3. APRENDIZAJE INDUCTIVO SUPERVISADO

Seguidamente se desarrollará el aprendizaje inductivo a partir de ejemplos en el sentido más habitual del término, en el que los ejemplos son vectores de pares atributo-valor. El objetivo de tal aprendizaje es construir procedimientos que proporcionen criterios para determinar el valor de un atributo especial, normalmente denominado clase, a partir de un subconjunto de los otros pares atributo-valor.

Se comenzará con uno de los más sencillos y hasta rudimentario que ayudará a fijar conceptos como el espacio de hipótesis, el *bias* y su transcendencia, etc.

3.1. EL ESPACIO DE VERSIONES Y EL ALGORITMO DE ELIMINACIÓN DE CANDIDATOS.

3.1.1. EL ESPACIO DE VERSIONES

Una hipótesis o función booleana que aproxima el concepto que se trata de aprender es, como se dijo, identificable con una función booleana (caso de solamente dos valores para el atributo especial clase; más brevemente dos clases: clase 0 y clase 1) definida desde el conjunto o dominio ejemplos.

$$h: \text{Dominio} \longrightarrow \{0,1\}$$

La definición que proporciona Mitchell [MITCHELL, 97, pga. 30] del espacio de versiones es la siguiente:

> **Definición II.3.1**. Sea D un conjunto de ejemplos descritos por un número finito de pares atributo= valor para unos atributos preestablecidos. Sea H el conjunto de las hipótesis que se pueden expresar como conjunción de esos pares atributo-valor. El espacio de versiones, denotado por $VS_{H,P}$, con respecto al espacio de hipótesis H y al conjunto de ejemplos de entrenamiento D, es el subconjunto de las hipótesis de H consistente con los ejemplos de entrenamiento de D.

Es decir; consiste en las distintas hipótesis que se pueden obtener como resultado de un proceso de aprendizaje que utilice ese lenguaje para la expresión de las hipótesis.

Obsérvese que el espacio de todas las hipótesis posibles para un dominio de ejemplos descrito por n atributos de, por ejemplo, m valores cada uno -para simplificar todos con igual número de valores- tiene un cardinal igual al número de subconjuntos del dominio -cada función booleana se identifica con un subconjunto- y por tanto del orden de 2^{m^n}, pues el dominio tendrá m^n elementos. Por otra parte, con el lenguaje que el espacio de versiones utiliza para las hipótesis solamente se consiguen distinguir a lo sumo $(m+1)^n$ subconjuntos (algunos pueden ser iguales) del dominio. Para obtener ese último valor se consideran los m valores posibles de cada atributo, más el caso de que no tenga valor asignado ese atributo en una cierta hipótesis, y se combinan las m+1 posibilidades de cada uno de los n atributos. Como $2^{m^n} \geq (m+1)^n$, los algoritmos que manejen el espacio de versiones para abarcar todas o mayor número de las hipótesis posibles deberán tener un lenguaje más potente que la simple conjunción de pares atributo-valor.

En el siguiente subapartado se estudiará el ***Algoritmo de Eliminación de Candidatos*** (Mitchell, 1983(?) [MITCHELL, 97, pga. 32] que explota el espacio de versiones tal y como se definió, por lo que tiene la limitación explicada.

3.1.2. EL ALGORITMO DE ELIMINACIÓN DE CANDIDATOS

El conjunto de la hipótesis posibles en el espacio de versiones, interpretadas como funciones booleanas con el dominio de todos los ejemplos posibles, D, tiene una estructura de retículo considerada la relación entre las hipótesis "***más general que"***.

> **Definición II.3.2.** La hipótesis h_j es *más general que* la hipótesis h_i si para cada ejemplo *d* del dominio D se verifica que *si* $h_i(d) = 1$ *entonces* $h_j(d) = 1$; en ese caso también se dice que h_i es *más específica que* la hipótesis h_j.

El *Algoritmo de Eliminación de Candidatos* procede desde la hipótesis más general y desde la más específica, representadas respectivamente por G_0= (_,_, ..., _) S_0= (∅,∅, ..., ∅), que delimitan el espacio completo de hipótesis con su relación de orden, y opera de acuerdo al siguiente procedimiento que básicamente consiste en partir de las hipótesis más general y de la más específica. Hipótesis ambas que actualiza para hacerlas coherentes con el primer ejemplo que incorpora y producir, como resultado, un conjunto de hipótesis generales maximales y otro de específicas maximales, coherentes todas ellas con el ejemplo que se incorpora. En posteriores pasos actualiza los conjuntos de hipótesis maximales para hacerlas coherentes con cada ejemplo incorporado. Cuando el ejemplo es positivo generaliza las hipótesis más específicas para incluir al ejemplo positivo. Cuando el ejemplo es negativo particulariza las más generales para excluir al ejemplo negativo. En cada momento se tienen todas las hipótesis coherentes con todos los ejemplos utilizados; por ello, al finalizar con todos, se tendría como resultado la acotación superior e inferior del espacio de hipótesis para el conjunto de ejemplos de entrenamiento utilizado. Desde estas acotaciones se pueden calcular todas las soluciones del espacio de versiones.

- **Algoritmo**

Comienzo
Inicializar G como el conjunto de hipótesis generales maximales en H;
Inicializar S como el conjunto de hipótesis específicas maximales en H;
.
Para cada ejemplo de entrenamiento d, **hacer**

- **Si** d es un ejemplo positivo
  - Borrar de G las hipótesis inconsistentes con d;
  - **Para cada** hipótesis s∈ S que no sea consistente con d;
    - Borrar s de S;
    - Añadir a S todos las generalizaciones minimales h de s tales que h sea consistente con d, y algún miembro de G es más general que h
    - Borrar de S todas las hipótesis que sean más generales que cualquier otra hipótesis de S
  - **Fin_para**
- **Fin_si**
- **Si** d es un ejemplo negativo
  - Borrar de S las hipótesis inconsistentes con d
  - **Para cada** hipótesis g∈ G que no sea consistente con d
    - Borrar g de G
    - Añadir a G todos las especializaciones minimales h de g tales que h sea consistente con d, y algún miembro de S es más específico que h
    - Borrar de G todas las hipótesis que sean menos generales que cualquier otra hipótesis de G
  - **Fin_para**
- **Fin_si**
-

Fin_para
Fin.

- **Criterio de parada del proceso anterior**

Por defecto

- **Si** no existen más ejemplos para procesar**:** ***insuficientes ejemplos para aprender el concepto*** (aunque quedan acotadas las posibles hipótesis solución).

O también:

- Si $G_i = S_i$ y es un conjunto unitario: ***concepto aprendido***.
- Si los conjuntos G_i y S_i se sobrepasan uno al otro**:** ***conjunto de ejemplos inconsistente***.

En el supuesto de tener suficientes ejemplos de entrenamiento, que no hubiese errores en ellos y que el espacio de hipótesis **contuviese** la hipótesis solución, está demostrado que este algoritmo **converge a la hipótesis correcta** que describe el concepto objetivo del aprendizaje como una conjunción de pares atributo = valor. Sin embargo, si no estuviese contenida la hipótesis solución en el espacio de hipótesis de este algoritmo; es decir que no se pueda expresar como conjunción de pares atributo = valor, no sería posible alcanzarla y tendríamos que contentarnos siempre con una solución de compromiso, al menos con esta versión rudimentaria del algoritmo. El motivo de este contratiempo es el *bias* del lenguaje utilizado para expresar las hipótesis; con las expresiones en forma de conjunciones de pares atributo-valor con las que es imposible abarcar el espacio de todas las hipótesis posibles.

- **Un ejemplo**

Ejemplo II.3.3. Considérese el conjunto, E, de ocho ejemplos que figuran en la siguiente tabla, descritos por los valores de cuatro atributos y una de las dos posibles clases:

Ejemplos	*Estatura*	*Pelo*	*Ojos*	*Clase*
$\mathbf{e_1}$	Baja	Rubio	Azules	+
$\mathbf{e_2}$	Alta	Rubio	Castaños	-
$\mathbf{e_3}$	Alta	Pelirrojo	Azules	+
$\mathbf{e_4}$	Baja	Moreno	Azules	-
$\mathbf{e_5}$	Alta	Moreno	Azules	-
$\mathbf{e_6}$	Alta	Rubio	Azules	+
$\mathbf{e_7}$	Alta	Moreno	Castaños	-
$\mathbf{e_8}$	Baja	Rubio	Castaños	-

INICIO: G_0= (_, _, _), S_0= (∅, ∅, ∅).

ITERACIÓN 1: Tras la incorporación del ejemplo $\mathbf{e_1}$:
G_1= (_, _, _),
S_1=(Baja, Rubio, Azules).

ITERACIÓN 2: Tras la incorporación del ejemplo $\mathbf{e_2}$:
G_{21}= (Baja, _, _), G_{22}= (_, _, Azules),
S_2= (Baja, Rubio, Azules).

ITERACIÓN 3: Tras la incorporación del ejemplo $\mathbf{e_3}$:
G_{31}= (_, _, Azules),
S_{31}= (_, _, Azules).

El concepto que ejemplifican los ocho ejemplos es imposible de expresar como una conjunción de pares atributo = valor. En realidad con los tres primeros ejemplos se ha aprendido un concepto que no encaja con los siguientes por lo que deducimos que, si no hay ruido ni ejemplos erróneos, el concepto objetivo es más complejo de lo debido para este algoritmo. Veremos seguidamente algoritmos que consiguen aprender de los ocho ejemplos por tener un lenguaje más rico para expresar las hipótesis.

Ejercicio. Generalícese el Algoritmo de Eliminación de Candidatos para que sea capaz de aprender hipótesis más complejas; es decir, que por ejemplo, en el caso anterior sea capaz de aprender el concepto ejemplificado por los ocho ejemplos que integran el conjunto de entrenamiento.

3.2. APRENDIZAJE DE REGLAS POR RECUBRIMIENTO: LA FAMILIA *AQ* Y *ARCO*.

Los algoritmos de recubrimiento tratan de obtener un conjunto de reglas que cubran o clasifiquen a todos los ejemplos positivos y excluyan a los negativos. Para aplicar esta filosofía a más de una clase, una vez obtenidas las reglas que cubren a los ejemplos de una *primera* clase, se vuelve a repetir el proceso sobre el conjunto de ejemplos de otra de las clases para obtener así una familia de reglas que cubra a una *segunda* clase de ejemplos. La aplicación reiterada de este proceder proporciona sucesivas familias disjuntas de reglas que cubren jerárquicamente a los ejemplos de clases distintas, exceptuando la clase que quedó como *última* que carece de reglas. Por extensión de este concepto trataremos también en este apartado los sistemas que obtienen directamente un conjunto de reglas que entre todas cubren el total de las clase. Seguidamente se describirán dos metodologías que ejemplifican los dos modelos de algoritmos de aprendizaje por recubrimiento.

3.2.1. LA FAMILIA *AQ*.

Esta familia engloba a una serie de sucesivos algoritmos en al que cada uno es una evolución y mejora de los anteriores. El primero de ellos data de 1969 y su autor es Mitchalski. Seguidamente se exponen las líneas generales del funcionamiento del primero de los algoritmos de la familia.

- **Algoritmo**

  1 ***Inicializar***
  - Divide los ejemplos en positivos y negativos

  2 Seleccionar semillas
  - Selecciona aleatoriamente un ejemplo positivo. Éste ese usará como una semilla para la generación de estrellas.

  3 Generar estrella
  - Genera una estrella reducida (descripción más general, a base de eliminar pares atributo valor de la descripción inicial de la semilla) a partir de la semilla distinguiéndola de los ejemplos negativos.
  - En el proceso de generación de estrellas, aplicar generalizaciones, tareas específicas o reglas heurísticas que sean apropiadas.
  - En la estrella obtenida, elegir la descripción más general preferida de acuerdo con un criterio de preferencia (LEF: *lexicographic evaluation function*).

  4 Comprobar si el recubrimiento está completo
  - Si la descripción, junto con descripciones previamente generadas, cubren completamente el conjunto de ejemplos positivos, entonces ir al paso 5.
  - En otro caso, borrar todos los ejemplos positivos cubiertos por la descripción generada a partir de los ejemplos positivos , y repetir el proceso principal desde el paso 2.

  5 Finalizar
  - Formar una disyunción de las descripciones generadas. Esta expresión forma una descripción completa y consistente del concepto.
  - Aplicar cualquier contracción o reformulación de las reglas para obtener una expresión posiblemente más simple.

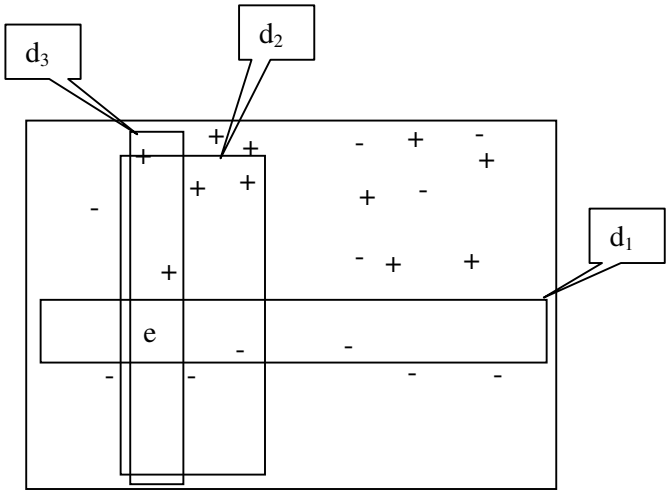


Figura 14: La descripción d_3 cubre el ejemplo e y ningún ejemplo negativo.

- **La generación de estrellas**

El algoritmo AQ es capaz de incorporar conocimiento para ayudar al desarrollo de la descripción de un concepto, este conocimiento puede incluir tanto afirmaciones generales como conocimiento específico del dominio. El conocimiento general puede consistir, por ejemplo, en reglas y restricciones que son aplicables a las propiedades generales de las clasificaciones. El conocimiento específico puede consistir, por el contrario, en reglas de inferencia para inferir nuevos descriptores o cuáles de ellos son más relevantes para la clasificación.

El algoritmo AQ incorpora este conocimiento por medio de una *anotación*. Una anotación de un descriptor simplemente es información o conocimiento sobre el descriptor, como puede ser: saber que un voltaje doméstico se encuentra en el intervalo 210 a 230 voltios, que es una magnitud continua, que está en relación con la intensidad y la resistencia, etc. Formalmente una anotación es un predicado, una variable o una función sobre el descriptor.

A pesar de las restricciones que se puedan introducir, el espacio de hipótesis pude resultar todavía inabarcable. La elección de los pares atributo valor adecuados entre todos los permitidos por las restricciones suele ser uno de los principales problemas de todo sistema. AQ lo resuelve por medio de una función de evaluación lexicográfica (LEF). LEF es una sucesión de pares de criterios de tolerancia que se aplican en orden lexicográfico para evaluar las clasificaciones. El procedimiento consiste en reducir el subconjunto de pares atributo valor hasta que únicamente quede una como remanente, o bien la sucesión de pares de criterios haya sido utilizada en su totalidad. Los criterios de preferencia incluyen:

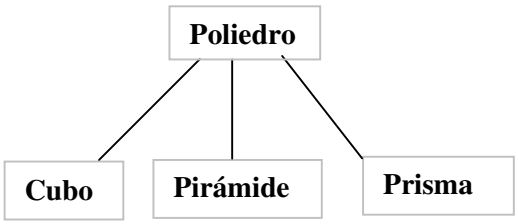
- Simplicidad global de la descripción con intención de mejorar la comprensión humana
- El costo de medición de los valores de ciertos descriptores
- Algunas medidas sencillas, tales como el número de pares atributo-valor en el descriptor.

Un propiedad importante de este enfoque es que el mismo sistema puede aprender descripciones características y discriminantes de las clases mediante una conveniente elección de los criterios de preferencia.

- **Generalización de reglas**

La metodología estrella incluye numerosas *generalizaciones*, *especializaciones* y *reformulación* de reglas que se utilizan para derivar nuevos descriptores que se incluirán en la descripción de las clases.

Las metarreglas de generalización transforman la descripción inicial en otra más general de forma que esta implique lógicamente a la primera. Generalmente se admiten pequeñas excepciones a esa implicación lo que hace que no se preserve la verdad sobre el conjunto de entrenamiento. La intención de esta permisión es aislar el ruido, las excepciones o simplemente el *sesgo* que posee la muestra de ejemplos si no fuese suficientemente representativa, y, así, el conocimiento extraído se comporte más correctamente frente a un conjunto test de ejemplos. Se distinguen dos tipos de generalizaciones; aquellas que utilizan los mismo pares atributo-valor que los ejemplos; por ejemplo la ***generalización por cierre de intervalo***, o aquellas que introducen nuevos pares atributo-valor; por ejemplo la ***generalización por escalada***. Este último proceso consiste en sustituir más de una par atributo valor por su supremo según una relación de conceptos externa y proporcionada generalmente por el usuario.



Con la relación de orden proporcionada por la figura anterior, si tuviésemos tres descripciones o reglas de una misma clase cuyos antecedentes fuesen del estilo de las siguientes:

$R_1 \equiv$ color= rojo, forma= cubo;
$R_2 \equiv$ color= rojo, forma= pirámide;
$R_3 \equiv$ color= rojo, forma= prisma;

se podrían generalizar en una sola que sería:

$R \equiv$ color= rojo, forma= poliedro.

- **Especialización de reglas**

Las especializaciones consiste en el proceso inverso del anterior y proporcionan descripciones que son implicadas por las descripción de partida.

- **Reformulaciones de reglas**

Las reformulaciones simplemente proporcionan descripciones diferentes sintácticamente pero equivalentes como fórmulas lógicas.

$$\textbf{Generalización} \Rightarrow \wedge \textbf{Descripción inicial}_i$$
$$\textbf{Descripción inicial} \Rightarrow \textbf{Especialización}$$
$$\textbf{Descripción inicial} \Leftrightarrow \textbf{Refomulación}$$

Figura 15: Resumen de la relación lógica de la descripción original con sus transformadas.

- **AQ11**

Es este algoritmo una implementación histórica de la metodología estrella y uno de los principales miembros de la familia AQ. Es un fiel representante de los algoritmos por recubrimiento. Su característica más destacable es el lenguaje utilizado para las descripciones, que está basado en el lenguaje de la lógica de predicados con ciertas propiedades añadidas; realmente en su propia evolución trabajó con dos lenguajes basados en el de la lógica de predicados ampliado: el lenguaje VL_1 y el lenguaje APC (*Annotated Predicate Calculus*) que con su riqueza -*bias*- proporcionan un amplio espacio de hipótesis. AQ11 trata de obtener la regla o descripción más general de una clase; es decir, la condición *necesaria* para pertenecer a esa clase. En términos más geométricos, con la descripción más general se expanden las fronteras de una clase hasta el máximo posible como se puede observar en al **figura 16**.

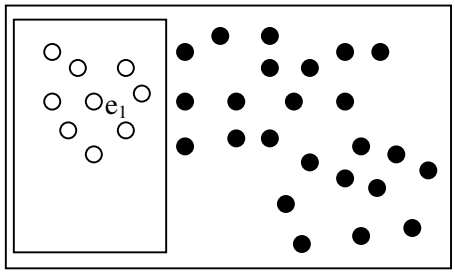


Figura 16: Descripción más general de la clase de la semilla e_1, frente al resto de las clases.

Obtenida la descripción más general de una clase se prescinde de sus ejemplos y se continua calculando al descripción más general de una de las clases remanentes frente al resto de ellas. Este proceder lleva a un solapamiento de las *regiones* del dominio de los ejemplos para las que no había ejemplos de entrenamientos (**figura 17**).

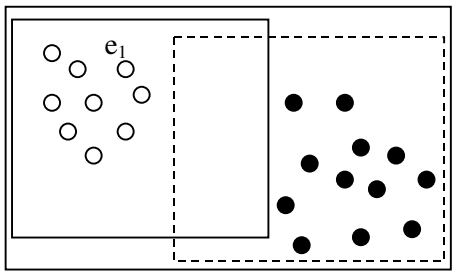


Figura 17: Solapamiento de dos descripciones más generales de clases diferentes.

AQ11 es capaz de hallar reglas de clasificación que no se solapen en virtud de la utilización del mismo lenguaje de representación para los ejemplos y para las reglas. De esa manera, el algoritmo puede tratar como datos de entrada tanto ejemplos como reglas, interpretadas estas últimas como instancias generalizadas. Con este método de funcionamiento se pueden incorporar las reglas descriptivas de una clase, halladas previamente, como información a tener en cuenta a la hora de calcular las reglas que describan la siguiente clase. AQ11 trata las descripciones de los conceptos o clases tratadas hasta un cierto momento como ejemplos negativos a la hora de hallar nuevas descripciones para sucesivas clases. Un pequeño problema se deriva de este tratamiento: las clases que se describen primero resultan sobregeneralizadas con respecto a las que se describen en lugares posteriores, **figura 18.**

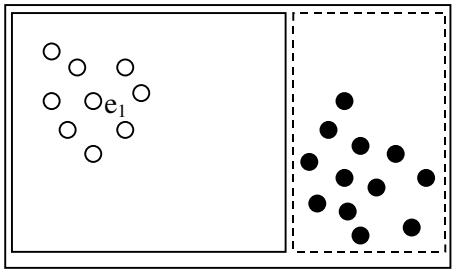


Figura 18: Sobregeneralización de la primera clase descrita frente a la segunda.

El conjunto de reglas o descripciones de máxima generalidad alcanzado por AQ11 para una clase se corresponde **aproximadamente** con el conjunto G del espacio de versiones (descripción maximal de las soluciones en el espacio de versiones). Por otra parte, con un criterio apropiado de preferencia AQ11 es capaz de calcular un conjunto S aproximado para cada clase. Con G y con S, AQ11 puede diferenciar las clasificaciones que realiza de un ejemplo en varios tipos: ***clasificación definida*** (la instancia es cubierta solamente por S), ***clasificación probable*** (la instancia es cubierta solamente por G) y ***clasificación múltiple*** (la instancia es cubierta por varios G de diferentes clases).

La metodología estrella de la familia AQ11 se utiliza también en sistemas de aprendizaje no supervisado como es la familia CLUSTER (concretamente en CLUSTER/2).

- **El sistema AQ15**

El sistema AQ15 es un exponente de los sistemas de aprendizaje supervisados e incrementales con memoria plena; es decir, con todos los ejemplos introducidos hasta el momento presentes en memoria. En su núcleo sigue la metodología estrella y el proceder de AQ11; sin embargo, contienen novedades necesarias para el tratamiento incremental.

3.2.2. *ARCO*.

ARCO (Algoritmo para el cálculo de un Recubrimiento de Coste Óptimo) es una herramienta de aprendizaje cuyo cometido inicial es la ordenación de reglas para optimizar el coste de su aplicación; por tanto, son necesarios costes asociados a las reglas. Desde una perspectiva abstracta trabaja con recubrimientos, puesto que una regla se puede identificar con el conjunto de ejemplos que clasifica. Considera un coste para cada uno de los conjuntos que lo constituyen y, definido un coste para recubrimientos, ARCO trata de encontrar el recubrimiento de coste óptimo. El coste definido para recubrimientos, en el caso de un recubrimiento de solamente tres conjuntos F_1,F_2 y F_3 con costes respectivos C_1, C_2 y C_3, es:

$$coste_1(L) = C_1 + (1 - p(F_1))C_2 + (1 - p(F_1) - p(^{c}F_1 \cap F_2))C_3.$$

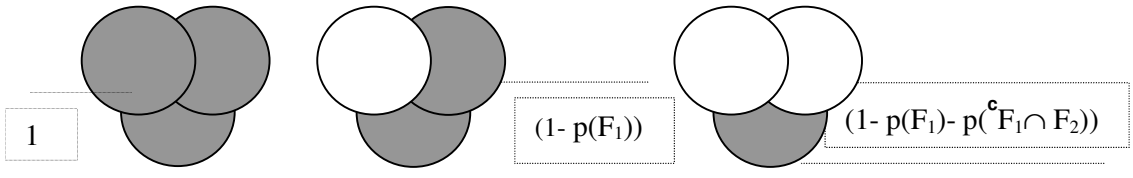


Figura 19: Conjuntos sobre los que se aplican las tres reglas jerárquicamente

En el caso general de ***n*** conjuntos en el recubrimiento, Fk_1,Fk_2 ... Fk_n, con sus respectivos costes Ck_1, Ck_2 ... Ck_n, la expresión del que denominaremos $coste_1$ es:

$$\mathbf{coste_1(L)= Ck_1+ (1- p(Fk_1))Ck_2 + (1- p(Fk_1)- p(^cFk_1 \cap Fk_2))Ck_3 + \ ... \ + (1- p(Fk_1)- p(^cFk_1 \cap Fk_2)- \ ... \ -p(^cFk_1 \cap ^cFk_2 \cap \ ... \ \cap ^cFk_{n-2} \cap Fk_{n-1}))Ck_n,}$$

- **La ordenación de reglas y los recubrimientos**

Si una regla se identifica con el conjunto de ejemplos que clasifica -algo que implícitamente se viene haciendo hasta aquí-, entonces un conjunto de reglas se identifica con un recubrimiento del conjunto de entrenamiento. Con esta perspectiva, está clara la relación de ARCO con el problemas de ordenar reglas asociadas a un conjunto de ejemplos.

Si se posee un conjunto de reglas que clasifican a un conjunto de ejemplos de una forma redundante, con ARCO se obtiene un recubrimiento con un coste (aproximado al)óptimo. Una elección adecuada de los costes individuales de las reglas permitirá dirigir el conjunto de reglas resultante -recubrimiento- hacia el óptimo, por ejemplo, en la precisión de la clasificación, o en su coste económico, o en la rapidez de aquella, etc. e incluso a una combinación de estos u otros intereses.

- **La generación de reglas y las familias determinantes**

Sin embargo, para constituir un sistema de aprendizaje completo hay que proporcionar una manera de obtener las reglas que constituyan el recubrimiento del que parte ARCO.

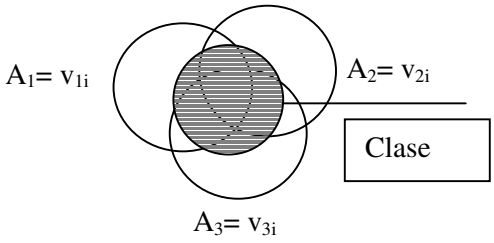


Figura 20: Interpretación de los pares atributo=valor de un ejemplo como intersección de conjuntos.

Con esa meta como objetivo, se considerará cada ejemplo como una regla *demasiado* específica y se generalizará. Para ello se identifica cada par atributo-valor del ejemplo con el conjunto de ejemplos que lo poseen. La intersección de esos conjuntos determinan un subconjunto de ejemplos que contendrá como elemento al ejemplo de partida y estará contenido en la clase del ejemplo. El objetivo que se persigue es concretar una subfamilia de pares atributo-valor, de entre los pares del ejemplo en cuestión, que determinen la misma intersección que el total o que, al menos, esté contenida en la clase del ejemplo.

Para enfocar el problema desde un punto de vista abstracto, denominaremos ***familia determinante*** a una familia de conjuntos que con su intersección determinan un cierto subconjunto, las familias determinantes pueden ser redundantes en cuanto al conjunto que determinan; es decir, que una cierta subfamilia determinen el mismo conjunto que la familia principal. Nuestro objetivo será obtener una de esas subfamilias y que, además, optimice un coste definido para familias determinantes. En un segundo paso nos conformaremos con que la subfamilia determine un subconjunto mayor que el determinado por la por la propia familia principal, pero contenido en la clase del ejemplo de partida.

El coste que se define para familia determinantes en el caso de solo tres miembros, y dados en el orden F_1,F_2 y F_3 con costes respectivos C_1, C_2 y C_3, es el siguiente:

$$coste_2(L) = C_1 + p(F_1)C_2 + p(F_1 \cap F_2)C_3.$$

En el caso general de ***n*** miembros, Fk_1,Fk_2 ... Fk_n, con sus respectivos costes Ck_1, Ck_2 ... Ck_n, la expresión del coste que denominaremos $coste_2$ es:

$$\mathbf{coste_2(L)= Ck_1+ p(Fk_1)Ck_2+ p(Fk_1 \cap Fk_2)Ck_3 + \ ... \ +p(Fk_1 \cap Fk_2 \cap \ ... \ \cap Fk_{n-1})Ck_n}$$

Mediante una dualidad que enunciaremos entre los conceptos de recubrimiento y familia determinante, se puede aplicar ARCO con unas ligeras modificaciones (y con las que pasa a denominarse DARCO: Dual de ARCO) para obtener una familia determinante de coste óptimo, e incluso ir más allá.

- **La dualidad entre familias determinantes y recubrimientos**

El concepto de familia determinante es dual del de recubrimiento respecto a las operaciones unión para el recubrimiento e intersección para la familia y esa dualidad se enuncia en la siguiente proposición:

> **Proposición II.3.4.** Sea los subconjunto finitos de índices $J \subset I$. $[F_j]_{j \in J}$ es un recubrimiento de coste óptimo de $\cup(F_i: i \in I)$, en el que F_i tiene un coste C_i para cada $i \in I$, si y solo si $[\overline{F}_j]_{j \in J}$ es una familia determinante de coste óptimo de $\cap(\overline{F}_i: i \in I)$, en el que para cada $i \in I$, $\overline{F}_i$ es el complementario de F_i y tiene asignado el coste C_i.

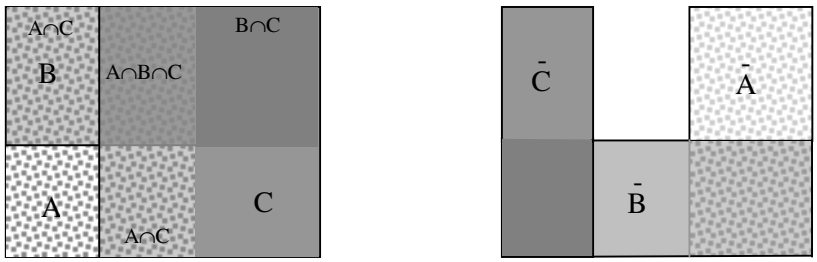


Figura 21.: Diagramas de una familia determinante y de un recubrimiento duales

Según este resultado las modificaciones que se deben realizar sobre ARCO son fundamentalmente sobre los conjuntos de entrada. Se deben de calcular sus complementarios y éstos, con los costes heredados respectivos, serán la entrada de ARCO. Los complementarios de los conjuntos de salida constituirán la familia determinante solución del proceso.

- **Preproceso de las entradas**

En le proceso de extracción de un subrrecubrimiento que realiza ARCO no importa tanto los ejemplos que cubre un conjunto como las subfamilias de ejemplos determinadas por las pertenencia a una familia maximal de conjuntos del recubrimiento de partida; es decir, aquellos subconjuntos de ejemplos que pertenecen a la intersección de una familia maximal de subconjuntos del recubrimiento. En la **figura 22** puede verse el cambio de perspectiva según se consideren los *elementos* de los conjuntos los ejemplos o las familias de ejemplos mencioanadas. Puede observarse que la topología del recubrimiento no se ve afectada por el cambio de interpretación de los conjuntos. Por tanto se realiza este pretratamiento de los conjuntos de entrada para transformar cada F_i en $F_{i\approx}$.

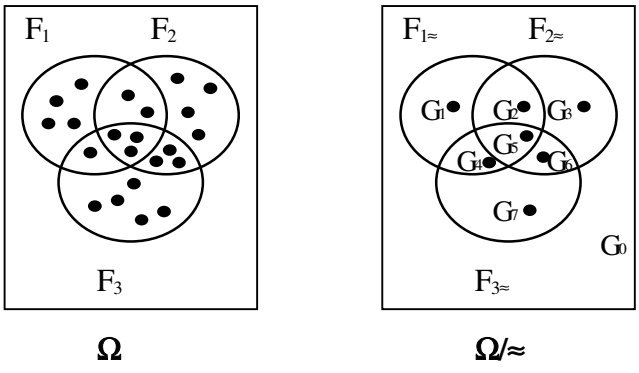


Figura 22: Ejemplos versus generadores

Entradas
Ω : conjunto de ejemplos,
R: recubrimiento

Comenzar ***Preproceso de los Datos de Entrada***
Ω/≈:= [ ]
Para cada F_i de R hacer $F_{i\approx}$:= { } **fin_para**
Para cada ejemplo e de Ω hacer
Construir una lista G con los conjuntos F_i de Ω tales que e ∈ F_i
Si G ∉ Ω/≈ **entonces** añadir G a Ω/≈ y añadir G a $F_{i\approx}$ **fin_si**
fin_para
devolver $R_{\approx}$ /*nuevo recubrimiento formado por todos los $F_{i\approx}$*/
fin.

- **El algoritmo ARCO**

Seguidamente desarrollamos el algoritmo ARCO

Entradas:
Recubrimiento: /*Familia de conjuntos, cada conjunto con un coste asociado*/
Costes
Conjunto de Elementos

Comenzar ARCO
Coste Inicial = 0
Coste Final = 0
Conjuntos Disponibles = Recubrimiento
hasta que Coste Inicial = Coste Final **hacer**
Conjuntos Elegidos = ∅
mientras Conjuntos Disponibles ≠∅ **hacer**
Elegir un conjunto F de disponibles /*mediante la función heurística de elección*/
añadir F a Conjuntos Elegidos
borrar F de Conjuntos Disponibles
Para cada H de Conjuntos Disponibles
borrar los generadores de H cubiertos por F
si H queda sin generadores **entonces**
borrar H de Conjuntos Disponibles
fin_si
fin_para
Actualizar Coste Final
fin-mientras
Familia Conjuntos := Conjuntos Elegidos
fin_hasta
Fin.

- **Arquitectura de la Herramienta**

La arquitectura de la *herramienta* se basa en una integración de los algoritmos **ARCO** y **DARCO**.

La idea básica de la *herramienta* es considerar cada ejemplo del conjunto de entrenamiento como una regla, regla que resultará de la máxima concreción y disjunta con las demás: **un solo ejemplo clasificado por cada regla y una única regla para clasificar cada ejemplo,** salvo que existan ejemplos repetidos. No podrán existir dos ejemplos iguales y en distinta clase. Es ésta una exigencia de coherencia mínima que suele ser habitual en los sistemas de aprendizaje basados en ejemplos.

El siguiente paso en el proceso es simplificar cada regla eliminando en cada una antecedentes no necesarios para determinar la clase u objetivo de la regla. La utilidad de **DARCO** en esta simplificación tiene dos versiones ligeramente diferentes. El objetivo inmediato ahora es la elección de los costes que valoren lo adecuado de las propiedades para determinar lo más acertada y concretamente la clase del consecuente de la regla que se simplifica.

Una vez simplificadas todas las reglas se procede a su selección y ordenación mediante **ARCO**. El objetivo de la selección y ordenación es obtener una familia de reglas con el mayor porcentaje posible de éxitos en la clasificación de cualquier ejemplo del mismo contexto que los ejemplos de entrenamiento. La valoración de lo interesante que consideremos una regla, de acuerdo a un cierto criterio que tenemos que fijar, se reflejará en los costes.

La arquitectura de la *herramienta* se resume gráficamente en la **figura 23**.

- **Asignación de Costes a los Antecedentes**

La elección de los propiedades adecuadas para constituir los antecedentes de una regla es un paso clave en cualquiera de los sistemas de aprendizaje basados en ejemplos de la literatura. Medidas de proximidad, medidas de entropía, probabilidades, *rough sets*, conjuntos borrosos, etc. son diferentes alternativas para comparar y elegir las propiedades que intervendrán en un regla. Todas las soluciones aportadas al problema tratan de suplir, mediante aproximaciones obtenidas con cálculos relativamente sencillos, valores cuyo conocimiento exacto nos llevaría a enfrentarnos a problemas de tipo exponencial.

La solución que se ha adoptado es de tipo probabilístico, lo que no cierra la posibilidad de utilizar otros que no se han explorado pero que podrían proporcionar resultados equiparables o incluso - por qué no- mejores.

> **Definición II.3.5.** Sea p_i una propiedad que aparece como antecedente de una regla R con consecuente la clase c_j, si se identifican cj y pi con los conjuntos de ejemplos que poseen esas propiedades, definimos el ***coste C_{ij} de la propiedad p_i en la regla R*** como: ***C_{ij}= 1-p(c_j /p_i),*** donde $p(c_j / p_i)$ es la probabilidad condicionada de que un ejemplo cualquiera que posea la propiedad pi también posea la clase c_j. Es decir, $p(c_j / p_i)$ es la probabilidad de éxito de la regla: $p_i \longrightarrow c_j$.

Se observa en la definición que una propiedad tendrá tantos costes distintos como clases existan. Es decir, aunque, por coherencia con el proceso que se va a realizar que trabaja regla a regla, se definió el coste de una propiedad para una regla, en realidad los costes dependen únicamente del tipo del consecuente de la regla y por tanto de las clases exclusivamente.

Por otra parte, si se toman las frecuencias relativas en el conjunto de entrenamiento como aproximación a las probabilidades y si volvemos a identificar una vez más una propiedad o clase con el conjunto de elementos que la poseen, se tiene que:

$$C_{ij} = 1 - (card(p_i \cap c_j) / card(p_i)).$$

La interpretación intuitiva -poco más allá se puede ir- de estos costes es sencilla. Se entiende que una propiedad o par atributo-valor colabora en mayor medida a determinar la clase en una regla si la mayor parte de los ejemplos que poseen la propiedad poseen también la clase. Como el concepto de familia determinante de coste óptimo conlleva la minimización del coste, interesa que los costes disminuyan con lo adecuado de las propiedades para determinar la clase de la regla; de ahí que pasemos al complementario, de manera que a cada propiedad le asociamos la probabilidad condicionada de que un ejemplo no posea la clase de la regla si posee la propiedad cuyo coste se valora.

La interpretación anterior de los costes puede originar un sinfín de discusiones pero otras alternativas probabilísticas también son igualmente discutibles y algunas más complicadas en los cálculos. La opción adoptada proporcionó buenos resultados en los experimentos.

En estos costes no se ha pretendido reflejar ninguna intención que proporcione un sentido específico al orden de las propiedades en la regla resultante tras aplicar **DARCO**. Realmente el orden influye en cuanto nos quedaremos con las primeras propiedades que determinen la clase

según la ordenación que realiza el algoritmo -y los costes asignados pretenden que se lleve a cabo con el menor número de propiedades- pero una vez ahí el orden de las propiedades definitivas no tiene significado especial. Qué duda cabe que se hubiera podido intentar combinar nuestro coste con otro de índole económico que reflejase, por ejemplo, lo costoso de verificar la propiedad sobre un ejemplo, pero en aras a la claridad preferimos la solución sencilla en oposición a la híbrida.

Precisamente por lo irrelevante de la ordenación final de las propiedades cabe plantearse un proceso añadido respecto al procedimiento standard de tratamiento de propiedades por parte de **DARCO**.

Por dualidad con los recubrimientos de coste óptimo, las familias determinantes de coste óptimo pueden presentar redundancias en el sentido de no ser necesaria la familia óptima completa para determinar la misma intersección. Sin embargo, la presencia de redundancias en ocasiones es necesaria para minimizar el coste, aunque la naturaleza heurística del algoritmo que calcula la

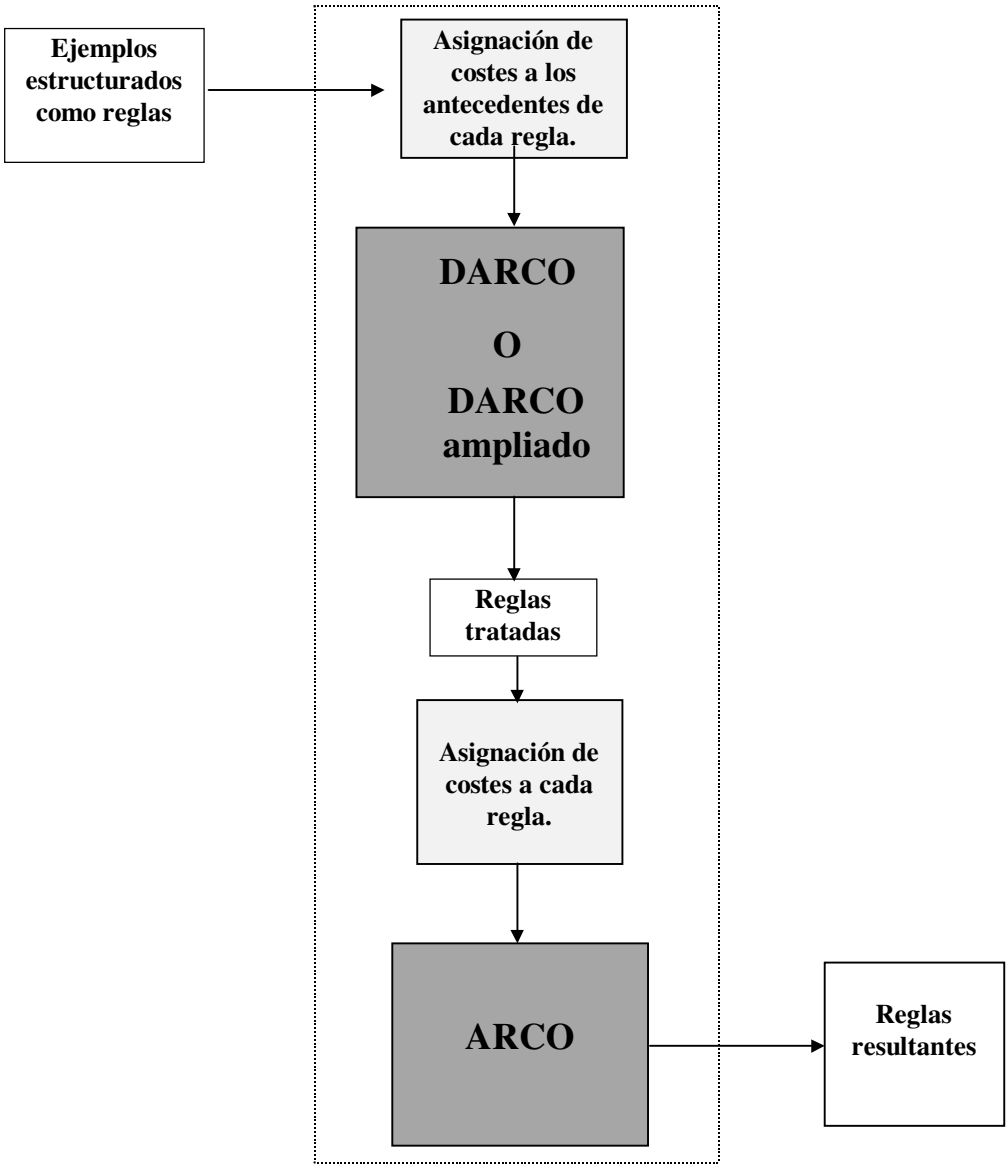


Figura 23: Arquitectura de la herramienta de aprendizaje.

familia hace que sea posible que aparezcan disfrazados en ella, como elementos económicamente necesarios, algunos que no son más que redundancias en todos los sentidos.

En el caso de tratar con propiedades parece interesante premiar la sencillez de la regla en número de antecedentes -así lo corroboró también la experiencia- frente a una regla más compleja y más concreta (aunque quizás más exacta), por lo que la eliminación de posibles redundancias de propiedades en el antecedente se convierte en un objetivo con prioridad. Como solución (provisional) para alcanzarlo proponemos una función que definiremos y que habría que incorporar a **DARCO** o a **DARCO ampliado** según la variante que utilizásemos. Esta función actúa sobre el recubrimiento de coste óptimo dual de la familia determinante de coste óptimo que saldría como solución.

Una vez tratado el recubrimiento los complementarios de sus nuevos conjuntos constituirán la familia determinante solución sin redundancias. Así, resulta que, aunque la necesidad de la función surgió al intentar realizar una depuración *más fuerte* de una familia determinante óptima, realmente actúa sobre recubrimientos donde también se podría explotar si hubiese justificación.

En caso de redundancias, el proceso de depuración supone que un conjunto es tanto más imprescindible en el recubrimiento de coste óptimo cuanto más al final está situado. Así, empezando por el último conjunto, y siguiendo correlativamente en orden ascendente, se van borrando la totalidad de los elementos cubiertos por un conjunto de todos los conjuntos anteriores a él. Si en la operación algún conjunto quedase sin elementos se borraría, y también se borrarían aquellos que quedasen solo con elementos que no cubren ***de hecho***; es decir, con elementos que son cubiertos por otros conjuntos anteriores en el recubrimiento. Para distinguir los elementos cubiertos ***de hecho*** por un conjunto de los que no lo son por ese conjunto, se les marca a éstos previamente con el valor FALSO. Al resto de los elementos de ese conjunto se les marca con el valor VERDAD. Se utiliza para ello un marcador booleano que se incorpora a todos los elementos.

- **Un Ejemplo**

Sean los conjuntos $F_1= \{e_1,e_2\}$, $F_2=\{e_2,e_3\}$, y $F_3=\{e_3,e_4\}$, con costes asociados C_1, C_2, y C_3. Supóngase que $[F_1\ F_2\ F_3]$ constituye un recubrimiento de coste óptimo de $\cup(F_i: 1\leq i \leq 3)$. Este recubrimiento es redundante si se prescinde de su coste. Aplicaremos el procedimiento de depuración de redundancias que expusimos.

Los conjuntos cocientes que maneja el algoritmo son:

$$F_{1\approx}= \{G_1,G_2\}, \qquad F_{2\approx}= \{G_2,G_3\}, \qquad F_{3\approx}= \{G_3,G_4\}.$$

En la **figura 24** aparecen una representación gráfica de los conjuntos y sus elementos o generadores.

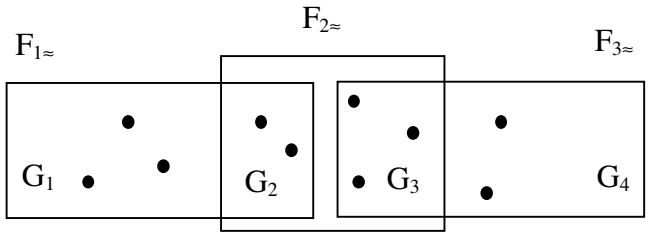


Figura 24: Supuesto recubrimiento de coste óptimo obtenido por ARCO

Un recubrimiento de coste óptimo en términos de elementos se podría visualizar de manera simplificada como:

$$[(F_{1\approx}\ (G_1,G_2))\ (F_{2\approx}\ (G_3))\ (F_{3\approx}\ (G_4))],$$

donde cada conjunto cociente aparece acompañado de los elementos que cubre **de hecho**; así por ejemplo, el elemento G_2 se considera cubierto por $F_{1\approx}$ por lo que no se considera como elemento cubierto por $F_{2\approx,}$ que tiene una posición posterior en el recubrimiento. Consideremos ahora el recubrimiento óptimo anterior con todos los elementos de sus conjuntos, diferenciados con FALSO los cubiertos **de hecho** de los otros marcados con VERDAD, quedaría:

$$[(F_{1\approx}\ ((G_1,\text{FALSO}),(G_2,\text{FALSO})))\ (F_{2\approx}\ ((G_2,\text{VERDAD}),(G_3,\text{FALSO})))\ (F_{3\approx}\ ((G_3,\text{VERDAD})\ (G_4,\text{FALSO})))].$$

Si ahora eliminamos elementos según el proceso descrito, se comenzaría eliminando todos los elementos de $F_{3\approx}$ de las listas de elementos de $F_{2\approx}$ y $F_{1\approx}$. El resultado es:

$$[(F_{1\approx}\ ((G_1,\text{FALSO}),(G_2,\text{FALSO})))\ (F_{2\approx}\ ((G_2,\text{VERDAD})))],$$

como $F_{2\approx}$ queda acompañado de elementos que no son abarcados **de hecho** por él; es decir, marcados con VERDAD, se elimina. El proceso concluye ahí, pues al considerar el conjunto $F_{1\approx}$ para repetir el mismo proceso el resto de la lista ya es vacía.
El recubrimiento depurado de redundacias es entonces: $[F_{1\approx}\ F_{3\approx}]$.
Restaría por recuperar los conjuntos F_1 y F_3 y devolver como solución: $[F_1\ F_3]$. •

- **4.2.2 Asignación de Costes a las Reglas**

En el proceso de simplificación anterior, inicialmente sin ningún tipo de inducción añadida, es deseable y frecuente que distintas reglas nacidas de la interpretación de los ejemplos como tales, una vez simplificadas originen la misma regla, y que por tanto la familia final de reglas resulta ampliamente redundante. También suele ser habitual en conjuntos de entrenamiento, distribuidos normalmente, que una vez eliminadas las reglas idénticas -salvo una representante- no sean necesarias todas las demás para clasificar los ejemplos primitivos. La elección de una subfamilia *suficiente* de reglas supone un problema de recubrimientos óptimos que se resolverá mediante **ARCO** con mejor o peor fortuna según la elección de los costes que se lleve a cabo para esas reglas.

La asignación de los costes tendrá como meta que resulte elegida una familia ordenada de reglas que clasifique con pocos errores. Se han excluido intencionadamente cualquier cariz en los costes que pudiesen mermar la efectividad de las reglas en el sentido dicho. Cualquier otro coste originado con intención diferente trastocaría el objetivo de este capítulo que, como se dijo, consiste tanto en demostrar la versatilidad de **ARCO** y **DARCO** como de plantear un *herramienta* de aprendizaje abierta pero con fundamentos sólidos.

Los experimentos llevaron a considerar dos tipos de costes para la elección de una familia de reglas *suficientes* como los de mejores rendimientos. Ambos son relativamente sencillos.

El primero consiste en asignar a todas las reglas **coste unitario**. Con este coste, en cada elección de una regla todas compiten, en lo que se refiere al coste, con igual importancia, por lo que su elección queda determinada por los ejemplos que abarca la regla en la posición que se está decidiendo y que no sean abarcados por reglas elegidas anteriormente. Así, la elección tenderá hacia un recubrimiento mínimo -o mejor minimal- de las ejemplos por las reglas.

El segundo tipo de coste barajado, y que en líneas generales fue el que produjo mejores resultados, consiste en considerar **el cardinal del conjunto de ejemplos clasificado por cada regla originalmente** como *peso* de la regla. Como ese cardinal grava inversamente a la importancia de las reglas según la intención que nos motiva, se tomó como coste de una regla el

inverso de ese cardinal. Como ya se indicó anteriormente, con estos costes en cada elección de una regla se valora, por medio del coste, la importancia original de la reglas que intervienen; es decir la cantidad de ejemplos que cubre independientemente de las que se hayan elegido anteriormente. Por otra parte en la elección se valora siempre la posición de la regla en la familia final, por lo que con este tipo de costes se hace una valoración ponderada de ambos factores: importancia de la regla individualmente e importancia de la regla en la posición que se discute.

Este último coste parece el más indicado para el tratamiento del ruido ya que está en consonancia con una de las hipótesis de trabajo: los ejemplos ruidosos son clasificados por reglas con pocos ejemplos originalmente.

La sencillez y el origen relativamente empírico de estos costes hace que, salvo la interpretación que se expuso de su repercusión en las elecciones de las reglas, no se puedan avalar de mejor manera que con las pruebas experimentales.

En la siguiente sección se desarrolla un ejemplo en el que se estudian casos concretos de los conceptos desarrollados hasta aquí sobre esta herramienta de aprendizaje.

- **Un Ejemplo Ilustrativo**

Para ilustrar el funcionamiento de la *herramienta* que se acaba de describir se recoge un ejemplo desarrollado por Quinlan [QUINLAN, 83, pag. 465] con su algoritmo ID3, precursor de C4.5. Los resultados que obtuvieron son casi análogos a los de aquel.

El conjunto de entrenamiento consiste en ocho ejemplos descritos por tres atributos más la clase. Los atributos, todos cualitativos, tienen rangos de diferentes tamaños pero para nosotros cada par atributo-valor corresponde a una propiedad; en total se manejan siete propiedades más dos clases. Los ejemplos aparecen en la siguiente tabla:

Ejemplos	*Estatura*	*Pelo*	*Ojos*	*Clase*
1	Baja	Rubio	Azules	+
2	Alta	Rubio	Castaños	-
3	Alta	Pelirrojo	Azules	+
4	Baja	Moreno	Azules	-
5	Alta	Moreno	Azules	-
6	Alta	Rubio	Azules	+
7	Alta	Moreno	Castaños	-
8	Baja	Rubio	Castaños	-

En primer lugar se identifican los ejemplos con reglas, obteniéndose así las siguientes:

Reglas		*Estatura*	*Pelo*	*Ojos*		*Clase*
$\mathbf{R_1}$	SI	Baja	Rubio	Azules	ENTOCES	+
$\mathbf{R_2}$	SI	Alta	Rubio	Castaños	ENTOCES	-
$\mathbf{R_3}$	SI	Alta	Pelirrojo	Azules	ENTOCES	+
$\mathbf{R_4}$	SI	Baja	Moreno	Azules	ENTOCES	-
$\mathbf{R_5}$	SI	Alta	Moreno	Azules	ENTOCES	-
$\mathbf{R_6}$	SI	Alta	Rubio	Azules	ENTOCES	+
$\mathbf{R_7}$	SI	Alta	Moreno	Castaños	ENTOCES	-
$\mathbf{R_8}$	SI	Baja	Rubio	Castaños	ENTOCES	-

Seguidamente se pasa a tratar los antecedentes de cada regla para obtener una familia determinante óptima en sentido ampliado; esta labor la realiza **DARCO (ampliado)**, previa asociación de un coste a cada antecedente de cada regla según el método que expusimos.

DARCO (ampliado) comienza calculando los costes a cada uno de los antecedentes de la regla sobre la que actúa según la fórmula que se propuso

Seguidamente **DARCO (ampliado)**, asocia a cada una las propiedades del antecedente los ejemplos que **no** poseen **ni** esa propiedad **ni** la clase de la regla en tratamiento.

En la regla R_1 del ejemplo intervienen los antecedentes o propiedades:

Estatura_Baja, Pelo_rubio, Ojos_azules.

Como el conjunto de ejemplos que poseen la estatura baja y la clase + es {1}, y el conjunto de ejemplos que poseen la propiedad estatura baja es{1,4,8}, el cálculo del primer coste se desglosa como sigue:

$$C_{11} = 1 - \frac{\mathbf{card}\{1\}}{\mathbf{card}\{1,4,8\}} = 1 - \frac{1}{3} = 0.6666\,.$$

Este coste es el que heredará ${}^{c}F'_{11}$. El resto de los costes para la regla 1 se calculan análogamente sin ninguna dificultad.

Con las demás reglas se realiza idéntico proceso. El resultado de estas dos primeras operaciones preparatorias para la regla R_1 es:

REGLA 1			
(Ni Estatura_baja ni clase +)	${}^{c}F'_{11}=$	{2,5,7}	coste $C_{11}= 0.6666$
(Ni Pelo_rubio ni clase +)	${}^{c}F'_{12}=$	{4,5,7}	coste $C_{12} = 0.5$
(Ni Ojos_azules ni clase +)	${}^{c}F'_{13}=$	{2,7,8}	coste $C_{13} = 0.4$

Con las demás reglas se realizan idénticos procesos. Los resultados son:

REGLA 2		
${}^{c}F'_{21}=$	{1}	$C_{21}= 0.4$
${}^{c}F'_{22}=$	{3}	$C_{22}= 0.5$
${}^{c}F'_{23}=$	{1,3,6}	$C_{23}= 0$

REGLA 3		
${}^{c}F'_{31}=$	{4,8}	$C_{31}= 0.6$
${}^{c}F'_{32}=$	{2,4,5,7,8}	$C_{32}= 0$
${}^{c}F'_{33}=$	{2,7,8}	$C_{33}= 0.4$

REGLA 4		
${}^{c}F'_{41}=$	{3,6}	$C_{41}= 0.33$
${}^{c}F'_{42}=$	{1,3,6}	$C_{42}= 0$
${}^{c}F'_{43}=$	{}	$C_{43}= 0.6$

REGLA 5		
${}^{c}F'_{51}=$	{1}	$C_{51}= 0.4$
${}^{c}F'_{52}=$	{1,3,6}	$C_{52}= 0$
${}^{c}F'_{53}=$	{}	$C_{53}= 0.6$

REGLA 6		
${}^{c}F'_{61}=$	{4,8}	$C_{61}= 0.6$
${}^{c}F'_{62}=$	{4,5,7}	$C_{62}= 0.5$
${}^{c}F'_{63}=$	{2,7,8}	$C_{63}= 0.4$

REGLA 7		
${}^{c}F'_{71}=$	{1}	$C_{71}= 0.4$
${}^{c}F'_{72}=$	{1,3,6}	$C_{72}= 0$
${}^{c}F'_{73}=$	{1,3,6}	$C_{73}= 0$

REGLA 8		
${}^{c}F'_{81}=$	{3,6}	$C_{81}= 0.33$
${}^{c}F'_{82}=$	{3}	$C_{82}= 0.5$
${}^{c}F'_{83}=$	{3,6}	$C_{83}= 0$

Para cada regla R_i, $1 \leq i \leq 8$, a su familia $\{{}^{c}F'_{ij}: 1 \leq j \leq 3\}$, de los conjuntos asociados a sus antecedentes, se le calcula un recubrimiento de coste óptimo, para lo que se toma como coste de un conjunto ${}^{c}F'_{ij}$ el coste de la propiedad correspondiente. Este cálculo se lleva a cabo mediante **ARCO** como parte integrante de su algoritmo dual -**DARCO**- según nuestra versión. Sobre el resultado obtenido se aplica el algoritmo que expusimos con objeto de simplificar el recubrimiento -si es posible aún más-, que en términos de la regla significaría una mayor reducción del número de antecedentes remanentes.

En el resultado de todas estas operaciones aparecen repeticiones de reglas originadas desde distintos ejemplos, como se puede observar seguidamente:

R_1	SI	ojos_azules y pelo_rubio	ENTONCES	clase +
R_2	SI	ojos_castaños	ENTONCES	clase -
R_3	SI	pelo_pelirrojo	ENTONCES	clase +
R_4	SI	pelo_moreno	ENTONCES	clase -
R_5	SI	pelo_moreno	ENTONCES	clase -
R_6	SI	ojos_azules y pelo_rubio	ENTONCES	clase +
R_7	SI	pelo_moreno	ENTONCES	clase -
R_8	SI	ojos_castaños	ENTONCES	clase -

Estas reglas, con los costes que se deben calcular ahora, constituirán la entrada del siguiente proceso que protagonizará **ARCO**, con ánimo de extraer un familia que clasifique -recubra- a todos los ejemplos a un coste óptimo.

Asignaremos como costes a las reglas los inversos del cardinal del conjunto de ejemplos asociado a cada una.

Entonces, los costes C_i, $1 \le i \le 8$, para las reglas anteriores y que heredarán los conjuntos de ejemplos F_i que llevan asociados son:

C_1= 0.5000	F_1= {1,6}	C_5= 0.3333	F_5= {4,5,7}
C_2= 0.3333	F_2= {2,7,8}	C_6= 0.5000	F_6={1,6}
C_3= 1.0000	F_3= {3}	C_7= 0.3333	F_7= {4,5,7}
C_4= 0.3333	F_4= {4,5,7}	C_8= 0.3333	F_8= {2,7,8}.

ARCO comienza calculando los conjuntos cociente $F_i/_{\approx}$, o más concisamente $F'_{i\approx}$, cuyos elementos G_j se denominaron generadores. Los resultados son:

$F_{1\approx}$= {G_1}, $F_{2\approx}$= {G_2,G_3}, $F_{5\approx}$= {G_3,G_5}, $F_{6\approx}$= {G_1},
$F_{3\approx}$= {G_3}, $F_{4\approx}$= {G_3,G_5}, $F_{7\approx}$= {G_3,G_5}, $F_{8\approx}$= {G_2,G_3}.

Los ejemplos constituyentes de los generadores se pueden apreciar en la **figura 25**, así como una panorámica gráfica de la distribución de los conjuntos de generadores $F_{i\approx}$.

El recubrimiento de coste óptimo obtenido corresponde a las siguientes reglas ordenadas, acompañadas de los ejemplos que cubren (resaltados los que clasifica cada una con ese orden):

R_2,	SI	ojos_castaños	ENTONCES	clase -; {**2**,**7**,**8**}
R_4,	SI	pelo_moreno	ENTONCES	clase -; {**4**,**5**,7}
R_1,	SI	ojos_azules y pelo_rubio	ENTONCES	clase +; {**1**,**6**}
R_3,	SI	pelo_pelirrojo	ENTONCES	clase + {**3**}.

Si se introduce una regla por defecto con el consecuente de la última regla se eliminará esta última regla y todas aquellas que la preceden correlativamente con la misma clase como consecuente; en este caso la R_1 y la R_3, resultando entonces la familia compactada:

R_2,SI	ojos_castaños	ENTONCES	clase -; {**2**,**7**,**8**}
R_4,SI	pelo_moreno	ENTONCES	clase -; {**4**,**5**,7}
		SI NO	clase +; {**1**,**3**,**6**}.

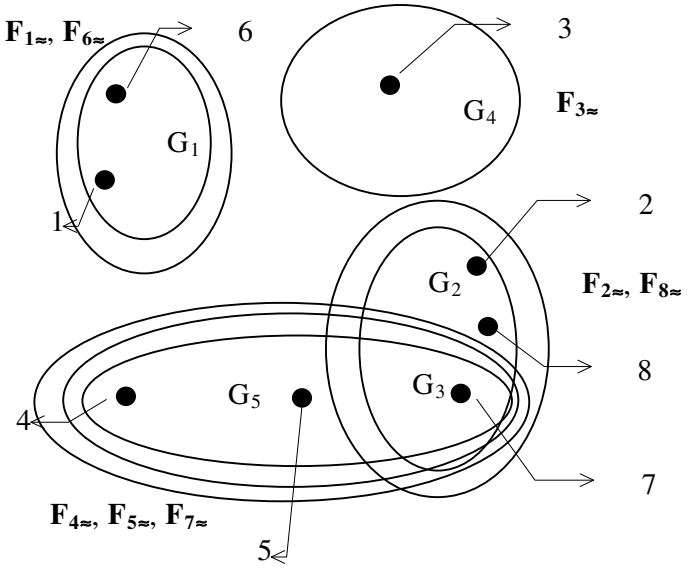


Figura 25.