Capítulo 2

Una metodología genética para el aprendizaje de reglas difusas

2.1 Un enfoque iterativo para el aprendizaje de conceptos

En la literatura podemos encontrar propuestas muy diferentes para el problema del aprendizaje de múltiples conceptos ([31, 48, 85, 86, 110, 126]). En un intento de clasificar las diferentes propuestas hemos considerado como un elemento fundamental la posible descomposición del problema. Así, consideramos la siguiente clasificación:

a) **Enfoques sin descomposición del problema.** En este primer caso, el aprendizaje de conceptos se considera como un problema indivisible, y el algoritmo trata de extraer un conocimiento que dé una respuesta global a todos ellos. En esta categoría podemos considerar los algoritmos de aprendizaje inductivo basados en redes neuronales, donde toda la información existente es suministrada a la red y esta trata de encontrar una disposición interna que sea capaz de dar una solución al problema.

b) **Enfoques con descomposición del problema.** En esta categoría, podemos encuadrar a aquellos algoritmos que asumen que el problema de la clasificación de los ejemplos se puede descomponer en un conjunto de subproblemas. En este caso, la solución al problema se construye en base a la agregación de las soluciones obtenidas en cada subproblema.

Dentro del enfoque por descomposición del problema, podemos encontrar en la literatura del aprendizaje automático diversas estrategias, tales como las que proporcionan los algoritmos basados en árboles de decisión ([125]). Sin embargo, nos centraremos en nuestro estudio en un tipo especial de estrategia, utilizada ampliamente en diversos algoritmos de aprendizaje, que denominaremos **estrategia iterativa**. Una estrategia iterativa es aquella donde el problema de aprender m conceptos se descompone en m subproblemas cada uno de ellos correspondiente al aprendizaje de un único concepto. Básicamente, esta estrategia fija la clase que se desea aprender y en base a los ejemplos del conjunto de entrenamiento y mediante procesos de generalización y especificación se va construyendo la descripción de la clase. Por tanto, una solución parcial, siguiendo esta estrategia, se compone de una o varias descripciones para un concepto. La solución global se construye como la agregación de las diferentes soluciones parciales.

Esquema 2.1 *Enfoque iterativo con un solo nivel*

0. Sean $X_1, X_2, \ldots, X_n$ las variables predictivas que son incluidas para aprender el comportamiento de un determinado sistema. Sea Y la variable de clasificación definida sobre un dominio discreto de m conceptos. Además, sea E un conjunto con k ejemplos que describen el comportamiento en el pasado de dicho sistema. Sean $GLOBAL$ y $PARCIAL$ dos conjuntos de reglas que inicialmente están vacíos.

1. Seleccionar un concepto B que aún no se ha aprendido.

2. Encontrar un conjunto de descripciones $PARCIAL$ que sean una solución al problema de clasificar los ejemplos del concepto B que se encuentran en el conjunto de entrenamiento E.

3. Añadir las nuevas descripciones al conjunto $GLOBAL$.

$$GLOBAL = GLOBAL \bigcup PARCIAL$$

4. Eliminar las descripciones de $PARCIAL$; $PARCIAL = \emptyset$.

5. Si existen más objetos que aprender, volver al paso 1., en otro caso devolver $GLOBAL$ como solución al problema de aprendizaje.

Normalmente, un algoritmo inductivo para el aprendizaje de reglas guiado por una estrategia iterativa responde a la estructura propuesta en el esquema 2.1. Las principales diferencias entre dos algoritmos inductivos que siguen esta estrategia se encuentran en la forma de definir el paso 2 del esquema anterior. Este paso implica dos características fundamentales en el algoritmo de aprendizaje:

a) un criterio para valorar si un determinado conjunto de descripciones se puede considerar como una solución para representar un determinado concepto, y

b) un algoritmo de búsqueda que permita encontrar dicho conjunto de descripciones.

En general, los algoritmos que siguen esta estrategia se apoyan en las condiciones de consistencia y completitud, propuestas en [114], como criterio para determinar si un conjunto de descripciones es solución al problema de aprender los conceptos. Estas dos condiciones, que ya se describieron en el capítulo anterior, proporcionan un fundamento lógico a los algoritmos de aprendizaje inductivo y aunque Michalski las propone como condición que debe satisfacer la solución global, se puede extender su aplicación de forma trivial a las soluciones parciales. Si todas las soluciones parciales satisfacen las condiciones de consistencia y completitud, entonces la solución global compuesta por la agregación de las soluciones parciales también las satisfacen.

En la literatura podemos encontrar trabajos, tales como [51, 157], que van un paso más alla en la descomposición del problema a aprender. Según el planteamiento de estos autores, el encontrar un conjunto de descripciones que sea una solución al problema de clasificar los ejemplos de un concepto, puede descomponerse a su vez, en un conjunto de subproblemas más simples. En este planteamiento, podemos considerar dos soluciones parciales de diferente orden: una solución parcial de primer orden, que corresponde a la solución parcial del esquema anterior y una solución parcial de segundo orden, que establece que una descripción es solución, si puede ser una descripción que pertenezca al conjunto de descripciones para un determinado concepto. Los algoritmos de aprendizaje que siguen esta filosofía mantienen básicamente la estructura que se propone en el esquema 2.2.

Esta nueva descomposición plantea dos problemas:

a) El número de descripciones necesarias para cada concepto no se conoce a priori.

b) Es necesario establecer un criterio para determinar si una descripción puede considerarse como una solución parcial de segundo orden.

Para dar respuesta a los anteriores problemas, en la mayoría de los casos, se vuelve a recurrir a la extensión de las condiciones de consistencia y completitud para determinar las soluciones de segundo orden. La condición de consistencia es fácilmente extensible, ya que si el conjunto de descripciones para un determinado concepto verifican esta condición, cada una de las descripciones particulares deben verificarla también. Por contra, la condición de completitud no es extensible en este sentido. Sin embargo, la agregación de las descripciones individuales si debe verificar la condición de completitud. Por consiguiente, la condición de completitud establece un criterio para determinar cuándo el conjunto de descripciones obtenidas son suficientes para una determinada clase.

Esquema 2.2 *Enfoque iterativo con dos niveles*

0. Sean $X_1, X_2, \ldots, X_n$ las variables predictivas que son incluidas para aprender el comportamiento de un determinado sistema. Sea Y la variable de clasificación definida sobre un dominio discreto de m conceptos. Además, sea E un conjunto con k ejemplos que describen el comportamiento en el pasado de dicho sistema. Sean $GLOBAL$ y $PARCIAL$ dos conjuntos de reglas que inicialmente estan vacíos.

1. Seleccionar un concepto B que aún no se ha aprendido.

2. Mientras $PARCIAL$ no sea una solución parcial de primer orden hacer:

    2.1. Encontrar una descripción d que sea solución parcial de segundo orden.

    2.2. Añadir d al conjunto de descripciones $PARCIAL$.

    $$PARCIAL = PARCIAL \bigcup d.$$

    2.3 Penalizar las descripciones de $PARCIAL$ para que no vuelvan a ser seleccionadas.

3. Añadir las nuevas descripciones al conjunto $GLOBAL$.

    $$GLOBAL = GLOBAL \bigcup PARCIAL$$

4. Eliminar las descripciones de $PARCIAL$; $PARCIAL = \emptyset$.

5. Si existen más objetos que aprender, volver al paso 1., en otro caso devolver $GLOBAL$ como solución al problema de aprendizaje.

Teniendo en cuenta lo anterior, tenemos criterios para determinar si una descripción es una solución parcial de segundo orden, y en base a ella, criterios para determinar si un conjunto de descripciones pueden considerarse como solución parcial de primer orden.

Definición 2.1 *Sea una descripción d para un determinado concepto T. Diremos que d es una descripción que proporciona una solución parcial de segundo orden al problema de describir T, si d verifica la condición de consistencia.*

Definición 2.2 *Sea un conjunto de descripciones $D = \{d_1, d_2, \ldots, d_m\}$ para un determinado concepto T. Diremos que D es una solución parcial de primer orden al problema de describir T, si $\forall d_i$, d_i es solución parcial de segundo orden y además D verifica la condición de completitud.*

El apartado 2.3. de la descripción de esta metodología es importante, ya que si las descripciones sobre un concepto que se han aprendido no son penalizadas, es posible que vuelvan a aparecer como solución. Una forma muy fácil de penalizar las reglas consiste en eliminar los ejemplos del conjunto de entrenamiento que son representados por alguna de las descripciones ya existentes en el conjunto $PARCIAL$. Esta forma de penalizar reglas es la más común en algoritmos que siguen esta metodología tales como los algoritmos de la familia AQ [115] o el algoritmo CN2 [19].

En esta metodología podemos encontrar soluciones parciales a dos niveles; en el primer nivel la solución esta formada por un conjunto de descripciones para un único concepto que satisface las condiciones de consistencia y completitud, mientras que en el segundo nivel la solución esta formada por una única descripción que satisface la condición de consistencia. En el siguiente apartado estudiamos más detalladamente algunas características que son deseables para las soluciones parciales de segundo nivel.

2.2 SLAVE: un algoritmo de aprendizaje inductivo basado en el modelo iterativo genético

En este capítulo se han descrito los elementos más importantes que constituyen un algoritmo inductivo de aprendizaje de conceptos basados en reglas atributo-valuadas. Además se han propuesto enfoques generales para la construcción de dichos algoritmos.

Un modelo iterativo genético de aprendizaje sigue el esquema 2.2, y usa un algoritmo genético para buscar las soluciones parciales de segundo orden del apartado 2.1.

El enfoque iterativo genético puede considerarse como un nuevo modelo para aprender con algoritmos genéticos, diferente a los modelos Michigan y Pittsburgh, que permite reducir el espacio de búsqueda gracias a la descomposición del problema y además permite que cada subproblema pueda aprenderse de forma paralela. Los primeros algoritmos que han utilizado este enfoque son SIA [157] y SLAVE [51]. Posteriormente, han surgido otros sistemas de aprendizaje con este enfoque tales como [21, 22].

El algoritmo genético necesario para obtener las soluciones parciales de segundo orden, representa en cada individuo de su población una única regla atributo-valuada al igual que la aproximación Michigan, pero a diferencia de ésta, solamente se considera la mejor regla de entre las obtenidas durante el proceso genético.

Podemos destacar dos ventajas fundamentales, comparando este modelo genético con los modelos Michigan y Pittsburgh:

- El modelo iterativo génetico, al igual que el modelo Pittsburgh, no requiere prácticamente ninguna adaptación especial del algoritmo de aprendizaje.

- El modelo iterativo génetico reduce el espacio de búsqueda considerablemente, debido fundamentalmente a su esquema de descomposición del problema. El algoritmo genético busca, en cada iteración, una única solución. En la tabla 2.1, donde N es el número de reglas necesario para describir el problema, K es el número de clases, n es el número de atributos y V es el número de valores por atributo, propuesta en [157] se muestra el tamaño del espacio de búsqueda para el algoritmo genético en cada una de las diferentes propuestas, y se pone de manifiesto la reducción del espacio de búsqueda en el modelo iterativo genético.

Modelo genético	Tamaño del espacio de búsqueda
Pittsburgh	$N^{K(V+1)^n}$
Michigan	$K(V+1)^n$
Iterativo	2^n

Tabla 2.1: Tamaño del espacio de búsqueda para los distintos modelos de aprendizaje con genéticos.

A continuación nos centraremos en describir el sistema de aprendizaje SLAVE (Structure Learning Algorithm in Vague Enviroment) [51], un algoritmo de aprendizaje inductivo, basado en el modelo genético iterativo, definido para extraer en términos de reglas atributo-valuadas el conocimiento de un sistema.

En las siguientes secciones, describimos cada uno de los elementos que componen este algoritmo de aprendizaje.

2.2.1 Representación del conocimiento

El objetivo de un algoritmo de aprendizaje inductivo consiste en establecer una hipótesis para explicar un conjunto de ejemplos que describen el comportamiento de un determinado sistema. El conocimiento que dicha hipótesis aporta para comprender a dicho sistema, viene determinado por el tipo de representación que el algoritmo de aprendizaje utiliza para almacenar la información. Así, si se usa una red neuronal, el algoritmo aprenderá el sistema, pero no es fácil determinar a partir de esta representación las relaciones existentes entre las distintas variables que intervienen en el sistema. Si se desea conocer dicha relación, es necesario utilizar otra representación del conocimiento. Las reglas atributo-valuadas son un tipo de representación del conocimiento que permiten establecer las relaciones existentes entre las variables involucradas en el sistema. El lenguaje DNF (forma normal disyuntiva modificada) propuesto en [115] y descrito en el primer capítulo, es un lenguaje útil para representar conceptos complejos en base a un conjunto disyuntivo de reglas de clasificación. La parte antecedente o descripción de la regla se compone de la conjunción de una o más variables, mientras la parte consecuente indica el concepto que debe ser asignado a un ejemplo cuando este se adapte a la parte condición. Además en la parte antecedente se permiten disyunciones internas, es decir, una misma variable puede contener una disyunción de valores de su dominio.

Ejemplo 2.1 *Sean X_1, X_2 y X_3 tres variables cuyos dominios asociados son D_1=\{rojo, verde, blanco, azul\}, D_2= \{esférico, cilíndrico, cúbico\} y D_3 =\{ 1, 2, 3, 4, 5\}respectivamente. Una regla válida en el lenguaje DNF podría ser:*

$$\textit{IF } (X_1 \textit{ is rojo}) \textit{ and } (X_2 \textit{ is esférico o cúbico}) \textit{ and } (X_3 \textit{ is 3 ó 4 ó 5})$$
$$\textit{THEN } Y \textit{ is } C$$

donde Y es la variable de clasificación y C es un valor de dicha variable.

La posibilidad dada a una variable de contener una disyunción de valores de su dominio, otorga a este modelo de regla una herramienta para generar nuevos valores, en base a la agregación de los mismos. Esta ventaja en la agrupación de valores, se pone de manifiesto en problemas donde existen variables con dominios asociados que mantienen algún tipo de estructura jerárquica o alguna ordenación entre sus valores. Por ejemplo, en el problema anterior, la variable X_3 tiene asociado un dominio ordenado. Esto implica, que la regla anterior, se puede reformular en el siguiente sentido:

IF (X_1 is rojo) and (X_2 is esférico o cúbico) and (X_3 is mayor o igual que 3) THEN Y is C

donde, mayor o igual que 3, *representa la agrupación de los valores* \{3,4,5\} *del dominio D_3 para la variable X_3.*

Esta agrupación de valores confiere al modelo la posibilidad de construir descripciones más comprensibles. Además, usando este tipo de lenguaje y mediante la agrupación de *valores, es posible determinar si una variable es útil para describir una clase. Si una variable toma como valor todo su dominio de definición, esto indica que esta variable no es necesaria para describir la clase. Por ejemplo, en la siguiente regla*

$$\begin{array}{c} \textit{IF } (X_1 \textit{ is rojo o verde o blanco o azul) and} \\ (X_2 \textit{ es cúbico) and} \\ (X_3 \textit{ is 1 o 2 o 3 o 4)THEN } Y \textit{ is } C \end{array}$$

la única variable relevante de la regla es X2, y la regla se puede reformular como:

$$\textit{IF } (X_2 \textit{ is cúbico) THEN } Y \textit{ is } C$$

Dada esta propiedad que ofrece el lenguaje DNF y que permite no sólo determinar la relación entre las variables predictivas y la variable de clasificación, sino que además permite establecer el subconjunto de variables predictivas que son necesarias para explicar la variable de clasificación, en el desarrollo de este algoritmo de aprendizaje usamos el siguiente modelo de regla basado en este lenguaje.

Sean n variables antecedente X_i, cada una con un referencial U_i y un dominio finito asociado D_i. Dada una variable consecuente Y con referencial V y un dominio finito asociado F, se propone el modelo de regla

$$\text{IF } X_1 \textit{ is } A_1 \textit{ and } \ldots \textit{ and } X_n \textit{ is } A_n \text{ THEN } Y \textit{ is } B$$

tal que el valor de la variable Y es B, donde $B \in F$ y el valor de la variable X_i is A_i, donde $A_i \in P(D_i)$ y $P(D_i)$ denota el conjunto de las partes D_i.

Por tanto, el conjunto de todas las posibles reglas es

$$\Delta = P(D_1) \times P(D_2) \times \ldots \times P(D_n) \times F.$$

Sobre este modelo de regla realizamos las dos siguientes consideraciones:

1. El modelo de regla podría contener más de una variable de clasificación, pero lo restringimos al uso de una única variable para facilitar tanto la comprensión de este modelo, como del resto de los componentes que constituyen este algoritmo de aprendizaje. Así, la extensión para trabajar con más de una variable de clasificación es directa.

2. En este modelo supondremos, sin pérdida de generalidad, que las variables X_i son variables lingüísticas en el sentido que toman valores sobre un dominio finito difuso D_i, y cada A_i se puede interpretar como una disyunción de etiquetas difusas. En función del tipo de dominio F de la variable consecuente podemos considerar dos

clases diferentes de clasificación: cuando el dominio toma valores sobre un referencial nominal o discreto y cuando el dominio toma valores sobre un referencial continuo. Cada uno de estos problemas requiere un tratamiento distinto, que será estudiado más adelante en esta memoria.

Las variables lingüísticas aportan dos características fundamentales desde el punto de vista del aprendizaje:

a) Permiten establecer discretizaciones flexibles sobre variables con referencial continuo.

b) Permiten dotar de semántica a dichas discretizaciones, de manera que cada etiqueta corresponde a la descripción de un término lingüístico. Esta característica es importante, si se desea que el conocimiento extraído de un determinado problema, sea fácilmente interpretable desde un punto de vista humano.

Una vez definida la herramienta de representación del conocimiento obtenido, es necesario establecer cómo se utilizará dicha herramienta. En la siguientes subsecciones se define el concepto de adaptación entre un ejemplo y una regla, que es necesario tanto para determinar en qué medida los ejemplos refuerzan la veracidad de una regla, cómo para determinar la clasificación de un ejemplo una vez extraído el conjunto de reglas.

2.2.2 Concepto de adaptación

Una vez determinado el modelo de regla a utilizar, es necesario establecer como dichas reglas serán utilizadas durante el proceso de predicción. El concepto clave para determinar la aplicación de las reglas es el de adaptación. La adaptación permite establecer si una regla se puede emplear para clasificar un ejemplo dado. Cuando trabajamos con ejemplos definidos de forma precisa y reglas atributo-valuadas clásicas, este concepto esta claramente definido. Si los valores asociados a las variables predictivas pertenecen al conjunto definido por la descripción de la regla, entonces dicha regla es aplicable para clasificar el ejemplo.

Sin embargo, cuando trabajamos con reglas atributo-valoradas difusas y/o ejemplos expresados de forma imprecisa, es necesario extender el concepto de adaptación, ya que no es posible una aplicación directa del concepto anterior. El primer paso para establecer esta extensión, es determinar la adaptación entre dos conjuntos difusos.

Definición 2.3 *Sean a y b dos conjuntos difusos sobre un mismo conjunto de referencia U, y * una t-norma. Definimos la compatibilidad entre a y b como la siguiente función:*

$$\sigma(a,b) = \sup_{x \in U}\{\mu_a(x) * \mu_b(x)\}$$

donde μ_c es la función de pertenencia al conjunto difuso c.

El modelo de regla descrito anteriormente permite disyunciones internas en la valoración de las variables del antecedente. Por tanto, debemos extender a conjuntos de conjuntos difusos.

Definición 2.4 *Sean Dom_1 y Dom_2 dos dominios constituidos por conjuntos difusos sobre un mismo referencial U. Sean $C_1 \subseteq Dom_1$ y $C_2 \subseteq Dom_2$ cada uno de ellos un conjunto de conjuntos difusos. Definimos la compatibilidad entre estos conjuntos como:*

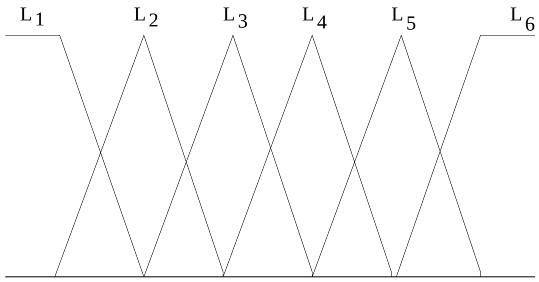
$$\sigma(C_1, C_2) = \sup_{a \in C_1} \sup_{b \in C_2} \sigma(a, b)$$

Usamos el mismo símbolo σ para ambas definiciones ya que la última definición es una extensión de la primera, y ambas coinciden cuando los conjuntos son unitarios.

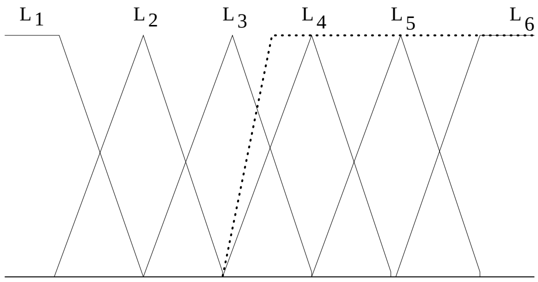
En base a estos conceptos, para calcular la adaptación entre un valor v (crisp o difuso) y un conjunto de conjuntos difusos $L \subseteq D$, ambos valores de una misma variable X con dominio difuso D, usamos la siguiente valoración:

$$Poss(v|L) = \frac{\sigma(v, L)}{\sigma(v, D)}$$

que establece el grado de adaptación entre X y v dado que $X = L$ y que se puede interpretar como una distribución de posibilidad.



(a) Un dominio difuso



(b) Envolvente convexa basada en una medida de posibilidad

Figura 2.1: Envolvente convexa de etiquetas difusas

$\boxed{\textbf{Ejemplo}}$ **2.2** *Sea X_i una variable lingüística cuyo dominio es el que se describe en la figura 2.1(a). Supongamos que el valor asignado a la variable X_i es $\{L_4, L_5, L_6\}$ en una determinada regla. Utilizando la anterior medida para calcular la adaptación entre un valor v y la valoración $\{L_4, L_5, L_6\}$ se puede interpretar como X_i* es mayor o igual que L_4 *ya que $Poss(v|\{L_4, L_5, L_6\})$ es la envolvente convexa normalizada que se describe en la figura 2.1(b).*

Es importante señalar que la normalización se añade para mantener la propiedad de agrupación de valores descrita en el ejemplo 2.1, ya que directamente los conjuntos difusos no tienen esa propiedad.

Sea e un ejemplo compuesto por diferentes características

$$e = (e_1, e_2, \ldots, e_n, c(e))$$

donde e_i es un conjunto difuso sobre el referencial U_i, y $c(e)$ es la clase asignada al ejemplo e.

Los ejemplos normalmente son valores crisp, sin embargo, el algoritmo de aprendizaje puede utilizar ejemplos difusos sin ninguna dificultad. Además, admitir ejemplos difusos es útil para incluir un grado de imprecisión en los datos. Por ejemplo, podemos utilizar ejemplos difusos cuando tenemos reglas imprecisas en las cuales no tenemos plena confianza y decidimos incluirlas como nuevos ejemplos. Además éstos aparecen cuando tenemos valores perdidos del tipo "no importa" y los reemplazamos por sus dominios completos (un conjunto de conjuntos difusos) los cuales representan que cualquier valor es posible.

En el ejemplo anterior, el valor *no importa* se reemplaza por $v = D$ y

$$Poss(D|L) = \frac{\sigma(D, L)}{\sigma(D, D)} = 1 \ \forall L \subseteq D$$

cuando trabajamos con etiquetas difusas normalizadas.

Para que el concepto de adaptación sea aplicable al modelo de regla definido, es necesario extenderlo al producto cartesiano.

Definición 2.5 *Sea $R_B(A)$ una regla cuya descripción es $A = (A_1, \ldots, A_n)$ $A_i \subseteq D_i$ y su parte consecuente es $B \in F$. Definimos la adaptación entre un ejemplo $e = (e_1, e_2, \ldots, e_n, c(e))$ y el antecedente de una regla como la combinación a través de una t-norma de las adaptación parcial de cada A_i con su componente asociada del ejemplo.*

$$U(e, A) = Poss(e_1|A_1) * Poss(e_2|A_2) * \ldots * Poss(e_n|A_n).$$

y definimos la adaptación entre un ejemplo y el consecuente de una regla como

$$U(e, B) = Poss(c(e)|B).$$

donde c(e) representa la clase o concepto asociado al ejemplo.

La función U establece una medida de adaptación entre la descripción de una regla y un ejemplo, teniendo en cuenta las medidas de posibilidad asociadas a cada variable de la descripción dada la evidencia soportada por el ejemplo. Estas medidas de posibilidad se combinan mediante una t-norma, ya que el nexo de unión entre variables en la descripción del modelo de regla empleado se realiza mediante el conectivo *y*. Por contra, la adaptación del ejemplo con el consecuente de la regla, es directamente la valoración definida anteriormente, al asumir en el modelo de regla que únicamente se admite una variable de clasificación. Para extender esta medida a un modelo de regla con más de una variable de clasificación, se aplica la extensión anteriormente propuesta para el cálculo de la adaptación con el antecedente de la regla.

En base a la definición del grado de adaptación entre un ejemplo y cada una de las reglas que forman parte del conocimiento extraido, se establece el modelo de inferencia. Podemos distinguir dos situaciones diferentes en función del tipo de variable de clasificación que se defina en el problema.

b) **Caso difuso**. La variable consecuente tiene referencial continuo o tiene un referencial crisp con un número elevado de elementos, en cualquier caso se utiliza un dominio difuso para discretizar el referencial. En la literatura se han propuesto distintos modelos de inferencia para este tipo de problemas ([34]). Para nuestro sistema hemos considerado el modelo de inferencia max-min, asociado al método de nitificación (defuzzification) de la media de las alturas ponderada por la adaptación y a la t-norma mínimo.

a) **Caso crisp**. La variable consecuente tiene referencial crisp y no necesita utilizarse un dominio difuso al tener un número suficientemente pequeño de elementos. En estos casos, el dominio suele corresponder al propio referencial.

   En este caso, se devuelve como clasificación, la clase de la regla que presenta mejor adaptación entre su antecedente y el ejemplo.

   Definición 2.6 *Dado un ejemplo e y un conjunto de reglas difusas $R_{B_j}(A^i)$, la regla que determina la clasificación de este ejemplo es $R_{B_s}(A^c)$ si verifica que*

   $$U(e, A^c) \geq U(e, A^i) \; / i = 1 \ldots p$$

   y B_s el valor consecuente de la regla con antecedente A^c, donde p es el número de reglas del conjunto de reglas.

   A diferencia del caso anterior donde las reglas cooperan para obtener una salida, en esta segunda situación las reglas compiten entre si y una única regla puede ser aplicada para obtener la clasificación del ejemplo. Por tanto, durante este proceso se puede producir una situación donde la adaptación no sea suficiente para determinar la salida y se producen conflictos. Un conflicto se produce, cuando existen dos o más reglas que representan diferentes conceptos y tienen el mismo mejor grado de adaptación entre sus antecedentes y el ejemplo. Por consiguiente, se hace necesario establecer mecanismos que resuelvan estos conflictos, de forma que asociados a la inferencia produzcan una mejor representación del sistema. Normalmente, los mecanismos de resolución de conflictos se establecen teniendo en cuenta cierto metaconocimiento que se extrae sobre el problema durante el proceso de aprendizaje, es decir, conocimiento que indica como usar el conocimiento extraido.

En la literatura, podemos encontrar diferentes mecanismos de resolución de conflictos [21, 83].

De lo anterior, se deduce que un algoritmo de aprendizaje inductivo que use el modelo de regla y el módulo de inferencia previamente descrito para el caso crisp, debe extraer dos tipos de conocimiento:

- El conocimiento que permite establecer las relaciones entre las variables predictivas y la variable de clasificación.
- El conocimiento que permite definir como se debe emplear el anterior conocimiento.

Para extraer este segundo tipo de conocimiento es necesario conocer el modelo general de aprendizaje que utiliza este modelo de regla. Así que más adelante se estudiarán detenidamente mecanismos de resolución de conflictos para el caso crisp.

2.2.3 Ejemplos positivos y negativos

Los algoritmos de aprendizaje inductivo basados en el modelo iterativo no pueden utilizar directamente el modelo de inferencia para determinar las reglas que se deben obtener para establecer una correcta descripción de un concepto. Por esta razón, estos sistemas establecen valoraciones para determinar la bondad de las reglas atendiendo a otros parámetros. Prácticamente la totalidad de estos algoritmos, tales como [51, 115, 157] y algunos otros basados en otros modelos como [31, 85, 125], utilizan como medida principal en dicha valoración, la capacidad que tiene cada descripción de representar correctamente los ejemplos asociados al concepto que se desea aprender en relación a los ejemplos que representa de otros conceptos y que se encuentran en el conjunto de entrenamiento. Estos son conocidos como ejemplos positivos y ejemplos negativos a una regla.

Debido a que la definición de ejemplo positivo y negativo esta estrechamente relacionada con el concepto de adaptación entre las reglas y los ejemplos, y a que SLAVE al trabajar con reglas difusas y ejemplos expresados de forma inexacta tiene una definición propia de adaptación, es necesario redefinir dichos conceptos para esta nueva situación.

La utilización de la función de adaptación definida en SLAVE, provoca una relajación de estos conceptos. En el caso de que la regla sea crisp, si un ejemplo afecta a una regla, entonces, o bien éste apoya completamente la veracidad de la regla, o bien la contradice completamente. En el caso de que las regla sea difusa, un mismo ejemplo puede apoyar y contradecir la veracidad de la regla en un cierto grado. Por consiguiente, la positividad o negatividad de los ejemplos a las reglas, se encuentra graduada por la medida de adaptación. Teniendo en cuenta esta situación, podemos definir el conjunto de ejemplos positivos y negativos a la misma.

Definición 2.7 *Dado un conjunto de entrenamiento E y una regla $R_B(A)$, diremos que el conjunto de ejemplos positivos para la regla $R_B(A)$ es el siguiente subconjunto difuso de E*

$$E^+(R_B(A)) = \{(e, U(e,A) * U(e,B)) \mid e \in E\}$$

*donde $A = (A_1, \ldots, A_n)$, y * es una t-norma.*

Definición 2.8 *Dado un conjunto de entrenamiento E y una regla $R_B(A)$, diremos que el conjunto de ejemplos negativos para la regla $R_B(A)$ es el siguiente subconjunto difuso de E*

$$E^-(R_B(A)) = \{(e, U(e,A) * U(e,\overline{B})) \mid e \in E\}$$

*donde $A = (A_1, \ldots, A_n))$, $\overline{B} = \{b \in F / b \neq B\}$ y * es una t-norma.*

Definición 2.9 *Dado un conjunto de entrenamiento E y una regla $R_B(A)$, diremos que el conjunto de ejemplos cubiertos por la regla $R_B(A)$ es el siguiente subconjunto difuso de E*

$$E(R_B(A)) = \{(e, U(e,A)) \mid e \in E\}$$

*donde $A = (A_1, \ldots, A_n))$, y * es una t-norma.*

Una vez que hemos definido los anteriores conjuntos, podemos determinar el número de ejemplos positivos y negativos para una regla. Así, notaremos $|J| = \sum_{u \in U} \mu_J(u)$, como el cardinal de un conjunto difuso J. Aplicando este cardinal a los anterior conjuntos, obtenemos las siguientes definiciones:

Definición 2.10 *En base a los conceptos anteriores, definimos:*

- *El número de ejemplos positivos a la regla $R_B(A)$: $n^+(R_B(A)) = |E^+(R_B(A))|$*
- *El número de ejemplos negativos a la regla $R_B(A)$: $n^-(R_B(A)) = |E^-(R_B(A))|$*
- *El número de ejemplos cubiertos por la regla $R_B(A)$: $n(R_B(A)) = |E(R_B(A))|$*

Las medidas aquí propuestas formarán la base para establecer la valoración de las reglas en el sistema de aprendizaje SLAVE.

2.2.4 Selección de reglas

Las soluciones parciales de segundo orden deben satisfacer la condición de consistencia. Sin embargo, la preservación de esta propiedad en la aplicación de los algoritmos a problemas reales es dificil, debido fundamentalmente a la aparición de inconsistencias, ruido o imprecisiones en el conjunto de entrenamiento. En estos casos, lo que parece más apropiado es buscar entre las descripciones posibles, aquellas que ésten más cerca de verificar la condición de consistencia. Una de las alternativas más comunes para evaluar esta cercanía consiste en graduar la verificación de la condición.

Para la gradación de la condición de consistencia proponemos una restricción estricta en el espacio de soluciones parciales de segundo orden, de manera que establecemos que descripciones son admisibles y que descripciones no lo son para la representación de un concepto. Esta separación en el espacio de soluciones se realiza teniendo en cuenta el nivel de distorsión que es admisible en una descripción.

Definición 2.11 *Definimos el conjunto de descripciones admisibles como solución parcial de segundo orden al conjunto Δ^k, tal que*

$$\Delta^k = \{R_B(A) \mid n^-(R_B(A)) < kn^+(R_B(A)) \text{ y } n^+(R_B(A)) > 0\}$$

donde $n^+(R_B(A))$ y $n^-(R_B(A))$ son el número de ejemplos positivos y negativos a la regla $R_B(A)$ y $k \in]0,1]$.

En la anterior definición, inicialmente propuesta en [51], el parámetro k establece el máximo porcentaje de ruido que es admitido para considerar a la descripción como una posible solución. En este sentido, el concepto de ruido se aleja de su descripción física como un pequeña distorsión en los datos y se generaliza a todos aquellos ejemplos que existen en el conjunto de entrenamiento que no permiten que una determinada descripción verifique la condición de consistencia. Llamaremos a k, **parámetro de consistencia** porque permite relajar la condición de consistencia variando dicho parámetro y por consiguiente, consideramos que cualquiera de las descripciones existentes en dicho conjunto son una solución parcial de segundo orden.

Además, realizamos un definición especial del conjunto de soluciones que satisfacen completamente la condición de consistencia que denominamos como

$$\Delta^0 = \{R_B(A) \mid n^-(R_B(A)) = 0 \text{ y } n^+(R_B(A)) > 0\}.$$

La siguiente proposición pone de manifiesto la relación existente entre dos espacios de soluciones admisibles.

Proposición 2.1 *Sea $R_B(A) \in \Delta$ una regla del espacio de posibles soluciones, y sean k_1 y k_2 dos parámetros de consistencia que verifican que $k_1 \leq k_2$, entonces*

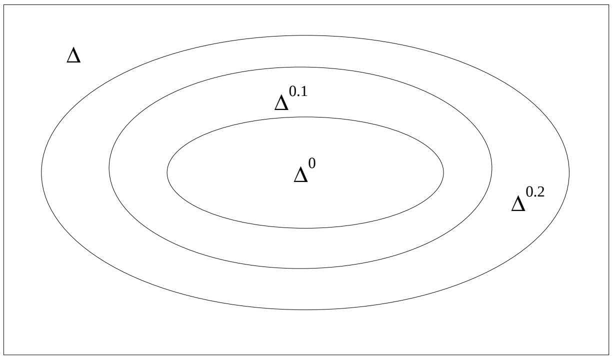


Figura 2.2: Estructura del espacio de soluciones parciales

$$Si\ R_B(A) \in \Delta^{k_1} \Rightarrow R_B(A) \in \Delta^{k_2}$$

Demostración. *Si $R_B(A) \in \Delta^{k_1}$ con $k_1 \neq 0$, entonces se verifica que $n^+(R_B(A)) > 0$ y que $n^-(R_B(A)) < k_1 n^+(R_B(A))$.*

Por ser $k_2 > k_1$, se satisface que $k_1 n^+(R_B(A)) < k_2 n^+(R_B(A))$.

Por tanto, $n^-(R_B(A)) < k_1 n^+(R_B(A)) < k_2 n^+(R_B(A))$, lo que implica que

$$R_B(A) \in \Delta^{k_2}$$

Proposición 2.2 *El conjunto de descripciones que satisfacen completamente la condición de consistencia es un subconjunto de cualquier espacio de soluciones parciales Δ^k, es decir,*

$$\Delta^0 \subseteq \Delta^k\ ,\ \forall k \in]0,1].$$

La propiedad anterior refleja que el espacio de soluciones Δ mantiene una estructura que relaciona cualesquiera dos subespacios de soluciones parciales a través de la operación $\subseteq$, de manera que dicho espacio tiene una estructura semejante a la propuesta en la figura 2.2. donde cualquier restricción Δ^k contiene las descripciones que satisfacen completamente la condición de consistencia.

El conjunto Δ^k presenta además otra propiedad que dota a dichas restricciones sobre el espacio global de soluciones parciales de consistencia interna. Antes de la presentación de dicha propiedad necesitamos el siguiente lema:

Lema 2.1 *Sean $R_{B_1}(A)$ y $R_{B_2}(A)$ dos reglas con $B_1 \neq B_2$, entonces*

$$n^-(R_{B_1}(A)) \geq n^+(R_{B_2}(A))$$

Demostración. *Usando la medida de adaptación tenemos que*

$$U(e,\overline{B_1}) = \frac{sup_{b \in \overline{B_1}} \sigma(c(e),b)}{\sigma(c(e),F)} \geq \frac{\sigma(c(e),B_2)}{\sigma(c(e),F)} = U(e,B_2)$$

Por tanto,

$$n^-(R_{B_1}(A)) = \sum_{e \in E} U(e,A) * U(e,\overline{B_1}) \geq \sum_{e \in E} U(e,A) * U(e,B_2) = n^+(R_{B_2}(A))$$

Proposición 2.3 *Si la regla $R_{B_1}(A) \in \Delta^k$ entonces la regla $R_{B_2} \notin \Delta^k$, $\forall B_2 \in F$, tal que $B_2 \neq B_1$.*

Demostración. *Supongamos que ambas reglas $R_{B_1}(A)$ y $R_{B_2}(A)$ pertenecen a Δ^k. Por tanto, ambas reglas satisfacen que:*

$$n^-(R_{B_1}(A)) < kn^+(R_{B_1}(A))$$

$$n^-(R_{B_2}(A)) < kn^+(R_{B_2}(A))$$

Usando en estas expresiones el lema anterior tenemos:

$$n^+(R_{B_2}(A)) \leq n^-(R_{B_1}(A)) < kn^+(R_{B_1}(A))$$

$$n^+(R_{B_1}(A)) \leq n^-(R_{B_2}(A)) < kn^+(R_{B_2}(A))$$

Por consiguiente se obtiene que

$$n^+(R_{B_2}(A)) < kn^+(R_{B_1}(A)) < k^2 n^+(R_{B_2}(A))$$

y estas expresión implica que

$$k^2 > 1$$

ya que $n^+(R_{B_2}(A)) > 0$, y este resultado es contradictorio con el rango en el que se mueve el parámetro $k \in [0,1]$.

La proposición anterior, nos asegura que si una descripción es admisible para representar un concepto, esa misma descripción es inadmisible para el resto de los conceptos. Por consiguiente, en el conjunto Δ^k no existen reglas contradictorias, para cualquier valor que tome el parámetro k.

A pesar de que la anterior restricción sobre el espacio de soluciones, permite resolver el problema de la ausencia de descripciones completamente consistentes, y por tanto, es aplicable a conjuntos de ejemplos con ruido presenta el siguiente inconveniente que ilustramos con el siguiente ejemplo:

Ejemplo 2.3 *Sean r_1 y r_2 dos reglas que pertenecen al espacio Δ^k con $k > 0$. Por tanto, ambas reglas son válidas como solución parcial. Supongamos que para un $k_1 \leq k$, se verifica que $r_1 \in \Delta^{k_1}$ pero $r_2 \notin \Delta^{k_1}$. Bajo esta situación la regla r_1 esta más cerca de satisfacer la condición de consistencia que la descripción r_2, sin embargo, ambas soluciones en el espacio Δ^k son igualmente deseables.*

La situación expuesta en el ejemplo, pone de manifiesto que además de discriminar entre soluciones admisibles y no admisibles, es necesario también discriminar dentro de las soluciones admisibles la cercanía a la verificación de la condición de consistencia. Con este planteamiento, proponemos una definición graduada de la condición de consistencia en base a dos párametros k_1 y k_2, de manera que si la regla pertenece a Δ^{k_1} la nueva condición se satisface a grado uno, si la regla no pertenece a Δ^{k_2} la condición se satisface a grado cero y entre ambas situaciones se establece el grado de satisfacción como una variación lineal entre ambos parámetros.

Definición 2.12 *Definimos el grado en el cuál una descripción $R_B(A)$ satisface la condición de consistencia débil como*

$$\Gamma_{k_1k_2}(R_B(A)) = \begin{cases} 1 & si\ R_B(A) \in \Delta^{k_1} \\ \frac{k_2 n^+(R_B(A)) - n^-(R_B(A))}{n^+(R_B(A))(k_2-k_1)} & si\ R_B(A) \notin \Delta^{k_1}\ y\ R_B(A) \in \Delta^{k_2} \\ 0 & en\ otro\ caso \end{cases}$$

donde $k_1, k_2 \in [0,1]$ y $k_1 < k_2$, $n^+(R_B(A))$, es el número de ejemplos positivos y $n^-(R_B(A))$ es el número de ejemplos negativos a la regla $R_B(A)$.

Los parámetros k_1 y k_2 se pueden interpretar como umbrales de ruido. El parámetro k_1 fija el nivel máximo de ruido admisible para considerar que una regla satisface completamente la condición de consistencia débil. En el otro extremo se situa el parámetro k_2, fijando el nivel mínimo de ruido a partir del cual una regla no se debe considerar como solución parcial de segundo orden.

Esta nueva definición graduada para la evaluación de la consistencia permite resolver el inconveniente puesto de manifiesto en el ejemplo anterior, pero es interesante comprobar si esta definición mantiene las buenas propiedades que presentan los conjuntos Δ^k descritos anteriormente. El lema siguiente nos sirve de base para plantear las propiedades de la condición de consistencia débil.

Lema 2.2 *Las reglas que son completamente consistentes satisfacen a grado uno la condición de consistencia débil.*

Demostración. *Las reglas que satisfacen a grado uno la condición de consistencia débil, son aquellas que pertenecen al conjunto de soluciones admisibles Δ^{k_1}. Aplicando el corolario 2.2, tenemos que el conjunto de reglas que satisfacen completamente la condición de consistencia Δ^0 son un subconjunto de cualquier Δ^k para cualquier $k \in [0,1]$. Por consiguiente, también son un subconjunto en especial de Δ^{k_1}.*

Proposición 2.4 *Dada una descripción A, existe un único concepto B_1 para el cuál la regla $R_{B_1}(A)$ tiene un grado de consistencia mayor estrictamente que cero, es decir, $\Gamma_{k_1k_2}(R_{B_1}(A)) > 0$ y $\Gamma_{k_1k_2}(R_{B_2}(A)) = 0$, $\forall B_2 \neq B_1$.*

Demostración. *Supongamos que* $\Gamma_{k_1k_2}(R_{B_1}(A)) > 0$. *Aplicando la definición de la condición de consistencia débil sabemos que* $R_{B_1}(A) \in \Delta^{k_2}$.

De la proposición 2.3 se deduce que si $R_{B_1}(A) \in \Delta^{k_2}$ *entonces* $R_{B_2}(A) \notin \Delta^{k_2}$ $\forall B_2 \neq B_1$.

Si $R_{B_2}(A) \notin \Delta^{k_2}$, *aplicando la definición de grado de satisfacción de la condición de consistencia débil, sabemos que* $\Gamma_{k_1k_2}(R_{B_2}(A)) = 0$, $\forall B_2 \neq B_1$.

En base a los lemas y proposiciones anteriores, se demuestra que la condición de consistencia débil, mantiene las propiedades de los conjuntos Δ^k en el sentido de que el conjunto de descripciones que son completamente consistentes tienen grado uno de consistencia débil y dada una descripción A existe un único concepto B para el cuál la regla $R_B(A)$ tiene un grado de consistencia mayor estrictamente que cero, es decir, no existen en el espacio de búsqueda reglas contradictorias.

2.2.4.1 Grado de completitud

Como ya se ha dicho en esta memoria, las soluciones parciales de segundo orden deben satisfacer la condición de consistencia. Este requisito que es necesario para mantener la verificación de dicha condición no parece suficiente si queremos extraer un conocimiento que aporte una comprensión del sistema que se trata de aprender. Supongamos que para cada ejemplo del conjunto de entrenamiento se construye una regla. Si suponemos además que no existen inconsistencias en el conjunto de ejemplos, cada descripción satisface la condición de consistencia, y el conjunto de todas las descripciones verifica también la condición de completitud. Por consiguiente este conjunto de descripciones es una solución parcial de primer orden. Llegados a este punto, cabe preguntarse ¿Se ha extraido algún conocimiento nuevo sobre el conjunto de ejemplos? En este sentido, parece lógico pedir que la solución parcial de segundo orden satisfaga alguna otra propiedad.

Una de las propiedades que pueden ser exigibles a un algoritmo de aprendizaje es que las soluciones parciales de primer orden contenga el mínimo número de descripciones posibles sobre el concepto que se desea aprender. Si extendemos esta idea a las soluciones de segundo orden podemos establecer que la mejor descripción es aquella que verificando la condición de consistencia este *más cerca* de satisfacer la condición de completitud. ¿Cómo evaluar la cercanía a la verificación de esta condición? La condición de completitud se satisface cuando todos los ejemplos del conjunto de entrenamiento que describen un determinado concepto se representan por al menos una de las descripciones obtenidas para dicho concepto. En base a esta característica definimos el concepto de grado de completitud para una regla.

Definición 2.13 *El grado de completitud de una regla* $R_B(A)$ *se define como*

$$\Lambda(R_B(A)) = \frac{n^+(R_B(A))}{n_B}$$

donde $n^+(R_B(A))$ es el número de ejemplos del conjunto de entrenamiento del concepto B que son cubiertos por la regla $R_B(A)$ y n_B es el número de ejemplos existentes en el conjunto de entrenamiento que describen el concepto B,

De la anterior definición se deduce que si $\Lambda(R_B(A)) = 1$ entonces la regla $R_B(A)$ satisface la condición de completitud, y por tanto, cuanto más cercano a 1 este este valor, más cerca se encuentra la descripción de satisfacer esta condición.

Por consiguiente, parece hasta el momento, que una regla que sea solución parcial de segundo orden debe satisfacer en alto grado tanto la condición de consistencia débil como el grado de completitud. Por consiguiente, si ambos factores son relevantes para establecer una medida de la bondad de la regla, es necesario articular algún criterio que permita combinar ambos factores. En el siguiente apartado estudiamos diferentes combinaciones de estos factores.

2.2.4.2 Combinación de consistencia y completitud

De acuerdo con las definiciones anteriores, debemos adaptar el criterio utilizado para establecer una solución parcial de segundo nivel. Este criterio debe tener en cuenta tanto la maximización del grado de consistencia débil, como el grado de completitud, pero ¿cómo combinarlos? Podemos aplicar varias posibilidades:

a) Establecer una evaluación funcional lexicográfica (EFL). Una EFL se define como secuencia de criterios:

$$EFL : (c_1, c_2, \ldots, c_s)$$

donde c_1 es un criterio elemental.

Dado un conjunto de descripciones $D = \{d_1, d_2, \ldots, d_m\}$ y una secuencia ordenada de criterios de evaluación $(c_1, c_2, \ldots, c_s)$, el EFL determina la descripción más preferible de la siguiente forma: En un primer paso, todas las descripciones se evaluan utilizando el criterio c_1. Todas aquellas descripciones que alcanzan el mismo máximo valor de satisfacción del criterio son retiradas del conjunto D. Las descripciones retiradas se evaluan según el criterio c_2 y se reducen de forma similar. El proceso continua hasta que únicamente una descripción es retirada y esta descripción es considerada como la mejor.

Por tanto, existen dos posibles evaluaciones funcionales lexicográficas considerando el grado de completitud y la condición de consistencia débil como dos criterios.

a1) **Considerar el orden** $(\Gamma_{k_1,k_2}, \Lambda)$ Esta ordenación provoca que se prefieran descripciones que satisfagan en un alto grado la condición de consistencia débil, frente a su grado de completitud. Este criterio de preferencia conserva la idea original, es decir, lo importante es que la regla sea consistente. Este criterio está sometido al mismo inconveniente que se describió anteriormente, se tenderá a seleccionar reglas que esten soportadas por pocos ejemplos y por consiguiente, serán necesarias muchas descripciones para representar un concepto.

a2) **Considerar el orden** $(\Lambda, \Gamma_{k_1,k_2})$ Esta ordenación prefiere descripciones que satisfagan la completitud frente a la consistencia. La aplicación de este EFL puede llevar a la obtención de pocas descripciones para los conceptos, pero excesivamente ruidosas.

b) Maximizar una combinación de ambas valoraciones mediante una función de agregación de información, ya que lo que se desea es una descripción que satisfaga en un alto grado la completitud y en un alto grado la condición de consistencia débil. Si utilizamos como función de agregación el producto, la mejor descripción será aquella que haga máximo esta combinación, es decir, aquella que verifique

$$\max_{A \in D} \{\Lambda(R_B(A)) \times \Gamma_{k_1 k_2}(R_B(A))\}$$

donde $D = P(D_1) \times P(D_2) \times \ldots \times P(D_n)$.

La combinación de ambos valores permite obtener descripciones que simultaneamente son completas y consistentes en el máximo grado posible. Además, utilizamos la t-norma producto, por ser la combinación a través de este operador muy interactiva frente a otras posibles propuestas.

2.2.4.3 Simplicidad y comprensibilidad

Utilizando el criterio anterior podemos encontrar varias soluciones que obtengan la máxima valoración. El módulo de selección de reglas tiene que decidir cual de ellas se devolverá como la mejor descripción para la clase que se está aprendiendo. Un criterio definido para resolver estas situaciones podría decir que cualquiera de ellas es una buena solución. Sin embargo, uno de los objetivos propuestos en la definición de este algoritmo es encontrar reglas que esten soportadas por los ejemplos, y sean fáciles de comprender. En este sentido, el criterio de decisión ante estas situaciones se debe establecer en función de la simplicidad de la regla. A continuación pasamos a formalizar estos criterios.

Definición 2.14 *Dada una regla $R_B(A)$ tal que $A = (A_1, A_2, \ldots, A_n)$ y cada $A_i \subseteq D_i$ representa el valor asignado a un variable X_i. Diremos que la variable X_i es irrelevante para describir el concepto B si A_i es la disyunción de todos los valores del dominio D_i, es decir,*

$$A_i = D_i$$

Definiremos además como $IRREV(A)$ al número de variables de la regla $R_B(A)$ que satisfacen la condición de irrelevancia y $REV(A)$ al número de variables de la regla $R_B(A)$ que no satisfacen la anterior condición.

Sobre la definición anterior debemos especificar una aclaración: cuando una variable X_i toma como valor A_i todo su dominio de definición D_i podemos eliminarla de la descripción de la regla gracias al concepto de adaptación definido en la sección 2.2.2. El uso de la normalización permite que la valoración asignada a cualquier valor sobre el conjunto completo de estiquetas sea

$$Poss(D_i|v) = 1 \ \forall v$$

y además

$$a * 1 = a \text{ para cualquier t-norma } *.$$

Podemos considerar que una regla es más simple cuanto mayor es el número de variables irrelevantes, puesto que esto conlleva la obtención de descripciones más generales de los conceptos. Por tanto, a partir del concepto de irrelevancia podemos definir el concepto de simplicidad.

Definición 2.15 *Dada una regla $R_B(A)$ tal que $A = (A_1, A_2, \ldots, A_n)$, diremos que el grado de simplicidad asociado a esta regla es:*

$$S(R_B(A)) = \frac{IRREV(A)}{n}$$

Cuando tratamos de extraer el conocimiento necesario para comprender el comportamiento de un sistema, podemos querer que dicho conocimiento represente dos tipos descripciones diferente con respecto al concepto:

a) Descripciones que permiten distinguir el concepto del resto de los que se desean aprender. Este tipo de descripciones se llaman **descripciones discriminantes**.

b) Descripciones que permitar conocer todas las variables que tienen influencia para describir este concepto. Este tipo de descripciones se llaman **descripciones características**.

Estos dos tipos de descripciones se asemejan a las propuestas en [115] y descritas en el primer capítulo, aunque nosotros proponemos un sentido distinto para las descripciones características. Según [115] las descripciones características únicamente deben verifican la condición de completitud, sin embargo, desde nuestra interpretación, estas descripciones deben verificar también la condición de consistencia.

Por tanto, de los anteriores tipos de descripciones, las que buscaremos serán las descripciones discriminantes minimales, es decir las que implican el menor número de variables relevantes y las descripciones características maximales, es decir, las descripciones consistentes que implican el menor número posible de variables irrelevantes.

Utilizando el concepto de simplicidad, podemos obtener ambos tipos de descripciones, de manera, que las descripciones discriminantes minimales se obtienen maximizando el valor de la simplicidad, mientras que las descripciones características máximales se pueden obtener mediante la minimización del concepto de simplicidad.

Por consiguiente, un algoritmo de aprendizaje que tenga en cuenta el concepto de simplicidad puede ser aplicable para obtener estos tipos de descripciones.

Una vez que hemos propuesto el concepto de simplidad de una regla, vamos a definir el concepto de comprensibilidad. Este concepto trata de evaluar la simplicidad de los valores asignados a una determinada variable en una regla. Veamos de forma intuitiva su comportamiento con el siguiente ejemplo:

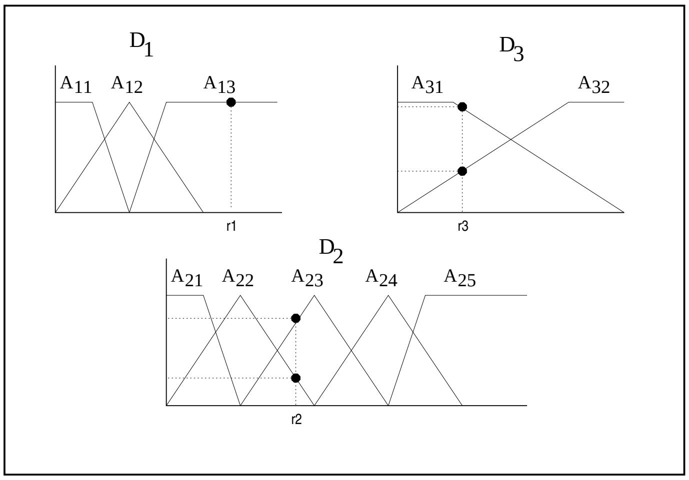


Figura 2.3: Dominio de las variables X_1, X_2, X_3

Ejemplo 2.4 *Sea X_2 una variable cuyo dominio ordenado D_2 es el definido en la figura 2.3. Supongamos los dos siguientes valores para la variable X_2: $X_2 = \{A_{33}, A_{34}, A_{35}\}$ y $X_2 = \{A_{33}, A_{35}\}$. El primer valor se puede expresar en la descripción de una regla como*

$$\ldots \; y \; X_2 \; es \; \{A_{33}, A_{34}, A_{35}\} \; y \; \ldots$$

pero, podemos reformularlo como

$$\ldots \; y \; X_2 \; es \; mayor \; o \; igual \; que \; A_{33} \; y \; \ldots$$

mientras el segundo valor sólo puede ser expresado como

$$\ldots \; y \; X_2 \; es \; \{A_{33}, A_{35}\} \; y \; \ldots$$

Si tuviéramos que decidir entre estos valores en una situación de igual valoración, podemos observar que el primer valor presenta una descripción más comprensible desde un punto de vista humano.

Por tanto, si los valores que toma una determinada variable siguen una secuencia continua sobre un dominio ordenado, el valor de dicha variable, se puede interpretar como la envolvente convexa de dichos valores. Este hecho, implica que los valores asociados a las variables, puedan ser agrupados aumentando la comprensibilidad de esta regla. En este sentido, definimos el concepto de valor agrupado para una variable.

Definición 2.16 *Dado un valor $A_i \subseteq D_i$ para una variable X_i, y siendo D_i un dominio ordenado asociado a la variable X_i, decimos que la variable X_i toma un valor estable A_i, sii el valor A_i esta constituido por una secuencia de valores primitivos adyacentes en el dominio D_i.*

Además, dada una regla $R_B(A)$ tal que $A = (A_1, A_2, \ldots, A_n)$ y cada $A_i \subseteq D_i$ representa el valor de un variable X_i, definimos como $EST(A)$ el número de variables X_i que satisfacen:

i) X_i tiene un dominio asociado ordenado D_i.

ii) El valor A_i que toma D_i es un valor estable.

A partir de esta definición, podemos establecer una valoración sobre la comprensibilidad de una regla:

Definición 2.17 *Dada una regla $R_B(A)$ definimos el grado de comprensibilidad de una regla como:*

$$CP(R_B(A)) = \frac{1 + EST(A)}{1 + p}$$

donde p es el número de variables que tienen un dominio asociado ordenado.

Con la medida propuesta establecemos un criterio de preferencia en el proceso de búsqueda, de manera que el sistema tenderá a obtener soluciones parciales de segundo orden que maximizen lo más posible la comprensibilidad de las asignaciones a las variables con dominio ordenado.

Teniendo en cuenta las definiciones de simplicidad y de comprensibilidad, cada posible descripción puede ser evaluada atendiendo a tres factores. Por tanto, utilizaremos una EFL, donde consideraremos como primer criterio la satisfacción de la combinación de consistencia y completitud, como segundo criterio la simplicidad de las misma, que podrá ser valorada, como la maximización o minimización del concepto de simplicidad y por último se establecerá el criterio de comprensibilidad por ser el menos importante de los tres.

Por consiguiente, dada una regla $R_B(d)$, definimos la valoración de la regla $R_B(d)$ como la función

$$\rho : \Delta \longrightarrow [0, 1]^3$$

$$\rho(R_B(d)) = (\Lambda(R_B(d)) \times \Gamma_{k_1k_2}(R_B(d)), S(R_B(d)), CP(R_B(d))).$$

Usaremos una evaluación funcional lexicográfica que para problemas que requieran descripciones discriminantes usará los criterios

$$EFL(\rho(R_B(A))) = (\max_{A\in D}\{ \Lambda(R_B(A)) \times \Gamma_{k_1k_2}(R_B(A))\}, \max_{A\in D}\{S(R_B(A))\}, \max_{A\in D}\{CP(R_B(A))\})$$

y para descripciones características usará

$$EFL(\rho(R_B(A))) = (\max_{A\in D}\{ \Lambda(R_B(A))\times\Gamma_{k_1k_2}(R_B(A))\}, \max_{A\in D}\{1-S(R_B(A))\}, \max_{A\in D}\{CP(R_B(A))\}).$$

Asumiremos a partir de este momento y para el resto de la memoria que utilizamos la formulación que permite encontrar descripciones discriminantes, a menos que explícitamente se diga lo contrario.

2.2.5 El algoritmo genético de SLAVE

Hasta el momento hemos descrito el criterio que utiliza SLAVE para determinar la mejor solución parcial de segundo orden. Como dijimos anteriormente, SLAVE es un sistema de aprendizaje basado en el modelo iterativo genético. Así, partiendo del esquema 2.2 usaremos un algoritmo genético en el paso 2.1. que nos permita encontrar una descripción que sea solución parcial de segundo orden. Por tanto, la arquitectura básica de SLAVE es la que se muestra en el esquema 2.3.

Vamos ahora a detallar cada una de las componentes básicas del algoritmo genético del apartado 2.1.:

- Representación de los elementos de la población.
- Generación de la población inicial.
- Función de evaluación.
- Los operadores genéticos.
- Condición de terminación.

2.2.5.1 Representación de un elemento de la población

Como se comento en el primer capítulo, la representación de la población en los algoritmos genéticos es un elemento clave en la operatividad del mismo. En nuestro caso, si la base de datos contiene n variables antecedente $X_1, \ldots, X_n$ y cada una de ellas tiene un dominio difuso asociado D_i con m_i etiquetas, entonces usamos el siguiente metodo de codificación

Esquema 2.3 *Arquitectura de SLAVE*

0. Sean $X_1, X_2, \ldots, X_n$ las variables predictivas que son incluidas para aprender el comportamiento de un determinado sistema. Sea Y la variable de clasificación definida sobre un dominio discreto de m conceptos. Además, sea E un conjunto con k ejemplos que describen el comportamiento en el pasado de dicho sistema. Sean $GLOBAL$ y $PARCIAL$ dos conjuntos de reglas que inicialmente estan vacíos.

1. Seleccionar un concepto B que aún no se ha aprendido.

2. Mientras $PARCIAL$ no represente suficientemente el concepto B hacer:

    2.1. Utilizar un algoritmo genético para encontrar una regla $R_B(d)$ que satisfaga que
    $$\rho(R_B(d)) = EFL(\rho(R_B(A)))$$
    sobre un subconjunto de los ejemplos de entrenamiento E, y $A \subseteq D$.

    2.2. Añadir d al conjunto de descripciones $PARCIAL$.
    $$PARCIAL = PARCIAL \bigcup d$$

    2.3 Penalizar las descripciones de $PARCIAL$ para que no vuelvan a ser selecionadas.

3. Añadir las nuevas descripciones al conjunto $GLOBAL$.
$$GLOBAL = GLOBAL \bigcup PARCIAL$$

4. Eliminar las descripciones de $PARCIAL$; $PARCIAL = \emptyset$.

5. Si existen más conceptos a aprender, volver al paso 1., en otro caso devolver $GLOBAL$ como solución al problema de aprendizaje.

binaria para los elementos de Δ como un vector de $m_1 + \ldots + m_n$ de componentes cero-uno, tal que

$$componente(m_1 + \ldots + m_{r-1} + s) = \begin{cases} 1 & \text{si el s-ésimo elemento del dominio } D_r \\ & \text{es un valor de la variable } X_r \\ 0 & \text{en otro caso} \end{cases}$$

Ejemplo **2.5** *Supongamos las siguientes tres variable X_1, X_2 y X_3, tal que sus dominios difusos asociados son los mostrados en la figura 2.3. En este caso, el vector 1010010111 representa el siguiente antecedente*

$$X_1 \ es \ \{A_{11}, A_{13}\} \ y \ X_2 \ es \ \{A_{23}, A_{25}\} \ y \ X_3 \ es \ \{A_{31}, A_{32}\}.$$

Como X_3 toma todos los valores de su dominio D_3, el antecedente anterior es equivalente a

$$X_1 \ es \ \{A_{11}, A_{13}\} \ y \ X_2 \ es \ \{A_{23}, A_{25}\}.$$

Este método de codificación binaria, donde la información representa la inclusión o no de los valores de los dominios de las variables, se utilizó en la primera versión de SLAVE [51] y se ha empleado en diversos sistemas de aprendizaje que trabajan con algoritmos genéticos. Por ejemplo, [157] utiliza la misma codificación de los elementos de la población, y en los sistemas GABIL [31] y GIL [85] una codificación semejante aunque adaptada al hecho de ser sistemas que trabajan con el modelo Pittsburgh.

Por otro lado, este tipo de codificación ha sido ampliamente utilizado por este tipo de sistemas de aprendizaje, ya que esta codificación es una forma simple y natural de recoger el lenguaje DNF modificado, que es el que suelen utilizar la mayoría de estos sistemas.

2.2.5.2 Generación de la población inicial

La población inicial es el punto de partida del proceso evolutivo. En el capítulo primero destacamos que la forma más habitual de generar esta primera población consiste en la construcción completamente aleatoria de los individuos. Sin embargo, desde el punto de vista de los algoritmos de búsqueda, cuanto más cercano se encuentre el punto de partida de las soluciones a alcanzar, menor debe ser el costo para encontrar dichas soluciones. En este sentido, en el algoritmo genético de SLAVE, la generación de la población inicial consiste en seleccionar ejemplos de forma aleatoria del conjunto de ejemplos de la clase que ha de ser aprendida y obtener para cada uno de ellos el antecedente más específico que mejor lo recubra, es decir, cada variable tomará como valor solamente una etiqueta, y esta etiqueta será aquella que presenta mayor grado de pertenencia con su correspondiente componente en el ejemplo. A continuación mostramos un ejemplo para construir en base a un ejemplo el antecedente más específico que mejor lo recubre.

$\boxed{\textbf{Ejemplo}}$ **2.6** *Sean X_1, X_2 y X_3 variables cuyos dominios difusos asociados son los descritos en la figura 2.3, y sea (r1,r2,r3) un ejemplo seleccionado de forma aleatoria del conjunto de ejemplos para una determinada clase. El antecedente más específico que mejor lo representa es:*

$$X_1 \ es \ A_{13} \ and \ X_2 \ es \ A_{23} \ and \ X_3 \ es \ A_{21}$$

cuyo representación binaria es 0010010010.

Cuando existen varias posibilidades, se selecciona aleatoriamente una de ellas.

Esta selección de la población inicial puede considerarse similar a la usada por los algoritmos de la familia AQ [109, 114], ya que para la generación del antecedente inicial para una clase, tomamos ejemplos de esta clase en el conjunto de entrenamiento como punto de partida, y el algoritmo genético puede considerarse como un proceso de generalización sobre los antecedentes seleccionados.

2.2.5.3 Función de evaluación (FITNESS)

El algoritmo genético de SLAVE presenta como entradas, un conjunto de ejemplos de entrenamiento, un conjunto de variables antecedente y una clase fija de la variable de clasificación. El objetivo del algoritmo genético consisten en encontrar la mejor solución parcial de segundo orden para esta clase, es decir, la mejor asignación de valores a las variables que permiten describir a dicha clase. Anteriormente, se han definido los criterios para valorar la bondad de las descripciones, que vienen determinadas por la aplicación de una evaluación funcional lexicográfica sobre la consistencia, completitud, simplicidad y comprensibilidad de las mismas.

Por tanto, la valoración de una descripción A se define como:

$$fitness(R_B(A)) = \rho(R_B(A))$$

sobre el conjunto de ejemplos de entrenamiento y el objetivo del algoritmo genético consiste en

$$EFL(fitness(R_B(A)))$$

con $A \in D$.

2.2.5.4 Los operadores genéticos

Para generar las nuevas poblaciones se usan un conjunto de operadores genéticos, que podemos clasificar en dos tipos: operadores genéticos básicos y operadores genéticos definidos para el problema del aprendizaje. A continuación pasamos a describir cada uno de ellos.

Operados genéticos básicos

- **Operador de selección lineal con elitismo**. Este es un modelo de selección que ordena los elementos de la población atendiendo a su valoración y asigna una probabilidad de selección a cada posición de la población. La selección de los individuos que pasarán a formar parte de la siguiente población se realiza mediante un muestro aleatorio simple. A este operador se le asocia la estrategia adicional que se conoce por elitismo ([29]), el cual asegura que el mejor individuo de una generación estará presente en la siguiente.

- **Operador de mutación.** Este es el clásico operador de mutación que cambia un gen de un elemento en la población con una cierta probabilidad.

- **Operador de cruce sobre dos puntos**. Este tipo de cruce fija dos puntos de corte entre dos elementos de la población e intercambia el segmento central como muestra la figura 2.4. Este operador de cruce es una variante del operador de cruce simple ([48, 72]).

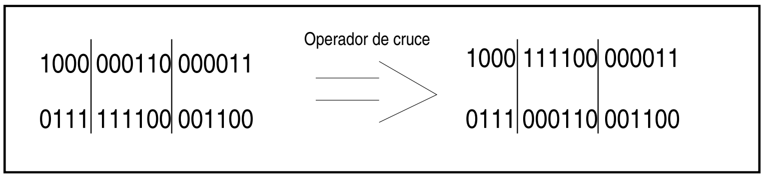


Figura 2.4: Operador de cruce sobre dos puntos

Operadores genéticos especificos para este problema

- **Operador de rotación**([55]). Este operador es una versión modificada del tradicional operador de inversión que provoca el incremento del nivel de diversidad en la población. Este operador toma un punto de corte en un elemento de la población e intercambia los dos segmentos como muestra la figura 2.5.

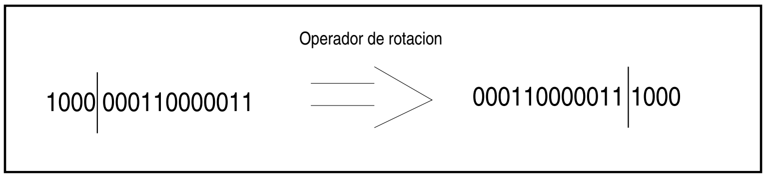


Figura 2.5: Operador de rotación

- **Operador de generalización**. Este operador trata de simplificar y aumentar la comprensibilidad de las reglas devueltas por el proceso genético. El operador generaliza las reglas de la población utilizando la misma idea que fué descrita para el criterio de comprensibilidad. Por consiguiente, este operador favorece la modificación en los individuos de la población para mejorar su comprensibilidad. Al igual que sucede con el criterio de compresibilidad, este operador afecta únicamente a aquellas variables que tiene un dominio ordenado asociado.

Previo a la descripción del operador, es necesario introducir algunos conceptos. En la definición 2.16 se establece el concepto de variable estable. Dicha definición se puede caracterizar en base a la representación genética propuesta de la siguiente manera:

Definición 2.18 *Diremos que una variable antecedente con dominio asociado ordenado es estable, si en la representación binaria de su valor aparece un única secuencia continua de unos.*

De igual forma, y por oposición a este concepto, definimos los dos siguientes conceptos:

Definición 2.19 *Diremos que una variable es inestable, si en la representación binaria de su valor existe más de una secuencia continua de unos.*

Definición 2.20 *Dada la representación genética de un valor de una variable inestable, llamaremos zona inestable a cada secuencia continuo de ceros.*

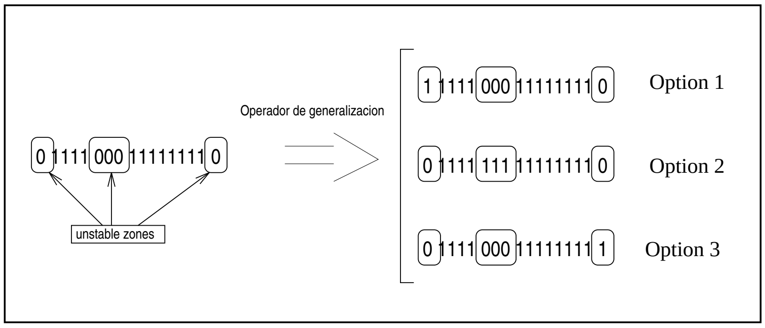


Figura 2.6: Operador de Generalización

El operador de generalización trata de obtener variables estables eliminando sus zonas inestables. Este operador se puede describir de forma algorítmica como indica el esquema 2.4.

La figura 2.6 muestra las zonas inestable del valor de una variable y además se muestran las tres posibles opciones de generalización.

- **Operadores AND y OR**. Son dos operadores de cruce especiales que tratan de representar procesos de generalización y especificación entre los individuos de la población. Al igual que en el operador de cruce definido anteriormente, estos dos operadores seleccionan dos individuos y establecen dos puntos de corte. Los segmentos centrales de ambos individuos se combinan bit a bit mediante la operación lógica AND en un caso, y mediante la operación lógica OR en el otro caso.

Esquema 2.4 *Enfoque iterativo con dos niveles*

1. Se selecciona aleatoriamente una variable con dominio asociado ordenado de un elemento de la población.
2. Si esta variable es inestable hacer lo siguiente
   - 2.1 Detectar sus zonas inestables y seleccionar aleatoriamente una de ellas.
   - 2.2 Cambiar la secuencia de ceros de la zona inestable selecionada por una secuencia de unos.
   - 2.3 Calcular la valoración del nuevo elemento usando la función de valoración. Si su valoración es igual o mejor que la del elemento original, la modificación se mantiene. En otro caso, se restablece el valor original de la variable.

2.2.5.5 Condición de terminación

La forma más habitual de establecer la condición de terminación en los algoritmos genéticos consiste en establecer un número fijo de iteraciones. En nuestro caso, con la idea de realizar búsquedas más adecuadas al tipo de descripciones que se intentan buscar hemos considerado una condición de terminación múltiple. En la implementación de esta condición distinguimos entre el caso en el que estamos aprendiendo la primera regla para una clase y el caso en que al menos ya tenemos alguna regla para describir dicha clase. En el primer caso realizamos una búsqueda más exhaustiva mientras para el segundo caso optamos por relajar el proceso de búsqueda.

El algoritmo genético termina devolviendo la mejor regla de la última población si alguna de las siguientes condiciones se verifica:

- el número de iteraciones es mayor que un límite fijo N_ITER_1,
- la valoración de la mejor regla no mejora durante un número fijo de iteraciones N_ITER_2, donde N_ITER_2 < N_ITER_1, y existen reglas obtenidas para esta clase,
- no hay todavía reglas obtenidas de esta clase, pero la función de valoración se mantiene durante un número fijo de iteraciones (N_ITER_2) y el mejor elemento de la población recubre al menos un ejemplo del conjunto de entrenamiento.

2.2.6 Penalización de reglas

Las subsecciones anteriores presentan definiciones y criterios para obtener soluciones parciales de segundo orden. Sin embargo, para que la agregación de estas soluciones pueda ser considerada como solucion parcial de primer nivel para algún concepto, es necesario establecer un criterio de penalización de las descripciones obtenidas según describe el paso 2.3. del esquema 2.3. Como en dicha sección dijimos, el criterio más extendido para penalizar reglas en este tipo de enfoques consiste en eliminar del conjunto de entrenamiento aquellos ejemplos que apoyan la veracidad de las descripciones del conjunto $PARCIAL$. La implementación de este criterio es muy fácil cuando trabajamos con ejemplos y reglas crisp, sin embargo, cuando trabajamos con ejemplos y reglas difusas, un ejemplo apoya la veracidad de la regla en un cierto grado. Por consiguiente, es necesario incluir una definición que permita determinar cuando un ejemplo apoya la veracidad de una regla en un grado suficiente, de manera que dicha regla sea útil para describir dicho ejemplo. Para definir esta extensión nos apoyamos en la siguiente definición.

Definición 2.21 *Sea E un conjunto de ejemplos y sea B un concepto. Definimos el conjunto de ejemplos de E que describen el concepto B como:*

$$E_B = \{e \in E \mid U(e, B) > 0\}$$

En base a la definición anterior podemos establecer el concepto de λ-recubrimiento

Definición 2.22 *Sea $R_B(A)$ una regla, E un conjunto de ejemplos y $\lambda \in [0,1]$. El λ-recubrimiento de la regla $R_B(A)$ sobre el conjunto E es el λ-corte de $E^+(R_B(A))$. Llamaremos a λ parámetro de recubrimiento.*

A partir de la anterior definición, podemos determinar el conjunto de ejemplos de un determinado concepto que aún permanecen en el conjunto de entrenamiento, y por tanto, a partir de este concepto fijar un criterio de penalización de reglas.

Dado un conjunto $PARCIAL=\{R_B(A^1), R_B(A^2), \ldots, R_B(A^j)\}$ de reglas ya aprendidas para el concepto B y un conjunto de ejemplos E, definimos E_B^j como el subconjunto de ejemplos que describen al concepto B sin considerar los ejemplos que son cubiertos por las reglas existentes en el conjunto $PARCIAL$, es decir,

$$E_B^j = \{e \in E_B \mid e \notin \bigcup_{i=1}^{j} E_\lambda^+(R_B(A^i))\}$$

Siguiendo este criterio de penalización de reglas, comprobar si un conjunto de descripciones constituye una solución parcial de primer orden es tan simple como comprobar que no existe ningún ejemplo del concepto que se trata de aprender en el conjunto de entrenamiento, despues de haber eliminado todos los ejemplos cubiertos por las descripciones ya obtenidas.

Definición 2.23 *Dado un conjunto $PARCIAL=\{R_B(A^1), R_B(A^2), \ldots, R_B(A^j)\}$ de reglas ya aprendidas para el concepto B y un conjunto de ejemplos E, diremos que el conjunto $PARCIAL$ verifica la condición de completitud para el concepto B, sii*

$$|E_B^j| = 0$$

La verificación de esta propiedad por un conjunto de descripciones para un ejemplo implica que dicho conjunto se puede considerar como una solución parcial de primer orden. Pero realmente, la condición anterior solamente es aplicable cuando el conjunto de ejemplos no contiene ni ruido ni inconsistencias. Por consiguiente, se hace necesario, también en este caso, relajar esta condición. A continuación se propone una nueva condición relajada sustituyendo la condición de completitud, cuya verificación indica que el concepto esta suficientemente representado por las descripciones ya obtenida. Esta nueva condición tiene en cuenta dos consideraciones:

1. Todos los ejemplos de un determinado concepto están cubiertos de forma adecuada por alguna de las descripciones obtenidas.
2. Existen ejemplos que no son cubiertos adecuadamente por las descripciones obtenidas, pero no es posible encontrar una nueva descripción útil.

Más formalmente, podemos definir la nueva condición de la siguiente manera:

Definición 2.24 *Dado un conjunto* $PARCIAL=\{R_B(A^1), R_B(A^2), \ldots, R_B(A^j)\}$ *de reglas ya aprendidas para el concepto* B*, y dada una nueva regla* $R_B(A^{j+1})$*, diremos que el conjunto* $PARCIAL$ *representa suficientemente el concepto* B*, si verifica al menos una de las siguientes condiciones:*

a) $|E_B^j| = 0$

b) Si al añadir $R_B(A^{j+1})$ *al conjunto* $PARCIAL$ *se satisface que:*

$$|E_B^j| - |E_B^{j+1}| = 0.$$

En este caso $R_B(A^{j+1})$ *no se incluye en el conjunto* $PARCIAL$*.*

La primera condición determina si el conjunto de reglas obtenido hasta el momento, $PARCIAL$, cumple la condición de completitud. La segunda condición comprueba si la última regla que se ha obtenido permite representar algún ejemplo que no sea cubierto por alguna de las reglas anteriores. Esta comprobación se realiza comparando el número de ejemplos que quedaban por representar, y los que después de incluir la regla aún quedan. Para que esta segunda condición cumpla su cometido correctamente, es necesario que la descripción obtenida sea en realidad la mejor solución parcial de segundo orden que se puede obtener sobre el conjunto de ejemplos que aún permanecen en el conjunto de entrenamiento.

En conclusión, un conjunto de reglas que describen un determinado concepto se considerada como una solución parcial de primer orden si satisface la anterior definición, y por tanto, el paso 2 del esquema 2.3 utiliza esta definición para comprobar si son necesarias más reglas para describir este concepto.

2.3 Mecanismo de resolución de conflictos

El mecanismo de resolución de conflictos trata de dar respuesta a las dificultades que se le presentan a un modelo de inferencia en el caso crisp, para tomar una decisión. Estas dificultades aparecen porque normalmente el algoritmo de aprendizaje extrae conocimiento sobre el comportamiento del sistema, pero no extrae información que le permita determinar como utilizar dicho conocimiento. Este metaconocimiento es importante, ya que permite comprender mejor el sistema que se ha aprendido.

Cuando trabajamos con reglas, es posible que dos o más reglas que describen diferentes conceptos sean aplicables ante una misma entrada al sistema. En este caso, es necesario establecer una norma de actuación para determinar cuál de las posibles reglas es la más adecuada.

Nosotros establecemos un mecanismo de resolución de conflictos basado en la confianza que las reglas nos ofrecen. Esta confianza se establece en función del comportamiento que las mismas presentaban sobre el conjunto de entrenamiento. En este sentido, proponemos una evaluación funcional lexicográfica para la resolución de los conflictos en base a los siguientes criterios:

Criterio 1: Adaptación entre el ejemplo y la regla.

Criterio 2: Confianza que la regla ofreció durante el proceso de aprendizaje.

Criterio 3: Simplicidad de la regla.

En cuanto al primer criterio, este viene determinado por la valoración obtenida de la adaptación entre ejemplo y regla. Con respecto al segundo criterio, la confianza de una regla debe estar relacionada con la capacidad de predicción de la misma. Dicha capacidad se puede medir en función de su comportamiento en la fase de aprendizaje. Con este fin, definimos el concepto de peso de una regla como la relación entre los ejemplos que representó correctamente sobre el conjunto de entrenamiento y todos los ejemplos cubiertos por ella.

Definición 2.25 *Sea E un conjunto de ejemplos y sea $R_B(A)$ una regla. Definimos el peso de la regla $R_B(A)$ como:*

$$w(R_B(A)) = \frac{n^+(R_B(A))}{n(R_B(A))}$$

El tercer criterio, mide la simplicidad de la reglas. Como se ha definido anteriormente en este capítulo, la simplicidad esta relacionada con el número de condiciones que un ejemplo debe verificar para respaldar la veracidad de una regla. Desde un punto de vista objetivo, parece sensato que si dos reglas son aplicables a una misma entrada, y no hay posiblidad de establecer cual de ellas es más adecuada en función de las medidas anteriores, se seleccione aquella que necesita verificar más condiciones. Por tanto, en este último criterio, se considerá la más adecuada, la regla con menor simplicidad entre las aplicables.

2.4 Pruebas experimentales

En esta sección estudiaremos el comportamiento del sistema de aprendizaje SLAVE y la posibilidad de incorporar algunas modificaciones en su implementación. Para este estudio, hemos seleccionado una amplia serie de bases de datos que se describen en el apéndice A. Estas bases de datos pueden considerarse como una extensa representación de las diferentes situaciones con las que se puede encontrar un algoritmo de aprendizaje inductivo, tales como valores perdidos, ruido e inconsistencias, un número elevado de variables antecedente, variable de clasificación crisp y variable de clasificación difusa.

Con el fin de establecer el comportamiento de SLAVE, sus resultados se comparan con los obtenidos por otros algoritmos de aprendizaje que se describen en el apéndice B. Cada uno de estos algoritmos se puede considerar como un representante de diferentes filosofías de aprendizaje.

La estrategia utilizada para la realización de la experimentación es la siguiente: cada base de datos esta dividida en una serie de particiones, donde el número de las mismas viene determinado por las siguientes características:

- Si la base de datos se ha obtenido en un único fichero que contiene toda la información, nosotros hemos procedido a realizar cinco particiones de la misma en dos ficheros de entrenamiento y prueba, donde cada fichero de entrenamiento contiene el 70 % de los ejemplos del fichero original, y el fichero de prueba contiene el 30 % restante.
- Si la base de datos contenía particiones de entrenamiento y prueba que han sido utilizadas por algunos autores para comprobar empíricamente el comportamiento de sus sistemas, hemos conservado para nuestras pruebas dichas particiones.

Sobre cada prueba se han extraído los siguientes parámetros: porcentaje de acierto sobre la partición de entrenamiento y la partición de prueba, número de reglas y tiempo de ejecución en segundos. Los resultados considerados para cada partición son la media sobre cada uno de estos parámetros sobre cinco ejecuciones. En las tablas, se representará para cada base de datos, la media obtenida sobre cada una de las particiones. Por consiguiente, el significado de cada valor de la tabla será el anteriormente definido, a menos que se diga explícitamente otra cosa.

Parámetro	Valor
Número máximo de iteraciones	300
Número máximo de iteraciones sin mejora	100
Número de individuos en la población	20
Probabilidad de mutación	0.01
Probabilidad de cruce	0.1
Parámetro de consistencia k_1	0
Parámetro de consistencia k_2	1
Parámetro de recubrimiento λ	0.8

Tabla 2.2: Parámetros fijados para la ejecución de SLAVE

Bases de datos	Entrenamiento	Prueba	N.Reglas	Tiempo
IRIS	97.63	95.01	5.6	85.2
MONK1	96.13	95.0	6	41
MONK2	90.06	69.95	12.2	97
MONK3	98.20	89.16	7.4	47
INFARTO	98.29	79.68	7.68	252.4
WINE	99.61	95.54	8.2	839.8
PIMA	79.89	79.99	20.8	1818
IONOSPHERE	99.65	81.52	44.6	15112
SOYBEAN	94.42	93.34	41	12314

Tabla 2.3: Resultados de la propuesta original para caso crisp

En este primer capítulo, estudiaremos experimentalmente los siguiente puntos:

1. Los resultados que proporciona SLAVE considerando únicamente como operadores géneticos el cruce sobre dos puntos y la mutación, comparando estos resultados con los obtenidos por los otros algoritmos de aprendizaje propuestos.

2. Las ventajas que aporta a SLAVE la inclusión de los otros operadores genéticos.

2.4.1 Propuesta original

Con la estrategia de experimentación anteriormente descrita, estudiamos el comportamiento de SLAVE considerando que el algoritmo genético encargado de extraer las reglas únicamente utiliza los operadores de cruce y mutación para evolucionar la población hacia la solución final. La tabla 2.2 muestra los parámetros fijados para la ejecución del algoritmo de aprendizaje, mientras que en la tabla 2.3 se muestran los resultados obtenidos para las diferentes bases de datos con consecuente crisp usando los anteriores parámetros.

	C4.5	CN2		LVQ
Bases de datos	Prueba	Prueba	Reglas	Prueba
IRIS	92.7	94.16	6.4	95.72
MONK1	75.6	100	9	76.62
MONK2	64.3	84.0	43	67.59
MONK3	92.6	91.7	15	85.19
INFARTO	81.4	76.16	5.8	56.41
WINE	93.2	89.76	8.8	72.60
PIMA	67.7	74.5	38	67.71
IONOSPHERE	86.5	83.96	16.4	87.97
SOYBEAN	97.9	98.6	32	91.75

Tabla 2.4: Resultados obtenidos con C4.5, CN2 y LVQ

Con el fin de establecer la bondad del algoritmo propuesto, comparamos estos resultados con los obtenidos con otros algoritmos de aprendizaje. Para realizar esta comparación en el caso crisp, hemos seleccionado tres algoritmos de aprendizaje muy conocidos y referenciados en la literatura para el problema de la clasificación. Estos algoritmos son C4.5, CN2 y LVQ cuyas características de implementación se comentan en el apéndice B. Para la experimentación, hemos ejecutado cada uno de estos algoritmos, usando los parámetros por defecto que indican sus implementaciones, sobre las mismas particiones definidas para SLAVE tomando de cada uno de ellos la media sobre su porcentaje de acierto en la predicción. Además, en CN2 que es un algoritmo de aprendizaje que expresa el conocimiento adquirido en forma de reglas, hemos añadido como parámetro de comparación adicional el número medio de las mismas que son necesarias sobre cada base de datos. En la tabla 2.4 se muestran estos resultados.

Los resultados reflejan que SLAVE obtiene el mejor resultado únicamente en dos de las nueve bases de datos propuestas (WINE y PIMA), aunque por el contrario, SLAVE únicamente obtiene el peor resultado de los cuatro algoritmos para la base de datos IONOSPHERE (figura 2.7). Este hecho pone de manifiesto que el modelo de regla difusa descrito en este capítulo, es un forma válida de representación del conocimiento que presenta un correcto funcionamiento en los problemas de clasificación comparable al de otros algoritmos de aprendizaje. Además, esta forma de representación permite que el conocimiento extraido sea más fácil de entender desde un punto de vista humano. Así, por ejemplo en una ejecución de C4.5 para la base de datos WINE obtenemos el siguiente árbol:

```
Att7 <= 0.96 : 3 (27.0)
Att7 > 0.96 :
|   Att13 <= 750 :
|   |   Att11 <= 0.68 : 3 (6.0)
|   |   Att11 > 0.68 :
```

```
|   |   |   Att7 <= 1.22 : 2 (2.0/1.0)
|   |   |   Att7 > 1.22 :
|   |   |   |   Att1 <= 13.17 : 2 (47.0)
|   |   |   |   Att1 > 13.17 :
|   |   |   |   |   Att1 <= 13.29 : 1 (2.0)
|   |   |   |   |   Att1 > 13.29 : 2 (4.0)
|   Att13 > 750 :
|   |   Att10 <= 3.4 : 2 (4.0)
|   |   Att10 > 3.4 : 1 (37.0)
```

Para la misma partición de esta base de datos, las reglas obtenidas por SLAVE son las siguientes:

```
 SI
      Att5 es {Muy Pequeno, Pequeno}
      Att13 es {Medio}
  ENTONCES
      Clase es Tipo1
---------------------------------------

 SI
      Att1 es Mayor o Igual que {Medio}
      Att5 es Menor o Igual que {Medio}
      Att7 es Mayor o Igual que {Pequeno}
  ENTONCES
      Clase es Tipo1
---------------------------------------

 SI
      Att10 es {Ex. Pequeno, Muy Pequeno}
  ENTONCES
      Clase es Tipo2
---------------------------------------

 SI
      Att1 es Menor o Igual que {Medio}
      Att2 es {Medio}
      Att10 es Menor o Igual que {Pequeno}
  ENTONCES
      Clase es Tipo2
---------------------------------------
```

```
SI
      Att11 es Menor o Igual que {Pequeno}
  ENTONCES
      Clase es Tipo3
```

Tomando ambas representaciones del conocimiento sobre un mismo sistema, normalmente la transmisión de dicho conocimiento entre dos personas esta mucho más cerca de la que proporciona SLAVE a la que proporciona C4.5.

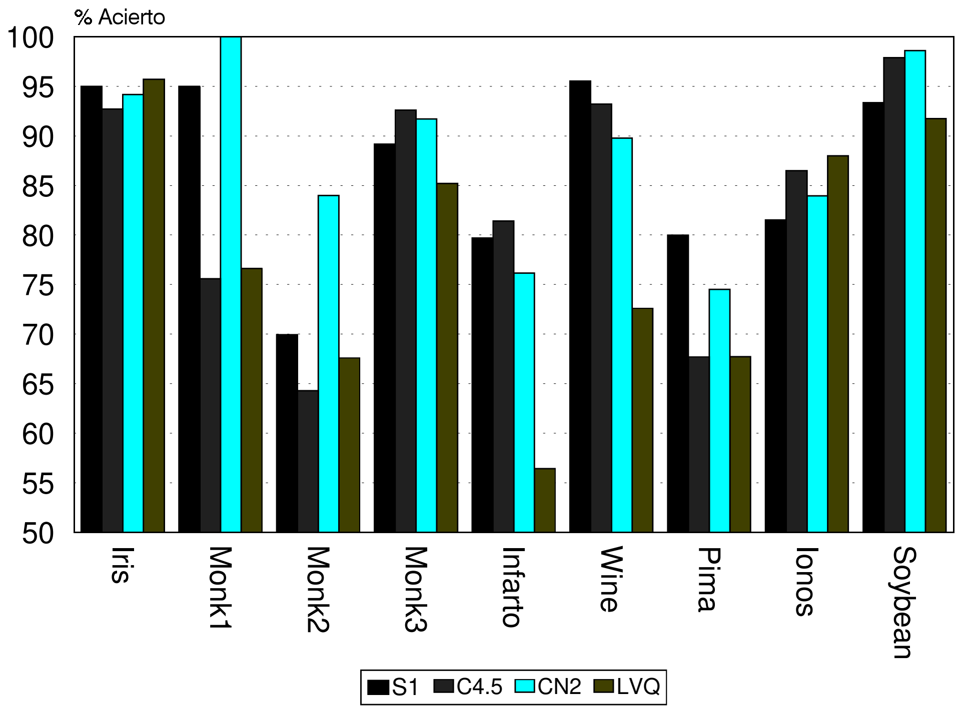


Figura 2.7: Comparativa del acierto en los conjuntos de prueba (1)

Centrándonos de nuevo en los resultados el número medio de reglas que obtiene SLAVE en cada base de datos es semejante al que proporciona CN2. Sin embargo, podemos detectar que para la base de datos IONOSPHERE el número de reglas que produce SLAVE es extremadamente alto comparandolo con el número de reglas de CN2. Esto indica que SLAVE con esta definición tiene un comportamiento aceptable con bases de datos que contienen un número no excesivamente alto de variables de entrada, y que dicho comportamiento empeora ante la presencia de un número elevado de las mismas.

También estamos interesados en comprobar el comportamiento del algoritmo propuesto ante bases de datos con consecuente difuso. Así, utilizando los mismos parámetros

Bases de datos	Entrenamiento	Prueba	N.Reglas	Tiempo
SIN	0.0026	0.0015	6	51
PARABOLA	0.0057	0.0016	2	34
3DPARABOLA	0.0066	0.0112	3	1676
MONTE	0.0038	0.0038	3	2408
PENDULO	22.5634	7.1376	6	83
RICE	0.1354	0.092	5	80.2

Tabla 2.5: Resultados de SLAVE para el caso difuso

Bases de datos	Entrenamiento	Prueba	N.Reglas
SIN	0.0038	0.0412	7
PARABOLA	0.0040	0.0021	7
3DPARABOLA	0.0032	0.0055	49
MONTE	0.0028	0.0024	49
PENDULO	1.2044	7.6490	13
RICE	0.0515	0.0604	43

Tabla 2.6: Resultados obtenidos con el algoritmo WM

definidos para la experimentación anterior, obtenemos los resultados que se muestran en la tabla 2.5.

Al igual que hicimos anteriormente para determinar la bondad del comportamiento de SLAVE en el caso crisp, lo realizaremos en el caso difuso, comparando SLAVE con algún algoritmo de aprendizaje para este tipo de problemas. Para ello seleccionamos el algoritmo de aprendizaje definido en [159] que denominaremos algoritmo WM. Para la ejecución de la parte experimental con WM asumiremos que las variables de entrada y de salida tienen la misma discretización que fué definida para la experimentación con SLAVE. Los resultados obtenidos con este algoritmo se presentan en la tabla 2.6.

A diferencia del caso crisp, en los resultados obtenidos por SLAVE en el caso difuso son comparativamente menos competitivos con los algoritmos especialmente diseñados para abordar estos problemas como es el algoritmo WM. A pesar de esto, podemos observar que el resultado de SLAVE para las bases de datos RICE y SIN es sensiblemente mejor que el obtenido por WM.

Un dato a destacar en esta comparación es el reducido número de reglas que necesita SLAVE para representar las distintas bases de datos que contrasta con el número de las mismas necesitado por WM. Además, se puede observar que esta diferencia entre los números de reglas aumenta en gran medida cuantas más variables de entrada se encuentren involucradas en la base de datos. Esta diferencia es debida a que ambos algoritmos utilizan diferentes modelos de reglas difusas, de manera, que el modelo de regla propuesto por SLAVE puede incluir un conjunto de reglas de modelo definido en el otro algoritmo. Este

Parámetro	Valor
Probabilidad del operador de rotación	0.1
Probabilidad del operador de generalización	0.1
Probabilidad del operador AND	0.05
Probabilidad del operador ORD	0.1

Tabla 2.7: Parámetros adicionales fijados para la ejecución de SLAVE

Bases de datos	Entrenamiento	Prueba	N.Reglas	Tiempo
IRIS	96.56	95.40	4.6	78
MONK1	96.77	94.45	6	42
MONK2	91.25	70.37	13.8	119
MONK3	97.21	92.64	6.8	44
INFARTO	98.39	79.98	7.2	241
WINE	99.56	95.93	8	809.2
PIMA	79.37	80.94	23	1757
IONOSPHERE	99.65	81.90	39.2	6064
SAT	96.26	94.15	40	11372

Tabla 2.8: Resultados de SLAVE caso crisp incluyendo los nuevos operadores genéticos

hecho, al igual que sucede en el caso crisp, permite una mayor compresión del conocimiento adquirido.

2.4.2 Propuesta original con los nuevos operadores genéticos

En esta segunda parte de la experimentación estamos interesados en estudiar que aporta la inclusión de los nuevos operadores genéticos definidos anteriormente en este capítulo. Con tal fin repetimos la experimentación anterior fijando como parámetros de SLAVE, además de los propuestos en la tabla 2.2 los que muestra la tabla 2.7.

Con esta configuración de partida, la tabla 2.9 muestra los resultados obtenidos para las bases de datos del caso difuso, mientras que la tabla 2.8 muestra los resultados con las bases de datos del caso crisp.

En ambos casos podemos observar que en general se produce una ligera mejora de los resultados obtenidos por SLAVE, siendo estos más acusados en el acierto en la partición de prueba y en la reducción del tiempo necesario para la extracción del conocimiento. Esta mejora en los resultados se debe a que la inclusión de los operadores genéticos provoca un comportamiento más estable del algoritmo de aprendizaje. En la figura 2.8, donde **S2** denota los resultados obtenidos por SLAVE con los nuevos operadores genéticos y **S1** los resultados obtenidos con la implementación anterior, podemos observar que SLAVE mantiene el mejor resultado sobre las bases de datos WINE y PIMA, y obtiene el mejor resultado en la base de datos MONK3.

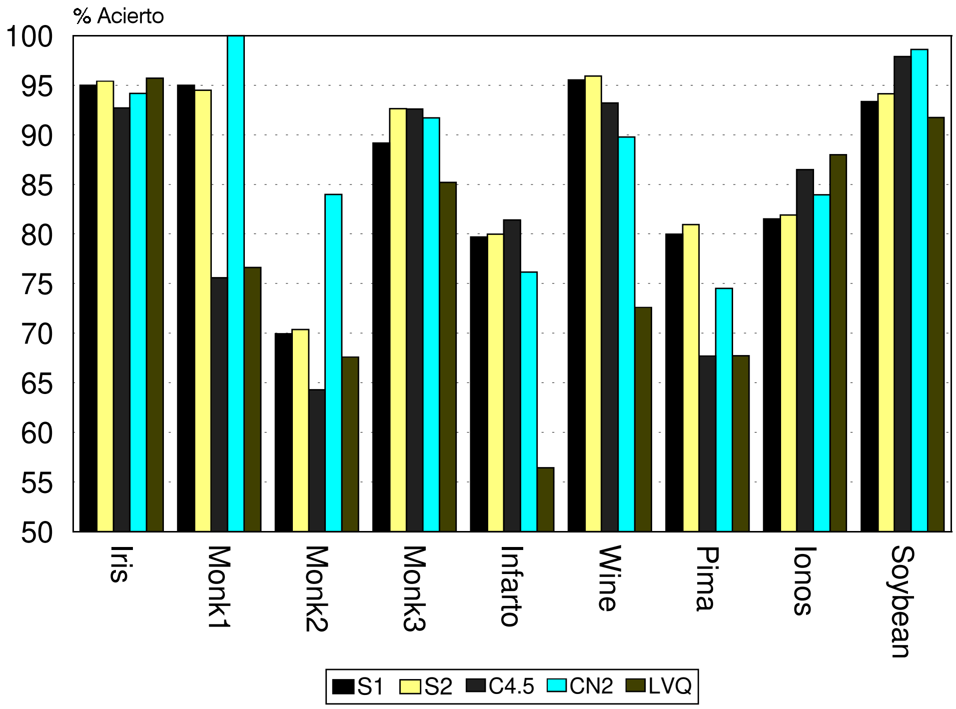


Figura 2.8: Comparativa del acierto en los conjuntos de prueba (2)

A la vista de los resultados, el dato que más resalta, es la gran reducción en el tiempo de ejecución obtenido para la base de datos IONOSPHERE, lo que pone de manifiesto que la aportación que pueden proporcionar estos operadores es mayor cuantas más variables de entrada se encuentren involucradas en el problema.

Por último, podemos extraer las siguientes conclusiones sobre esta parte experimental:

- Los resultados obtenidos por SLAVE son similares a los que producen otros algoritmos de aprendizaje, aunque en términos generales son ligeramente peores.

- La inclusión de los nuevos operadores genéticos mejoran ligeramente los resultados anteriores en términos de capacidad de predicción, y notablemente en cuanto al número de reglas y al tiempo de respuesta.

En el siguiente capítulo proponemos nuevas vías para mejorar el comportamiento del algoritmo de aprendizaje SLAVE.

Bases de datos	Entrenamiento	Prueba	N.Reglas	Tiempo
SIN	0.0021	0.0015	6	51
PARABOLA	0.0051	0.0043	2	34
3DPARABOLA	0.0064	0.0112	3	1676
MONTE	0.0038	0.0041	3	2408
PENDULO	5.0690	6.3751	5	83
RICE	0.315	0.0105	6.2	80.2

Tabla 2.9: Resultados de SLAVE para el caso difuso incluyendo los nuevos operadores genéticos