Capítulo 3

Colaboración entre las soluciones parciales

3.1 Inconvenientes del modelo iterativo

El modelo iterativo descompone el problema en tantos subproblemas como conceptos se desean aprender, de manera, que la solución global se construye como la agregación de las descripciones obtenidas para cada concepto. Para que esta forma de construir la solución global proporcione una descripción correcta de los conceptos que se desean aprender, es necesario que no existan interacciones entre los distintos conceptos, es decir, las descripciones de los conceptos deben ser independientes entre sí.

La agregación de las soluciones parciales proporciona una buena solución global cuando cada solución parcial de primer orden satisface la condición de completitud y cada solución parcial de segundo orden satisface la condición de consistencia. El objetivo de forzar la satisfacción de la condición de consistencia es para garantizar que

$$\bigcap_{i=1}^{j} A_i = \emptyset.$$

donde A_i es la descripción de la regla i-ésima. Sin embargo, como la satisfacción de esta propiedad se restringe a los ejemplos del conjunto de entrenamiento, lo único que se puede asegurar cuando se verifica la condición anterior, es que no existe un ejemplo de ninguna clase que satisfaga la descripción de ninguna regla de cualquier otra clase en el conjunto de ejemplos.

Bajo estas circunstancias, podemos asegurar que no existe interacción entre las distintas soluciones parciales sobre el conjunto de entrenamiento, y por consiguiente, la agregación de las mismas es una buena solución global.

Pero, como ya se comentó anteriormente, en problemas extraídos del mundo real, donde los conjuntos de ejemplos normalmente contienen ruido o están sujetos a descripciones imprecisas, es difícil mantener las anteriores condiciones. La no verificación de la condición de completitud y fundamentalmente la no satisfacción de la condición de consistencia en las soluciones parciales, provoca que la construcción de la solución global como agregación de estas soluciones parciales contenga una descripción imprecisa del sistema que se desea aprender, en el sentido de que varias reglas de diferentes conceptos son igualmente aplicables sobre determinadas zonas del espacio de los ejemplos, es decir, existen al menos dos reglas cuyas descripciones verifican que

$$A_1 \bigcap A_2 \neq \emptyset.$$

En estas situaciones, parece conveniente mantener la solución global más cercana a la satisfacción de las dos condiciones (consistencia y completitud), aún a costa de ser menos exigentes con su satisfacción en las soluciones parciales. Para tener en cuenta este extremo, es necesario que el algoritmo de aprendizaje tenga una visión menos local del problema y contemple desde un punto de vista más global el aprendizaje de los diferentes conceptos.

Esta visión global del problema, no sólo es útil ante la presencia de ruido e imprecisión. Aún conociendo que no existe ruido ni imprecisión en el conjunto de ejemplos, esta visión global puede en un momento dado, sacrificar la satisfacción de las dos condiciones a costa de mejorar la comprensibilidad de las descripciones parciales manteniendo lo más posible la capacidad de predicción sobre el conjunto de ejemplos.

Un ejemplo típico de esta situación se presenta cuando un concepto se puede describir como un caso particular de un concepto superior.

Ejemplo 3.1 *Supongamos que se desea clasificar un conjunto de aves en base a su capacidad de volar. Sea el conjunto de ejemplos el siguiente:*

Ave	*Concepto*
Aguila	*Vuela*
Albatros	*Vuela*
Buitre	*Vuela*
Cuervo	*Vuela*
Petirrojo	*Vuela*
Golondrina	*Vuela*
Pingüino	*No Vuela*
Gaviota	*Vuela*
Paloma	*Vuela*
Flamenco	*Vuela*

Si cada concepto se aprenden de forma independiente, se pueden obtener las dos siguientes descripciones que verifican las condiciones de consistencia y completitud:

1a) Si el ave es un Pingüino entonces No Vuela.

1b) Si el ave es un Aguila o es un Albatros o es un Buitre o es un Cuervo o es un Petirrojo o es una Golondrina o es una Gaviota o es una Paloma o es un Flamento entonces Vuela.

Sin embargo, tomando una visión más global de problema, se podría sugerir la siguiente clasificación:

2a) Si el ave es un Pingüino entonces No Vuela.

2b) Si es cualquier ave entonces Vuela.

Estas dos nuevas reglas son más faciles de entender desde un punto de vista humano, ya que se describe la situación del Pingüino, como un caso particular de ave que es incapaz de volar. Sin embargo, la descripción (2b) no satisface la condición de consistencia, y por consiguiente, vulnera ser una solución parcial de segundo orden.

Si nos detenemos a analizar el segundo conjunto de descripciones, no parece tan relevante que la descripción 2b no satisfaga la condición de consistencia si estas situaciones se tienen en cuenta durante el proceso de inferencia. Si a la entrada del sistema de inferencia se le presenta un ejemplo que es cubierto por varias descripciones de diferentes conceptos, la salida del mismo debería ser el concepto asociado a la descripción más específica de entre las descripciones aplicables.

En el caso anterior, ante el ejemplo "Pingüino", ambas descripciones (2a y 2b) son aplicables, pero la descripción 2a esta contenida en la descripción 2b, por tanto, es más específica que esta. En consecuencia, la salida diría que ese ave No Vuela.

De todas maneras, en la mayoría de las situaciones, no es fácil que el conjunto de descripciones satisfaga un orden completo con respecto a la operación $\subseteq$. Es decir, nos podemos encontrar con que para una misma entrada dos descripciones de dos conceptos diferentes son aplicables, sin que entre ellos exista un relación de contenido, aunque si de intersección no vacía.

Basándonos en esto, podemos caracterizar las reglas que pueden provocar conflictos durante el proceso de inferencia en función de las intersecciones entre las descripciones.

Definición 3.1 *Sean dos reglas $R_{B_1}(A_1)$ y $R_{B_2}(A_2)$ que describen dos conceptos diferentes B_1 y B_2. Diremos que estas dos reglas pueden entrar en conflicto durante el proceso de inferencia, si la intersección entre sus descripciones no es vacía, es decir,*

$$A_1 \bigcap A_2 \neq \emptyset$$

Además, en base al concepto de intersección entre descripciones, podemos determinar cuando una regla es una excepción a una regla más general.

Definición 3.2 *Sean dos reglas $R_{B_1}(A_1)$ y $R_{B_2}(A_2)$ que describen dos conceptos diferentes. Diremos que la regla $R_{B_2}(A_2)$ es una excepción a la regla $R_{B_1}(A_1)$ si y sólo si,*

$$A_1 \bigcap A_2 = A_2$$

Además, en estas circunstancias diremos que la regla $R_{B_1}(A_1)$ es una regla más general que $R_{B_2}(A_2)$ o que la regla $R_{B_2}(A_2)$ es una regla más específica que $R_{B_1}(A_1)$.

Si durante el proceso de inferencia entran en conflicto reglas que satisfacen la anterior definición, es decir, existe una regla que es más específica que el resto de las reglas aplicables, parece que la entrada propuesta satisface una excepción a las reglas más generales, y por tanto, la salida del proceso de inferencia debe coincidir con el concepto asociado a esta regla más específica.

Como ya se describió en el capítulo anterior, el mecanismo de resolución de conflictos definido en SLAVE, toma como primer criterio el peso asociado a las reglas. El objetivo que se persigue con este criterio, es dar prioridad a las reglas que presentaron una mayor capacidad de predicción sobre el conjunto de entrenamiento.

En base a lo anterior, una propiedad deseable que debería cumplir el sistema para garantizar un correcto funcionamiento, es que si existen dos reglas para dos conceptos diferentes, y una de ellas es una excepción a la otra, entonces la regla más específica debe tener un peso mayor que la más general. Sin embargo, esta propiedad no la satisface SLAVE.

Ejemplo **3.2** *Supongamos que el algoritmo de aprendizaje devuelve un conjunto de reglas para describir el comportamiento de un determinado sistema. Supongamos además que $R_{B_1}(A_1)$ y $R_{B_2}(A_2)$ son dos de las reglas extraídas por el algoritmo para dos conceptos distintos que verifican que:*

- $A_1 \bigcap A_2 = A_1$*, es decir, la regla $R_{B_1}(A_1)$ es una excepción a la regla $R_{B_2}(A_2)$*
- $\omega(R_{B_2}(A_2)) > \omega(R_{B_1}(A_1))$ *y por consiguiente el peso asociado a la regla más general es mayor que el asociado a la regla más específica.*

Bajo estas condiciones, todo ejemplo que sea cubierto por la regla $R_{B_1}(A_1)$, lo es también por la regla $R_{B_2}(A_2)$. Al tener la regla más general mayor peso que la más específica, en caso de conflicto siempre se seleccionará la regla $R_{B_2}(A_2)$, por lo que podemos asegurar que la regla $R_{B_1}(A_1)$ no será seleccionada nunca durante el proceso de predicción. Por tanto, dicha regla no es útil para describir el sistema y puede ser eliminada del conjunto de reglas.

Podría parecer que el problema que pone de manifiesto el ejemplo anterior parte de una inadecuada definición de los criterios establecidos en el mecanismo de resolución de conflictos. Sin embargo, este problema plantea un error en la propia concepción del algoritmo de aprendizaje. Desde el punto de vista del aprendizaje, no parece lógico que la excepción a una regla tenga menor capacidad de predicción que la regla más general, por que si esto ocurre, es que la excepción planteada no es cierta.

El aprendizaje individual y aislado de cada una de las reglas que son necesarias para describir el sistema es el desencadenante de este tipo de problemas. En consecuencia, para que este tipo de inconvenientes no aparezcan en los algoritmos de extracción automática de conocimiento, es necesario establecer relaciones de colaboración-competición entre las reglas que componen dicho conocimiento.

Por consiguiente, si se desea que se establezcan relaciones de colaboración-competición entre las reglas aprendidas es necesario tener una visión más global del problema. Obviamente, esta concepción más global del problema del aprendizaje esta intimamente relacionada con el conocimiento de como las reglas se comportarán durante el proceso de predicción, es decir, con el conocimiento del uso de las reglas aprendidas.

De entre las formas posibles de dotar a los algoritmos basados en el modelo iterativo de esta capacidad de visión más global del problema, en este capítulo proponemos dos posibles vías para solucionarlo: la primera consiste en añadir al algoritmo de aprendizaje un módulo de refinamiento de teorías como fase final del proceso de aprendizaje. La entradas a este módulo de refinamiento son el conjunto de ejemplos y el conjunto de reglas aprendidas. El objetivo del refinamiento consiste en mejorar el conocimiento aprendido en varios sentidos: capacidad de predicción, comprensibilidad de las descripciones, reducción de las descripciones, etc.

La segunda vía consiste en modificar el algoritmo de aprendizaje para establecer relaciones de colaboración-competición entre las soluciones parciales. La modificación propuesta consiste básicamente en incorporar la información que se conoce hasta un momento dado, del comportamiento que tendrán las reglas durante el proceso de predicción para evaluar la bondad de las siguientes reglas a aprender.

En las dos siguientes secciones se estudia más detenidamente cada una de estas alternativas.

3.2 Algoritmo de refinamiento de teorías difusas

Dado un conocimento incompleto y/o incorrecto (teoría) que trata de describir el comportamiento de un determinado sistema, un algoritmo de refinamiento de teorías trata de modificar este conocimiento inicial para producir un mejor funcionamiento del sistema.

En nuestro caso, el conocimiento inicial sobre el sistema se describe como una base de reglas difusas del tipo DNF (teoría difusa) y se determina la mejora de dicho conocimiento utilizando un conjunto de ejemplos del sistema que se desea aprender. El objetivo del algoritmo de refinamiento consiste en modificar el conjunto inicial de reglas para obtener una mejor adaptación sobre el conjunto de ejemplos, una mayor comprensibilidad de las descripciones y una reducción, si es posible, del número de reglas.

Los operaciones básicas de este tipo de algoritmos, cuando el conocimiento se representa mediante reglas, son los procesos de especificación, generalización, adición y eliminación de reglas. En consecuencia, la principal diferencia entre dos algoritmos de refinamiento de reglas es la heurística usada por el algoritmo para aplicar estos procesos. Por ejemplo, el algoritmo EITHER [120] mejora el conjunto de reglas causando el mínimo número de cambios sobre el conocimiento inicial. Sin embargo, podemos encontrar otros algoritmos como JoJo [37] donde su heurística solamente se guía por la precisión del conjunto de reglas sobre el conjunto de ejemplos. Estos dos sistemas se diseñaron para trabajar con reglas crisp.

Por las razones anteriomente expuestas, en [55] se propone un algoritmo de refinamiento, que asociado a SLAVE, trata de verificar que la unión de las reglas individuales puede considerarse como una buena solución global. La asociación de estos dos algoritmos recibe el nombre de SLAVE+R, y presenta una estructura semejante a la propuesta en la figura 3.1. En trabajos posteriores, tales como [57, 58], se modifica la la definición original de SLAVE+R, proponiéndose dos versiones diferentes de dicho algoritmo de refinamiento, uno para el caso difuso y otro para el caso crisp. Además, se estudia el comportamiento del algoritmo de refinamiento de teorías propuesto con el conocimiento extraído por otros algoritmos de aprendizaje.

Los elementos básicos del algoritmo de refinamiento de teorías son:

a) **El modelo de regla**. Este elemento se hereda de SLAVE y es una característica importante de este sistema porque permite deteminar las variables relevantes para cada regla.

b) **La heurística**. La manera especial en la cual se combinan los procesos de especificación, generalización, adición y eliminación de reglas.

c) **El proceso de inferencia**. Podemos distinguir dos procesos diferentes de inferencia: caso crisp y caso difuso. Como se muestra en el capítulo anterior, SLAVE trata ambos tipos de problemas de aprendizaje. Para el primero de ellos, SLAVE usa una función para determinar la compatibilidad entre un ejemplo y el conjunto de reglas y utiliza un mecanismo especial de resolución de conflictos basado en una medida sobre la capacidad de predicción de cada regla. Por consiguiente, el algoritmo de refinamiento supone que el conjunto de reglas que se ofrece como entrada está ordenado siguiendo este criterio. Para problemas con consecuente difuso, el orden de las reglas es menos importante ya que la salida se obtiene mediante la combinación de las reglas. En este caso, se usa un proceso de inferencia y un método de nitificación.

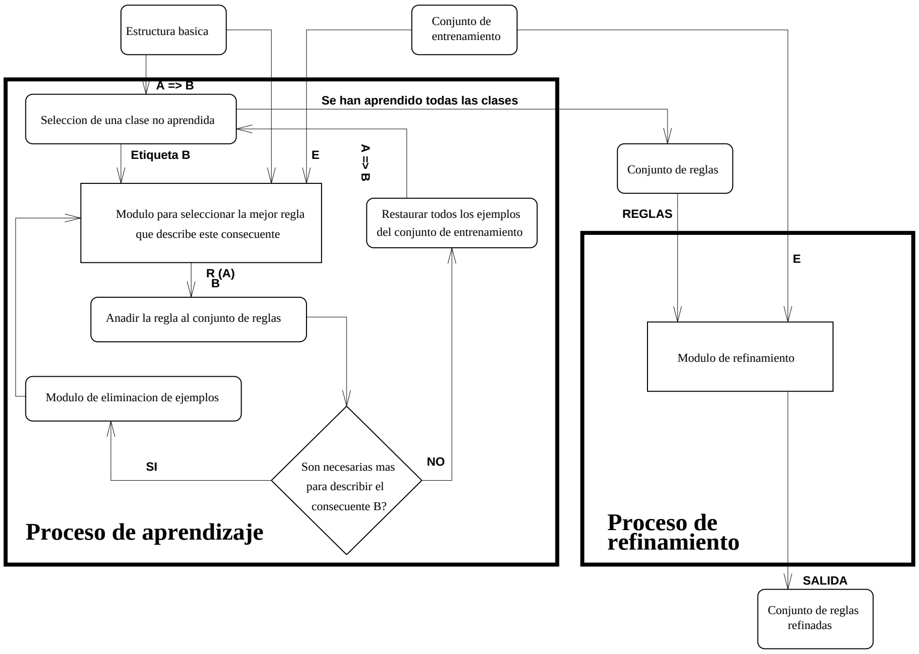


Figura 3.1: Estructura del algoritmo de aprendizaje SLAVE+R

En los siguientes apartados, describimos las características fundamentales del algoritmo de refinamiento. En primer lugar definimos la propuesta para el caso crisp y posteriormente la propuesta para el caso difuso.

3.2.1 Refinamiento de reglas difusas con consecuente crisp

El algoritmo de refinamiento intenta mejorar el conjunto de reglas obtenido por SLAVE, reduciendo los problemas descritos previamente. El algoritmo se compone fundamentalmente de dos tareas. La primera de ellas consiste en mejorar la precisión de cada regla. Con este propósito, se usa un proceso de especificación, el cual intenta que cada regla recubra el mayor número de ejemplos correctamente clasificados de la regla original sin recubrir sus ejemplos mal clasificados (precisión). La aplicación de esta tarea, puede provocar que algunos de los ejemplos mal clasificados por la regla original, pasen a estar no clasificados por ninguna. Por tanto, la siguiente tarea intenta, mediante un proceso de generalización, recubrir estos ejemplos no clasificados. Se utilizan dos mecanismos para recubrir estos ejemplos: o bien generalizando las reglas ya existentes o bien añadiendo nuevas reglas que recubran específicamente estos ejemplos. Las dos tareas anteriores se repiten hasta que se satisfaga una condición de terminación. La figura 3.2 muestra los diferentes pasos de este algoritmo.

El refinamiento usa una función heurística y una estrategia de ascensión de colinas para seleccionar la acción más prometedora en cada paso del algoritmo. Consideramos

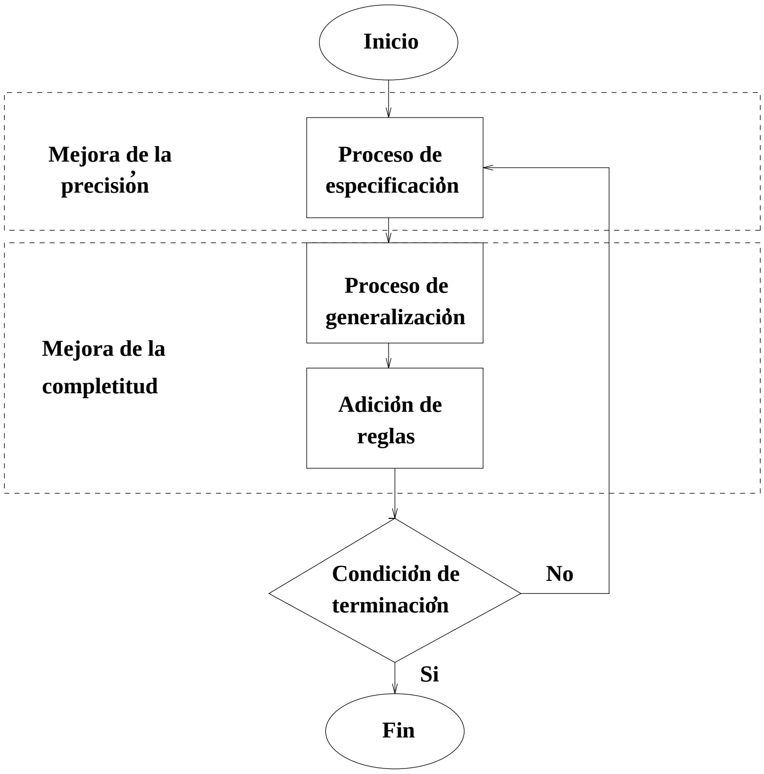


Figura 3.2: Algoritmo de refinamiento

una función que mide la capacidad de predicción global de un conjunto de reglas sobre el conjunto de ejemplos. Esta función determina la precisión que ofrece el proceso de inferencia de SLAVE (descrito en el capítulo anterior) sobre el conjunto de entrenamiento.

La componente heurística del algoritmo de refinamiento seleciona las reglas en el orden previamente descrito en el conjunto de reglas. Sin embargo, no se ha considerado ninguna ordenación especial para las variables y sus valores y se toma sobre ellas un orden por defecto.

A continuación, describimos cada uno de los pasos del algoritmo de refinamiento de una forma más detallada.

1) Mejorar la precisión de las reglas

En este paso, tratamos de mejorar la capacidad de predicción del conjunto de reglas obtenido por SLAVE sobre el conjunto de entrenamiento, incrementando la precisión de cada regla.

Obtener un conjunto de reglas, donde todas las reglas sean completamente precisas no es posible en todos los casos. Por ejemplo, cuando trabajamos con conjuntos de ejemplos que contienen ruido, puede ocurrir que un subconjunto de ellos se pueda clasificar correctamente, pero no podemos decir nada sobre el resto. Por esto, la precisión de la regla establece un grado en términos de la relación existente entre

Esquema 3.1 *Mejora de la precisión de las reglas*

1. Se selecciona la regla más relevante.
2. Se selecciona una variable relevante y se hace lo siguiente:
   - 2.1. Se selecciona un valor activo.
   - 2.2. Se desactiva este valor.
   - 2.3. Si se mantiene o se mejora la precisión global, entonces la modificación se acepta. En otro caso, el valor se vuelve a activar.
   - 2.4. Si hay más valores activos, ir al paso 2.1.
   - 2.5. Si todos los valores de la variable están no activos, entonces esta regla se elimina del conjunto de reglas.
   - 2.6. Si hay variables relevantes aún no seleccionadas, ir al paso 2.
3. Si hay quedan más reglas, seleccionar la siguiente regla e ir al paso 2., en otro caso el proceso termina.

aciertos y fallos que produce una regla sobre el conjunto de ejemplos. Asociando este grado con el concepto de completitud, el primer objetivo del algoritmo de refinamiento consiste en encontrar un buen conjunto de reglas, en el cual cada regla debe tener un grado de completitud tan alto como sea posible.

SLAVE tiene tendencia a generalizar las reglas tanto como los parámetros de consistencia se lo permiten. Por esta razón, el primer paso del algoritmo de refinamiento intenta contrarestar este comportamiento mediante la especificación de las reglas. El objetivo es reducir el número de fallos en cada regla. Así, el proceso empieza especificando las reglas en el orden en el que aparecen en el conjunto de reglas.

Diremos que un valor para una variable está activo si este valor aparece para esa variable en la descripción de la regla. Por contra, diremos que un valor para una variable está no activo o desactivado si este valor no aparece para esta variable en la descripción de la regla. De esta manera, una operación de especificación se puede considerar como el cambio a no activo de un valor que previamente estaba activo. Diremos en este caso que el valor se desactiva.

Siguiendo la notación anterior, el esquema 3.1 describe este proceso.

Debemos destacar que el proceso anterior permite eliminar reglas del conjunto de reglas. Por consiguiente, la eliminación de reglas se considera como un caso particular de especificación que se produce cuando todos los valores de una variable están desactivados.

Otra consideración a destacar del proceso anterior es la justificación del orden elegido para realizar la especificación. Supongamos que la teoría que deseamos refinar está constituida por las reglas que se proponen en el ejemplo 3.2. Con esta configuración, a la regla más general le corresponde una posición menor que a la regla más específica, ya que su peso es superior. Por tanto, es posible que la especificación del antecedente de la regla más general provoque una mejora en el sistema al hacer útil a la más específica. Sin embargo, si se aplica el orden contrario en el proceso de especificación, el sistema terminará eliminando la reglas más especificas, con lo que el algoritmo de refinamiento pierde parte de la información que le proporcionó el algoritmo de aprendizaje.

2) **Mejorar la completitud generalizando las reglas**

En este segundo paso, el algoritmo de refinamiento intenta mejorar la completitud de las reglas usando un proceso de generalización ordenado. En este proceso, se añaden nuevos valores a la descripción de cada una de las reglas. El proceso acepta la generalización de una regla cuando la capacidad de predicción global del conjunto de reglas se mantiene o mejora la del conjunto de reglas sin la modificación. La mejora de la capacidad de predicción en este proceso tiene en cuenta las siguientes dos situaciones:

a) Los ejemplos no clasificados pueden ser cubiertos correctamente mediante la generalización de alguna de las reglas.
b) Los ejemplos que son clasificados incorrectamente por una regla con un orden menor, pueden ser clasificados correctamente cuando se generaliza alguna de las reglas con un orden mayor.

Por consiguiente, este paso del algoritmo mejora tanto la completitud como la precisión de algunas reglas de la teoría.

Este proceso tiene una estructura similar al paso anterior, pero se usa el orden inverso para la selección de las reglas, es decir, se empieza por las reglas de menor orden y se termina con las reglas de mayor orden. El esquema 3.2 muestra el algoritmo que describe el funcionamiento de este proceso.

La razón de utilizar el orden inverso en este proceso es la siguiente: el proceso de inferencia en caso de conflicto toma las reglas que aparecen en las primeras posiciones del conjunto de reglas. Una regla que ocupa la posición T en el conjunto de reglas puede incluir en su descripción, las descripciones de todas las reglas que tienen una posición inferior a T, por supuesto, si esto no implica una reducción en la capacidad de predicción del sistema. Este hecho provoca que las primeras reglas tiendan a ser más específicas que las que ocupan las últimas posiciones en el conjunto de reglas. Por tanto, las últimas reglas tiene una tendencia mayor a ser generalizadas. Por

Esquema 3.2 *Mejora de la completitud de las reglas*

1. Se selecciona la regla menos relevante.
2. Se selecciona una variable relevante y se hace lo siguiente:
   - 2.1. Se selecciona un valor no activo.
   - 2.2. Se activa este valor.
   - 2.3. Si se mantiene o se mejora la precisión global, entonces la modificación se acepta. En otro caso, el valor se vuelve a desactivar.
   - 2.4. Si hay más valores no activos, ir al paso 2.1.
   - 2.5. Si hay variables relevantes aún no seleccionadas, ir al paso 2.
3. Si aún quedan más reglas, seleccionar la siguiente regla e ir al paso 2., en otro caso el proceso termina.

consiguiente, esta reglas están más cerca de sus descripciones definitivas y permiten mayor flexibilidad con respecto a la generalización de las reglas de orden inferior.

3) **Mejorar la completitud añadiendo nuevas reglas**

El siguiente paso en el algoritmo, consiste en añadir nuevas reglas para recubrir los ejemplos que no se pudieron recubrir por ninguna de las reglas en los pasos anteriores. Este paso añade la regla más específica con mejor adaptación para cada ejemplo no cubierto del conjunto de entrenamiento. La forma de construcción de esta regla se describe en el ejemplo 2.6 del capítulo 2.

4) **Condición de terminación**

Los pasos anteriores se repiten hasta que el conjunto de reglas es estable. En el algoritmo, se determina que un conjunto de reglas es estable cuando el número de reglas y la precisión se mantienen en dos iteraciones consecutivas.

3.2.2 Refinamiento para reglas con consecuente difuso

El algoritmo de refinamiento descrito anteriormente es válido cuando la variable de clasificación es crisp. Sin embargo, cuando la variable de clasificación es difusa, es necesario modificar el algoritmo de refinamiento, ya que los problemas de este tipo tienen un mecanismo de inferencia diferente. El modelo de inferencia propuesto para el caso difuso contempla la colaboración entre las reglas para determinar la salida del sistema. Esta característica es una de las diferencias fundamentales entre ambas implementaciones. Mientras en el caso crisp, se realiza un proceso previo de ordenación de reglas atendiendo al peso de las mismas, en el caso difuso, no se aplica este proceso, y se toma el orden en el que las reglas aparecen en el conjunto de reglas.

Por esta razón, definimos una readaptación al caso difuso del refinamiento de teorías. Básicamente, el algoritmo mantiene la estructura del anterior (figura 3.2), pero incorporando dos nuevos componentes: un mecanismo de inferencia para este tipo de reglas y una medida de error que establece la bondad del conjunto de reglas.

El mecanismo de inferencia coincide con el descrito en SLAVE para este tipo de problemas, es decir, el sistema de inferencia max-min y la media de las alturas como método de nitificación, usando el operador mínimo como t-norma [34].

En base a este sistema de inferencia, definimos una medida de error que establece la precisión del conjunto de reglas sobre el conjunto de ejemplos. Esta medida de error es una modificación sobre la media del error cuadrático que se define como:

$$Error = \sum_{i=1}^{n} \frac{(y_i - y_i^1)^2}{2n}$$

donde y_i es la salida del sistema, y_i^1 es la salida que propone el mecanismo de inferencia usando el conjunto de reglas y n es el número de ejemplos.

Por tanto, el algoritmo de refinamiento para el caso de variables de clasificación difusa es también un algoritmo heurístico que intenta maximizar la bondad del conjunto de reglas minimizando la medida de error.

Los pasos del algoritmo son los siguientes:

1) **Mejorar la precisión de las reglas:** Cuando trabajamos con problemas de variable consecuente crisp, la precisión establece una relación entre el número de ejemplos correctamente clasificados por una regla y el número total de ejemplos que son cubiertos por la misma. En el caso difuso se debe extender este concepto, ya que la salida del mecanismo de inferencia depende de la colaboración de las reglas que son aplicables. Por esta razón, en el caso difuso, definimos la precisión de una regla como la media de los errores que produce dicha regla en las salidas donde ésta se ha usado.

   Desde este punto de vista, este paso consiste en mejorar la precisión de la regla mediante procesos simples de especificación que permiten eliminar valores irrelevantes de la descripción de cada regla. En este sentido, un proceso de especificación trata de eliminar valores de las variables que provocan colaboraciones anómalas entre las reglas durante el proceso de inferencia. Además, es posible que reglas que provoquen un alto nivel de distorsión en las salidas del sistema (es decir, que tengan un nivel bajo de precisión) pueden ser eliminadas del conjunto de reglas. Esto es posible, ya que al igual que en el caso crisp, la eliminación de reglas se considera un proceso especial de especificación, donde todos los valores de una determinada variable son desactivados.

El comportamiento de este primer paso del algoritmo de refinamiento coincide con el propuesto en el esquema 3.1, considerando la nueva interpretación del concepto de precisión de las reglas.

2. **Aumentar el grado de influencia de las reglas en la interpolación de la salida:** En este paso, al contrario que en el anterior, tratamos de incrementar la interacción entre las distintas reglas mediante un proceso de generalización. Este incremento de la interacción entre las reglas intenta potenciar la colaboración, con el objetivo de aumentar la precisión global de la teoría. El proceso consiste en añadir un nuevo valor a una variable antecedente siempre y cuando la precisión global se mantenga o mejore estrictamente. De tal forma, que sólo se permite que una regla forme parte del conjunto de reglas que influyen en la interpolación para un subconjunto de ejemplos, si su participación no provoca un aumento del error global.

   El funcionamiento de este paso es semejante al propuesto en el esquema 3.2 teniendo en cuenta el cambio de significado del concepto de precisión de las reglas, al igual que en el paso anterior.

3. **Añadir nuevas reglas:** Supongamos que mediante el proceso anterior la interacción entre las distintas reglas no puede mejorarse generalizando algunas de las reglas existentes. En este caso, es posible que se necesiten nuevas reglas para decrementar los errores de las salidas. Este paso es el encargado de introducir estas nuevas reglas para reducir el error, cuando el paso anterior, no es capaz de realizarlo.

   El problema que surge en este paso es: ¿Qué regla es la que debemos añadir al conjunto de reglas?

   Responder a esta pregunta implica establecer un criterio para decidir entre todas las posibles reglas que se pueden introducir. Nuestro objetivo consiste en encontrar el conjunto de reglas que produzcan, en media, el menor error posible sobre el conjunto de entrenamiento. Con esta finalidad, establecemos un criterio basado en la elección de la regla más específica (es decir, la que menor distorsión provocará en el conjunto de reglas) que reduce el error del ejemplo peor representado del conjunto de entrenamiento. En otras palabras, construimos a partir del ejemplo que posee el máximo error individual, una regla que trate de reducir el mismo. En el caso de que exista más de un ejemplo que tenga el máximo error individual se tomará aleatoriamente cualquiera de ellos. La nueva regla se construye de la siguiente manera:

   – como parte condición: el antecedente más específico que mejor recubra a dicho ejemplo.

– como parte consecuente: el valor del dominio que menor error global proporcione, al ser incluida esta regla en el conjunto de reglas ya existentes.

Esta filosofía de actuación asume la siguiente idea: la introducción de una regla que mejore el máximo error individual, también mejorará el error medio global.

Este paso va introduciendo reglas hasta que se encuentra con un ejemplo que no mejora ni el máximo error individual del ejemplo, ni el error medio global. La regla que satisface esta condición no es introducida en la base de reglas.

3.2.2.1 Un ejemplo del funcionamiento del algoritmo de refinamiento

Supongamos que queremos realizar una modelización cualitativa de un sistema descrito por la función $h(x, y) = x^2 + y^2$ (3.3(a)). La figura 3.3(b) representa la superficie aprendida usando el algoritmo SLAVE a través de un conjunto de ejemplos del sistema original. Podemos observar que las reglas propuestas por SLAVE aproximan bastante correctamente la zona central de la superficie, pero no así los extremos de la gráfica.

El algoritmo de refinamiento intenta corregir la función, disminuyendo el error. En la figura 3.3(c) se muestra la superficie después del primer paso. Aparentemente, esta nueva superficie presenta una aproximación peor que la inicial aunque el error es mejor. Este aspecto es debido a la influencia negativa o insuficiente de algunas reglas en el proceso de interpolación. En las siguientes iteraciones del algoritmo, se va suavizando este comportamiento, que inicialmente es brusco.

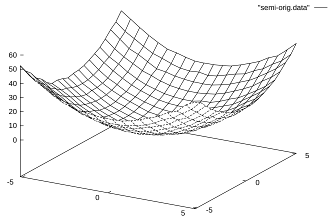
Despues de este paso, podemos ver la evolución del algoritmo de refinamiento como un proceso de aproximación suave a la superficie original. En la figura 3.3(d) podemos observar que la superficie experimenta una sensible mejora después del proceso de generalización. Esto es debido al aumento de las reglas que influyen positivamente en la interpolación sobre los subconjuntos de ejemplos mal representados.

La figura 3.3(e) muestra el comportamiento del conjunto de reglas después del paso 3. En esta gráfica podemos observar que las reglas existentes no eran suficientes para aproximar los extremos de la gráfica, pero después de incluir las nuevas reglas, esta aproximación se mejora.

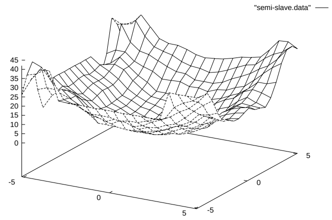
En este ejemplo, la gráfica después del cuarto paso no se muestra debido a que en este problema no existen variables irrelevantes, y por lo tanto, este paso devuelve la misma superficie que se obtuvo en el paso 3.

Para este ejemplo, el algoritmo de refinamiento se repitió ocho veces antes de hacer cierta la condición de terminación. En la figura 3.3(f) podemos ver la superficie resultante después de la ejecución del algoritmo. Se puede observar que es bastante similar a la superficie original, y que mejora notablemente la superficie de partida (figura 3.3(b)).

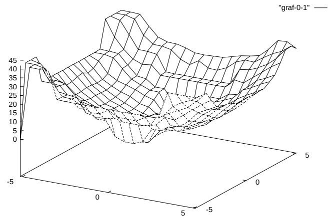
En la identificación de sistemas se suele usar un modelo de regla donde cada variable antecedente toma uno y sólo un valor de su dominio [22, 68, 159]. El modelo usado



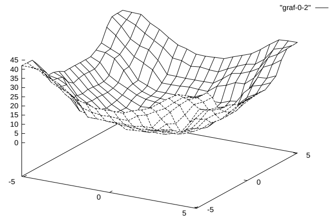
(a) Función $h(x,y) = x^2 + y^2$



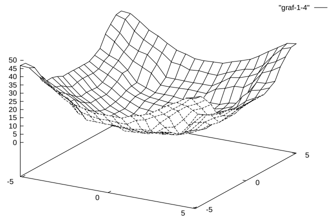
(b) Aproximación obtenida por SLAVE



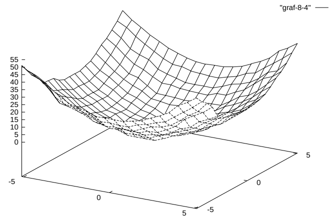
(c) Resultado después del paso 1



(d) Resultado después del paso 2



(e) Resultado después del paso 3



(f) Resultado final obtenido por el refinamiento

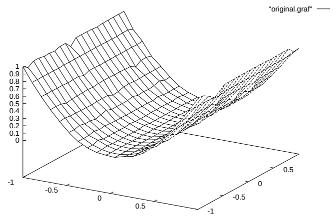
Figura 3.3: Funcionamiento del algoritmo de refinamiento

en SLAVE es más general, ya que el modelo anterior se puede incluir como un caso particular, presentando además la ventaja de obtener definiciones más simples. En el capítulo 2, mostramos las ventajas de este tipo de reglas como herramienta para detectar la relevancia de las variables antecedente para cada valor de la variable consecuente. En la figura 3.4 mostramos una función no lineal con variables irrelevantes para comprobar si el algoritmo de refinamiento mantiene las variables relevantes obtenidas por SLAVE. La función utilizada para esta prueba es

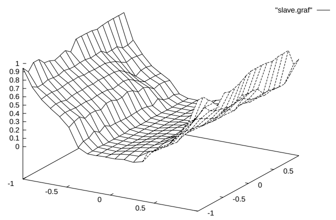
$$f(x,y) = x^2$$

(figura 3.4(a)) para el cual SLAVE devuelve 11 reglas (figura 3.4(b)). SLAVE+R (figura 3.4(c)) refina las reglas de SLAVE y devuelve 6 reglas donde la variable y se considera irrelevante para todas las reglas.

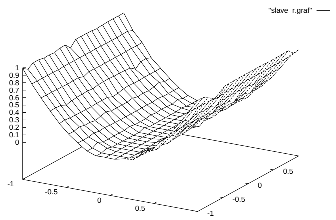
Por tanto, el algoritmo de refinamiento mantiene la tendencia a seleccionar, entre todas las reglas que permiten una correcta identificación del sistema, las que tienen una descripción más simple, es decir, que involucren el menor número posible de variables antecedente en su descripción.



(a) Función $f(x, y) = x^2$



(b) Resultado obtenido por SLAVE



(c) Resultado obtenido por SLAVE+R

Figura 3.4: Conservación de la simplicidad en el algoritmo de refinamiento

3.2.3 Estudios experimentales

Una vez definidos los algoritmos de refinamiento para el caso crisp y el caso difuso, pasamos a analizar el comportamiento real de dicho algoritmo. Para ello, realizamos una comparación entre los resultados que se obtienen al incluir dicho algoritmo de refinamiento como un paso más del proceso de aprendizaje con los resultados previamente obtenidos. Además, realizamos un estudio más detallado de la combinación de estos dos algoritmos en la aplicación práctica de un problema particular.

Bases de datos	Entrenamiento	Prueba	N.Reglas	Tiempo
IRIS	96.97	94.02	3.7	86
MONK1	100	100	4	48
MONK2	91.60	69.30	13.2	156
MONK3	97.21	92.64	5.4	57
INFARTO	98.69	81.56	4.9	323
WINE	100	89.69	5.3	2242
PIMA	82.99	77.08	16.2	3821
IONOSPHERE	100	83.23	16	12107
SOYBEAN	98.23	96.72	33	15903

Tabla 3.1: Resultados de SLAVE+R en el caso crisp

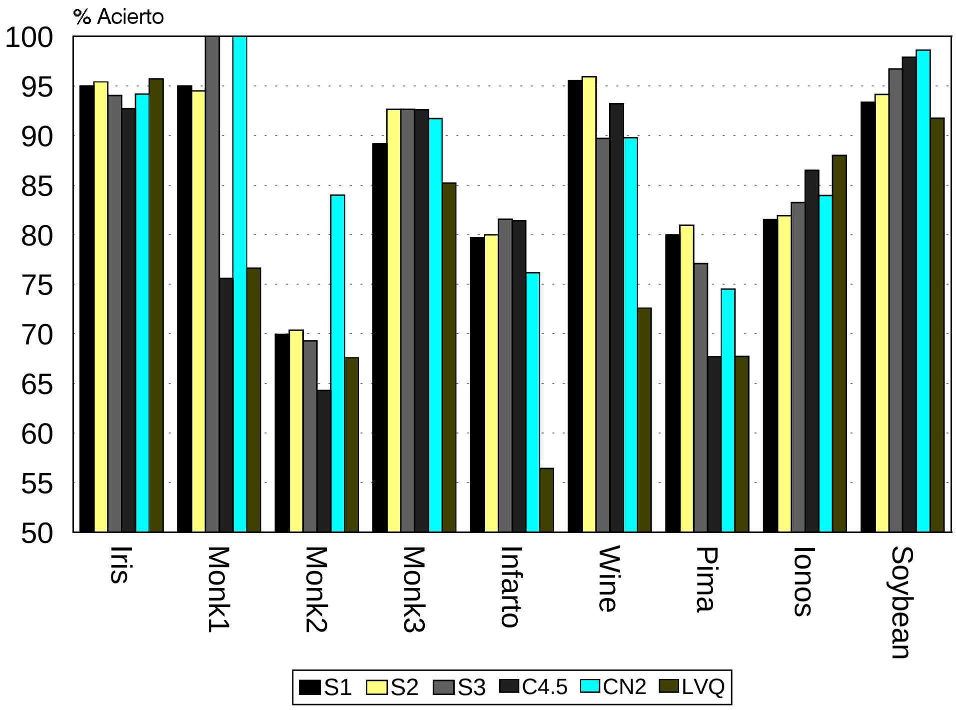
3.2.3.1 Comportamiento del algoritmo de refinamiento

En esta sección estudiamos los resultados que obtiene el algoritmo de aprendizaje SLAVE si se considera el algoritmo de refinamiento de teorías, definido anteriormente, como el último paso del algoritmo de aprendizaje. Para esta experimentación, se ha tomado como entrada al algoritmo de refinamiento las reglas obtenidas por SLAVE incluyendo los nuevos operadores, cuyos resultados se mostraron en la tabla 2.8. En la tabla 3.1 se muestran los resultados que se obtienen al incluir el algoritmo de refinamiento.

Debemos destacar que la medida de tiempo asociada a cada base de datos se obtiene como la suma del tiempo que consumió SLAVE para aprender el conjunto de reglas más el tiempo consumido por el algoritmo de refinamiento de teorías. Por esta razón, el tiempo de respuesta de SLAVE+R es peor que el obtenido al aplicar aisladamente SLAVE. Si comparamos estas diferencias de tiempo, podemos observar que el algoritmo de refinamiento tiene un tiempo de respuesta aceptable sobre bases de datos que contienen un número pequeño de reglas, mientras que este tiempo aumenta bastante cuando el número de reglas es grande.

Con respecto al número de reglas, podemos observar que el algoritmo de refinamiento lo reduce en todas las bases de datos propuestas, lo que conlleva que el conjunto final de reglas sea en términos generales más simple.

Sobre el porcentaje de acierto de los nuevos conjuntos de reglas podemos decir que hay un aumento en términos generales de los mismos, tanto sobre los conjuntos de entrenamiento como sobre los conjuntos de prueba (figura 3.5), aunque dicho aumento es más evidente sobre los primeros. Este hecho es razonable, ya que al igual que la mayoría de los algoritmos de refinamiento de teorías, su objetivos es ajustar los más posible sus reglas al conjunto de entrenamiento. Este comportamiento hace que los algoritmos de refinamiento tiendan de forma desmedida al sobreaprendizaje de los ejemplos del conjunto de entrenamiento, lo que ocasiona en la mayoría de las situaciones una reducción en la capacidad de predicción correcta sobre los ficheros de prueba. Sin embargo, pro-



S1	Algoritmo SLAVE descrito en el capítulo 2
S2	Algoritmo SLAVE descrito en el capítulo 2, incluyendo nuevos operadores genéticos
S3	Algoritmo SLAVE+R

Figura 3.5: Comparativa del acierto en los conjuntos de prueba (3)

Bases de datos	Entrenamiento	Prueba	N.Reglas	Tiempo
IRIS	97.09	94.87	3.6	26
MONK1	100	100	4	28
MONK2	92.90	66.20	16	203
MONK3	98.36	88.89	8	142
INFARTO	96.73	80.86	4.2	125
WINE	100	84.27	9.4	3134
PIMA	83.33	67.71	14	5503
IONOSPHERE	100	79.46	19	25448
SOYBEAN	97.96	92.14	36	30732

Tabla 3.2: Resultados usando el refinamiento como un algoritmo de aprendizaje

baremos experimentalmente, que esta tendencia hacia el sobreaprendizaje se reduce si el algoritmo de refinamiento parte del conjunto de reglas que propone SLAVE. Para ello, debemos resaltar que el algoritmo de refinamiento de teorías se comporta como un algoritmo de aprendizaje si parte inicialmente de una teoria nula, es decir, de un conjunto que no contiene ninguna regla. Así, siguiendo la misma estrategia para la confección de los experimentos, en la tabla 3.2 mostramos los resultados obtenidos por el algoritmo de refinamiento sobre las bases de datos propuestas, si lo utilizamos como un algoritmo de aprendizaje.

Estos resultados ponen de manifiesto lo comentado anteriormente, es decir, un aumento en la capacidad de clasificación sobre el conjunto de entrenamiento, pero una reducción en la capacidad de predicción con acierto sobre los conjuntos de prueba, siendo en algunas bases de datos esta diferencia bastante significativa. Por otro lado, el tiempo de respuesta del algoritmo de refinamiento empeora sobre todo en aquellas bases de datos que requieren más reglas para describir su comportamiento.

El algoritmo de refinamiento, parece un complemento adecuado para el algoritmo SLAVE que proporciona un comportamiento aceptable de los conjuntos de reglas obtenidos, mitigando su tendencia al sobreaprendizaje. En [58] se propone un estudio para comparar como afecta este algoritmo de refinamiento si se incorpora como fase final de algunos algoritmos de aprendizaje clásicos. En concreto el estudio se realizo sobre el algoritmo C4.5 comprobandose que se mejoraban los resultados de dicho algoritmo, pero que estos eran peores que los que propociona SLAVE+R.

Hasta el momento hemos puesto de manifiesto que asociar el algoritmo de refinamiento de teorías a SLAVE provoca en el caso crisp una mejora significativa en la capacidad de clasificación. Ahora, trataremos de comprobar si dicha mejora se refleja tambien cuando trabajamos con bases de datos con consecuente difuso. Así, siguiendo la filosofía anterior, y tomando como base los conjuntos de reglas obtenidos con SLAVE cuyo resultados se mostraron en la tabla 2.9 se propone aplicar a dichos conjuntos de reglas el algoritmo de refinamiento. Los resultados obtenidos se muestran en la tabla 3.3.

Bases de datos	Entrenamiento	Prueba	N.Reglas	Tiempo
SIN	0.0012	0.0011	7	322
PARABOLA	0.0034	0.0039	3	132
3DPARABOLA	0.0017	0.0076	8	10237
MONTE	0.0031	0.0029	5	18401
PENDULO	5.3104	1.21	5	102
RICE	0.0099	0.0112	7.8	305

Tabla 3.3: Resultados de SLAVE+R para el caso difuso

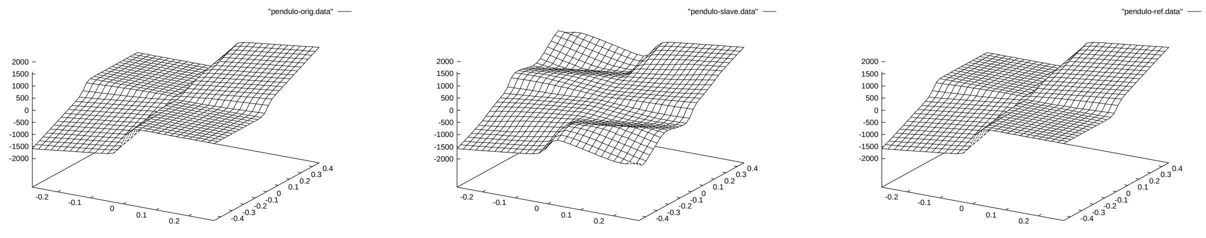
Al igual que sucede en el caso crisp, los resultados muestran una mejora en el comportamiento de los conjuntos de reglas obtenidos, tanto sobre los conjuntos de entrenamiento como sobre los conjuntos de prueba. Además, todas las consideraciones establecidas anteriormente para el caso crisp son aplicables al caso difuso excepto una. A diferencia del caso anterior, el refinamiento para el caso difuso tiene una tendencia a incluir más reglas que las que propone la ejecución aislada de SLAVE. Esta hecho refleja que el sistema de evaluación de reglas incluido en SLAVE tiene problemas para obtener el número adecuado de reglas en el caso difuso. Dejando a un lado esta situación, podemos concluir que asociar, en el caso difuso, el algoritmo de refinamiento a SLAVE produce resultados que son comparables a los que proponen otros algoritmos diseñados específicamente para resolver este tipo de problemas (ver tabla 2.6).

De entre los distintos resultados obtenidos, destacan por su bondad los obtenidos sobre la base de datos PENDULO. Para esta base de datos, SLAVE+R obtiene un número reducido de reglas simples que muestra la tabla 3.4.

1	If θ is NS and ω is NS Then f is NS
2	If θ is PS and ω is PS Then f is PS
3	If ω is ZR Then f is ZR
4	If θ is NM or NS and ω is NM Then f is NM
5	If θ is PS or PM and ω is PM Then f is PM

Tabla 3.4: Conjunto de reglas obtenido por SLAVE+R para el problema del péndulo

Además, podemos comprobar, como muestra la figura 3.6 que dicho conjunto de reglas produce una mejora significativa sobre la obtenida con SLAVE y que se asemeja considerablemente con la función original que describe el comportamiento del péndulo.



(a) Superficie original

(b) Superficie obtenida con SLAVE

(c) Superficie obtenida con SLAVE+R

Figura 3.6: Superficies obtenidas para el problema del péndulo

3.3 Incorporación de colaboración/competición entre reglas en SLAVE

El algoritmo de aprendizaje SLAVE determina la bondad de una regla en términos del número de ejemplos positivos y negativos que recubre dicha regla. Frente a esta posición, el algoritmo de refinamiento de teorías difusas, descrito anteriormente, determina la bondad de una regla en función de su comportamiento en el proceso de inferencia.

En el primer criterio, las reglas se aprenden de forma aislada, mientras que usando el segundo, se establecen relaciones de colaboración/competición entre las reglas, lo cual permite la determinación de un conocimiento más apropiado al problema. Por tanto, parece conveniente, si se quiere dotar a los algoritmos que usan el modelo iterativo de una visión más global durante el proceso de aprendizaje, el establecer relaciones de colaboración/competición entre las reglas. Para dotarlo de esta nueva visión, es necesario que la evaluación de las reglas no tenga en cuenta únicamente el recubrimiento de ejemplos positivos y negativos, sino que además se considere cual será su comportamiento durante el proceso de inferencia.

La incorporación de información sobre el comportamiento de las reglas en el proceso de inferencia no se puede realizar de forma directa en los algoritmos de aprendizaje que utilizan el modelo iterativo, ya que en un estado intermedio del proceso de aprendizaje, sólo se conocen las reglas aprendidas hasta el momento para un subconjunto de conceptos, desconociéndose absolutamente las reglas que se obtendrán para el resto. Por esta razón, no es posible realizar una inferencia de la misma forma como se hace en el algoritmo de refinamiento, donde se conoce el conjunto completo de reglas.

Sin embargo, una vez que se selecciona una regla en el modelo iterativo, se puede determinar como se comportará ésta durante el proceso de inferencia sobre un subconjunto de ejemplos del conjunto de entrenamiento. Si esta información se tiene en cuenta para aprender las siguientes reglas, se dota al modelo de la capacidad de establecer relaciones de colaboración/competición que permiten obtener una descripción más apropiada del sistema. En definitiva, si el algoritmo de refinamiento de teorías supone la introducción de colaboración entre soluciones parciales a través de un postproceso del conjunto de reglas obtenidas por SLAVE, ahora queremos modificar la propia estructura de SLAVE para que pueda incorporar información sobre la posible colaboración/competición de las reglas.

En consecuencia, el planteamiento que se desarrollará en las siguientes subsecciones consiste en modificar el algoritmo de aprendizaje SLAVE para que en la evaluación de las nuevas reglas se contemple la información de las reglas aprendidas previamente. Dado que esta información esta íntimamente relacionada con el mecanismo de inferencia asociado, deberemos definir dos aproximaciones diferentes de SLAVE, una para el caso crisp y otra para el caso difuso.

3.3.1 Colaboración/competición en el caso crisp

El método de inferencia asociado al caso crisp, establece una competición entre las diferentes reglas aplicables para determinar una respuesta. En el modelo iterativo, debido a que las reglas se aprenden de forma aislada, estas adolecen de esta componente competitiva. Este aislamiento provoca que no se tenga en cuenta el comportamiento de las reglas ya aprendidas sobre las nuevas reglas que se deben aprender.

Supongamos que se han aprendido T reglas para un determinado concepto y se desea aprender la siguiente regla T+1 para este mismo concepto. La información que maneja el algoritmo para aprender esta nueva regla se reduce a los ejemplos que las T reglas ya aprendidas no recubrieron. Sin embargo, se desconoce cómo las reglas ya aprendidas para otros conceptos, afectan a estos ejemplos. Además, cuando trabajamos con ejemplos y reglas difusas, la falta de información es más grave, ya que los ejemplos son cubiertos por las reglas en un cierto grado.

Por tanto, para problemas con consecuente crisp es importante conocer como se comportan las reglas ya aprendidas sobre cada ejemplo del conjunto de entrenamiento, para que esta información se tenga en cuenta durante el aprendizaje de nuevas reglas.

Con este fin, definimos el concepto de máximo grado de recubrimiento de un ejemplo por un conjunto de reglas, que nos permite establecer el grado de representación que ofrecen el conjunto de reglas a un ejemplo concreto del conjunto de entrenamiento.

Definición 3.3 *Sea e un ejemplo del conjunto de entrenamiento y $REGLAS$ un conjunto de reglas. Definimos* **el máximo grado de recubrimiento** *del ejemplo e^B como:*

$$Q(REGLAS, e) = \max_{R_B(A) \in REGLAS} \{U(e, A)\}$$

Los valores de $Q(REGLAS, e_i)$ obtenidos para todos los ejemplos e_i del conjunto de entrenamiento E, nos permiten conocer el grado de representación de las reglas sobre el conjunto E. Estas valoraciones obtenidas teniendo en cuenta todas las reglas que componen el conocimiento extraído hasta el momento no aportan demasiada información. Sin embargo, dado que SLAVE fija un concepto para aprender la mejor descripción que lo representa, podemos establecer en base a la definición anterior nuevas valoraciones que son representativas de como se comportará el proceso de inferencia sobre cada uno de los ejemplos. Así, fijado un concepto B, podemos dividir el conjunto de reglas obtenido hasta el momento en dos subconjuntos: el conjunto de reglas que describen el concepto B (R_B) y el conjunto de reglas que describen el resto de los conceptos ($R_{\overline{B}}$), de manera que REGLAS $= R_B \bigcup R_{\overline{B}}$,

La definición anterior, aplicada aisladamente a cada uno de estos dos subconjuntos, nos proporciona información sobre el comportamiento de las reglas sobre cada uno de los ejemplos del conjunto de entrenamiento. Así, si e^B es un ejemplo del conjunto de entrenamiento que pertenece al concepto B, definimos el **máximo grado de recubrimiento positivo** de dicho ejemplo como el máximo grado de recubrimiento que ofrece el subconjunto de reglas R_B sobre el ejemplo e^B, es decir ,

$$Q(R_B, e^B) = \max_{R_B(A) \in R_B} \{U(e^B, A)\}$$

Esta medida nos ofrece el mayor grado de compatibilidad entre el ejemplo e^B y el conjunto de reglas que permiten clasificar correctamente este ejemplo.

En el mismo sentido, podemos definir **el máximo grado de recubrimiento negativo** del ejemplo e^B como

$$Q_{\overline{B}}(R_{\overline{B}}, e^B) = \max_{R_{\overline{B}}(A) \in R_{\overline{B}}} \{U(e^B, A)\}$$

donde $Q_{\overline{B}}(R_{\overline{B}}, e^B)$ representa el mayor valor de compatibilidad entre e^B y el conjunto de reglas que clasificarán incorrectamente a dicho ejemplo.

Estos valores, que se van modificando a medida que se aprenden nuevas reglas durante el proceso iterativo, permiten determinar el comportamiento que las reglas producirán sobre un subconjunto de ejemplos del conjunto de entrenamiento en el proceso de predicción, de manera que si

$$Q_B(R_B, e^B) > Q_{\overline{B}}(R_{\overline{B}}, e^B)$$

podemos decir que con el conjunto de reglas que poseemos actualmente, ese ejemplo será clasificado correctamente, mientras que si

$$Q_B(R_B, e^B) < Q_{\overline{B}}(R_{\overline{B}}, e^B)$$

dicho ejemplo será incorrectamente clasificado.

Pero, ¿qué ocurre si $Q_B(R_B, e^B) = Q_{\overline{B}}(R_{\overline{B}}, e^B)$? En esta situación, se producirá un conflicto durante el proceso de inferencia para determinar la salida de este ejemplo. Por consiguiente, para determinar la regla aplicable, es necesario tener en cuenta el mecanismo de resolución de conflictos que usará este proceso de inferencia.

Como ya se describió anteriormente, el primer criterio utilizado por el mecanismo de resolución de conflictos está asociado al peso de las reglas. Incorporando esta nueva información a cada uno de los ejemplos, podemos introducir las siguientes definiciones:

Definición 3.4 *Sea e^B un ejemplo del conjunto de entrenamiento que describe el concepto B. Sea REGLAS el conjunto de reglas aprendidas hasta el momento, definido como $REGLAS = R_B \bigcup R_{\overline{B}}$, donde R_B es el conjunto de reglas que se han aprendido para el concepto B, mientras $R_{\overline{B}}$ es el conjunto de reglas que se han aprendido del resto de los conceptos.*

- *Diremos que un ejemplo e^B está correctamente clasificado si satisface al menos una de las dos siguientes condiciones:*

  i) $Q_B(R_B, e^B) > Q_{\overline{B}}(R_{\overline{B}}, e^B)$.
  ii) $Q_B(R_B, e^B) = Q_{\overline{B}}(R_{\overline{B}}, e^B)$ *y* $\omega(Q_B(R_B, e^B)) > \omega(Q_{\overline{B}}(R_{\overline{B}}, e^B))$.

- *Diremos que un ejemplo e^B está incorrectamente clasificado si no satisface ninguna de las condiciones anteriores.*

donde $\omega(Q_B(R_B, e^B))$ y $\omega(Q_{\overline{B}}(R_{\overline{B}}, e^B))$ representan el mayor peso de entre las reglas que alcanzan el máximo grado de recubrimiento positivo y negativo respectivamente.

Teniendo en cuenta la información que se introduce en el proceso con las anteriores valoraciones, proponemos la definición de dos nuevos conceptos: aciertos y fallos.

Consideraremos un ejemplo como un acierto para una nueva regla, si el ejemplo es correctamente clasificado, mientras que consideraremos un ejemplo como fallo para una nueva regla, si el ejemplo es incorrectamente clasificado. Antes de establecer una definición formal de estos dos conceptos, es necesario estudiar algunas consideraciones que son provocadas por la dependencia del orden existente durante el proceso de aprendizaje de los conceptos, y que se producen al incluir esta nueva información.

En el capítulo 2 se describió que una de las ventajas que aporta el modelo iterativo, es que es posible el aprendizaje de los conceptos de forma paralela, ya que en principio, no se establece ninguna dependencia entre ellos. Este hecho, implica que no sea relevante el orden en el cual se aprenden los conceptos. Sin embargo, la introducción de información sobre el comportamiento de las reglas en el proceso de inferencia, supone el establecimiento de dependencias entre las descripciones de los conceptos. Por tanto, esta modificación incluida en el método iterativo, provoca que el conjunto de reglas resultante dependa estrechamente del orden en el que los conceptos se aprenden.

Dado que el orden es importante, debemos considerar algunas situaciones que surgen por la propia filosofía del modelo iterativo, que ilustramos con el siguiente ejemplo:

Ejemplo **3.3** *Supongamos que aplicamos un algoritmo de aprendizaje que sigue el modelo iterativo. Supongamos además que se han aprendido las descripciones necesarias para representar un determinado concepto B_1 y se ha obtenido la primera regla para otro concepto B_2. Tomemos de entre los ejemplos del conjunto de entrenamientos dos ejemplos, e^{B_1} que es un ejemplo del concepto B_1 cuyas descripciones ya han sido aprendidas y e^{B_3} que es un ejemplo del concepto B_3 cuyas descripciones no se han aprendido todavia. Supongamos además que debido a la influencia de la única regla del concepto B_2 resulta que los ejemplos e^{B_1} y e^{B_3} son clasificados incorrectamente.*

Ante esta situación, aunque ambos ejemplos son clasificados incorrectamente con el conocimiento obtenido hasta el momento, sabemos que e^{B_1} es un ejemplo incorrectamente clasificado, independientemente del conjunto de reglas que se obtengan para el resto de los conceptos, ya que no será posible incorporar una nueva regla para el concepto B_1. Sin embargo, e^{B_3} es un ejemplo cuya situación de clasificación incorrecta es temporal, ya que existe la posibilidad de que se seleccione una regla para el concepto B_3 que permita clasificarlo correctamente.

Este comportamiento implica que los ejemplos que son correctamente clasificados durante el proceso de aprendizaje, solamente pueden conocerse cuando se obtenga el conjunto definitivo de regla, mientras que algunos de los ejemplos que son incorrectamente clasificados pueden conocerse durante el proceso de aprendizaje, si los ejemplos pertenecen a un concepto que ya ha sido aprendido.

La situación anterior abre las dos siguientes interrogantes:

a) ¿Deben cuantificarse al mismo nivel los ejemplo que con el conocimiento actual son incorrectamente clasificados, que los ejemplos que son de forma definitiva incorrectamente clasificados?

b) ¿No sería apropiado introducir una nueva conducta a la hora de establecer la medida de bondad de las reglas que sea más restrictivo con los ejemplos de los conceptos ya aprendidos y menos exigente con los ejemplos de los conceptos que quedan por aprender?

En la definición de los conceptos de aciertos y fallos, damos respuesta a los dos interrogantes anteriores, penalizando más severamente las incorrecciones definitivas de los ejemplos y optando por un criterio de aprendizaje sobre los conceptos, de manera, que los primeros conceptos que se aprenden tienen una tendencia superior a especificar que los últimos. Así, establecemos las siguientes definiciones:

Definición 3.5 *Sea e^B un ejemplo del conjunto de entrenamiento que verifica pertenecer al concepto B. Sea $REGLAS$ el conjunto de reglas aprendidas hasta el momento, definido como $REGLAS = R_B \bigcup R_{\overline{B}}$, donde R_B es el conjunto de reglas que se han aprendido para el concepto B, mientras $R_{\overline{B}}$ es el conjunto de reglas que se han aprendido del resto de los conceptos.*

- *Diremos que* e^B **es un acierto para la regla** $R_c(d)$ **sii** $c = B$ *y se satisface una de las siguientes condiciones:*

  i) $U(e^B, d) > Q(REGLAS, e^B)$

  ii) $U(e^B, d) = Q(REGLAS, e^B)$ *y* $\omega(R_c(d)) > \omega(Q(REGLAS, e^B))$

- *Diremos que* e^B **es un fallo provisional para la regla** $R_c(d)$ *sii* $c \neq B$ *y se satisface la siguiente condición:*

  i) $U(e^B, d) > Q_B(R_B, e^B)$ *y* $Q_B(R_B, e^B) = 0$

- *Diremos que* e^B **es un fallo definitivo para la regla** $R_c(d)$ *sii* $c \neq B$ *y se satisface una de las siguiente condiciones:*

  i) $U(e^B, d) > Q_B(R_B, e^B)$ *y* $Q_B(R_B, e^B) > 0$

  ii) $U(e^B, d) = Q_B(R_B, e^B)$, $Q_B(R_B, e^B) > 0$ *y* $\omega(R_c(d))) > \omega(Q_B(R_B, e^B)$

Por consiguiente, el sistema determina que un ejemplo es un acierto para la nueva regla que se propone si el ejemplo representa al concepto que se intenta aprender, y dicha regla será la utilizada en el proceso de inferencia para clasificar a este ejemplo, considerando las reglas que se han aprendido hasta el momento. Por contra, se establecen dos definiciones para considerar a un ejemplo como un fallo de la regla. Ambas consideran que, sin ser el ejemplo representativo del concepto que se está aprendiendo, la adaptación entre el ejemplo y la nueva regla es mayor que la adaptación con las reglas de su clase. El elemento diferenciador entre ambas definiciones lo establece el valor $Q_B(R_B, e^B)$. Si $Q_B(R_B, e^B) = 0$ solamente puede indicar dos cosas:

- El conjunto $R_B = \emptyset$, lo que indica, o bien que todavía no se han explorado las reglas para este concepto en el algoritmo de aprendizaje, o bien que se han explorado, pero no se ha encontrado ninguna regla que permita describir a dicho concepto.

- El ejemplo e^B no tiene adaptación con ninguna de las reglas obtenidas para clasificar el concepto B.

En cualquier caso, estos fallos asociados a la regla se consideran provisionales, bien por que todavía no se ha explorado el concepto B, o bien, porque se ha explorado y no ha sido posible representar a este ejemplo.

Por otro lado, si $Q_B(R_B, e^B) > 0$ indica inequivocamente que el concepto B ha sido explorado por el sistema de aprendizaje, y por eso, si la nueva regla tiene una adaptación superior con el ejemplo que la obtenida por las reglas de su clase, este fallo no podrá ser corregido por el algoritmo y será por consiguiente un fallo definitivo.

A pesar de lo dicho anteriormente, no podemos evaluar únicamente la bondad de una regla como relación entre aciertos y fallos por la siguiente razón: la evaluación de una regla se realiza en un punto intermedio del proceso de aprendizaje, por lo que la inclusión de nuevas reglas pueden alterar fuertemente su estimación de aciertos. Por consiguiente, dicha estimación de los aciertos de una regla debe asemejarse lo más posible a la evaluación real de los mismos al final del proceso de aprendizaje. Este hecho pone de manifiesto la necesidad de graduar los aciertos y los errores de una regla en la medida en que estos sean más o menos definitivos. La función de adaptación entre el antecedente de una regla y un ejemplo establece una medida del grado de compatibilidad entre ambos. Cuanto mayor sea este grado de compatibilidad entre ambos, con más certeza podemos asegurar que dicha regla será utilizada en el proceso de predicción para clasificar a dicho ejemplo. Por tanto, utilizaremos esta medida para cuantificar tanto los aciertos como los fallos. En base a lo anteriormente argumentado, establecemos las siguientes definiciones:

Definición 3.6 *Sea E un conjunto de ejemplos de entrenamiento, sea REGLAS el conjunto de reglas aprendidas hasta el momento y sea $R_c(d)$ una nueva regla que se desea incorporar al conjunto REGLAS. Definimos:*

- *Número de aciertos de la regla $R_c(d)$*

$$N^+(R_c(d)) = \sum_{e^B \in E} U(e^B, d) \mid B = c \text{ y } e^B \text{ es un acierto para la regla } R_c(d)$$

- *Número de fallos provisionales de la regla $R_c(d)$*

$$N^-(R_c(d)) = \sum_{e^B \in E} U(e^B, d) \mid B \neq c \text{ y } e^B \text{ es un fallo provisional o definitivo para la regla } R_c(d)$$

- *Número de fallos definitivos de la regla $R_c(d)$*

$$ND^-(R_c(d)) = \sum_{e^B \in E} \Upsilon \mid B \neq c \text{ y } e^B \text{ es un fallo provisional o definitivo para la regla } R_c(d)$$

donde e^B es un ejemplo del conjunto de entrenamiento que representa al concepto B y $\Upsilon \in [0,1]$ es un valor constante para la penalización de los errores definitivos.

Para ilustrar las anteriores definiciones, proponemos el siguiente ejemplo:

Ejemplo 3.4 *Sea $E = \{e_1^{B_1}, e_2^{B_1}, e_3^{B_1}, e_1^{B_2}, e_2^{B_2}, e_3^{B_2}, e_4^{B_2}, e_1^{B_3}, e_2^{B_3}\}$ un conjunto de entrenamiento que describe tres clases diferentes B_1, B_2 y B_3. Sea REGLAS el conjunto de*

	$e_1^{B_1}$	$e_2^{B_1}$	$e_3^{B_1}$	$e_1^{B_2}$	$e_2^{B_2}$	$e_3^{B_2}$	$e_4^{B_2}$	$e_1^{B_3}$	$e_2^{B_3}$
Q_B	0.3	0.8	0.3	0.1	0.7	0.0	0.0	0,0	0.0
$\omega(Q_B)$	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.0	0.0
$Q_{\overline{B}}$	0.2	0.5	0.8	0.4	0.3	0.5	0.6	0.0	0.0
$\omega(Q_{\overline{B}})$	0.6	0.6	0.6	0.6	0.6	0.6	0.0	0.0	0.0

Tabla 3.5: Grados de recubrimiento y peso de las reglas asociados a cada ejemplo de E dado el conjunto de reglas REGLAS

reglas obtenido hasta el momento y supongamos que los valores de los máximos grados de recubrimiento positivo y negativo, junto con los respectivos pesos asociados a estos valores para cada ejemplo son los que se proponen en la tabla 3.5.

Con estas condiciones, queremos evaluar una nueva regla $R_{B_2}(A)$ para la clase B_2 asumiendo que todas las reglas necesarias para aprender el concepto B_1 ya se han aprendido y se encuentran en el conjunto REGLAS. La compatibilidad de cada uno de los ejemplos con esta nueva reglas y el peso de la misma se muestra en la tabla 3.6.

	$e_1^{B_1}$	$e_2^{B_1}$	$e_3^{B_1}$	$e_1^{B_2}$	$e_2^{B_2}$	$e_3^{B_2}$	$e_4^{B_2}$	$e_1^{B_3}$	$e_2^{B_3}$
$U(e_i, A)$	0.3	0.6	0.4	0.6	0.5	0.7	0.3	0.5	0.0
$\omega(R_{B_2}(A))$	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.0

Tabla 3.6: Compatibilidad de cada ejemplo con la regla $R_{B_2}(A)$.

Para determinar los valores $N^+(R_{B_2}(A))$, $N^-(R_{B_2}(A))$ y $ND^-(R_{B_2}(A))$ necesitamos establecer previamente los aciertos y los errores que provoca la inclusión de la nueva regla. Así, tenemos que:

- *El ejemplo $e_1^{B_1}$ no es un fallo para la nueva regla, ya que aunque el máximo grado de recubrimiento positivo coincide con el grado de compatibilidad de esta regla, el peso asociado a la primera es superior. Por tanto, la inclusión de esta regla, no impide que este ejemplo sea correctamente clasificado.*
- *Tampoco el ejemplo $e_2^{B_1}$ no es un fallo para esta regla, debido a que la adaptación con una regla de su clase es mejor que la que proporciona esta nueva regla.*
- *Por contra, el ejemplo $e_1^{B_3}$ si es un fallo que introduce esta regla, ya que no es de la clase de la regla que se introduce, y la adaptación de esta es mejor que la que se tenía sobre dicho ejemplo.*
- *El ejemplo $e_3^{B_1}$ aunque debería ser un fallo para esta regla, no es contabilizado como tal, ya que por la configuración de valores y por ser B_1 una clase ya aprendida, este ejemplo ya ha sido computado como un error definitivo para una regla anterior de*

la clase B_2. Por esta razón, los errores definitivos no se toman en consideración para evaluar la bondad de las nuevas reglas.

- *Los ejemplos $e_1^{B_2}$ y $e_3^{B_2}$ son aciertos de esta regla, por mejorar la anterior compatibilidad que ofrecian las reglas de su clase, y superar la compatibilidad que ofrecen las reglas del resto de las clases.*
- *Los ejemplos $e_2^{B_2}$ y $e_4^{B_2}$ no pueden computarse como aciertos para esta regla, ya que para el primero de ellos existe otra regla de su clase que lo representa mejor y para el segundo, el grado de compatibilidad negativo es superior al que ofrece esta regla.*

En definitiva, tenemos que $e_1^{B_2}$ y $e_3^{B_2}$ son aciertos para esta regla, y $e_1^{B_3}$ es el fallo que introduce esta regla. Además, debemos considerar que el fallo que introduce la regla es provisional, ya que las reglas para representar las clase B_3 no se han aprendido todavía.

Una vez establecidos los aciertos y los fallos, podemos calcular el número de aciertos y fallos que introduce la regla de la siguiente forma:

$$N^{+}(R_{B_2}(A)) = U(e_1^{B_2}, A) + U(e_4^{B_2}, A) = 0.6 + 0.7 = 1.3$$

$$N^{-}(R_{B_2}(A)) = U(e_1^{B_3}, A) = 0.5$$

$$ND^{-}(R_{B_2}(A)) = 0$$

Si suponemos que la regla $R_{B_2}(A)$ es la mejor solución parcial de segundo orden y debe incluirse en el conjunto REGLAS, entonces la tabla anterior se modificaría tal como muestra la table 3.7.

	$e_1^{B_1}$	$e_2^{B_1}$	$e_3^{B_1}$	$e_1^{B_2}$	$e_2^{B_2}$	$e_3^{B_2}$	$e_4^{B_2}$	$e_1^{B_3}$	$e_2^{B_3}$
Q_B	0.3	0.8	0.3	0.6	0.7	0.7	0.3	0,0	0.0
$\omega(Q_B)$	0.7	0.7	0.5	0.7	0.5	0.5	0.5	0.0	0.0
$Q_{\overline{B}}$	0.3	0.6	0.8	0.4	0.3	0.5	0.6	0.5	0.0
$\omega(Q_{\overline{B}})$	0.5	0.5	0.6	0.6	0.6	0.6	0.6	0.5	0.0

Tabla 3.7: Modificación de la tabla de recubrimientos

Tras la modificación, mirando la tabla de recubrimientos 3.7 podemos determinar que ejemplos serán correctamente clasificados si usamos el conjunto REGLAS obtenido hasta el momento. Si el máximo grado de recubrimiento positivo es mayor que máximo grado de recubrimiento negativo para un ejemplo o si ambos grados son iguales, pero el peso asociado al positivo es mayor, podemos asegurar que ese ejemplo se clasificará correctamente. Por contra, si las anteriores condiciones no se satisfacen, entonces dicho ejemplo no será correctamente clasificado.

El hecho de considerar las medidas definidas anteriormente para evaluar la bondad de una regla, provoca dos ventajas sustanciales sobre la anterior versión del algoritmo de aprendizaje SLAVE:

a) Los ejemplos sólamente se computan como positivos o negativos para una regla, cuando estos causan un cambio significación en el proceso de clasificación teniendo en cuenta las reglas que se aprendieron previamente.

b) El algoritmo de aprendizaje, manteniendo la información anterior, conoce hasta donde puede recubrir los ejemplos negativos para que estos no sean considerados como fallos o errores en el proceso de clasificación. Esta información dota al algoritmo de la capacidad de conocer hasta donde puede generalizar las reglas sin perjudicar la clasificación de los ejemplos, por consiguiente, esta forma de evaluar permite establecer límites a la generalización de las reglas.

Así, consideramos la inclusión de estas medidas para definir la función de evaluación de reglas en el algoritmo genético, alterando la definición del primero de sus criterios. Hasta este momento, hemos considerado como primer criterio de evaluación el producto entre el grado de consistencia débil de la regla por su grado de completitud. La definición anterior tomaba en consideración el número de ejemplos positivos y negativos para determinar el grado de consistencia. Ahora, alteramos dicha definición, cambiando el número de ejemplos positivos por el número de aciertos de la regla, y el número de ejemplos negativos por el número de fallos de la siguiente forma:

$$\Gamma^N_{k_1 k_2}(R_B(A)) = \begin{cases} 1 & \text{si } R_B(A) \in \Delta^{k_1} \\ \frac{k_2 N^+(R_B(A)) - (N^-(R_B(A)) + ND^-(R_B(A)))}{N^+(R_B(A))(k_2 - k_1)} & \text{si } R_B(A) \notin \Delta^{k_1} \text{ y } R_B(A) \in \Delta^{k_2} \\ 0 & \text{en otro caso} \end{cases}$$

donde $k_1, k_2 \in [0,1]$ y $k_1 < k_2$ son los parámetros de consistencia débil, $N^+(R_B(A))$, es el número de aciertos, $N^-(R_B(A))$ es el número de fallos y $ND^-(R_B(A))$ es el número de fallos definitivos de la regla $R_B(A)$. Además, en esta definición modificamos el concepto de conjunto de descripciones admisibles Δ^k de la siguiente manera:

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

Debemos destacar que esta nueva definición del grado de consistencia débil conserva todas las propiedades de la definición anterior de la misma que se describieron en el capítulo anterior.

Por consiguiente, la modificación introducida en el primer criterio de evaluación de reglas queda como $(\Gamma^N_{k_1,k_2}(R) \times \Lambda(R))$, manteniendose la misma definición para el resto de los criterios.

Bases de datos	Entrenamiento	Prueba	N.Reglas	Tiempo
IRIS	97.83	95.93	4.8	74
MONK1	100	100	4.2	36
MONK2	94.67	83.19	14.2	125
MONK3	99.02	92.12	7.2	55
INFARTO	99.86	84.62	4.2	148
WINE	99.67	95.79	5	612
PIMA	85.45	76.56	31	2661
IONOSPHERE	99.84	73.53	53.6	9698
SOYBEAN	96.15	96.90	30	10190

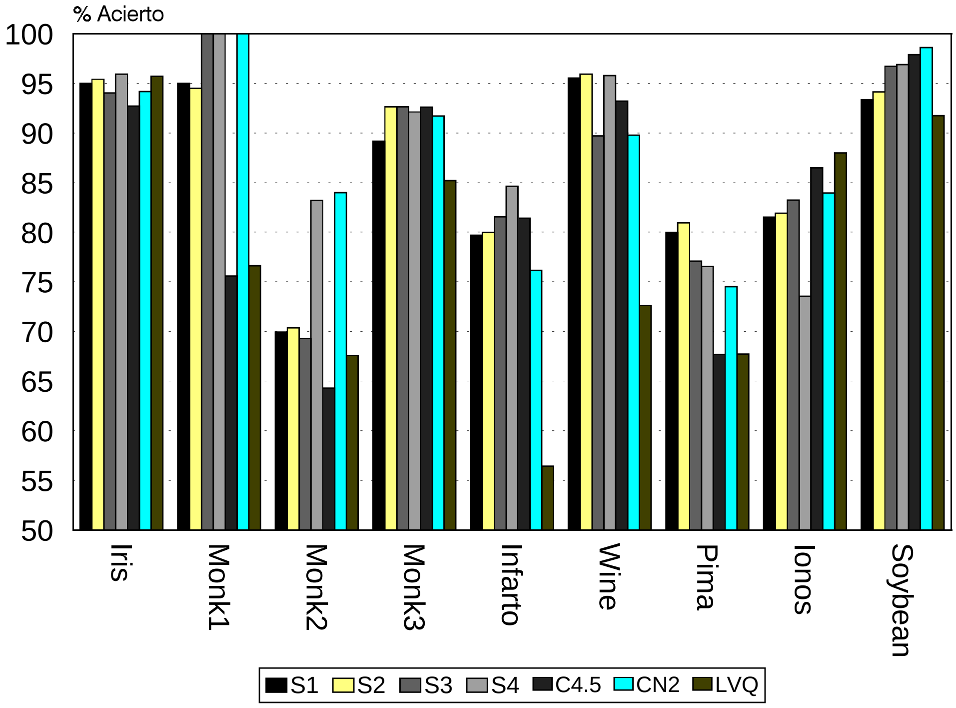
Tabla 3.8: Resultados con la nueva versión de SLAVE

3.3.1.1 Pruebas experimentales

A continuación, estudiaremos empíricamente los efectos que produce esta modificación con respecto a la versión anterior de SLAVE y que ventajas e inconvenientes introduce con respecto a la inclusión como paso final del proceso de aprendizaje de un algoritmo de refinamiento de teorías. Con tal fin, realizamos la ejecución de la nueva versión de SLAVE tomando como parámetros del mismo los que se propusieron en la tabla 2.2 incluyendo como constante de penalización de los errores definitivos $\Upsilon = 1$. Los resultados de la experimentación se muestran en la tabla 3.8.

De la comparación de estos resultados con los obtenidos por SLAVE+R, que se muestran en la tabla 3.1, podemos extraer las siguientes conclusiones:

- Aparece una significativa mejora en la capacidad de predicción con acierto del sistema sobre los conjuntos de prueba, si exceptuamos la base de datos IONOSPHERE (figura 3.7).
- El porcentaje de acierto sobre los conjuntos de entrenamiento son bastante parecidos, lo que indica que la modificación incluida en SLAVE potencia su capacidad de representación sobre el conjunto de entrenamiento a diferencia de la versión anterior.
- La nueva versión necesita un mayor número de reglas para representar los conceptos. Este hecho se hace patente fundamentalmente en aquellas bases de datos donde se encuentran involucradas muchas variables de entrada. En este sentido, parece que el sistema tiene problemas para generalizar las reglas, cuando aparece un gran número de ellas. En el siguientes capítulo estudiamos incorporar una mejora a SLAVE que le permita realizar una selección más adecuada de las variables.
- Sin lugar a dudas, la mejora más sustancial sobre la anterior versión de SLAVE y sobre SLAVE+R viene dada por el tiempo de respuesta del nuevo algoritmo, que se reduce de forma sustancial.



S1	Algoritmo SLAVE descrito en el capítulo 2
S2	Algoritmo SLAVE descrito en el capítulo 2, incluyendo nuevos operadores genéticos
S3	Algoritmo SLAVE+R
S4	La nueva propuesta de SLAVE

Figura 3.7: Comparativa del acierto en los conjuntos de prueba (4)

Básicamente, podemos considerar que la modificación incluida en SLAVE mejora en todos los aspectos a las versiones propuestas anteriormente excepto en el número de reglas. Por esta razón, desde este momento asumiremos para el resto del desarrollo de esta memoria, que cuando nos refiramos al algoritmo de aprendizaje SLAVE para el caso crisp, estaremos aludiendo a la modificación propuesta en la evaluación de la bondad de las reglas. Además, debemos destacar que el algoritmo de refinamiento de teorías expuesto en este capítulo sigue considerandose como una alternativa válida para el refinamiento de las reglas obtenidas por cualquier algoritmo de aprendizaje, y en concreto con la nueva versión propuesta para SLAVE.

3.3.2 Información necesaria para consecuente continuo

Como mencionamos anteriormente, el modelo de aprendizaje iterativo consiste en dado un concepto o valor de la variable consecuente, buscar la mejor descripción que permita representar a dicho consecuente. Como hemos visto anteriormente, cuando el dominio de la variable consecuente no contiene solapamiento entre sus distintos valores, el proceso de inferencia asociado establece una competición entre las reglas para determinar la salida del sistema.

Sin embargo, en aquellos problemas que presentan solapamiento en los valores de la variable consecuente, el proceso de aprendizaje no sólo debe establecer competición entre los distintos conceptos, sino que en algunos casos, se hace necesario establecer relaciones de competición entre ellos. En el siguiente ejemplo, ilustramos esta situación.

Ejemplo 3.5 *Supongamos que se desea aprender el conjunto de reglas que permita clasificar unas determinadas enfermedades en función de un conjunto de síntomas. Supongamos además que la variable consecuente, es una variable cuyos valores tienen la siguiente estructura:*

Cardiaca

Infarto

Soplo

Respiratoria

Gripe

Resfriado

En base a esta estructura, parece evidente que se deben establecer dos tipos de criterios en el proceso de aprendizaje. Por ejemplo, se debe establecer una competición entre valores tales como Cardiaca *y* Respiratoria*, o entre* Infarto *y* Soplo *o entre* Gripe *y* Resfriado*, mientras que sobre los valores* Cardiaca *e* Infarto *se debe establecer un colaboración entre las descripciones para ambos conceptos.*

En base a lo anterior, podemos decir que un ejemplo existente en el conjunto de entrenamiento que describa el concepto Infarto *no debe ser considerado como un ejemplo negativo al concepto* Cardiaca*. Por consiguiente, el proceso de aprendizaje debe tener en cuenta que todo* Infarto *es una enfermedad* Cardiaca*, y por tanto, para aprender esta última, será necesario que se establezca una competición entre ambos conceptos.*

Esta diferencia en el planteamiento del aprendizaje de los diferentes conceptos, permite estar más cerca de extraer un conocimiento válido para el problema, si el proceso de inferencia asociado presenta una actitud semejante.

Para aclarar aún más este punto, proponemos el siguiente ejemplo:

Ejemplo 3.6 *Supongamos que se desea obtener el conocimiento necesario para describir un sistema, donde el dominio de la variable consecuente está discretizado mediante cuatro intervalos con solapamiento como muestra la figura 3.8.*

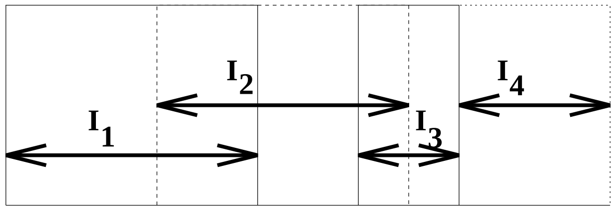


Figura 3.8: Dominio intervalar con solapamientos

Si deseamos aprender las reglas que permiten describir el intervalo I_1, podemos optar por dos comportamientos:

1. *Considerar como aciertos todos los ejemplos cuyo valor de la variable consecuente pertenezca el intervalo* I_1*, y considerar como errores todos los ejemplos* e_i *tal que* $e_i \in I_j$*, donde* $j = 2, 3, 4$.

2. *Considerar como aciertos todos los ejemplos cuyo valor de la variable consecuente pertenezca el intervalo* I_1*, y considerar como errores todos los ejemplos que no pertenezcan a este intervalo.*

Con la primera interpretación, estamos caracterizando los ejemplos que pertenecen al intervalo I_1 *que no pertenecen al intervalo* I_2*, es decir, se establece una compentencia directa entre estos dos intervalos. Este comportamiento, que es apropiado cuando no existe solapamiento entre los valores de la variable consecuente, provoca que el algoritmo de aprendizaje no tome ninguna postura sobre los ejemplos que pertenezcan a* $I_1 \bigcup I_2$.

Por contra, la segunda interpretación considera que los ejemplos que pertenecen a $I_1 \bigcup I_2$*. son ejemplos positivos tanto para el intervalo* I_1 *como para el intervalo* I_2.

La segunda de las interpretaciones es la más razonable en estos casos, que además coincide con la filosofia original de SLAVE: "Encontrar la mejor descripción que representa a un concepto". Sin embargo, en la propuesta inicial de SLAVE (que se desarrolló en el capítulo anterior) solamente se distinguía entre caso crisp y caso difuso durante el proceso de inferencia. Esta cirscunstancia, provoca que el proceso de aprendizaje utilizado para el caso difuso, mantiene una actitud de competición entre los conceptos, mientras que el proceso de inferencia requiere de la colaboración de las reglas para obtener una salida correcta.

Por consiguiente, ahora nos centraremos en corregir esta deficiencia en la concepción del aprendizaje para el caso difuso, readaptando el algoritmo para que además mantenga información del comportamiento del sistema durante el proceso de inferencia.

3.3.2.1 Aciertos y fallos en el caso difuso

Anteriormente, hemos ilustrado como debe comportarse el algoritmo de aprendizaje cuando existe algún tipo de solapamiento entre los valores de la variable consecuente. Este comportamiento es similar cuando el dominio de la variable consecuente se componen de un conjunto de etiquetas difusas. Al igual que en el caso crisp, bajo estas premisas, podemos establecer los conceptos de acierto y fallo en la clasificación de un concepto.

La segunda interpretación del ejemplo anterior, nos da la idea básica para la descripción de estos dos nuevos conceptos.

Definición 3.7 *Diremos que una regla* $R_B(A)$ *acierta en la clasificación de un ejemplo* e*, si y solo si, tanto la descripción de la regla como el consecuente de la misma tiene adaptación con el ejemplo, es decir,*

$$U(e, A) > 0 \text{ y } U(e, B) > 0$$

Definición 3.8 *Diremos que una regla $R_B(A)$ falla en la clasificación de un ejemplo e, si y solo si, la descripción de la regla tiene adaptación con el ejemplo y el ejemplo no tiene adaptación con el consecuente de la misma es decir,*

$$U(e, A) > 0 \ y \ U(e, B) = 0$$

A partir de las definiciones anteriores, podemos establecer las definiciones de número de aciertos y número de fallos de una regla como la suma de los mismos de manera que dado un conjunto de ejemplos E y una regla $R_B(A)$, definimos el número de aciertos de la regla sobre el conjunto de ejemplos, como el cardinal del conjunto de aciertos, es decir,

$$N_{ac}^{+}(R_B(A)) = |\{e \in E \mid U(e, A) > 0 \text{ y } U(e, B) > 0\}|$$

y definimos el número de fallos de la regla sobre el conjunto de ejemplos, como el cardinal del conjunto de fallos, es decir,

$$N_{fa}^{-}(R_B(A)) = |\{e \in E \mid U(e, A) > 0 \text{ y } U(e, B) = 0\}|$$

Al igual que en la modificación propuesta para el caso crisp, el número de aciertos y errores de las reglas permiten determinar la bondad de la regla. A diferencia de la propuesta anterior, en el caso difuso, no es necesario graduar la confianza del acierto, ya que las reglas no compiten entre sí, sino que colaboran para obtener la salida del sistema.

Para el caso crisp, la modificación de SLAVE considera como una información muy importante el peso de las reglas. En ese contexto, el peso permite determinar la importancia de la regla y entra a formar parte como un elemento competitivo fundamental en el mecanismo de resolución de conflictos. En el caso difuso, este elemento deja de ser relevante, al obtenerse la salida en base a la competición de las distintas reglas aplicables. En esta situación, el elemento importante a considerar del proceso de inferencia es el error que se comete en las salidas del sistema. Este elemento para a formar parte en un segundo término, de la evaluación de la bondad del conjunto de reglas obtenido hasta el momento. De tal forma, que para la evaluación de las reglas se utiliza una EFL cuyo primer criterio se rige por el grado de satisfacción de la consistencia y completitud aplicados a los aciertos y fallos de las reglas, y como segundo criterio el error cometido hasta el momento por las salidas del sistema.

Al igual que sucede con el caso crisp, no es posible aplicar directamente el proceso de inferencia en el mismo sentido en el que se usa en el algoritmo de refinamiento de teorías, ya que sólo se conoce una porción de reglas que describen el sistema. Por esta razón, para determinar la medida del error, únicamente se toman del conjunto de entrenamiento, aquellos ejemplos que tienen alguna adaptación con los conceptos que han sido aprendidos. Por consiguiente y a semejanza del caso crisp, el orden en el cual se aprenden los conceptos es importante, pero a diferencia de este, donde es posible estudiar diferentes ordenes debido a la independencia de los conceptos, en este caso únicamente dos ordenes parecen apropiados, el considerar el orden creciente de las etiquetas, y el decreciente. En SLAVE asumiremos que se usa el orden creciente de las etiquetas.

Asumiendo que las etiquetas difusas se aprenden de forma creciente, y estamos intentando clasificar los ejemplos de la etiqueta t-ésima de la variable consecuente, podemos aplicar el método de inferencia sobre el conjunto de reglas obtenido hasta el momento para conocer el error que se comete sobre aquellos ejemplos que pertenecen a las t primeras etiquetas. Actuando de esta manera, cada vez que se aprende una nueva regla, no sólo se intenta máximizar el número de aciertos, sino que además el algoritmo trata de encontrar la regla que proporcionando el mismo número de aciertos, reduce el error con respecto a los ejemplos asociados a las etiquetas del consecuente que ya han sido aprendidas. Sin embargo, para que el proceso de reducción del error tenga un comportamiento más flexible y permita una mejora sustancial del mismo, es necesario que aquellos ejemplos que tienen pertenecia con una etiqueta de orden inferior a la que se intenta aprender y que tiene adaptación con el antecedente de la regla que se propone no sean considerados como error. Para aclarar este último extremo proponemos el siguiente ejemplo:

Ejemplo 3.7 *Supongamos que en el conjunto de entrenamiento tenemos un ejemplo e que tiene adaptación únicamente con dos etiquetas de la variable consecuente L_1 y L_2. Supongamos además que el algoritmo de aprendizaje obtuvo una regla que lo consideraba como un acierto para la etiqueta L_1, pero sin embargo, para ninguna de las reglas obtenidas de la etiqueta L_2 este ejemplo fué considerado como un acierto. Si el algoritmo ahora intenta aprender una nueva regla para la etiqueta L_3, es posible que si esta nueva regla tiene adaptación con el ejemplo anteriormente mencionado, la diferencia se reduzca entre la salida que propociona el conjunto de reglas y la salida real para este ejemplo. Como el ejemplo no pertenece en ningún grado a la etiqueta L_3, por tanto, dicho ejemplo debería considerarse como un fallo para la nueva regla, pero este hecho ocasiona que la incorporación de ninguna regla pueda reducir el error que produce este ejemplo. Por esta razón, los ejemplos con pertenencia a una etiqueta de orden menor, no se consideran como ejemplos.*

También podría suceder que el error al incluir la nueva regla aumentase. En dicho caso, el propio sistema tiene recursos para intentar obtener otra regla que no recubriera a este ejemplo, a menos que este hecho sea imposible. Por consiguiente, penalizar a estos ejemplos va en perjuicio de la capacidad de representación que es capaz de obtener el algoritmo sobre el sistema que se desea aprender.

En definitiva, la nueva versión de SLAVE para el caso difuso conlleva dos modificaciones sobre la evaluación de la bondad de la regla. La primera de ellas, al igual que la modificación para el caso crisp, se sustituyen los valores de número de ejemplos positivos y número de ejemplos negativos por el número de aciertos y número de fallos en el caso difuso respectivamente para obtener el grado de consistencia débil de la regla. La segunda modificación involucra la inclusión de un nuevo criterio que mide el error cometido en las salidas hasta el momento. La evaluación de este criterio se considera el más importante

Bases de datos	Entrenamiento	Prueba	N.Reglas	Tiempo
SIN	0.0016	0.0014	7	26
PARABOLA	0.0030	0.0037	3	12
3DPARABOLA	0.0026	0.0048	7	597
MONTE	0.0021	0.0020	13	971
PENDULO	0.1204	0.1680	6	38
RICE	0.0052	0.0112	7.5	77

Tabla 3.9: Resultados de SLAVE para el caso difuso con la nueva función de evaluación de reglas

en la aplicación de la EFL después del producto entre el grado de consistencia débil y el grado de completitud.

3.3.2.2 Pruebas experimentales

A continuación mostramos las pruebas experimentales realizadas para comprobar el comportamiento del nuevo sistema. En la tabla 3.9 se muestran los resultados obtenidos tras la ejecución de SLAVE sobre las diferentes bases de datos propuestas.

Comparando estos resultados con los obtenidos por el algoritmo WM (tabla 2.6) podemos observar que para las bases de datos seleccionadas la modificación incluida en SLAVE obtiene mejores resultados, es decir, reduce de forma significantiva el error. Además, el número de reglas necesario para representar el conocimiento aprendido es también también menor. Así, si la versión inicial de SLAVE tan sólo mostraba buenos resultados en el caso crisp, la nueva versión del sistema tiene buenos resultados tanto en el caso crisp como en el caso difuso. Obviamente mejora a la versión previa de SLAVE, los resultados son mejores que los obtenidos por otros algoritmos sobre las bases de datos de prueba.

Por otro lado, si comparamos estos resultados con los obtenidos por SLAVE+R (tabla 3.3) no podemos afirmar de forma clara, que uno de ellos tenga un comportamiento mejor que el otro. Sin embargo, si asumimos que la diferencia en cuanto al error cometido entre ambos algoritmos no es significativo, entonces, en este caso, el tiempo de respuesta que ofrece esta versión modificada de SLAVE es significativamente menor.

Por tanto, al igual que se comento anteriormente, tampoco podemos descartar el algoritmo de refinamiento de teorías, como una vía válida para mejorar el comportamiento del conjunto de reglas, también en el caso de problemas con consecuente difuso. Sin embargo, nos fijamos como objetivo para el resto de memoria, mejorar en la medida de lo posible el funcionamiento de esta nueva propuesta en la evaluación de las reglas que se ha incluido en el algoritmo de aprendizaje SLAVE.