto de valores de pertenencia es necesario realizar de forma iterativa la operación \( \max(0, x+y-1) \) sobre un conjunto de valores \( \{a, b, c, ..., n\} \):

\[ \max(0, \max(\max(... \max(0, m+n-1), ...), b-1) + a-1) \]

La ejecución de la t-norma de L. implica la realización de tres operaciones:

- Sumar \( x \) y \( y \)
- Restar 1
- Comparar con 0

Para simplificar el proceso, se hace uso del principio de dualidad (3) de la t-norma y de la t-conorma de L. Para ello, en vez de computar la t-norma, se evalúa la t-conorma con los valores complementados. Esto significa utilizar valores complementados en la evaluación del operador: \( \min(1, 1-x+1-y) \).

\[ \min(0, \min(\min(... \min(0, 1-m+1-n), ...), 1-b) + 1-a) \]

La ejecución de esta operación implica sólo dos pasos:

- Sumar \( x \) y \( y \)
- Comparar con 1.
2 MODELO DE PROCESO

El objetivo del circuito es realizar la evaluación de la inferencia de la forma más rápida posible y utilizando el mínimo número de puertas. Una característica adicional es que el mismo circuito se pueda emplear para valores de pertenencia codificados con un número arbitrario de dígitos.

Para ello se va a plantear el desarrollo de un sistema segmentado o pipeline que realice el proceso empleando técnicas dígito-serie [2], [3], [7], [8], [10]. La unidad elemental que se emplea como modulo de proceso bit-slice de evaluación del operador \( w^* \) es la siguiente: (Figura 1).

Figura 1 Ejemplo de operador dígito a dígito

Donde A y B son palabras que codifican el valor de verdad complementado y R es el resultado de la t-norma complementada. Los dígitos de cada palabra se evalúan de forma sucesiva. Para la evaluación sobre un conjunto de palabras la estructura de proceso es la siguiente:

Figura 2 Utilización de operadores sobre dígito para procesar distintas palabras

Para ejecutar el operador \( w^* \) hay que ejecutar dos operaciones: suma y comparación. Pero hay que tener en cuenta que la comparación no sólo significa obtener un resultado de comparación, sino de selección de un valor u otro en función de la comparación de ambos. Es una comparación de dos pasos:

- Evaluación de la comparación.
- Selección del valor adecuado en función de la comparación.

Para comparar dígito a dígito dos palabras e ir proporcionando el resultado final sin esperar a la comparación de todos los dígitos es necesario realizar la comparación comenzando por el más significativo. En cambio, la operación de suma se puede realizar ofreciendo los resultados finales según se va operando dígito a dígito sólo si se empieza a procesar desde el dígito menos significativo. Para poder compatibilizar ambas operaciones en un sistema de proceso dígito-serie es necesario emplear una notación alternativa al sistema binario: notación signo-dígito [1]. Esta notación permitirá realizar el proceso de comparación y suma comenzando por el dígito más significativo.

2.1 REPRESENTACIÓN Y PROCESO DE VALORES DE PERTENENCIA

En notación signo-dígito [1], si se tiene una base de codificación \( r \) y dos valores de pertenencia \((x,y)\) de \( n \) dígitos normalizados entre \([0, 2^r - 1]\), estos pueden representarse como \((x \cdot r^x + 1, y \cdot r^y + 1)\) para permanecer en los intervalos de codificación. La complementación se convierte en un cambio de signo (inversión del bit más significativo) de cada dígito del código (propiedad de la notación signo-dígito).

El proceso de suma de dos valores de pertenencia es el siguiente:

\[
x + y = x \cdot r^x + 1 + y \cdot r^y + 1 = x + y \cdot r^y + 2 \equiv x + y - r^y + 1 \quad (9)
\]

Hay que retocar el resultado final para que el valor obtenido sea realmente el que se busca, ya que por defecto el resultado se encuentra modificado por un factor de \((r^y + 1)\). Por lo tanto, es necesario realizar la normalización de la salida del sumador. La normalización se realiza empleando otro sumador serie con una de sus entradas constante y con valor \(-r \cdot l\) (para cada dígito de la palabra).

De esta forma el circuito que ejecuta el operador \( w^* \) se configura como un sumador en dos pasos o etapas, una etapa que realiza la suma en sí y otra etapa que realiza la normalización del resultado intermedio. Este sumador, tal y como aquí se plantea tiene los siguientes problemas:

- Después de cada suma, el resultado emplea más dígitos que los realmente necesarios para codificar la palabra. Exactamente un dígito más. Debido a que se emplean dos sumadores, el sistema intenta ocupar dos dígitos más.
- El operador incluye la comparación con el valor 1. Esta comparación se va a traducir en evitar el overflow del resultado cuando se realiza la operación de suma y mantener los valores saturados.
- Es necesario encadenar varios de estos operadores.

Los dos primeros problemas están relacionados: es necesario evitar la propagación por encima del número de dígitos necesarios para codificar la palabra y por encima del valor lógico 1, que es una sucesión de dígitos con el máximo valor.

Esto implica la necesidad de propagar hacia atrás el acarreo de los dígitos más significativos a los dígitos menos significativos cuando los primeros se encuentran saturados, en vez de añadir un nuevo dígito. Esta estrategia permite mantener el valor máximo de salida como el 1 lógico.

Evitar dicha propagación se puede conseguir en la segunda etapa del sumador (la de normalización) ya que la primera etapa precisa de un dígito adicional para codificar el resultado ya que este se encuentra en el intervalo \([-2r + 2, 2r - 2]\). La nueva configuración de los módulos de sumador serie es la siguiente:

328
En esta figura aparece un nuevo módulo etiquetado como A₃. Tiene las señales \{ t', w', s' \}, que tienen la misma definición que las señales no marcadas, y la señal c, que se emplea para propagar hacia atrás la saturación de los dígitos más significativos. Se define como sigue:
- **c = 2;** Si no hay saturación.
- **c = 0;** Si hay saturación pero no hay propagación hacia atrás.
- **c = 1;** Si hay saturación positiva y propagación hacia atrás de 1.
- **c = -1;** Si hay saturación negativa y propagación hacia atrás de -1.

### 2.2 MODELO DE PROCESO CON BASE 3


#### Módulo A₁

| \( t = 0 \) | \( -2 \leq x + y < 2 \) |
| \( t = 1 \) | \( x + y = 2 \) |
| \( t = -1 \) | \( x + y \leq -2 \) |
| \( w = x + y - 3 \cdot t \) |

#### Módulo A₂

| \( s = w + t \) |

#### Módulo A₃

If \( c = 2 \) -- sin saturación de los dígitos más significativos

- If \(-2 < x + 2 < 2 \Rightarrow 4 < x < 0 \Rightarrow -2 \leq x < 0 \)
  - \( t = 0 \)
  - \( c = 2 \)
  - \( w = x + 2 \)
- If \( x \geq 0 \)
  - \( t = 1 \)
  - \( c = 2 \)
  - \( w = x - 1 \)

If \( c = 0 \) -- saturación de los dígitos más significativos

- \( t = 0 \)
- If \(-4 < x < 0 \Rightarrow -2 \leq x < 0 \)
  - \( c = 2 \)
  - \( w = x + 2 \)
- Else If \( x = 0 \)
  - \( c = 0 \)
  - \( w = 2 \)
- Else

---

### 2.3 EL VALOR INICIAL

Las fórmulas anteriores no reflejan el problema del valor inicial. Debido a que hay propagación hacia adelante de los resultados intermedios, es necesario guardar un ciclo de reloj de distancia entre la entrada de dos palabras consecutivas. Hay que tener en cuenta que, la suma de los dos dígitos más significativos de la entrada \( A_n \) y \( B_n \) producen dos resultados. El primero es \( R_n \) y el segundo es la posible propagación adelantada \( R_{n+1} \).

Este significa un dígito más en el resultado, lo que se pretende evitar. Por lo tanto, este dígito adicional ha de volver a sumarse al penúltimo dígito: \( R_n + R_{n+1} \). Esta suma ha de realizarse únicamente en la operación del primer dígito. De ahí que en la Figura 3 se represente una alimentación de la señal \( t \) hacia la señal \( c \). De esta forma, cualquier propagación que se produzca del dígito más significativo hacia adelante se anula en la segunda etapa del L-operator.

### 3 DISEÑO DE LOS MÓDULOS

El circuito que implementa el L-operator tiene la siguiente forma:

Figura 4 Módulos del circuito L-operator

Todos los módulos tienen entradas adicionales de resetz (reset a nivel bajo) y de clk. Las diferencias entre el circuito que se muestra en la Figura 4 con el circuito teórico de la Figura 3 son las siguientes:

- En esta última figura los registros se consideran incluidos en los módulos de donde procede la señal en las que estaban incluidos.
- Se ha sustituido en último módulo denominado A2 por un nuevo módulo denominado A4. La diferencia
funcional entre ambos no existe. El módulo A4 se diferencia del A2 en que se realiza en el diseño una optimización diferente para los casos especiales que no se contemplan en el A2.
Todos los módulos han sido diseñados en VHDL. Los siguientes apartados muestran los circuitos sintetizados empleando Aurora de Viewlogic.

3.1.1 Módulo A1
A continuación se muestra el esquemático sintetizado del módulo A1.

3.1.2 Módulo A2
A continuación se muestra el esquemático sintetizado del módulo A2.

3.1.3 Módulo A3
El esquemático sintetizado del módulo A3 es el módulo más complejo ya que es el que realiza la normalización y evita la propagación del resultado en un dígito adicional previniendo el overflow.

3.1.4 Módulo A4
El módulo A4 ajusta el acarreo producido por la normalización.

3.2 PRUEBAS DE INTEGRACIÓN
Los módulos A1 al A4 se integran para formar una unidad de proceso que permite el proceso de cinco valores de inferencia codificados con un número de dígitos variable. El módulo que integra todos los circuitos necesarios para dicho proceso de inferencia sobre cinco variables de entrada se ha denominado GLMG. El plan de pruebas de integración del módulo GLMG se realiza introduciendo distintos juegos de 5 palabras (valores de inferencia). Las palabras están nombradas de la 'a' a la 'e'. En esta prueba tienen 6 dígitos más un dígito adicional de separación entre palabras introducidas. Este dígito ha de encontrarse al final de la palabra. Se introducen 3 juegos de test que son los siguientes:

<table>
<thead>
<tr>
<th>Tabla 1 Juegos de Test</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Juego 1</strong></td>
</tr>
<tr>
<td>a 0 -2 0 -2 0 -2 -2 -2</td>
</tr>
<tr>
<td>b 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>c -2 -2 -2 -2 -2 -2 -2</td>
</tr>
<tr>
<td>d 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>e -2 0 -2 0 0 -2 0 0 0</td>
</tr>
<tr>
<td><strong>Juego 2</strong></td>
</tr>
<tr>
<td>a -2 -2 -2 -2 -2 -2 -2</td>
</tr>
<tr>
<td>b -2 -2 -2 -2 -2 -2 -2</td>
</tr>
<tr>
<td>c -1 -1 -1 -1 -1 -1 -1</td>
</tr>
<tr>
<td>d -1 -1 -1 -1 -1 -1 -1</td>
</tr>
<tr>
<td>e -2 -2 -2 -2 -2 -2 -2</td>
</tr>
<tr>
<td><strong>Juego 3</strong></td>
</tr>
<tr>
<td>a 0 0 -2 -2 -2 -2 -2 -2</td>
</tr>
<tr>
<td>b -2 -2 -2 -2 -2 0 0 0</td>
</tr>
<tr>
<td>c -1 -1 -1 -1 -1 -1 -1</td>
</tr>
<tr>
<td>d -1 -1 -1 -1 -1 -1 -1</td>
</tr>
<tr>
<td>e -2 -2 0 0 0 -2 -2 -2</td>
</tr>
</tbody>
</table>

El resultado que se ha de obtener de los tres juegos introducidos es:

<table>
<thead>
<tr>
<th>Tabla 2 Resultado previsto del juego de test</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Juego 1</strong></td>
</tr>
<tr>
<td><strong>Juego 2</strong></td>
</tr>
<tr>
<td><strong>Juego 3</strong></td>
</tr>
</tbody>
</table>
La pruebas de integración del módulo GLMG se muestran en los cronogramas que aparecen a continuación. Esta simulación prueba que el funcionamiento del módulo es el correcto:

Figura 9 Cronogramas de prueba del módulo GLMG

4 SÍNTESIS DEL CIRCUITO

La síntesis del circuito se ha realizado sobre FPGA’s de ACTEL familia ACT3. Concretamente sobre una A1460A de 6.000 puertas equivalentes. El dispositivo programado no llega a ocupar un 20% de la capacidad de la pastilla. (80 registros y 200 módulos lógicos). De los retardos entre registros y la información de setup/hold entre registros se determina que el periodo mínimo de reloj será de 40 ns, lo que garantiza una velocidad de reloj de 25 MHz.

Figura 10 Placa de prueba de la FPGA de Lukasiewicz

5 CONCLUSIONES

El circuito etiquetado como Luka00 permite realizar la inferencia por el método de Lukasiewicz con cinco valores de inferencia de entrada. Los valores de inferencia pueden estar codificados con un número arbitrario de dígitos. La velocidad del circuito implementado en una FPGA de ACTEL 1460A estándar (velocidad lenta) es de 25 MHz y tiene un tamaño aproximado entre las 1000 y las 1500 puertas lógicas.

La velocidad de inferencia es igual a la velocidad del circuito dividida entre el número de dígitos empleados para codificar los valores de pertenencia más uno. Como los dígitos en base 3 notación signo-dígito equivalen a 2,3 bits, la velocidad del circuito es aproximadamente igual a: \( \text{Freno} \times 2.3 / (\text{bits_valores_pertenencia+1}) \).

En las simulaciones presentadas, se han empleado valores con 6 dígitos, lo que equivale a valores de 13 bits. En este caso la velocidad de inferencia es igual a 4 millones de inferencias por segundo.

La latencia depende del número de entradas y no del número de dígitos por entrada. Es igual a 4 \( \times \) (número de variables-1). En este caso, la latencia es de 16 ciclos de reloj (640 ns).

El rendimiento se puede mejorar utilizando una arquitectura sistólica, en vez de utilizar un circuito serie. Esta es una posibilidad que deje abierta el diseño presentado. Se ganaría velocidad pero, por otro lado, el tamaño del circuito sería mucho mayor, para el caso que nos ocupa incluso hasta seis veces mayor.

6 AGRADECIMIENTOS

Este trabajo ha podido realizarse en el marco del proyecto de la CICYT 96-1393-C06-2 titulado “Microprocesador Borroso de Altas Prestaciones”. También deseamos agradecer la inestimable ayuda de nuestro compañero David Madroño Baeza en las tareas de laboratorio.
7 BIBLIOGRAFÍA


Un Modelo de Selección para Problemas de Decisión con Múltiples Expertos e Información Lingüística Multi-Granular

Francisco Herrera Triguero  
Dpto. de Ciencias de la Computación e I.A.  
E.T.S. de Ingeniería Informática  
Universidad de Granada  
18071 - Granada  
e-mail: herrera@decsai.ugr.es

Enrique Herrera Viedma  
Dpto. de Ciencias de la Computación e I.A.  
E.T.S. de Ingeniería Informática  
Universidad de Granada  
18071 - Granada  
e-mail: viedma@decsai.ugr.es

Luis Martínez López  
Dpto. de Informática  
Escuela Politécnica Superior  
Universidad de Jaén  
23071 - Jaén  
e-mail: martin@ujaen.es

Resumen

En este trabajo estudiamos los problemas de decisión con múltiples expertos, asumiendo que éstos proporcionan sus preferencias sobre las alternativas mediante valores lingüísticos evaluados en conjuntos de etiquetas con distinta granularity y/o semántica, esto es, mediante información lingüística multi-granular. En este contexto de decisión, se presenta un modelo de selección compuesto por dos pasos con objeto de obtener el conjunto solución de alternativas. Primero, se realiza la fusión de la información lingüística multi-granular expresada por los expertos, obteniéndose las preferencias lingüísticas colectivas sobre las alternativas. Y segundo, se realiza la selección de las mejores alternativas a partir de los valores de preferencia colectivos.

Palabras clave: Decisión con múltiples-expertos, información lingüística, multi-granularidad, grado de selección.

1 Introducción

En muchas actividades de decisión nos encontramos aspectos que no son fácilmente calificables mediante valores precisos, ya sea por su propia naturaleza (aspectos cualitativos: "belleza", "comfortabilidad", etc...) o simplemente porque en ese momento no está disponible o es muy costoso conseguir un valor exacto, por lo que un valor aproximado es suficiente. En tales circunstancias, es normal manejar y representar los aspectos cualitativos como términos lingüísticos mediante variables lingüísticas [12], es decir, variables cuyos valores no son números sino palabras o frases en un lenguaje natural o artificial. Este enfoque lingüístico ha sido utilizado por diversos autores para resolver problemas de decisión [4, 2, 5, 9, 11].

En el enfoque lingüístico es muy importante determinar la "granularidad de la incertidumbre", es decir, la cardinalidad del conjunto de términos lingüísticos utilizado para expresar la información. Según el grado de incertidumbre que un experto tiene al calificar un fenómeno, el conjunto de etiquetas elegido para expresar su opinión tendrá más o menos términos. Cuando existen distintos expertos con diferentes grados de incertidumbre sobre un fenómeno, pueden utilizar conjuntos de etiquetas con distinta granularity y/o semántica para expresar sus preferencias.

En este trabajo, consideramos problemas de decisión con múltiples expertos que dan sus preferencias sobre un conjunto de alternativas usando valores lingüísticos evaluados sobre conjuntos de etiquetas con distinta granularity y/o semántica, esto es, mediante información lingüística multi-granular. Presentamos un modelo de selección que obtiene el conjunto solución de alternativas siguiendo dos pasos:

1. Fusión de la información lingüística multi-granular. En esta fase, se obtiene una preferencia lingüística colectiva sobre cada alternativa mediante la fusión de las preferencias lingüísticas multi-granulares individuales dadas por los expertos sobre cada una de las alternativas. El esquema de fusión sigue las dos siguientes fases:
   (a) Hacer uniforme la información lingüística multi-granular. Consiste en expresar las preferencias lingüísticas multi-granulares individuales en un único conjunto básico de etiquetas (CBE). Cada valor lingüístico multi-granular suministrado por los expertos se representa como un conjunto difuso en el CBE.
   (b) Cálculo de las preferencias lingüísticas colectivas. Para cada alternativa se calcula la preferencia lingüística colectiva de acuerdo a las preferencias de todos los expertos.

2. Selección de las mejores alternativas. A partir de los valores colectivos calculamos una relación de preferencia difusa usando la Teoría de la Posibilidad aplicada a los conjuntos difusos definidos en CBE. Finalmente, sobre esta relación aplicamos una función de selección para obtener las mejores
El trabajo se estructura como sigue: en la Sección 2 presentamos el problema de decisión en contexto lingüístico; en la Sección 3 se muestra el modelo de selección; en la Sección 4 se da un ejemplo; y por último, apuntamos algunos comentarios finales.

2 El Problema de Decisión con Múltiples Expertos e Información Linguística Multi-Granular

Consideramos un problema de decisión en el cual tenemos un conjunto finito de alternativas \( X = \{x_1, ..., x_n\} \) (\( n \geq 2 \)) que son calificadas según un conjunto finito de expertos \( P = \{p_1, ..., p_m\} \) (\( m \geq 2 \)). Cada experto \( p_j \) da un vector de utilidad con un valor lingüístico de preferencia \( \mathbf{p}^{ij} \) para cada alternativa \( x_i \). Asumimos que cada experto \( p_j \) puede usar diferentes conjuntos de etiquetas \( S_j \) para expresar sus preferencias. Por lo tanto, para cada experto \( p_j \), el vector de utilidad se define como un subconjunto lingüístico de selección sobre el conjunto \( X \) de alternativas valorado lingüísticamente en \( S_j \):

\[
\mathbf{p}^{ij} \rightarrow (p_1^{ij}, ..., p_m^{ij}) \quad \mathbf{p}^{ij} \in S_j
\]

\( S_j = \{s_{0j}, ..., s_{kj}\} \quad j \in \{1, ..., m\} \)

donde \( k_j + 1 \) es la granularidad de \( S_j \).

Cada conjunto \( S_j \) es definido como un conjunto finito y totalmente ordenado de términos lingüísticos que representan un valor posible de una variable lingüística en el sentido usual [1].

La semántica de cada etiqueta está dada por números difusos definidos sobre el intervalo [0,1], descritos por funciones de pertenencia trapezoidales lineales, representadas por la tupla \((x_0, x_1, x_2, x_3)\), donde los parámetros \( x_1, x_2 \) indican el intervalo en el que la función de pertenencia vale 1.0 y \( x_0, x_3 \) indican los límites izquierdo y derecho del soporte de la función de pertenencia. La etiqueta del centro representa una incertidumbre de "aproximadamente 0.5" y el resto de etiquetas está distribuido simétricamente a ambos lados de la misma [1].

3 Modelo de Selección

Aquí desarrollamos cada paso del modelo de selección para problemas de decisión con múltiples expertos e información lingüística multi-granular presentado en la introducción.

3.1 Fusión de la Información Lingüística Multi-granular

En este fase se obtienen las preferencias lingüísticas colectivas sobre las alternativas de acuerdo a las preferencias lingüísticas multi-granulares individuales expresadas por los expertos. La técnica de fusión de información lingüística multi-granular que nos permite hallar los valores colectivos se desarrolla en los dos siguientes pasos:

1. Hacer uniforme la información lingüística multi-granular.

2. Cálculo de las preferencias lingüísticas colectivas.

3.1.1 Hacer Uniforme la Información Lingüística Multi-Granular

Para poder manejar la información lingüística multi-granular la representamos uniformemente transformándola a un único conjunto de etiquetas, CBE, que notamos como \( S_T \).

Antes de definir el proceso de conversión a \( S_T \), hemos de seleccionar el CBE. Este debe ser un conjunto de etiquetas lingüísticas que nos permita mantener el grado de incertidumbre asociado a cada experto y la capacidad de discriminación para expresar los valores de preferencia. Con vistas a cumplir este objetivo, buscamos los conjuntos de máxima granularidad en \( \{S_j, \forall j\} \). Cuando existe un único conjunto de etiquetas con máxima granularidad, se elige como CBE. Si encontramos dos o más conjuntos de máxima granularidad, entonces el CBE es seleccionado dependiendo de la semántica de estos conjuntos de etiquetas:

1. Si todos los conjuntos de etiquetas de máxima granularidad tienen la misma semántica, entonces \( S_T \) es cualquiera de ellos.

2. Si existen algunos conjuntos de etiquetas con diferente semántica, entonces \( S_T \) será un conjunto de etiquetas especial con un número de términos superior al que una persona es capaz de discriminar o distinguir (como mucho 11 ó 13 [6]). Definimos un conjunto de etiquetas especial con 15 términos y la siguiente semántica (ver Figura 1):

\[
\begin{array}{cccc}
\#_0 & (0,0,07) & \#_1 & (0,07,15) \\
\#_2 & (07,15,22) & \#_3 & (15,22,29) \\
\#_4 & (22,29,36) & \#_5 & (29,36,43) \\
\#_6 & (36,43,5) & \#_7 & (43,5,58) \\
\#_8 & (5,58,65) & \#_9 & (58,65,72) \\
\#_{10} & (65,72,79) & \#_{11} & (72,79,86) \\
\#_{12} & (79,86,93) & \#_{13} & (86,93,1) \\
\#_{14} & (93,1,1) & & \\
\end{array}
\]

Una vez seleccionado el CBE, \( S_T \), definimos una función de transformación de información lingüística multi-granular, que expresa la información de cada valor lingüístico \( \mathbf{p}^{ij} \in S_j \) como un conjunto difuso sobre \( S_T \).

Definición 1. Sean \( A = \{p_0, ..., p_T\} \) y \( S_T = \{c_0, ..., c_g\} \) dos conjuntos de etiquetas, con \( g \geq...
Una función de transformación de información lingüística multigranular, \( \tau_{AS_T} \), se define como:

\[
\tau_{AS_T} : A \rightarrow F(S_T)
\]

\[
\tau_{AS_T}(l_i) = \{ (c_k, \alpha^k) / k \in \{0, \ldots, g \} \}
\]

\[
\alpha^k = \max_y \{ \mu_i(y), \mu_{c_k}(y) \}
\]

donde \( F(S_T) \) es el conjunto de conjuntos difusos definidos en \( S_T \), \( \mu_i(y) \) y \( \mu_{c_k}(y) \) las funciones de pertenencia de las etiquetas \( l_i \) y \( c_k \) respectivamente.

El resultado de \( \tau_{AS_T} \), para cualquier valor lingüístico de \( A \) es un conjunto difuso definido en términos del conjunto de etiquetas \( S_T \).

**Ejemplo 1.** Sea \( A = \{ l_0, l_1, l_2, l_3, l_4 \} \) y \( S_T = \{ c_0, c_1, c_2, c_3, c_4, c_5, c_6 \} \) dos conjuntos de etiquetas, con 5 y 7 términos (Figura 2) y con las siguientes semánticas asociadas:

<table>
<thead>
<tr>
<th>( A )</th>
<th>( l_0 )</th>
<th>( l_1 )</th>
<th>( l_2 )</th>
<th>( l_3 )</th>
<th>( l_4 )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(0, 0, 25)</td>
<td>(0, 0, 16)</td>
<td>(0, 25, 5)</td>
<td>(0, 16, 34)</td>
<td>(0, 34, 5)</td>
</tr>
<tr>
<td>( S_T )</td>
<td>(0, 0, 16)</td>
<td>(0, 16, 34)</td>
<td>(0, 34, 5)</td>
<td>(34, 5, 66)</td>
<td>(5, 66, 84)</td>
</tr>
</tbody>
</table>

![Figura 2: Conjuntos de etiquetas A y S_T](image_url)

Veamos cómo expresar las etiquetas \( l_0 \) y \( l_1 \) en términos del conjunto \( S_T \). Los conjuntos difusos asociados a \( l_0 \) y \( l_1 \) después de aplicar \( \tau_{AS_T} \) son:

\[
\tau_{AS_T}(l_0) = \{ (c_0, 1), (c_1, .58), (c_2, .18), (c_3, 0), (c_4, 0), (c_5, 0) \}
\]

\[
\tau_{AS_T}(l_1) = \{ (c_0, .39), (c_1, .85), (c_2, .85), (c_3, .39), (c_4, 0), (c_5, 0), (c_6, 0) \}
\]

Por tanto, para hacer uniforme la información lingüística multi-granular seleccionamos \( S_T \) y aplicamos el conjunto de funciones de transformación de información lingüística multigranular \( \{ \tau_{S_jS_T}, j \in \{1, \ldots, m \} \} \) a los vectores de utilidad suministrados por los distintos expertos. Entonces cada valor lingüístico \( p^{ij} \) se representa mediante un conjunto difuso definido en \( S_T = \{ c_0, \ldots, c_7 \} \), caracterizado por la siguiente expresión:

\[
\tau_{S_jS_T}(p^{ij}) = \{ (c_0, \alpha^{ij}_0), \ldots, (c_7, \alpha^{ij}_7) \}
\]

De este modo, el vector de utilidad de cada experto \( p_j \) se representa como un vector de conjuntos difusos en \( S_T \):

\[
\tau_{S_jS_T}(p^{ij}) = \{ r^{ij}_0, \ldots, r^{ij}_7 \}
\]

Para simplificar la notación, cada conjunto difuso \( \tau_{S_jS_T}(p^{ij}) \) lo representamos como \( r^{ij} \), por lo que el vector de utilidad del experto \( p_j \) se representa como:

\[
( r^{ij}_0 , \ldots, r^{ij}_7 ) \text{ con } r^{ij} = \{ \alpha^{ij}_0 , \ldots, \alpha^{ij}_7 \}
\]

**3.1.2 Cálculo de las Preferencias Lingüísticas Colectivas**

En este paso del proceso de decisión la información aportada por un experto \( p_j \) sobre una alternativa \( x_i \), \( p^{ij}_k \), está definida como un conjunto difuso \( r^{ij}_k \) sobre \( S_T \). Por tanto, para obtener una preferencia lingüística colectiva sobre una alternativa, \( x_k \), debemos agregar estos conjuntos difusos \( r^{ij}_k, \forall j \). La preferencia lingüística colectiva sobre la alternativa \( x_k \) la notamos como \( r^k \), y es un nuevo conjunto difuso definido sobre \( S_T \) de acuerdo a la siguiente expresión:

\[
r^k = \{ \alpha^k_0, \ldots, \alpha^k_7 \}
\]

con función de pertenencia

\[
\alpha^k_k = f(\alpha^{ij}_0, \ldots, \alpha^{ij}_7), \ k \in \{0, \ldots, g\}
\]

donde \( f \) es un "operador de agregación".

Por tanto, el resultado de este paso del modelo de selección es un conjunto de preferencias lingüísticas colectivas, donde cada valor colectivo para cada alternativa es obtenido según las valoraciones individuales expresadas por todos los expertos sobre dicha alternativa,

\[
( r^1, \ldots, r^n )
\]

En la siguiente subsección, mostramos cómo encontrar el conjunto solución de alternativas a partir de las evaluaciones colectivas.
3.2 Selección de las Mejores Alternativas

El objetivo del modelo de selección es encontrar un conjunto de alternativas que contenga las mejores, de acuerdo a las preferencias de todos los expertos. En este caso las preferencias son conjuntos difusos sobre CBE, $r^i$. Entonces, tenemos que definir un método de selección, que aplicado directamente sobre las preferencias (conjuntos difusos), nos permita obtener la solución. Ésta no es una tarea fácil, pues hemos de comparar conjuntos difusos. Para resolverla cambiamos la representación de las preferencias lingüísticas colectivas basadas en conjuntos difusos por una representación basada en relaciones de preferencia difusas. Usamos el método de comparación de números difusos en contexto posibilitivo descrito en [3]. Específicamente, aplicamos una modificación del grado de posibilidad de dominancia sobre números difusos propuesto en [3], para que actúe sobre conjuntos difusos $r^i$ definidos en un universo discreto (el conjunto básico de etiquetas $S^r$). Este método de selección queda definido por los dos siguientes pasos:

1. Calcular una relación de preferencia difusa.
2. Aplicar un grado de selección a la relación de preferencia difusa para ordenar las alternativas y seleccionar la(s) mejor(es).

1. Obtener una Relación de Preferencia Difusa.

La siguiente definición se usa para comparar números difusos.

Definición 2 [3]. Sean $u$ y $v$ dos números difusos, el grado de posibilidad de dominancia de $u$ sobre $v$ es:

$$P(u \geq v) = \max_{z} \min_{y \in S^u} \{\mu_u(z), \mu_v(y)\}$$

Sin embargo, nosotros tenemos que ordenar conjuntos difusos en un universo discreto, $S_r$. En la siguiente definición adaptamos el grado de posibilidad de dominancia para poder trabajar en $S_r$.

Definición 3. Sean $x_i, x_j \in X (i \neq j)$ dos alternativas con sus respectivos conjuntos difusos de preferencia $r^i, r^j \in F(S_r)$, entonces el grado de preferencia de $x_i$ sobre $x_j$, $b_{ij}$, se obtiene según la siguiente expresión:

$$b_{ij} = \max_{c_l} \min_{c_k \in c_l} \{\mu_{r^i}(c_l), \mu_{r^j}(c_k)\}$$

donde $\mu_{r^i}(c_l) = a^i_l$ y $\mu_{r^j}(c_k) = a^j_k$.

Aplicando esta definición sobre todos los posibles pares ($i \neq j$) de las alternativas, obtenemos una relación de preferencia difusa $B = [b_{ij}]$.

2. Aplicación de un Grado de Selección.

Para finalizar, el modelo de selección calcula el conjunto solución de alternativas aplicando un grado de selección sobre la relación de preferencia difusa, $B$. En [8] se presenta una muestra de los distintos grados de selección que se pueden usar. Usando uno de ellos ordenamos las alternativas y seleccionamos aquella(s) con valor máximo en su grado de selección.

En la siguiente sección presentamos un ejemplo particular de aplicación de este modelo general de selección.

4 Ejemplo

Supongamos que tenemos una asesoría bursátil que recibe el encargo de hacer un estudio para invertir una cantidad de dinero de la forma más rentable. Existen cuatro posibles opciones de inversión:

- $x_1$ es una compañía de automóviles,
- $x_2$ es una compañía alimenticia,
- $x_3$ es una compañía de ordenadores,
- $x_4$ es una compañía armamentos.

La asesoría bursátil tiene un grupo de departamentos para consultar sus decisiones.

- $p_1$ es el departamento de análisis de riesgos,
- $p_2$ es el departamento de análisis de crecimiento,
- $p_3$ es el departamento de análisis medioambiental,
- $p_4$ es el departamento de análisis socio-político.

Cada uno es dirigido por un experto. Los expertos usan diferentes conjuntos de etiquetas para expresar sus preferencias lingüísticas sobre el conjunto de las alternativas. En particular:

- $p_1$ utiliza el conjunto de etiquetas $A$ de 9 términos.
- $p_2$ utiliza el conjunto de etiquetas $B$ de 7 términos.
- $p_3$ utiliza el conjunto de etiquetas $C$ de 5 términos.
- $p_4$ utiliza el conjunto de etiquetas $D$ de 9 términos.

<table>
<thead>
<tr>
<th>Conj. Etiquetas A</th>
<th>Conj. Etiquetas B</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_0$ (0 0 .12)</td>
<td>$b_0$ (0 0 .16)</td>
</tr>
<tr>
<td>$a_1$ (.12 .25)</td>
<td>$b_1$ (.16 .33)</td>
</tr>
<tr>
<td>$a_2$ (.12 .25 .37)</td>
<td>$b_2$ (.16 .33 .5)</td>
</tr>
<tr>
<td>$a_3$ (.25 .37 .5)</td>
<td>$b_3$ (.33 .5 .66)</td>
</tr>
<tr>
<td>$a_4$ (.37 .5 .62)</td>
<td>$b_4$ (.5 .66 .83)</td>
</tr>
<tr>
<td>$a_5$ (.5 .62 .75)</td>
<td>$b_5$ (.66 .83 .1)</td>
</tr>
<tr>
<td>$a_6$ (.62 .75 .87)</td>
<td>$b_6$ (.83 1)</td>
</tr>
<tr>
<td>$a_7$ (.75 .87 .1)</td>
<td></td>
</tr>
<tr>
<td>$a_8$ (.87 1 1)</td>
<td></td>
</tr>
</tbody>
</table>
obtenemos los siguientes conjuntos difusos sobre $S_T$:

$\sum^{n}_{i=1} w_i b_i$

Donde $W = \{w_1, \ldots, w_n\}$ es un vector de pesos, tal que $w_i \in [0, 1]$ y $\sum i w_i = 1$. $B$ es el vector ordenado asociado a $A$, donde $b_i \in B$ es el $i$-ésimo mayor valor en $A$.

Nuestro interés es alcanzar soluciones que expresen la opinión de la mayoría de los expertos. En este sentido, los pesos $w_i$ para la agregación se pueden calcular a partir de la función que describe a un cuantificador lingüístico proporcional creciente $Q$ mediante la siguiente expresión [10]:

$w_i = Q(i/m) - Q((i - 1)/m), i = 1, \ldots, m$.

con

$Q(t) = \begin{cases} 
0 & \text{si } t < a \\
\frac{t-a}{b-a} & \text{si } a \leq t \leq b \\
1 & \text{si } t > b
\end{cases}$

con $a, b, t \in [0, 1]$. En este caso, notaremos al operador OWA guiado por el cuantificador lingüístico $Q$, como $F_Q$.

En este ejemplo utilizamos el cuantificador "tantos como sea posible" con parámetros $(a = 0.5, b = 1)$, y por tanto con el vector de pesos $W = \{0.0, 0.5, 5\}$. Entonces, las preferencias lingüísticas colectivas que obtenemos son:

$\sum^{n}_{i=1} w_i b_i$

1.2 Calcular las preferencias lingüísticas colectivas.

A partir de los $r^I$ calculamos las preferencias lingüísticas colectiva sobre cada alternativa, usando el operador OWA guiado por un cuantificador lingüístico.

**Definición 4** [10]. Se $A = \{a_1, \ldots, a_n\}$ un conjunto de valores a ser agregados, el operador Ordered Weighted Averaging (OWA), $F$ se define como,

$F(a_1, \ldots, a_n) = \sum W b_i$

Donde $W = \{w_1, \ldots, w_n\}$ es un vector de pesos, tal que $w_i \in [0, 1]$ y $\sum i w_i = 1$. $B$ es el vector ordenado asociado a $A$, donde $b_i \in B$ es el $i$-ésimo mayor valor en $A$.

Nuestro interés es alcanzar soluciones que expresen la opinión de la mayoría de los expertos. En este sentido, los pesos $w_i$ para la agregación se pueden calcular a partir de la función que describe a un cuantificador lingüístico proporcional creciente $Q$ mediante la siguiente expresión [10]:

$w_i = Q(i/m) - Q((i - 1)/m), i = 1, \ldots, m$, con

$Q(t) = \begin{cases} 
0 & \text{si } t < a \\
\frac{t-a}{b-a} & \text{si } a \leq t \leq b \\
1 & \text{si } t > b
\end{cases}$

con $a, b, t \in [0, 1]$. En este caso, notaremos al operador OWA guiado por el cuantificador lingüístico $Q$, como $F_Q$.

En este ejemplo utilizamos el cuantificador "tantos como sea posible" con parámetros $(a = 0.5, b = 1)$, y por tanto con el vector de pesos $W = \{0.0, 0.5, 5\}$. Entonces, las preferencias lingüísticas colectivas que obtenemos son:

$\sum^{n}_{i=1} w_i b_i$
2. Selección de las Mejores Alternativas.
Se realiza también en dos pasos:

2.1 Cálculo de la relación de preferencia difusa. Obtengamos la siguiente relación de preferencia difusa a partir de los valores colectivos:

\[
\begin{pmatrix}
   - & .31 & .52 & .12 \\
   .82 & - & .52 & .27 \\
   .45 & 0 & - & 0 \\
   .27 & .27 & .27 & -
\end{pmatrix}
\]

2.2 Aplicación del grado de selección de no dominancia. A cada alternativa \( x_i \), le calculamos su grado de selección de no dominancia \( NDD_i \).

Definición 5 [7]: Sea \( B = [b_{ij}] \) una relación de preferencia difusa definida sobre el conjunto de alternativas \( X \). Para la alternativa \( x_i \), su grado de no dominancia \( NDD_i \), es definido como sigue:

\[ NDD_i = \min_{x_j}[1 - b_{ji}, j \neq i] \]

donde \( b_{ji} \) se calcula mediante la siguiente expresión,

\[ b_{ji} = \max\{b_{ji} - b_{ij}, 0\}, \]

y representa el grado para el cual \( x_i \) es estrictamente dominado por \( x_j \).

Entonces, primero calculamos la relación de preferencia estricta \( B^* \):

\[
\begin{pmatrix}
   - & 0 & .07 & 0 \\
   .51 & - & .52 & 0 \\
   .00 & 0 & - & 0 \\
   .15 & 0 & .27 & -
\end{pmatrix}
\]

y luego, los grados de no dominancia:

\[ \{NDD_1 = .49, NDD_2 = 1, NDD_3 = .84, NDD_4 = 1 \} \]

Por tanto, el conjunto solución de alternativas obtenido es \( X^{ND} = \{x_2, x_4\} \), siendo las mejores opciones la compañía alimenticia y la armamentística.

5 Comentarios Finales

En este trabajo hemos presentado un modelo de selección para problemas de toma de decisión con múltiples expertos los cuales expresan sus preferencias usando distintos dominios lingüísticos.

Este modelo es útil en problemas de decisión donde los expertos provienen de distintas áreas de conocimiento o tienen distintos grados de conocimiento sobre el problema. La técnica de fusión de información lingüística multi-granular usada puede ser aplicada en muchas otras áreas como diagnóstico, recuperación de información, etc.

Referencias


