Capítulo 4

El problema de la selección de características

4.1 La selección de características

El objetivo de los algoritmos de aprendizaje inductivos consiste en establecer una hipótesis que sea capaz de explicar el comportamiento de un sistema. Los algoritmos clásicos de aprendizaje automático suelen tener asociados al proceso de inducción criterios que tratan de encontrar estructuras no excesivamente complejas que permitan describir correctamente al sistema. Los dos criterios más comunes son los basados en la filosofía *Occam's Razor* [8] o en la obtención de *la descripción de longitud mínima* [133].

Con la inclusión de estos criterios, se intenta obtener un comportamiento ideal del algoritmo inductivo, de manera que este sea capaz de seleccionar el subconjunto de características de entrada que proporcione la mejor descripción del sistema. Encontrar la estructura minimal es en la mayoría de los casos un problema NP-duro. Por tanto, los algoritmos que tratan este problema, son conducidos por funciones heurísticas a través del espacio de las hipótesis posibles.

Abordando el problema de esta manera, la selección de características implica encontrar un conjunto de características "bueno" bajo algún punto de vista y debe ser valorado mediante una función objetivo. Normalmente, estas funciones establecen la bondad en términos del tamaño de las estructuras obtenidas para representar la conducta del sistema.

Por consiguiente, la selección de las características relevantes y la eliminación de las irrelevantes es uno de los problemas centrales de los algoritmos de aprendizaje automático [95]. La detección de las variables irrelevantes conlleva no sólo la reducción del espacio de búsqueda de las posibles descripciones válidas para representar los conceptos, sino además la obtención de soluciones más comprensibles.

Planteada la selección de características como un problema de búsqueda heurística, cada estado en el espacio de búsqueda especifica un subconjunto de variables posibles.

La estructura de este espacio, según se propone en [95], sugiere un método que debe tomar cuatro decisiones básicas para determinar la naturaleza del proceso de búsqueda heurístico.

La primera de ellas, debe determinar el punto de partida en el espacio de búsqueda, que establece la dirección que inicialmente tomará la búsqueda. La segunda de las decisiones está relacionada con la organización de la búsqueda. Dado que una búsqueda exhaustiva del espacio es imposible, debido a la explosión combinatoria del mismo, se han propuesto aproximaciones más aplicables a este problema. Entre ellas, podemos mencionar las técnicas greedy, o métodos de escalada que permiten moverse en el espacio en base a decisiones locales. La tercera determina la estrategia utilizada para evaluar las diferentes alternativas de subconjuntos de variables. Básicamente, podemos considerar dos métodos generales, los métodos por filtro y los métodos por envoltura. Los primeros determinan el conjunto de variables relevantes antes de aplicar el proceso de inducción, mientras que los segundos generan un conjunto de variables candidatas durante el proceso de inducción, y en base a los resultados obtenidos evalúan el conjunto de variables.

La última decisión hace referencia a los criterios que deben ser definidos para detener el proceso de búsqueda, determinando el subconjunto de variables relevantes al problema.

De lo anterior se deduce que la selección de un conjunto minimal de características está estrechamente relacionado con el concepto de relevancia. En la literatura podemos encontrar diversas definiciones de este concepto. Por ejemplo la definición siguiente fue propuesta en [1] y define la relevancia para variables que toman valores booleanos, asumiendo que no existe ruido.

Definición 4.1 *Diremos que una variable X_i es relevante para un determinado concepto C, si X_i aparece en toda fórmula booleana que representa a C. En otro caso, diremos que es irrelevante.*

Frente a la definición anterior, en [87] se define de forma más general el concepto de relevancia, en base a la siguiente afirmación: *Las características son relevantes si sus valores varían sistemáticamente por la pertenencia a una clase.*

Definición 4.2 *Diremos que X_i es relevante si y sólo si existen algunos x_i e y para los cuales $p(X_i = x_i) > 0$ y*

$$p(Y = y|X_i = x_i) \neq p(Y = y).$$

donde x_i es un valor del dominio de la variable X_i, siendo Y la variable concepto e y uno de los valores que puede tomar dicha variable.

Esta definición establece que una variable es relevante si el hecho de conocer el valor que se le asigna cambia la estimación para la variable concepto. En definitiva, la relevancia se relaciona con la dependencia condicionada entre la variable de clasificación y la variable predictiva.

La anterior definición falla en algunas situaciones para obtener la relevancia y se puede mejorar de la siguiente forma:

Definición 4.3 *Sea S_i el conjunto formado por todas las características excepto X_i, es decir, $S_i = X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_m$, y sea s_i una asignación de valores para todas las características de S_i.*

Diremos que X_i es relevante si y sólo si existen algunos x_i, s_i e y para los cuales $p(X_i = x_i, S_i = s_i) > 0$ y

$$p(Y = y|X_i = x_i, S_i = s_i) \neq p(Y = y|S_i = s_i).$$

Es decir, X_i es relevante si la probabilidad para alguna de sus posibles asignaciones, dada el resto de las características, puede cambiar cuando eliminamos el conocimiento sobre el valor de la variable X_i.

Esta última definición está propuesta en [87]. En este trabajo se define la relevancia desde dos puntos de vista: la anterior que se denomina *relevancia fuerte* y una nueva definición más relajada de relevancia, denominada *relevancia débil*.

Definición 4.4 *Una característica X_i es débilmente relevante si y sólo si no es fuertemente relevante y existe un subconjunto de características S_i^1 de S_i y algún x_i, s_i^1 e y con $p(X_i = x_i, S_i^1 = s_i^1) > 0$ tal que*

$$p(Y = y|X_i = x_i, S_i^1 = s_i^1) \neq p(Y = y|S_1^1 = s_i^1).$$

La relevancia fuerte implica que la característica es indispensable ya que si no se considera para la descripción del sistema, se perderá capacidad de predicción. Por el contrario, la relevancia débil implica que la característica puede contribuir algunas veces para aumentar la capacidad de predicción.

SLAVE aborda la selección de características manteniendo básicamente la idea propuesta en la definición de relevancia débil, ya que el algoritmo afronta la determinación de la relevancia o irrelevancia de las características desde un punto de vista local, sin tener en cuenta cómo estas características incidieron en las descripciones previamente aprendidas, ni de como incidirán en las que quedan por aprender. Por tanto, en el planteamiento de SLAVE, las variables relevantes se buscan individualmente para cada regla, de manera que, fijado un concepto a aprender, el algoritmo trata de encontrar el menor subconjunto de variables y la mejor asignación de valores a dichas variables para describir dicho concepto.

En comparación con otros modelos genéticos para el aprendizaje de reglas, el modelo iterativo de SLAVE permite obtener respuestas rápidas debido a la reducción del espacio de búsqueda. A pesar de ello, el problema del tiempo de respuesta continúa cuando se tiene un número alto de variables antecedente posibles. En estos casos el espacio de búsqueda sigue siendo excesivamente grande.

El algoritmo genético de SLAVE elimina variables irrelevantes para un valor fijo de la variable de clasificación. Sin embargo, la codificación de las variables irrelevantes permanece en el cromosoma y está expuesta a modificaciones provocadas por los operadores genéticos. Con esta situación, al algoritmo le cuesta un gran esfuerzo evolutivo determinar que una variable es irrelevante. Por el contrario, el esfuerzo genético para interpretar una variable irrelevante como relevante es muy pequeño.

Esta diferencia de esfuerzo se incrementa exponencialmente con el número de valores que constituyen los dominios de las variables, y con el número de variables que intervienen en el problema.

En este capítulo nos planteamos una modificación del algoritmo genético de SLAVE para aumentar su capacidad para determinar variables irrelevantes y mejorar su tiempo de respuesta para encontrar un subconjunto de variables relevantes.

4.2 Modelos de selección de subconjuntos de características

Según la clasificación que propone Langley en [95], atendiendo a la estrategia utilizada para evaluar las distintas alternativas de subconjuntos de características, podemos considerar la existencia de dos modelos: los modelos por filtro (filter model) y los modelos por envoltura (wrapper model).

El modelo por filtro recibe este nombre porque el subconjunto de características irrelevantes para el problema es eliminado de la descripción del mismo antes de aplicar el algoritmo inductivo. Este tipo de modelo tiene una estructura semejante a la que se propone en la figura 4.1. De entre los algoritmos descritos en la literatura que siguen este modelo, podemos destacar a FOCUS y FOCUS-2 [1, 2].

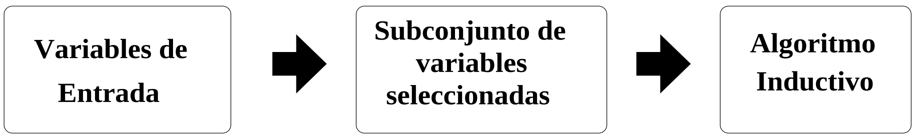


Figura 4.1: Esquema de un modelo por filtro

Frente al modelo por filtro, podemos encontrar otra estrategia de actuación que fija como objetivo del algoritmo inductivo la tarea adicional de determinar el subconjunto de características relevantes, de manera, que la evaluación de las variables seleccionadas se realiza de forma simultánea a la evaluación de la bondad de las soluciones que se desean extraer. El modelo asociado a la utilización de esta estrategia es conocido como modelo por envoltura y presenta una estructura semejante a la propuesta en la figura 4.2. Por tanto, los algoritmos de aprendizaje inductivo que siguen este modelo, son los que en [115] se clasifican como algoritmos que aprenden la relevancia parcial de los atributos, y ya comentamos en el capítulo primero de esta memoria. Por el contrario, los algoritmos que siguen el modelo por filtro son los algoritmos que aprenden la relevancia completa sobre todos los atributos que se proponen en el problema. Podemos destacar de entre los algoritmos que siguen este segundo modelo, OBLIVION [96] que combina este modelo con el método del vecino más cercano.

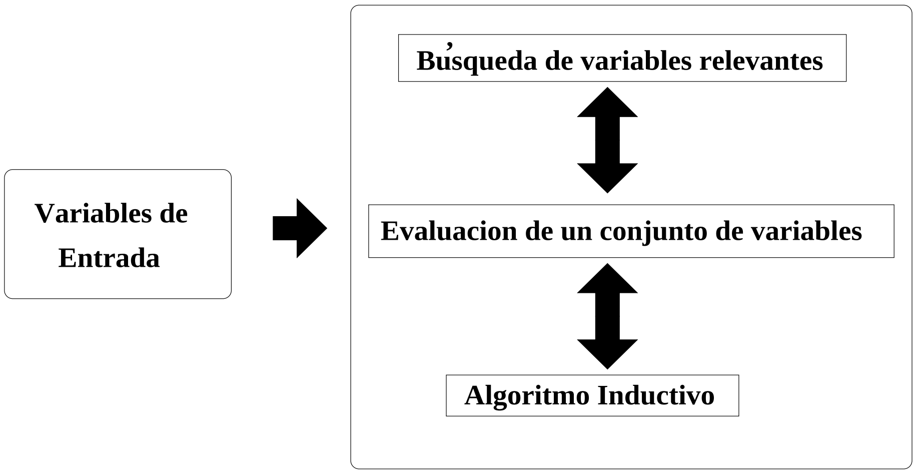


Figura 4.2: Esquema de un modelo por envoltura

Desde nuestro punto de vista, los algoritmos de refinamiento de teorías deberían incluirse como un modelo de selección de características, ya que algunos de ellos tienen como objetivo reducir el número de variables que intervienen en el proceso de clasificación de un concepto. En este sentido, el algoritmo de refinamiento de teorías difusas propuesto en el capítulo 2 de esta memoria, podría considerarse como un modelo de selección de características, ya que trata no sólo de mejorar el comportamiento del sistema en cuanto a su capacidad de predicción, sino que además intenta aumentar la comprensibilidad y simplicidad de las reglas, que son conceptos que involucran la selección de las características relevantes.

La estrategia utilizada por SLAVE en la selección de las variables relevantes para un determinado concepto, está claramente incluído dentro del modelo por envoltura. Debemos destacar, que a diferencia de otros algoritmos que siguen este modelo, SLAVE plantea la relevancia desde un punto de vista temporal, es decir, como una situación puntual que aparece en el proceso de aprendizaje, sin que ello implique nada definitivo para el resto del proceso evolutivo. Esto significa que en un momento dado de la evolución, una variable puede considerarse como irrelevante para un concepto en un individuo concreto de la población. Sin embargo, esta situación no implica que dicha variable sea considerada como irrelevante para el resto del proceso, ya que una simple aplicación de un operador genético puede considerar a dicha variable como relevante en las siguientes iteraciones.

Otra característica de SLAVE es que la bondad del subconjunto de características seleccionadas se valora conjuntamente con la asignación asociada a cada una de estas variables, aplicándose el criterio de simplicidad para tratar de reducir lo más posible dicho conjunto. Por tanto, la selección de características está perfectamente integrada con los criterios utilizados para la evaluación de las reglas.

4.3 Algoritmo genético de dos niveles

Conceptualmente, podemos considerar que SLAVE se propone resolver dos tareas de aprendizaje: determinar el subconjunto de variables que son necesarias para explicar un concepto y encontrar la mejor asignación de valores a dichas variables. Estas dos tareas que están muy relacionadas, en la definición de SLAVE se plantean conjuntamente, de manera, que se utiliza una misma representación y un mismo proceso para ambas. Este hecho conlleva que el proceso de selección de variables se entienda como un efecto colateral de la tarea de asignación de valores a variables, y por consiguiente, el algoritmo centra su esfuerzo en esta segunda tarea.

Como ya se dijo en este capítulo, por la forma de representación y por la conducta de SLAVE, determinar la irrelevancia de una variable requiere un costo evolutivo grande que crece exponencialmente con el número de etiquetas definidas para la variable, y con el número de variables que intervienen en el problema. Este hecho se pone de manifiesto cuando SLAVE se aplica a bases de datos con un número elevado de variables. Por ejemplo, en las pruebas experimentales realizadas hasta el momento, se puede comprobar que SLAVE obtiene un número excesivo de reglas para la base de datos IONOSPHERE, que provoca que la capacidad de predicción del conjunto de reglas obtenido no tenga un comportamiento tan bueno como podría esperarse.

Para resolver este problema, proponemos una nueva definición del algoritmo genético de SLAVE, donde los individuos estan constituidos por dos subestructuras que representan soluciones de una mayor complejidad. Cada subestructura se define mediante una representación apropiada a cada una de las tareas de aprendizaje. Por esta razón, denominaremos al algoritmo genético resultante, algoritmo genético con dos niveles.

Existen trabajos en la literatura que abordan el problema de la búsqueda de soluciones complejas a través de la descomposición de la representación en subestructuras. Sobre cada subestructura se plantea un algoritmo de búsqueda con una heurística propia, mientras que la evaluación de la bondad de la solución se realiza sobre la yuxtaposición de todas las subestructuras. Algunos investigadores han propuesto sistemas de búsqueda que siguen esta filosofía. En concreto, en el campo de los algoritmos genéticos podemos destacar los trabajos [122, 123] donde se describen las ventajas que proporciona esta forma de actuar en los procesos de búsqueda realizados a través de algoritmos genéticos, y en especial, la aplicación de este modelo al problema del aprendizaje. Cuando se utilizan algoritmos genéticos, o en general, estrategias evolutivas, a esta forma de actuar se la conoce como *coevolución*.

Por tanto, un algoritmo genético con dos niveles, mantiene dos representaciones diferentes: una para determinar el subconjunto de variables relevantes para un determinado concepto, y otro para encontrar la mejor asignación entre valores y variables para dicho concepto. Sobre cada representación se establece un proceso de coevolución genética, donde cada nivel tiene sus propios operadores genéticos, pero la bondad de la solución se establece en base a la colaboración entre los dos niveles que componen un individuo, como muestra la figura 4.3.

En la nueva definición del módulo de selección de reglas de SLAVE, distinguimos dos niveles diferentes en el proceso de búsqueda:

- El nivel de variable
- El nivel de valor

El primero de ellos, codifica qué variables de las que inicialmente se proponen en el problema se consideran relevantes para formar parte de la descripción de un determinado concepto. Esta información se va modificando durante el proceso evolutivo. El nivel de valor, codifica los valores particulares, tanto para las variables que se consideran relevantes como para las irrelevantes, aunque el código asociado a las variables no relevantes en este nivel no se considera para establecer la bondad de las reglas.

Esta separación entre ambas tareas de aprendizaje, que las sitúa a un mismo nivel de importancia durante el proceso, puede ser muy importante para problemas que involucran muchas variables, ya que cuando el algoritmo genético obtiene la mejor descripción para un concepto, puede seleccionar un subconjunto de variables y establecer la búsqueda usando esta información. Además, durante el proceso evolutivo los subconjuntos de variables seleccionadas pueden cambiar mediante la inclusión de nuevas variables o la eliminación de algunas de ellas que inicialmente fueron seleccionadas. De esta forma, el esfuerzo que requiere el sistema para seleccionar o eliminar una de las variables, es equivalente al esfuerzo que realiza el algoritmo para añadir o eliminar la asignación de un valor a una variable.

Para definir el algoritmo genético en dos niveles, es necesario describir las tres componentes siguientes:

1. *Una representación genética de las soluciones al problema,*
2. *una forma para crear la población inicial de soluciones, y*
3. *los operadores genéticos definidos para cada nivel.*

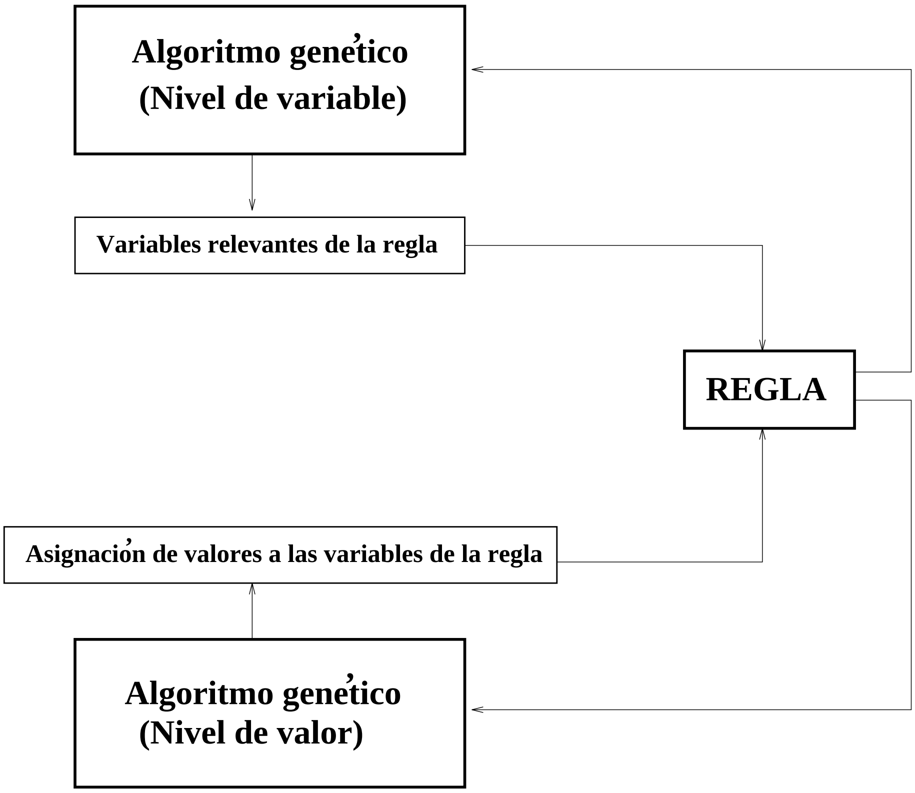


Figura 4.3: Esquema de SLAVE con dos niveles

Unicamente es necesario describir estas tres componente, ya que el modelo de coevolución mantiene la misma función descrita en el capítulo anterior para valorar la bondad de las reglas. Esta función de evalución es válida, ya que valora las mismas propiedades de las descripción propuestas por los individuos de la población, donde ahora, cada individuo esta compuesto por dos estructuras con significado diferente.

Lo mismo sucede con el operador de selección de la nueva población, ya que esta operación se realiza simultaneamente sobre las dos estructuras que componen cada individuo, esto es, la selección se realiza sobre individuos completos.

A continuación describimos más detalladamente las tres componentes que se modifican al realizar esta redefinición del algoritmo genético.

4.3.1 Representación de un elemento de la población

Supongamos que tenemos n posibles variables antecedente $X_1, \ldots, X_n$ y cada una de ellas tiene un dominio difuso asociado D_i con m_i componentes. La codificación genética necesita información sobre las variables seleccionadas y los valores asignados a estas variables. Así, necesitamos una codificación genética con las dos estructuras siguientes:

- Un cromosoma de variable
- Un cromosoma de valor

El cromosoma de variable (VAR) está constituido por un vector binario de longitud n (el número de variables) donde el valor uno indica que dicha variable se selecciona para la descripción de la regla, mientras que el valor cero indica que la variable no se selecciona.

El cromosoma de valor (VAL) usa la codificación binaria implementada en SLAVE, y ya mostrada en el capítulo 2.

Ejemplo **4.1** *Supongamos que tenemos tres variables , X_1, X_2 y X_3, tales que sus dominios difusos asociados son los mostrados en la figura 2.3 del capítulo 2. En este caso el código:*

$$VAR\ 101\ \ VAL\ 1110010111$$

representa al siguiente antecedente:

$$X_1 \ is\ \{A_{11}, A_{12}, A_{13}\} \ and\ X_3 \ is\ \{A_{31}, A_{32}\}.$$

Como X_1 toma todos los valores de su dominio D_1, el antecedente anterior es equivalente a

$$X_3 \ is\ \{A_{31}, A_{32}\}.$$

La variable X_2 fué eliminada de la descripción en el nivel de variable, mientras que X_1 fué eliminada en el nivel de valor.

Un aspecto importante a destacar, es que cuando el estado de una variable cambia de ser seleccionada a no ser seleccionada, el código relativo a dicha variable permanece. La razón es que si este estado cambia de nuevo, la variable tiene un valor concreto, que además coindice con el valor que tenía dicha variable antes de su primer cambio de estado.

4.3.2 La generación de la población inicial

La generación de la población inicial se debe hacer a dos niveles. A nivel de variable, el cromosoma de variable se construye usando una *probabilidad de activación inicial* $p \in [0, 1]$, donde p es la probabilidad de que una variable se seleccione para la descripción de la regla. Para no obtener inicialmente un antecedente excesivamente general, es conveniente que esta probabilidad tome un valor cercano a uno, dejando que sea el proceso evolutivo el que determine la configuración de la estructura más apropiada para el antecedente.

En el nivel de valor, el cromosoma se obtiene seleccionando aleatoriamente un ejemplo del concepto que se desea aprender y obteniendo el antecedente más especifico que mejor lo representa, como ya se mostró en el capítulo 2.

$\boxed{\textbf{Ejemplo}}$ **4.2** *Sean X_1, X_2 y X_3 tres variables cuyos dominios asociados corresponden a los que aparecen el la figura 2.3, y sea (r1,r2,r3) un ejemplo selecionado aleatoriamente del conjunto de entrenamiento para un determinado concepto. El antecedente más específico que mejor lo recubre es*

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

cuya representación binaria es 0010010010. *Tomando 0.9 como valor de la probabilidad de activación inicial, supongamos que obtenemos la configuración 101 para el nivel de variable. Por tanto, la codificación genética es*

$$VAR \ 101 \ \ VAL \ 0010010010.$$

Este proceso se repite para un número fijo de ejemplos seleccionados aleatoriamente para confeccionar la población inicial.

4.3.3 Operadores genéticos

El modelo de coevolución que se propone para resolver este problema, define un conjunto de operadores genéticos para cada una de las subestructuras que constituyen un individuo de la población. Por tanto, en este caso, nos encontramos con dos conjuntos de operadores genéticos, uno para el nivel de variable y otro para el nivel de valor.

Los operadores del nivel de valor coinciden con los propuestos en la versión de SLAVE presentada en los capítulos anteriores. Por consiguiente, únicamente nos queda describir los operadores del nivel de variable.

Este nivel mantiene una codificación binaria, para la cuál definimos los dos operadores genéticos estandar, es decir, la operación de mutación y la operación de cruce sobre dos puntos.

Estos conjuntos de operadores realizan su función paralelamente sobre ambas estructuras. Esto implica que la aplicación de los operadores genéticos se realiza de forma independiente entre los dos niveles.

4.4 Pruebas experimentales

Con la introducción de un algoritmo genético de dos niveles, tratamos de mejorar el tiempo de respuesta en la búsqueda de las soluciones parciales. Con este fin, probamos el comportamiento del algoritmo sobre las bases de datos crisp y los parámetros del algoritmo que se han empleado en las experimentaciones anteriores. La tabla 4.1 muestra los

Bases de datos	Entrenamiento	Prueba	N.Reglas	Tiempo
IRIS	97.47	95.80	4.2	50
MONK1	100	100	4.0	43
MONK2	94.67	83.19	15.2	109
MONK3	99.23	92.64	7.2	50
INFARTO	99.10	84.05	4.8	47
WINE	99.49	95.19	6	153
PIMA	85.45	76.56	17	2661
IONOSPHERE	97.22	90.91	9	1228
SOYBEAN	97.10	97.5	34	954

Tabla 4.1: Resultados con la nueva versión de SLAVE

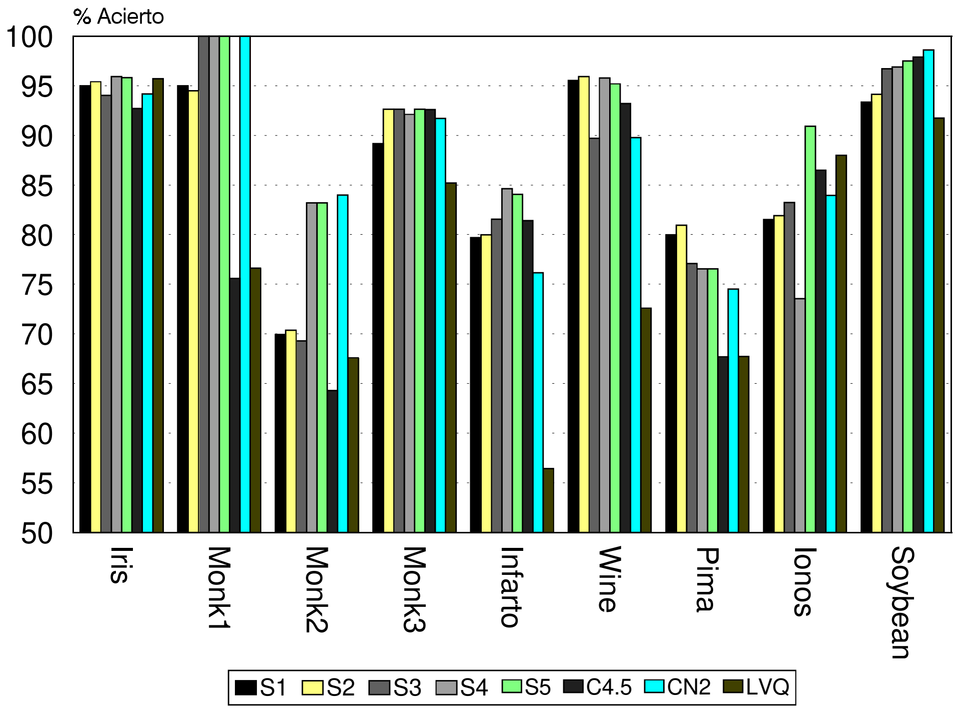
resultados que obtiene este algoritmo, asumiendo que la probabilidad de mutación y la probabilidad de cruce del nivel de variable coinciden con las definidas sobre el nivel de valor, que el valor de la probabilidad de activación definida para generar la población inicial es 0.9 y que el resto de parámetros, son los definidos para los anteriores experimentos.

De dicha tabla, podemos destacar que para las bases de datos que contienen pocas variables, los resultados obtenidos son semejantes a los que obtienen en la versión anterior de SLAVE, como muestra la figura 4.4, e incluso el tiempo de respuesta del algoritmo no varía sustancialmente. Sin embargo, para las bases de datos con un número mayor de variables, se producen mejoras significativas. Por ejemplo, se produce una reducción considerable en el tiempo de respuesta del algoritmo para las bases de datos WINE, PIMA e IONOSPHERE. Además, también se reduce el número de reglas obtenidas para las bases de datos PIMA e IONOSPHERE, que pasa en la primera de 31 a 17 y en la segunda de 53.6 a 9. Conviene destacar el aumento en el porcentaje de acierto obtenido sobre la base de datos IONOSPHERE que pasa de un 73.53 % a un 90.91%, que es la de mayor número de variables.

Por consiguiente, la descomposición de la búsqueda de las soluciones utilizando dos algoritmos de búsqueda que, de forma paralela, tratan de encontrar una solución parcial de segundo orden, parece una estrategia interesante que obtiene los mejores resultados cuando el problema planteado tiene un número elevado de variables.

4.5 Algoritmo genético de dos niveles con información

Los resultados experimentales de la sección anterior muestran que la aplicación de la coevolución a los dos niveles definidos para la representación de las soluciones provoca



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	Algoritmo SLAVE descrito en el capítulo 3
S5	Algoritmo SLAVE dos niveles sin información

Figura 4.4: Comparativa del acierto en los conjuntos de prueba (5)

mejoras en el comportamiento del algoritmo de aprendizaje. Más que en la capacidad de predicción, esta mejora se produce en el tiempo de respuesta del algoritmo de aprendizaje. Además, dicho tiempo mejora en comparación con las definiciones anteriores del algoritmo cuanto mayor sea el número de variables implicadas en el proceso de aprendizaje. Por consiguiente, parece que este algoritmo es apropiado para aprender grandes bases de datos, donde muchas variables están involucradas en la descripción del problema, y posiblemente muchas no sean relevantes.

Si nos detenemos en el proceso de generación de la población inicial de soluciones, podemos observar que para la configuración del nivel de valor, el sistema selecciona un ejemplo del conjunto de entrenamiento. Por consiguiente, el sistema aprovecha información sobre el problema, para generar este nivel. Por el contrario, en la configuración del nivel de variable, el algoritmo no contempla información sobre el problema que intenta resolver, y el proceso se realiza de forma aleatoria, en base a una probabilidad de activación inicial. En este sentido, podemos entender que el algoritmo, al generar por primera vez el nivel de variable, no aprovecha cierto conocimiento que se puede extraer del conjunto de entrenamiento. La adquisición de este conocimiento puede ser importante, ya que desde el principio del proceso evolutivo, sitúa el conjunto de soluciones iniciales en un camino más cercano a las posibles soluciones deseables.

En un principio, la incorporación de información extraída del conjunto de entrenamiento, puede reducir aún más el tiempo de respuesta del algoritmo. Este hecho provoca que la convergencia hacia las soluciones deseables se realice más rápidamente.

También es posible que, en función del tipo de información utilizada, esta sea útil, no sólo para la generación de la población inicial, sino que sea aplicable durante el proceso evolutivo. Si la información establece una valoración a priori de cada variable para determinar si debe o no formar parte de la descripción de la regla para un determinado concepto, está puede considerarse como parte constituyente de cada individuo y evolucionar para determinar el subconjunto de variables necesarias en dicha descripción.

A continuación, proponemos la incorporación al algoritmo genético de dos niveles de información obtenida sobre el conjunto de entrenamiento. Esta información permitirá determinar la relevancia de las variables predictivas. La interpretación de la información nos lleva a dos modelos.

4.5.1 Medidas de información

Estamos interesados en usar una medida de la relevancia inicial de las variables para determinar un concepto particular. Esta medida determinará el grado de influencia de cada variable para representar el valor de cada clase. En un segundo paso, el proceso genético, usando un conjunto de operadores, alterará este valor, y por tanto, modificará el criterio para determinar el conjunto de variables que deben activarse en la representación de cada clase.

Usamos la siguiente medida de información[93, 161]

$$I(X,Y) = \sum_x \sum_y p(x,y)\ log_2(\frac{p(x,y)}{p(x)p(y)})$$

donde p es una medida de probabilidad.

Esta medida define un rango diferente para cada variable. Para facilitar la comparación de los resultados en todas las variables, es necesario que todas las medidas trabajen en un mismo rango. Para este propósito, usamos la siguiente normalización

$$\tau(X,Y) = \frac{I(X,Y)}{H(X,Y)}$$

donde H(X,Y) es la entropía de Shannon sobre dos variables, definida como

$$H(X,Y) = \sum_x \sum_y p(x,y)\ log_2 p(x,y).$$

La medida estima la dependencia entre las variables X e Y en el sentido siguiente: valores de $\tau(X,Y)$ cercanos a cero indican un alto grado de independencia entre ambas variables, mientras que valores cercanos a uno, muestran un alto grado de dependencia funcional entre ellas.

En el algoritmo de aprendizaje de SLAVE, se seleccionan reglas fijando un concepto concreto. Por tanto, necesitamos restringir la medida anterior para el concepto que se desea aprender, y definimos

$$\tau_C(X) = \frac{I(X,Y=C)}{H(X,Y=C)}$$

donde X es una variable predictiva e Y es la variable de clasificación, siendo C un concepto particular o clase.

De forma similar a $\tau(X,Y)$, $\tau_C(X)$ determina la dependencia o independencia de la variable X para representar el concepto C de la variable de clasificación. En el algoritmo genético interpretamos este valor como una probabilidad inicial de activación de la variable, es decir, usamos $\tau_C(X)$ como la probabilidad con la que la variable X debe ser considerada como relevante para el concepto C, y $1-\tau_C(X)$ como la probabilidad con la que la variable X no debe tenerse en cuenta para dicho concepto.

El hecho de que los dominios de las variables estén compuestos por etiquetas lingüísticas, obliga a establecer un modelo de cálculo para estas probabilidades. En [54] se propone una metodología general para el cálculo de creencias usando información difusa. Con esta formulación, la probabilidad de que la variable X tome un valor a_i de su dominio $\{a_1, a_2, \ldots, a_t\}$ se define como

$$p(X=a_i) = \frac{1}{m}\sum_{j=1}^{m}(\frac{\mu_{a_i}(e_j)}{\sum_t \mu_{a_t}(e_j)})$$

donde m es el número de ejemplos del conjunto de entrenamiento E, e_j es un ejemplo de dicho conjunto de entrenamiento y μ_w es la función de pertenencia a un conjunto difuso w. La fórmula anterior asume que todos los ejemplos del conjunto de entrenamiento son no difusos. La probabilidad bidimensional se define de forma similar a la fórmula anterior, pero requiere la información de dos variables que serán combinadas a través de una t-norma

$$p(X = a_i, Y = b_j) = \frac{1}{m} \sum_{k=1}^{m} \left(\frac{\mu_{a_i}(e_k) * \mu_{b_j}(e_k)}{\sum_{t,h} \mu_{a_t}(e_k) * \mu_{b_h}(e_k)} \right)$$

donde el dominio de la variable Y es $\{b_1, b_2, \ldots, b_r\}$. Desde ahora, asumiremos que la t-norma utilizada para las fórmulas anteriores es el mínimo.

4.5.2 Añadir las medidas de información a SLAVE

Como anteriormente describimos, la modificación consiste en encontrar una forma de asignar los valores iniciales de relevancia a cada una de las variables predictivas, dependiendo de la información que proporciona el conjunto de entrenamiento. El proceso evolutivo determinará los valores de relevancia más apropiados para cada variable. En la sección anterior, se definieron las medidas de información que se usarán como probabilidades de activación.

La representación de los individuos de la población del nuevo algoritmo genético es similar a la que ya se propuso anteriormente. Por tanto, la representación consta de dos niveles (un nivel de variable y un nivel de valor) para todos los individuos de la población. Sin embargo, el nivel de variable de un elemento de la población estaba compuesto por un vector binario, donde el valor 1 en la variable X implicaba la activación de dicha variable y 0 la no consideración de la variable. Ahora, usaremos un nivel con codificación real en el cuál el nivel de variable contiene los valores τ_C, que se mueven en el intervalo [0,1]. Este valor se interpretará como la probabilidad inicial de activación de cada variable.

Podríamos transformar la probabilidad de activación obtenida en base a la información contenida en el conjunto de entrenamiento en un vector binario, y usarlo en la misma forma que se propuso en la versión anterior. Sin embargo, deseamos que los cambios de relevancia o irrelevancia de una variable se realicen de forma suave, es decir, durante la ejecución, el proceso evolutivo debe definir la tendencia hacia la relevancia o irrelevancia de cada variable.

Además, debemos destacar una ventaja fundamental que aporta la obtención de las medidas de información sobre el nivel de valor. Supongamos que X es una variable y x es uno de los valores que puede tomar dicha variable, y sean Y e y la variable de clasificación y un valor de dicha variable respectivamente. Si al observar el conjunto de entrenamiento resulta que no aparece ninguna ocurrencia donde simultáneamente aparezca que $X = x$ e $Y = y$, esto nos indica que el valor x de la variable X no es útil para determinar el concepto y. En este sentido, utilizamos la probabilidad bidimensional para determinar los valores de cada variable que no aparecen para los conceptos particulares que se desean aprender. Los valores de las variables que verifican tener probabilidad conjunta cero sobre algún concepto de la variable de clasificación, pueden ser eliminados y por tanto no se tienen en cuenta para encontrar la descripción de dicho concepto. El hecho de no considerar estos valores implica una reducción del nivel de valor, y por consiguiente una reducción del espacio de búsqueda.

Al optar por definir un nivel de variable con codificación real que mantiene las probabilidades de activación de cada variable, nos encontramos con la necesidad de articular algún mecanismo para determinar si una variable se considera activa o no en la evolución de una regla. De entre las diversas posibilidades que se pueden sugerir, optamos por definir dos formas alternativas de manejo de dicha información para su estudio. A estas dos alternativas las denominamos:

- **Aproximación por copia**

  En la primera alternativa, la relevancia de una variable para determinar su utilidad en la representación de un concepto depende tan solo de su probabilidad de activación. En la generación de la población inicial, el nivel de variable se inicializa con los valores τ_C para cada variable. El nivel de valor se genera al igual que las versiones anteriores, tomando aleatoriamente un ejemplo de dicho concepto del conjunto de entrenamiento.

  La diferencia fundamental se produce en la evaluación del antecedente. En este caso, dependiendo de la probabilidad individual de activación, algunas variables se seleccionan para la descripción de la regla, y se crea un nivel virtual con un nivel de variable binario, es decir, cada individuo, mantiene sus dos niveles, y durante el proceso de evaluación se le añade un nivel más que almacena qué variables fueron consideradas como relevantes y cuales no. El proceso para determinar si una variable X_i estará en la descripción de la regla para el concepto C se realiza de la siguiente forma:

  Esquema 4.1 *Proceso de selección de variables activas en la aproximación por copia*

  1. *Se genera aleatoriamente un valor t en el intervalo [0,1]*
  2. *Si $t \leq \tau_C(X_i)$ entonces la variable X_i se considerará para la evaluación de la regla. En otro caso, esta variable no se tendrá en cuenta en la evaluación de la regla.*

  Sin embargo, la codificación real original del nivel de variable se conserva, y es esta la utilizada para el proceso evolutivo, es decir, la copia binaria que se genera sobre el nivel de variable únicamente se usa para la evaluación de la descripción.

$\boxed{\textbf{Ejemplo}}$ **4.3** *Supongamos que tenemos tres variables, X_1, X_2 y X_3; los dominios difusos asociados con cada una de ellas se muestran en la figura 2.3. Supongamos además que los valores de información obtenidos para el concepto C de la variable de clasificación son:*

$$\tau_C(X_1) = 0.5,\ \tau_C(X_2) = 0.7,\ \tau_C(X_3) = 0.1$$

donde además, se ha calculado que

$$p(X_1 = A_{11}, Y = C) = 0,\ p(X_2 = A_{24}, Y = C) = 0,$$

$$p(X_3 = A_{25}, Y = C) = 0.$$

El proceso de evaluación necesita un nivel virtual de variable binario. Supongamos que, aplicando el esquema anterior, se ha obtenido la configuración 110. Con esta información, tenemos la siguiente configuración:

$$VAR\ (0.5, 0.7, 0.1)\ VAR(110)\ VAL\ ((01)(110)(11))$$

y para la evaluación consideraremos la siguiente descripción:

$$X_1 \ is\ \{A_{13}\}\ and\ X_2\ is\ \{A_{21}, A_{22}\}.$$

- **Aproximación por umbral de activación**

  Esta segunda aproximación propone la codificación de un umbral de activación dentro del nivel de variable. Con esta nueva representación de los elementos de la población, una variable X_i se considerará como relevante para la construcción de la descripción de una regla, si $\tau_C(X_i) \geq \tau_{umbral}$, donde τ_{umbral} representa el umbral de activación de las variables del j-ésimo elemento de la población. En otro caso, la variable no se tendrá en cuenta en la descripción de la regla.

  Inicialmente, el umbral de activación se define aleatoriamente para cada elemento de la población, donde τ_{umbral} toma un valor aleatorio en el intervalo

  $$[\min_i \tau_C(X_i), \max_i \tau_C(X_i)]$$

  Los valores τ_{umbral} son modificados por los operadores genéticos durante el proceso evolutivo como el resto de los componentes del nivel de variable.

$\boxed{\textbf{Ejemplo}}$ **4.4** *Utilizando las mismas variables del ejemplo anterior, el código:*

$$VAR\ (0.5, 0.7, 0.1)\ \tau_{umbral}\ 0.6\ VAR(110)\ VAL\ ((01)(110)(11))$$

representa el siguiente antecedente

$$X_2 \ is \ \{A_{21}, A_{22}\}.$$

La modificación del nivel de variable que pasa de tener codificación binaria a tener codificación real, obliga a la inclusión de nuevos operadores genéticos. Para este nivel incluimos dos operadores genéticos, uno para la mutación y otro para el cruce, que han sido ampliamente utilizados en la construcción de algoritmos genéticos con codificación real. Los operadores genéticos son la mutación no uniforme [106], que altera un gen con una cierta probabilidad que varía a medida que la evoluación del genético avanza, y el operador de cruce BLX_α [35], que permite la combinación de dos cromosomas para generar un nuevo cromosoma.

Como consideración final, diremos que los valores de τ_C se recalculan cada vez que el algoritmo de aprendizaje lanza un algoritmo genético para encontrar la mejor descripción, ya que debido al comportamiento de SLAVE, cuando se encuentra una regla para un concepto, los ejemplos del conjunto de entrenamiento que son cubiertos por dicha regla, se eliminan del mismo y por consiguiente la información que contiene el conjunto de entrenamiento después de este proceso cambia.

4.6 Pruebas experimentales

La inclusión en el nivel de variable de la información que aportan los ejemplos del conjunto de entrenamiento sobre la incidencia de cada una de las variables para representar un determinado concepto, permite la generación de una población inicial que se encuentra más cerca de las soluciones que se desean obtener. Este hecho, combinado con la reducción del espacio de búsqueda en el nivel de valor debido a la eliminación de aquellos valores de las variables que no tienen valor cero de probabilidad conjunta con el concepto que se aprende, debería tener como consecuencia una mejora en el tiempo de respuesta del algoritmo de aprendizaje. En esta parte experimental, estudiaremos el comportamiento de las dos aproximaciones definidas anteriormente que manejan dicha información para comprobar que se mejora el tiempo necesario en el aprendizaje de la base de reglas.

En la primera parte de esta sección nos centraremos exclusivamente sobre las bases de datos con consecuente crisp, ya que las bases de datos con consecuente difuso tiene un número pequeño de variables antecedente, y los resultados obtenidos por estas nuevas versiones son semejantes a los obtenidos en el capítulo anterior. Utilizando como parámetros de entrada los ya definidos en los anteriores experimentos. Además, en esta experimentación debemos incluir dos parámetros adicionales que son $\alpha = 0.05$ para el cruce BLX_α y una probabilidad de mutación sobre el nivel de variable que será de 0.001. En la tabla 4.2 se muestran los resultados obtenidos utilizando la aproximación por copia, mientras que la tabla 4.3 muestra los resultados obtenidos con la aproximación por umbral de activación.

Bases de datos	Entrenamiento	Prueba	N.Reglas	Tiempo
IRIS	97.82	96.12	4.2	40
MONK1	100	100	4.0	37
MONK2	94.67	83.19	15.2	109
MONK3	99.23	92.63	7.2	50
INFARTO	99.10	84.05	4.8	47
WINE	99.62	95.55	6	104
PIMA	85.69	75.00	21	2064
IONOSPHERE	96.22	93.12	9	791
SOYBEAN	96.52	99.45	33	1056

Tabla 4.2: Resultados de SLAVE dos niveles con copia

Bases de datos	Entrenamiento	Prueba	N.Reglas	Tiempo
IRIS	97.51	96.21	4	27
MONK1	100	100	4.0	30
MONK2	94.67	82.73	14.8	95
MONK3	99.50	95.32	7.6	46
INFARTO	99.72	86.12	3.7	27
WINE	98.50	96.76	5.2	41
PIMA	82.81	76.87	17	1107
IONOSPHERE	94.89	92.88	4	485
SOYBEAN	97.50	99.65	27	116

Tabla 4.3: Resultados de SLAVE dos niveles con umbral de activación

Estos resultados prueban de forma efectiva que las suposiciones descritas anteriormente son ciertas, ya que para ambas aproximaciones se puede observar una considerable reducción del tiempo de respuesta del sistema, aunque está es más patente en las resultados de la aproximación por umbral de activación.

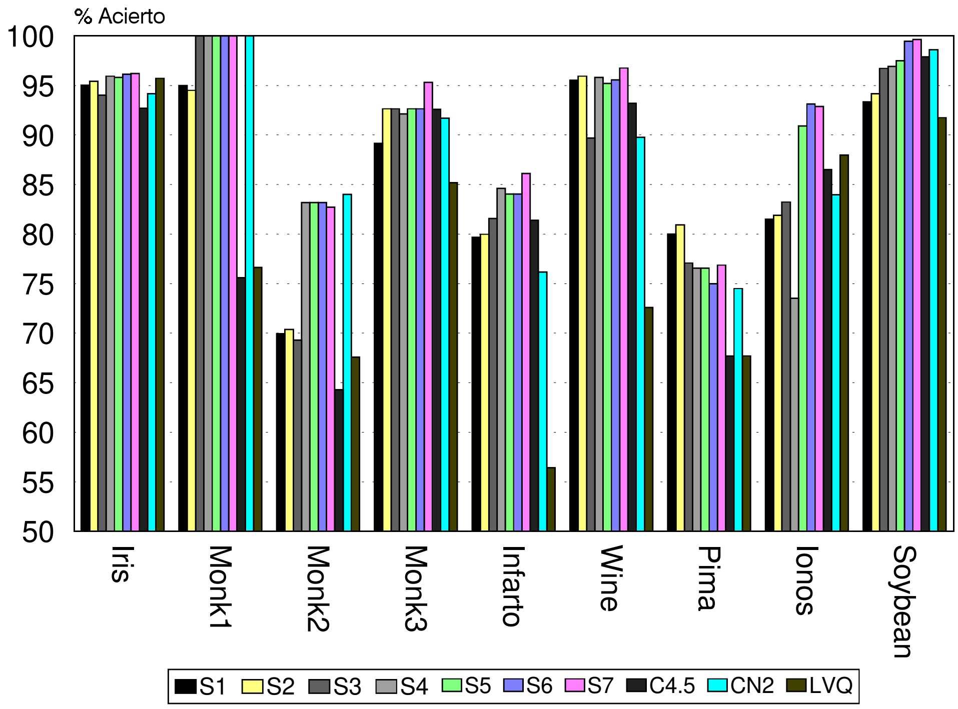
Sobre la capacidad de predicción, podemos observar que aparece un ligero empeoramiento sobre los conjuntos de entrenamiento. Sin embargo, los resultados sobre los conjuntos de prueba son muy parecidos a los mejores obtenidos por las diferentes versiones de SLAVE, e incluso, en algunos casos estos son mejores. Además, tampoco podemos observar diferencias apreciables entre el número de reglas que estas dos aproximaciones obtienen sobre la anterior experimentación con SLAVE. Sin embargo, son destacables los resultados obtenidos sobre la base de datos IONOSPHERE, ya que para ambas alternativas se mejora la predicción sobre el conjunto de prueba y el número de reglas obtenido por la aproximación por umbral de activación es sorprendentemente pequeño en comparación con los resultados anteriores.

Con la prueba experimental que se ha desarrollado en este apartado, no existe una evidencia clara de que una aproximación se comporte mejor con respecto a la otra en términos del porcentaje de aciertos sobre los conjuntos de entrenamiento y prueba. Sin embargo, con respecto al tiempo de respuesta, la aproximación por umbral de activación sí presenta un comportamiento claramente mejor. Por esta razón, incluimos esta aproximación como componente de la nueva versión de SLAVE.

En la figura 4.5 se muestran los resultados medios obtenidos por las distintas versiones de SLAVE propuestas en esta memoria, así como los resultados de los tres algoritmos de aprendizaje seleccionados para compara su comportamiento. Podemos observar que la versión de SLAVE dos niveles por umbral de activación obtiene los mejores resultados en porcentaje de acierto sobre los conjuntos de prueba de todas las bases, excepto sobre la base de datos IONOSPHERE.si los comparamos con las versiones anteriores de SLAVE propuestas en esta memoria. Además, si la comparamos con los otros algoritmos de aprendizaje, también podemos observar que obtiene el mejor resultado para todas las bases de datos, siendo únicamente superada por el algoritmo CN2 en la base de datos MONK2.

Los resultados presentados ponen de manifiesto que la utilización de la coevolución como planteamiento general para los procesos de búsqueda en los algoritmos de aprendizaje son una alternativa válida, en el sentido de que mantienen la capacidad de predicción y mejoran sustancialmente el tiempo de respuesta del algoritmo. Además, este tiempo de respuesta del algoritmo mejora aún más cuando se utiliza la información que aporta el conjunto de ejemplos de entrenamiento.

Si bien SLAVE obtiene prácticamente el mejor resultado para todas las bases de datos propuestas, ahora estamos interesados en comprobar si este buen comportamiento se mantiene ante bases de datos que contienen un número elevado de variables donde muchas de ellas son irrelevantes. Dado que es difícil encontrar bases de datos con estas características, planteamos una nueva experimentación sobre algunas de las bases de datos ya



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	Algoritmo SLAVE descrito en el capítulo 3
S5	Algoritmo SLAVE dos niveles sin información
S6	Algoritmo SLAVE dos niveles por copia
S7	Algoritmo SLAVE dos niveles por umbral de activación

Figura 4.5: Comparativa del acierto en los conjuntos de prueba (6)

		IRIS+	SOYBEAN+
SLAVE	Prueba	93.58	99.53
	N. Reglas	7.8	28.7
	Tiempo	132	1553
CN2	Prueba	83,04	92.13
	N. Reglas	80	173
	Tiempo	272	5660
C4.5	Prueba	92.56	97.9
	Tiempo	19	155
LVQ	Prueba	92,89	90.68
	Tiempo	175	2477

Tabla 4.4: Resultados obtenidos sobre las nuevas bases de datos con consecuente crisp

utilizadas, incluyendo un número elevado de variables irrelevantes.

Con este proposito, seleccionaremos dos bases de datos con consecuente crisp (IRIS y SOYBEAN) y una con consecuente difuso (MONTE) y generaremos las tres siguientes base de datos:

- **IRIS+**: Esta base de datos contiene las cuatro variables predictivas originales de la base de datos IRIS a las que se le añaden 95 variables. La nuevas variables son continuas y toman valores aleatorios en el intervalo [0,1]. Los conjuntos de ejemplos se generan de la siguiente forma: por cada ejemplo que aparece en la base de datos original se generan diez copias del mismo en la nueva base de datos de manera que, las variables originales conservan sus valores originales mientras que el resto de variables toman un valor aleatorio para cada una de las copias. De esta forma, la base de datos IRIS+ contiene 99 variables predictivas, y diez veces más ejemplos que la base de datos IRIS.

- **SOYBEAN+**: Esta base de datos se genera de forma semejante a la base de datos IRIS+ pero incluyendo 50 variables irrelevantes más sobre las 35 que posee el conjunto de ejemplos original. Por consiguiente, SOYBEAN+ contiene 85 variables predictivas y diez veces más ejemplos que SOYBEAN.

- **MONTE+**: La generación de esta base de datos es semejante a la anteriormente propuesta, incluyen en este caso 10 nuevas variables y generando tres copias por cada ejemplo del conjunto original.

Además, debemos especificar que la generación de los conjuntos de entrenamiento y prueba de estas nuevas bases de datos se generan a partir de los conjuntos de entrenamiento y prueba ya definidos sobre cada una de las bases de datos originales y utilizados en la parte experimental de esta memoria.

En la tabla 4.4 se muestran los resultados obtenidos en esta experimentación para las bases de datos IRIS+ y SOYBEAN+. Podemos observar que SLAVE continua obteniendo los mejores resultados en porcentaje de acierto sobre los conjuntos de prueba. Este hecho pone de manifiesto que el algoritmo SLAVE dos niveles por umbral de activación tiene un comportamiento robusto ante la presencia de variables irrelevantes. Por ejemplo, en una de las ejecuciones realizadas sobre la base de datos IRIS+ se obtuvo el siguiente conjunto de reglas:

```
 SI
       NS55 es {Alto}
       NS58 es {Bajo,Medio}
       Sepal_l es {Alta}
       Sepal_w es {Baja,Media}
       Petal_l es {Media,Alta}
       Petal_w es {Media}
   ENTONCES
       Clase es Versicoluor
--------------------------------------

 SI
       NS06 es {Bajo,Medio}
       Petal_w es {Media}
   ENTONCES
       Clase es Versicoluor
--------------------------------------

 SI
       Petal_l es {Muy_Baja,Baja}
       Petal_w es {Muy_Baja}
   ENTONCES
       Clase es Setosa
--------------------------------------
  SI
       Sepal_w es Menor o Igual que {Alta}
       Petal_l es {Alta,Muy_Alta}
       Petal_w es Mayor o Igual que {Media}
   ENTONCES
       Clase es Virginica
--------------------------------------

 SI
       Sepal_w es Menor o Igual que {Alta}
```

Bases de datos	Entrenamiento	Prueba	N.Reglas	Tiempo
MONTE	0.0019	0.0022	17	2401

Tabla 4.5: Resultados obtenidos sobre la base de datos MONTE+

```
    Petal_l es {Baja,Media}
    Petal_w es {Baja,Media}
ENTONCES
    Clase es Versicoluor
--------------------------------------
```

En esta base de reglas, únicamente aparecen variables irrelevantes afectando en un grado mímino a las dos reglas más especificas (las variables irrelevantes incluidas artificialmente son NS06 NS55 y NS58), lo que refleja el buen comportamiento del proceso de selección de características incluido en SLAVE.

Otro hecho que pone en evidencia la robusted de SLAVE en la selección de variables con respecto a CN2 lo refleja el número de reglas. La diferencia entre el número de reglas obtenidos por estos dos algoritmos es significativa. Sin embargo, el hecho más significativo surge en la comparación de los tiempos de respuesta. Si bien, para todas las versiones anteriores, el tiempo de respuesta de SLAVE era ampliamente peor que los que proporcionan C4.5, CN2 y LVQ, cuando el número de variables es muy elevado el tiempo de respuesta de SLAVE mejora en comparación con estos algoritmos. En concreto, en las bases de datos propuestas, SLAVE obtiene mejor tiempo de respuesta que CN2 y que LVQ.

Este buen comportamiento de SLAVE es aplicable también a problemas con consecuente difuso, como muestra el resultado obtenido sobre la base de datos MONTE+ en la tabla 4.5 [1].

Como conclusión final de este capítulo podemos afirmar, en base a las pruebas experimentales realizadas, que con esta versión de SLAVE se satisfacen los objetivos propuestos en este trabajo. En el siguiente capítulo nos planteamos dotar a SLAVE de una mayor capacidad de aproximación al comportamiento real de los sistemas. Con tal objetivo, se propone un nuevo modelo de regla que incluye modificadores semánticos que permiten modificar la semántica de las etiquetas iniciales, de manera que las descripciones que se obtengan se acerquen lo más posible a la información que aportan los ejemplos.

[1] No comparamos estos resultados con los obtenidos por el algoritmo WM ya que la implementación que tenemos de dicho algoritmo no soporta el número de variables antecedentes que posee esta base de datos.