Universitat Politècnica de València
Departamento de Sistemas Informáticos y Computación

Multi-paradigm Learning of Declarative Models

Aprendizaje multiparadigma de modelos declarativos

Autor:
Cèsar Ferri Ramírez
Director:
José Hernández Orallo
María José Ramírez Quintana

30 de mayo de 2003

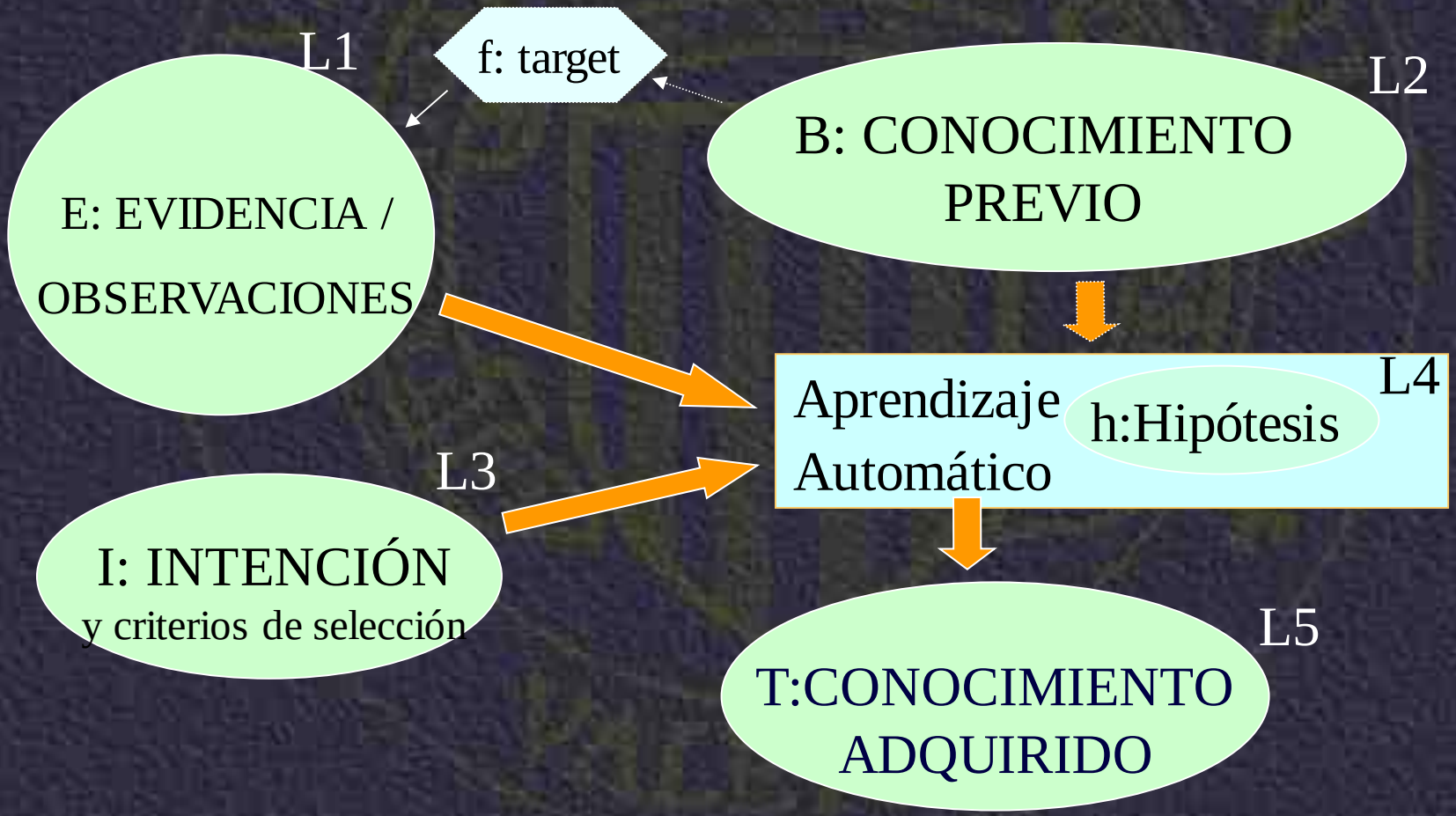
ÍNDICE DE LA EXPOSICIÓN.

1. Introducción
    1.1. Marco.
    1.2. Antecedentes.
    1.3. Objetivos.
    1.4. Metodología.
2. Inducción de Programas Lógico-Funicionales.
3. Multi-árboles de decisión
4. Aprendizaje Sensible al coste
5. Aportaciones Principales.
6. Conclusiones.

1. INTRODUCCIÓN.

1.1. Marco.

Extracción automática de conocimiento



1. INTRODUCCIÓN.

1.1. Marco.

Extracción automática de conocimiento

Terminología:

- *E*: Evidencia, Observaciones, Ejemplos, Casos, Datos.
- *B*: Conocimiento Previo o de Base, (Background Knowledge).
- *h*: Hipótesis, *H* (*L4*): espacio de hipótesis.
- *T*: Teoría, Modelo o Conocimiento Adquirido.
- *I*: Intención, interés, objetivo o expectativas del aprendizaje.

- *f: Target* Theory or Function: Teoría o función a aprender (desconocida siempre para el que aprende e incluso para todo el mundo).

1. INTRODUCCIÓN.

1.1. Marco.

Aprendizaje Supervisado

Cada observación incluye un valor de la clase a la que corresponde. Se aprende un clasificador. Además la clase (dominio de la función) es discreta

1. INTRODUCCIÓN.

1.2. Antecedentes.

Métodos de aprendizaje proposicionales

Métodos basados en lenguajes de representación limitados: reglas proposicionales. No pueden expresar relaciones entre atributos. Imposible capturar conceptos relacionales.

- Árboles de decisión
- Aprendizaje Bayesiano
- Reglas
- ....

1. INTRODUCCIÓN.

1.2. Antecedentes.

Árboles de Decisión

Árboles de Decisión (ID3 (Quinlan), C4.5 (Quinlan), CART).
Algoritmo Divide y Vencerás:

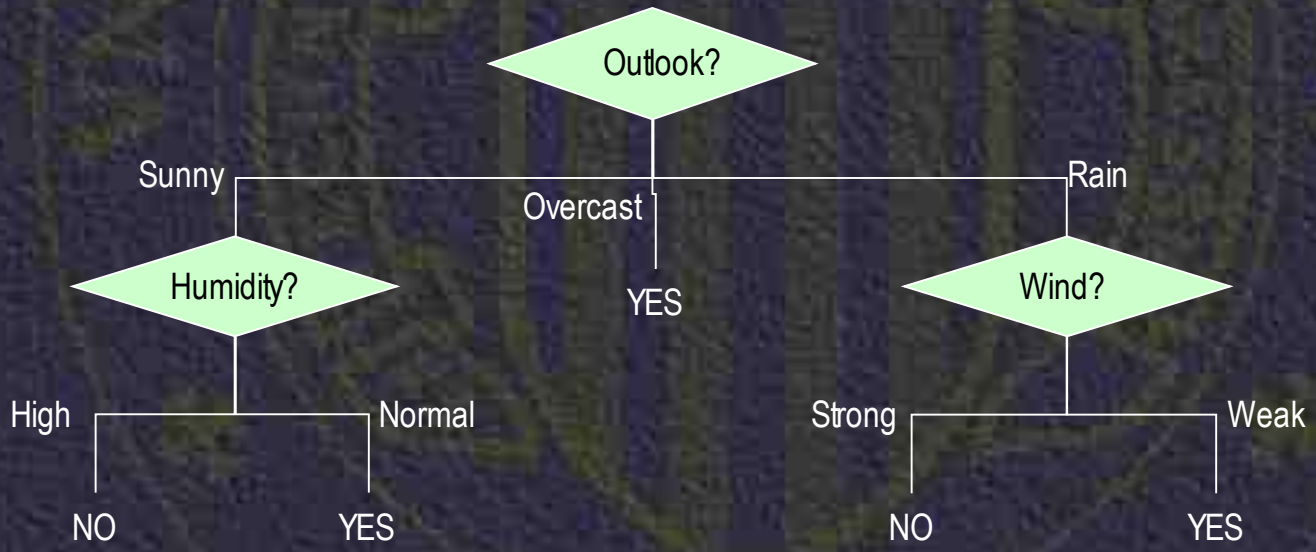
1. Se crea un nodo raíz con S:= todos los ejemplos.
2. Si todos los elementos de S son de la misma clase, el subárbol se cierra. Solución encontrada.
3. Se elige una condición de partición siguiendo un criterio de partición (split criterion).
4. El problema (y S) queda subdivido en dos subárboles (los que cumplen la condición y los que no) y se vuelve a 2 para cada uno de los dos subárboles.

1. INTRODUCCIÓN.

1.2. Antecedentes.

Árboles de Decisión

Árboles de Decisión.
Ejemplo C4.5 con datos discretos:



Representación Lógica:
(Outlook=Sunny AND Humidity=Normal) OR (Outlook=Overcast)
OR (Outlook=Rain AND Wind=Weak)

1. INTRODUCCIÓN.

1.2. Antecedentes.

Limitaciones del aprendizaje proposicional

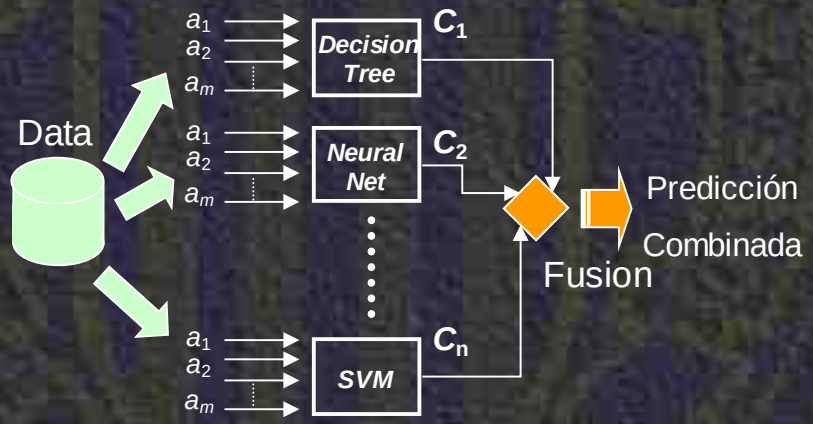
- No se pueden expresar relaciones entre atributos.
- Imposibilidad de aprender conceptos recursivos o con funciones auxiliares.
- Dificultad de trabajar con problemas con estructura compleja (datos semi-estructurados, listas..)

1. INTRODUCCIÓN.

1.2. Antecedentes.

Métodos Multi-clasificadores

Mejoran la precisión mediante la combinación de varios clasificadores.



Combinación simple

Diferentes políticas de generación: *Boosting, Bagging*
Diferentes tipologías: Simple, Stacking, Cascading

1. INTRODUCCIÓN.

1.2. Antecedentes.

Métodos Multi-clasificadores

Aunque, generalmente, la combinación de modelos supone una mejora en la precisión, esta técnica tiene importantes desventajas:

- El modelo combinado no es comprensible
- Necesidad de una lato grado de recursos para la generación y el uso del modelo combinado

1. INTRODUCCIÓN.

1.2. Antecedentes.

Aprendizaje Relacional

Programación Lógico Inductiva (ILP)

ILP se basa en la utilización de lenguajes de primeri orden (cláusulas de Horn) para la representación del conocimiento

p(X,Y,b) :- q(f(X,Y), c)

1. INTRODUCCIÓN.

1.2. Antecedentes.

Limitaciones de ILP

- Las funciones son simuladas artificialmente
- Problemas en el aprendizaje sensible al coste en problemas multi-clase
- No es posible la utilización del orden superior

1. INTRODUCCIÓN.

1.3. Objetivos.

Diseño y desarrollo de nuevos métodos de aprendizaje que permitan la obtención de modelos precisos y comprensibles desde en conjunto de datos que recojan experiencias previas.

- Modelos más expresivos que sean capaces de tratar con programas lógico-funcionales

- Modelos que utilicen las ventajas de los multi-clasificadores para mejorar su precisión

- Modelos que tengan un mejor comportamiento en contextos donde existan costes asociados al aprendizaje

2. IFLP

2.1. Programación Lógico-Funcional

Programación Lógico- Funcional

Sistemas de reescritura de términos condicionales (CTRS)

$l = r \Leftarrow e_1, \ldots, e_n$ with $n \geq 0$

Subsume Programación Lógica y Programación Funcional

- Más expresividad en comparación de los lenguajes funcionales
- Un evaluación más eficiente con respecto a los lenguajes lógicos

2. IFLP

2.2. Operadores

Inverso de unificación → *generalización*
Inverso de narrowing → "*narrowing inverso*"

Generalización Consistente Restrictiva (CRG)

La equación e = { $l_1 = r_1$ } es *una CRG de e' con respecto a E^+ y E^- y la teoría T = B ∪ P sii :*

1) *e es una RG de e',*

2) *no existe una derivación de narrowing ($s \hookrightarrow^*_{T \cup e} t$) tal que: s=t ∈ E^-, (consistencia con E^-)*

3) *para cada ecuación l = r ∈ E^+, no puede computarse una forma normal constructora desde l diferente a r con respecto a {T ∪ e}. (consistencia con E^+)*

2. IFLP

2.2. Operadores

Reverse Narrowing ($\leftarrow\!\!\!\!\neg$)

Se puede efectuar un paso de narrowing inverso desde t a t '($t \leftarrow\!\!\!\!\neg_{\theta} t'$) si

1. $u \in O(t)$,
2. $l = r$ is una variante nueva de una regla de P
3. $\theta = mgu(t_{|u}, r)$
4. $t' = \theta(t[l]_u)$.

Narrowing Reverso+ CRG = Narrowing Inverso.

Generación de reglas con Narrowing Inverso

r=s *(regla receptora)*
t=u *(regla emisora)*

1) $s \leftarrow\!\!\!\!\neg_{\theta} s'$ w.r.t t=u
2) CRG(r=s')

2. IFLP

2.2. Operadores

Ejemplo

(E_1^+)	0 + 0 = 0	(E_1^-)	s(0) + 0 = 0
(E_2^+)	s(0) + s(0) = s(s(0))	(E_2^-)	0 + 0 = s(0)
(E_3^+)	0 + s(0) = s(0)	(E_3^-)	s(0) + s(0) = s(0)
(E_4^+)	s(s(0)) + s(0) = s(s(s(0)))	(E_4^-)	s(0) + 0 = s(s(0))
		(E_5^-)	s(s(0)) + s(0) = 0

Ecuación e_a = { X + s(0) = s(X) } is a CRG of E^+_2, E^+_3, E^+_4
Ecuación e_b = { X' + 0 = X' } is a CRG of E^+_1

De la ecuación e_a ={X + s(0) = s(X)} selecciona t= s(X). Podemos utilizar una variante nueva de e_b ={X'' + 0 = X''}.

Dos ocurrencias:
$u_1 = 1$ retorna t'_1= s(X + 0) $u_2 = \varepsilon$ retorna t'_2= s(X) + 0
resultando en dos ecuaciones:
$e_{a,1}$ ={X + s(0) = s(X+0)} $e_{a,2}$ ={X + s(0) = s(X)+0}

Obviamente las dos derivan a e_a utilizando e_b.
Lo mismo sucede tras CRG: $e'_{a,1}$={X + s(Y) = s(X+Y)}

2. IFLP

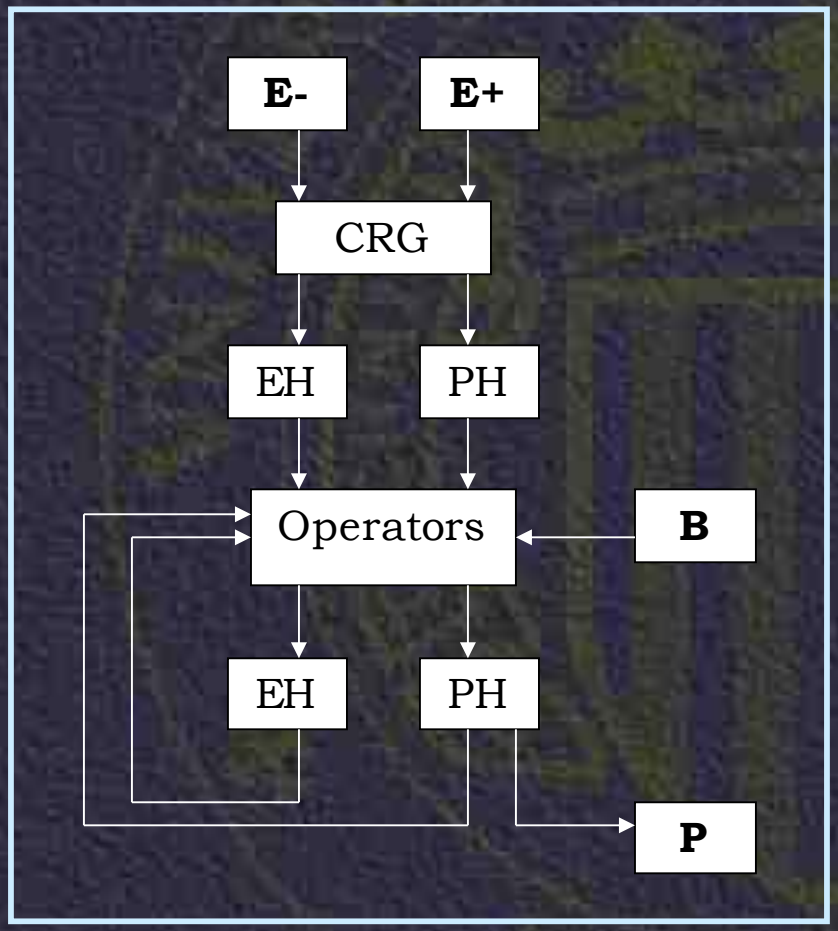
2.3. Sistema FLIP

El sistema FLIP es una implementación del marco IFLP:

- Parser sencillo
- Operador Narrowing
- Operador de Narrowing Inverso
- Operador CRG

2. IFLP

2.3. Sistema FLIP



- ✓ Entrada : **E+, E-, B**
- ✓ Salida: **P**

2. IFLP

2.3. Sistema FLIP

Programa	Pasos
sum (s(X),Y) = s (sum(X,Y)) sum (0,Y) = 0	2
prod(s(X),Y) = sum(prod(X,Y),Y) prod(0,X) = 0	2
fact(s(X)) = prod(fact (X) , s(X)) fact(0) = 0	2
app(· (X , Y) , Z) = ·(app (X ,Z), Y) app(λ , X) = X	2
rev(· (X,Y) = app (rev (X), · (λ,Y)) rev(λ) = λ	3
sort(· (X,Y)) = inssort(Y, sort (X)) sort(λ) = λ	2

2. IFLP

2.3. Sistema FLIP

• Dado el siguiente documento en *eXtended Mark-up Language* (XML) sobre los clientes rentables de una compañía de seguros de coches:

< ! customer of gratification level 30%! />
<goodc> <name> john </name> <has_children/> </goodc>

<goodc> <married/><teacher/><has_cellurar> </goodc>

<goodc> <sex>male</sex><teacher/><name> jimmy </name></goodc>
....

• Cada uno de los clientes pueden ser convertidos en términos funcionales:

goodc(: (: ([],name (john), has_children)) = 30

goodc(: (: (: ([],married), teacher) has_cellurar)) = 30

goodc(: (: (: ([],sex(male)), teacher), name (jimmy))) = 30
.....

2. IFLP

2.3. Algoritmo Incremental

Sistema Incremental: Los ejemplos se suministran gradualmente

A partir de cada uno, el sistema induce, mantiene o revisa las hipotesis.

Razones para usar lar incrementalidad:

- Imposible tener todos los ejemplos al mismo tiempo
- Limitaciones de memoria
- Eficiencia

2. IFLP

2.3. Algoritmo Incremental

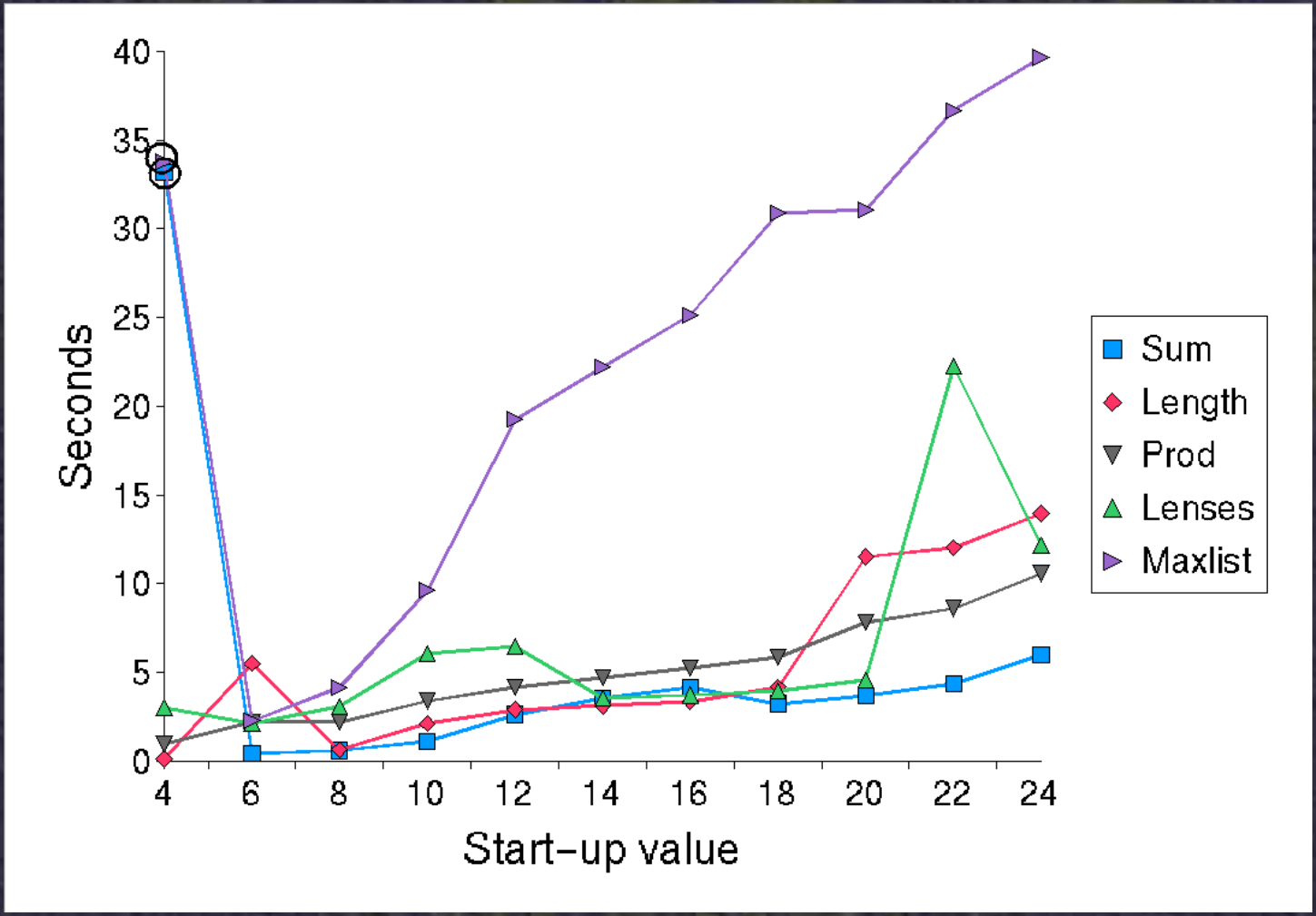
Para cada nuevo ejemplo suministrado:

- si *e* es un ejemplo positivo: E^+_n, probar para cada programa $P_i \in PH$:
  1. Éxito: $(P_i \not\models e)$
  2. Novedad: $(P_i \not\models e \wedge \forall e' \in OE^+_n : P_i \cup \{e\} \not\models e')$:
     Revisión de *PH* y *EH* para cubrir la nueva evidencia.
  3. Anomalía: $(P_i \not\models \neg e)$:
     Eliminar todo P_i de *PH* no confluente o incosistente y filtrar *EH*.
     Generar los CRG's de *e* en *EH* y extender *PH* con las nuevas reglas.

· Si es un ejemplo negativo: E^-_n, comprobar la consistencia de cada programa $P_i \in PH$ y actuar según los casos como si se diera un Éxito o una Anomalía .

2. IFLP

2.3. Algoritmo Incremental



2. IFLP

2.4. Conclusiones

Ventajas del marco IFLP:

- Capacidad de tratar directamente problemas con estructura compleja (documentos semi-estructurados XML)
- Posibilidad de incluir tipos y funciones en el aprendizaje
- Aprendizaje sensible al coste en funciones multi-clase

Limitaciones:

- Problemas con un número grande de ejemplos/atributos
- Evidencias con ruido o inconsistentes
- Atributos reales

3. Aprendizaje de Multi-árboles de Decisión

3.1. Introducción

Árboles de decisión

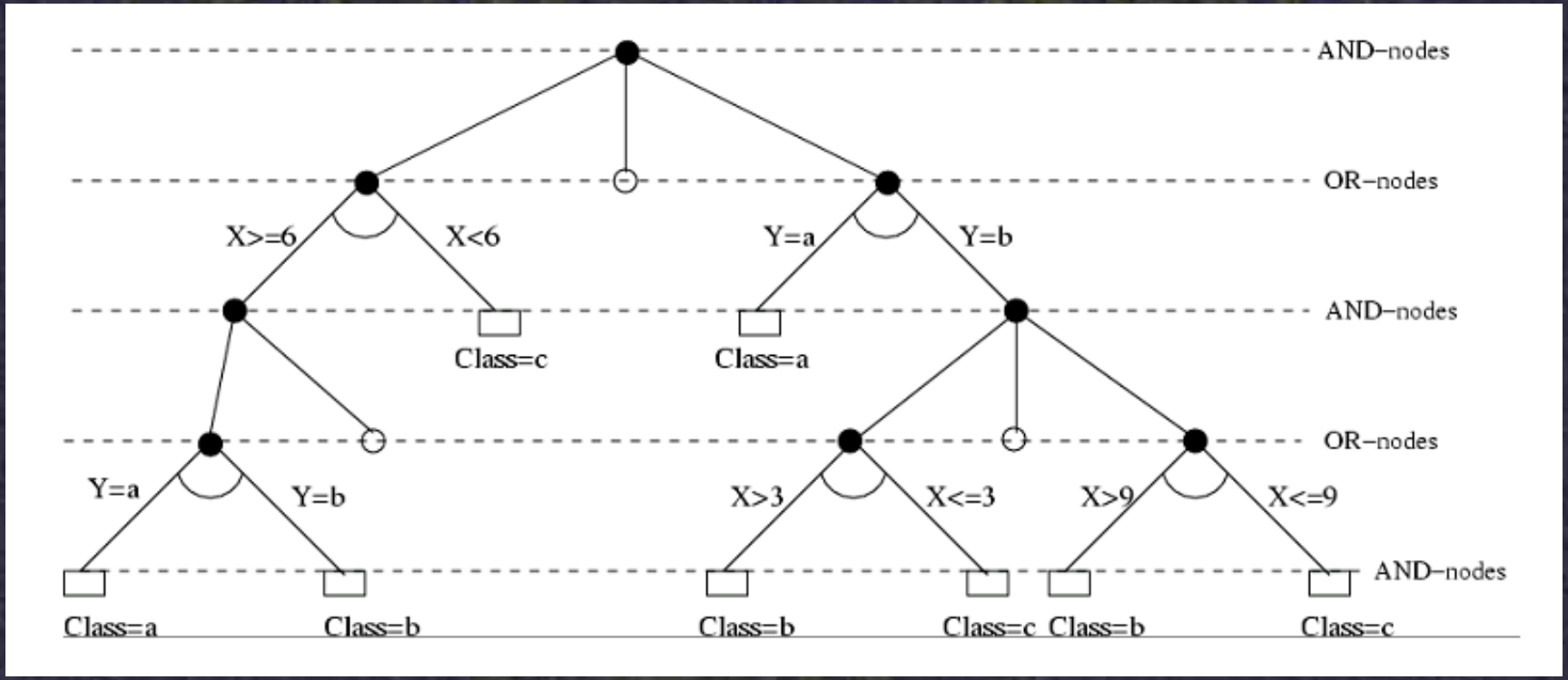
- Capacidad de abordar problemas de medio o gran tamaño
- Comprensibilidad y precisión del modelo aprendido

La estrategia voraz de construcción hace que se rechacen otras soluciones plausibles y posiblemente mejores

3. Aprendizaje de Multi-árboles de Decisión

3.1. Introducción

La idea de los Multi-árboles de decisión es alamcenar las particiones descartadas (nodos suspendidos), para (a posteriori) continuar la exploración en alguno de estos nodos suspendidos.



3. Aprendizaje de Multi-árboles de Decisión

3.1. Introducción

Multi-Árboles de decisión

- Contienen al mismo tiempo varios modelos
- Pueden considerarse un árbol AND/OR
- Modelos comparten algunos nodos
- Número de modelos crece exponencialmente con el número de nodos suspendidos explorados
- Nuevos criterios de construcción:
  - Criterio de selección de nodos suspendidos
  - Criterio de selección de soluciones

3. Aprendizaje de Multi-árboles de Decisión

3.2. Estrategias de Construcción

Criterio de selección de nodos suspendidos

- Rival absoluto
- Rival ratio
- Rival ratio con corrección de profundidad
- Aleatorio
- Aleatorio con corrección de profuncidad
- El más superior
- El más inferior

Criterio de selección de soluciones

- La más simple (Occam)
- Selección de k-soluciones

3. Aprendizaje de Multi-árboles de Decisión

3.2. Estrategias de Construcción

Precisión de las solución Occam (100 nodos explorados) según la estartegia de construcción

28 datasets	1ª	Inferior	Rival	Aleatorio	Rival Ratio	Superior
Media Geométrica	76.73	77.13	77.22	76.75	77.16	76.95

Precisión y tamaño de las solución Occam según tamaño del multi-árbol

28 datasets	1	50	100	150	200
Media Geométrica	76.73	77,17	77.22	77,21	77.16
Nº de reglas	85,54	83,14	81,87	81,03	80,32

3. Aprendizaje de Multi-árboles de Decisión

3.3. Multi-árboles como Multi-clasificadores

Un multi-árbol contiene varias hipótesis al mismo tiempo. Si combinamos de estas hipótesis el multi-árbol puede verse como un método multi-clasificador

- La combinación de las hipotesis se realiza en el interior de la estructura (fusión local)
- Las hipótesis comparten partes comunes (conjuntos compartidos)

3. Aprendizaje de Multi-árboles de Decisión

3.3. Multi-árboles como Multi-clasificadores

Métodos de fusión local

- Media aritmética
- Media geométrica
- Máximo
- Mínimo
- Producto
- Suma
- El más inferior

Métodos de trasnformación de vectores

- Diferencia
- Cuadrado
- Mayoría
- Buen perdedor
- Mal perdedor

3. Aprendizaje de Multi-árboles de Decisión

3.3. Multi-árboles como Multi-clasificadores

Forgetting: Técnica de filtro de nodos no explorados.
Mejora el uso de memoria y tiempo.

Métodos de forgetting:

- Constante
- Logarítmico
- Logarítmico ponderado por la profundidad

3. Aprendizaje de Multi-árboles de Decisión

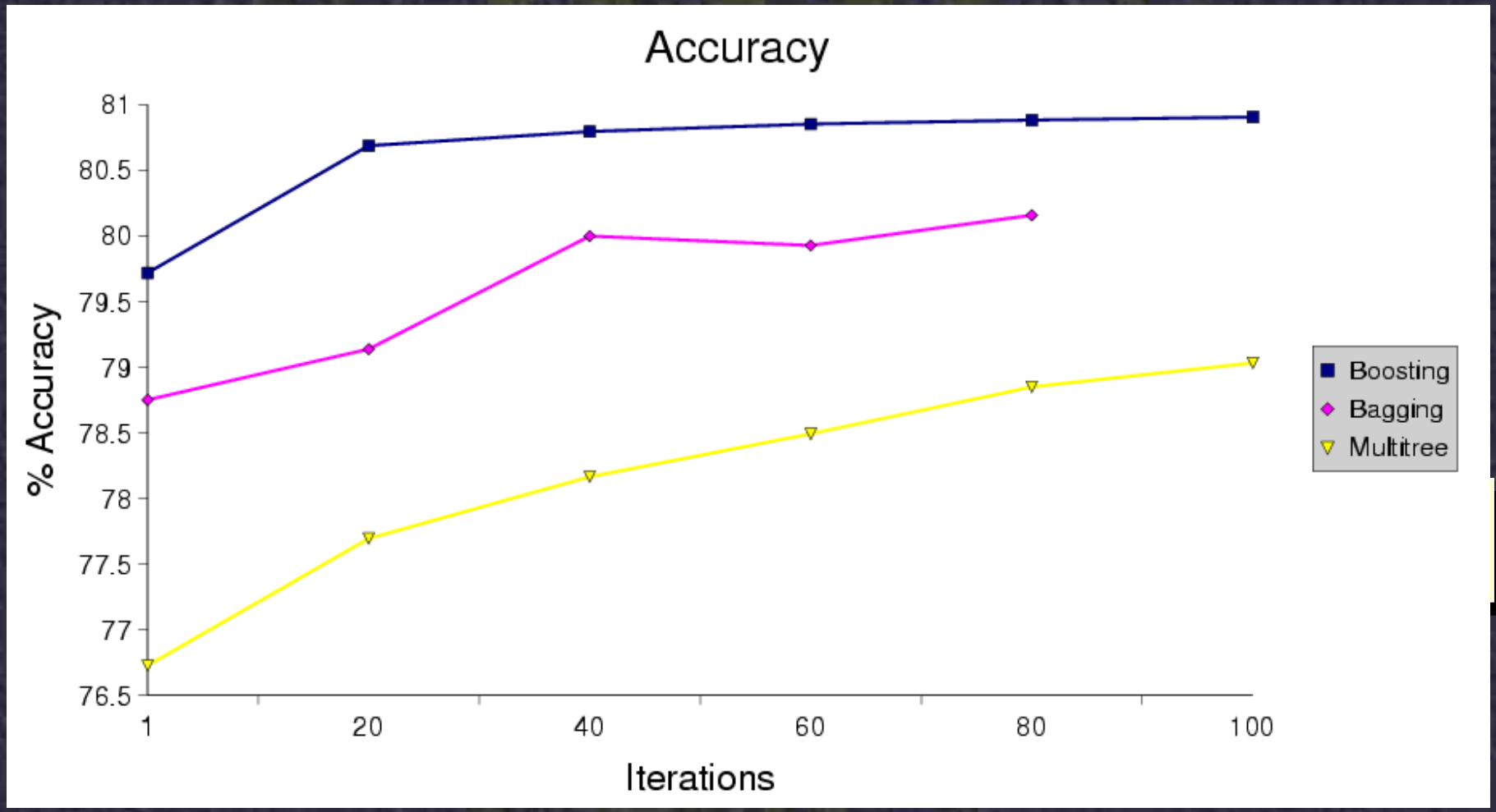
3.3. Multi-árboles como Multi-clasificadores

Precisión de la solución combinada según tamaño del multi-árbol

28 datasets	1	50	100	150	200
Media Geométrica	76.73	78,27	79.03	79,12	79.25

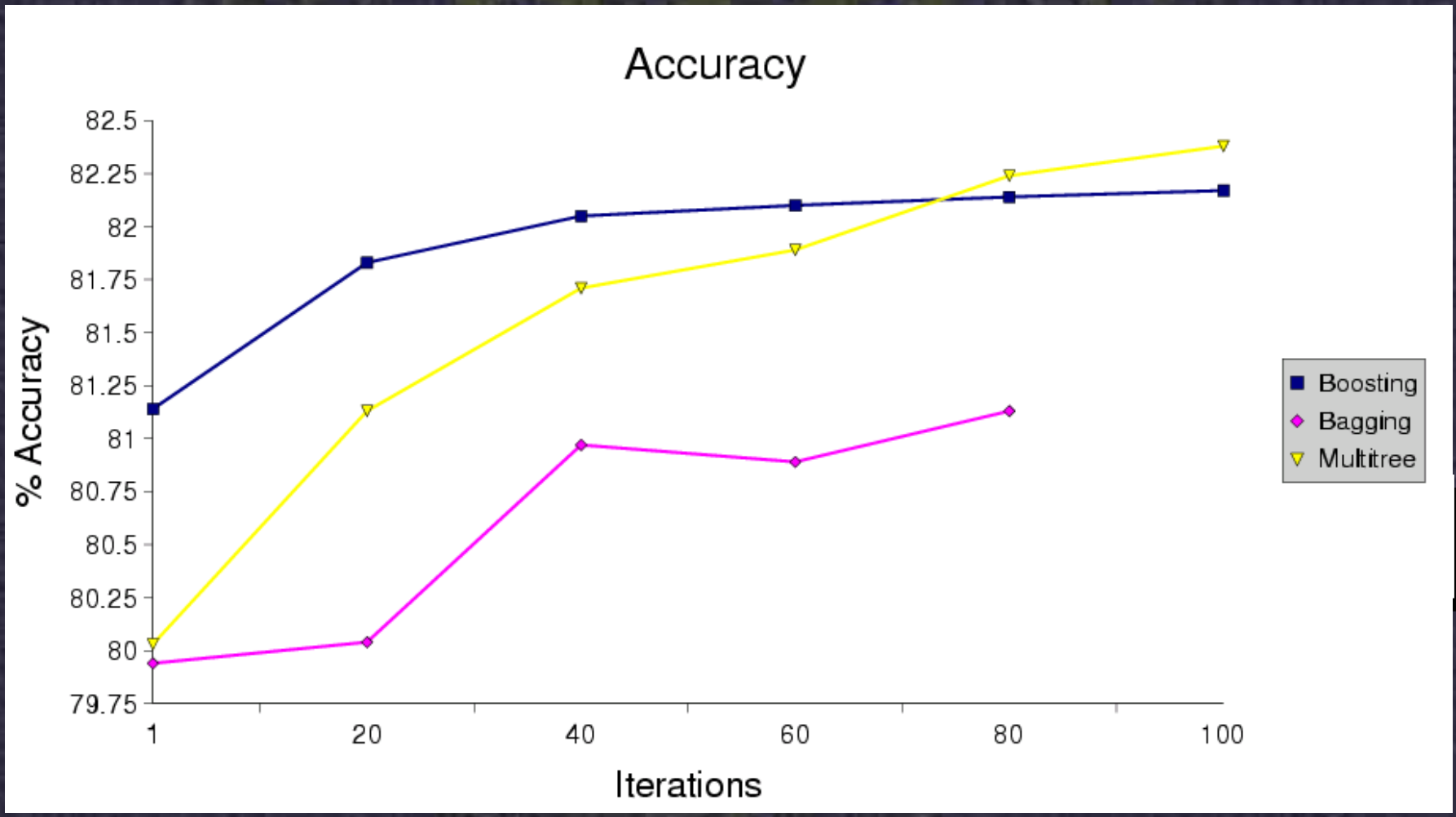
3. Aprendizaje de Multi-árboles de Decisión

3.3. Multi-árboles como Multi-clasificadores



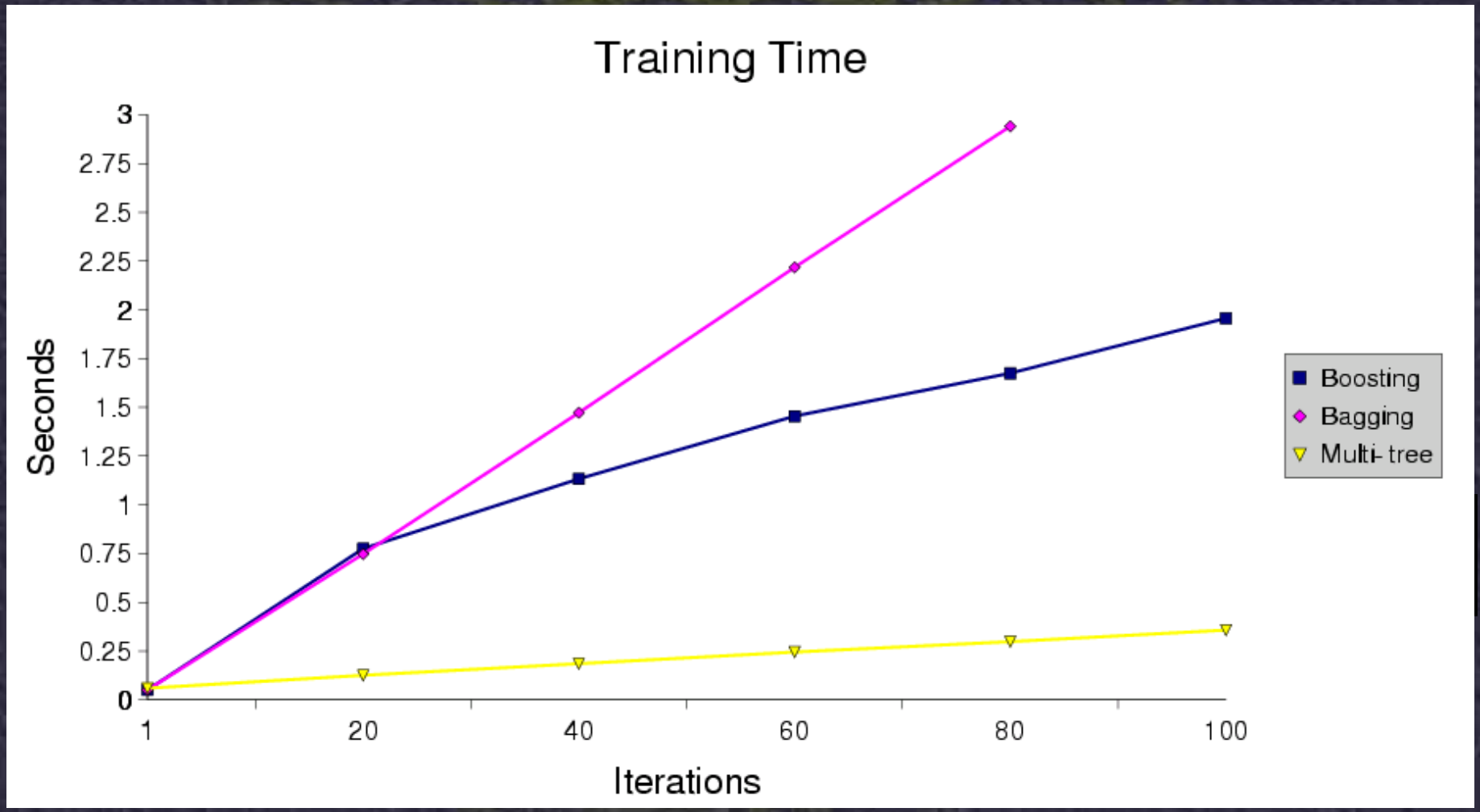
3. Aprendizaje de Multi-árboles de Decisión

3.3. Multi-árboles como Multi-clasificadores



3. Aprendizaje de Multi-árboles de Decisión

3.3. Multi-árboles como Multi-clasificadores



3. Aprendizaje de Multi-árboles de Decisión

3.3. Multi-árboles como Multi-clasificadores

Conclusiones

- Multi-árboles permiten construir modelos más eficientes y simples mediante la exploraciçon de las opciones no consideradas por el algortimo clásico
- El algoritmo de aprendizaje de un multi-árbol puede verse como un algoritmo anytime, ya que mejora el modelo aprendido con respecto al tiempo dedicado al aprendizaje
- Dado que contienen varias hipótesis, también puede considerarse como un método multi-clasificador, pero con una importante ventaja: un mejor tratamiento de recursos computacionales

3. Aprendizaje de Multi-árboles de Decisión

3.4. Archetyping

Principales problemas de métodos Multi-clasificadores

- **Costes Computacionales:** Se necesita gran cantidad de recursos y tiempo para construir y almacenar el conjunto de modelos

- ***Throughput*:** La aplicación del modelo combinado es lenta

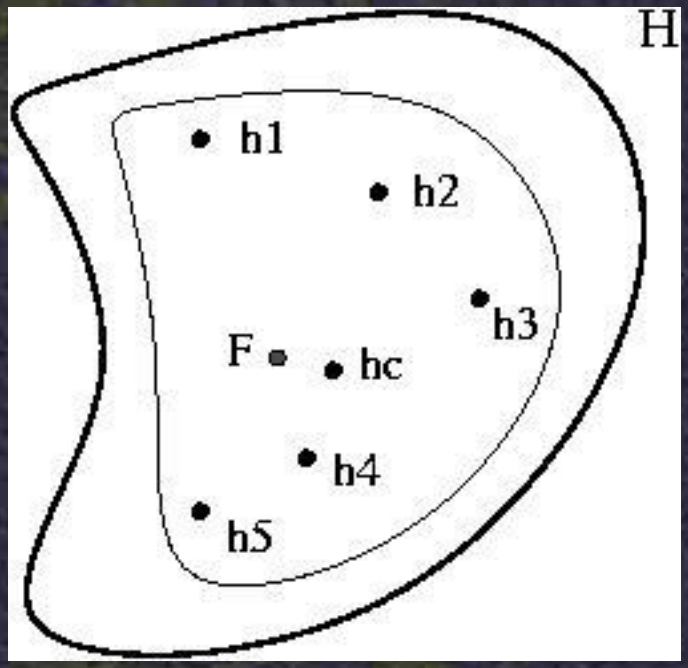
- **Comprensibilidad**: El modelo combinado se comporta como una caja negra

3. Aprendizaje de Multi-árboles de Decisión

3.4. Archetyping

• La clave es hallar un modelo comprensible desde la combinación de *n* modelos sin perder demasiada precisión

• Una posibilidad es seleccionar la mejor hipótesis del conjunto acuerdo con su similitud a la combinación



3. Aprendizaje de Multi-árboles de Decisión

3.4. Archetyping

• Para establecer la similitud semántica utilizamos medidas de discrepancia (o similitud) entre dos clasificadores

$$\theta = \sum_{i=1}^{C} \frac{M_{i,i}}{n}$$

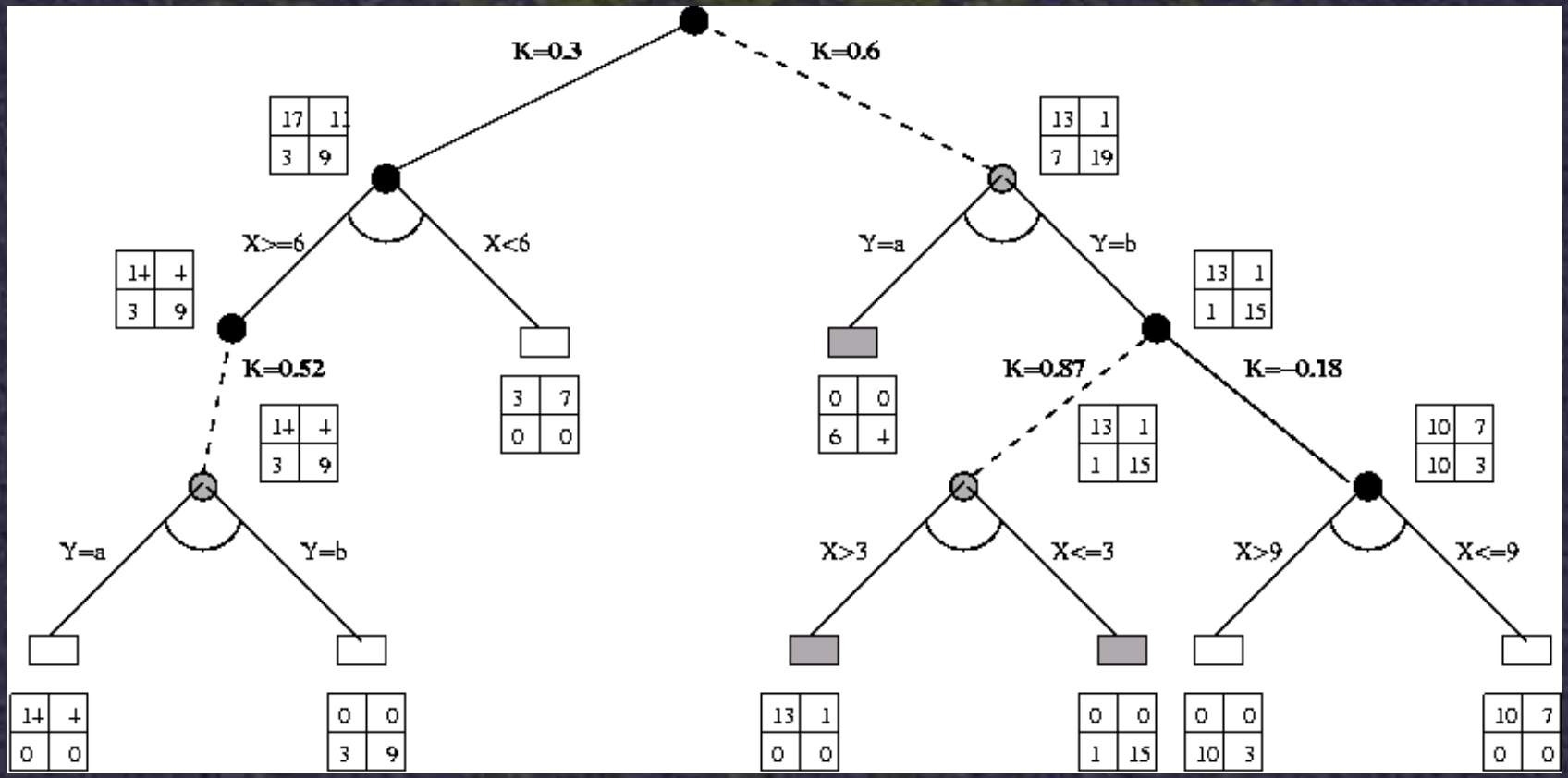
$$\kappa = \frac{\theta - \theta_2}{\theta - 1}$$

$$Q = \frac{\prod_{i=1}^{C} M_{i,i} - \prod_{i=1, j=1, i \neq f}^{C} M_{i,j}}{\prod_{i=1}^{C} M_{i,i} + \prod_{i=1, j=1, i \neq f}^{C} M_{i,j}}$$

• Además para computar estas medidas se utiliza un dataset inventado. De esta manera se computa la similitud con más exactitud

3. Aprendizaje de Multi-árboles de Decisión

3.4. Archetyping



3. Aprendizaje de Multi-árboles de Decisión

3.4. Archetyping

- El alto número de modelos en el mutli-árbol dificulta el computo de la similaridad de cada hipótesis con respecto a la combinada
- Una solución más optima es calcular la similaridad de cada nodo, y entonces, extraer una solución

1. Etiquetar el dataset inventado con la hipótesis
2. Rellenar una matriz de contingencia *M* en cada hoja del multi-árbol usando el dataset inventado ya etiquetado
3. Se Propaga hacia arriba la matriz de contingencia:
   - Para cada nodo AND se acumula la matriz de contingencia de sus nodos hijos: $M = M_1 + M_2 + .. + M_i$
   - Para cada nodo OR, se computa la similaridad de sus hijos, y la matriz *M* del nodo más similar se copia en el nodo padre. Se marca el nodo seleccionado.

Comentar volant:

la utilidad es más que obvia

3. Aprendizaje de Multi-árboles de Decisión

3.4. Archetyping

Resultados

- El procedimiento definido extrae la hipótesis más similar a la combinada si se utiliza θ como medida de similaridad (teorema 11)

- El procedimiento definido no extrae la hipótesis más similar a la combinada si se utiliza κ como medida de similaridad (contraejemplo)

3. Aprendizaje de Multi-árboles de Decisión

3.4. Archetyping

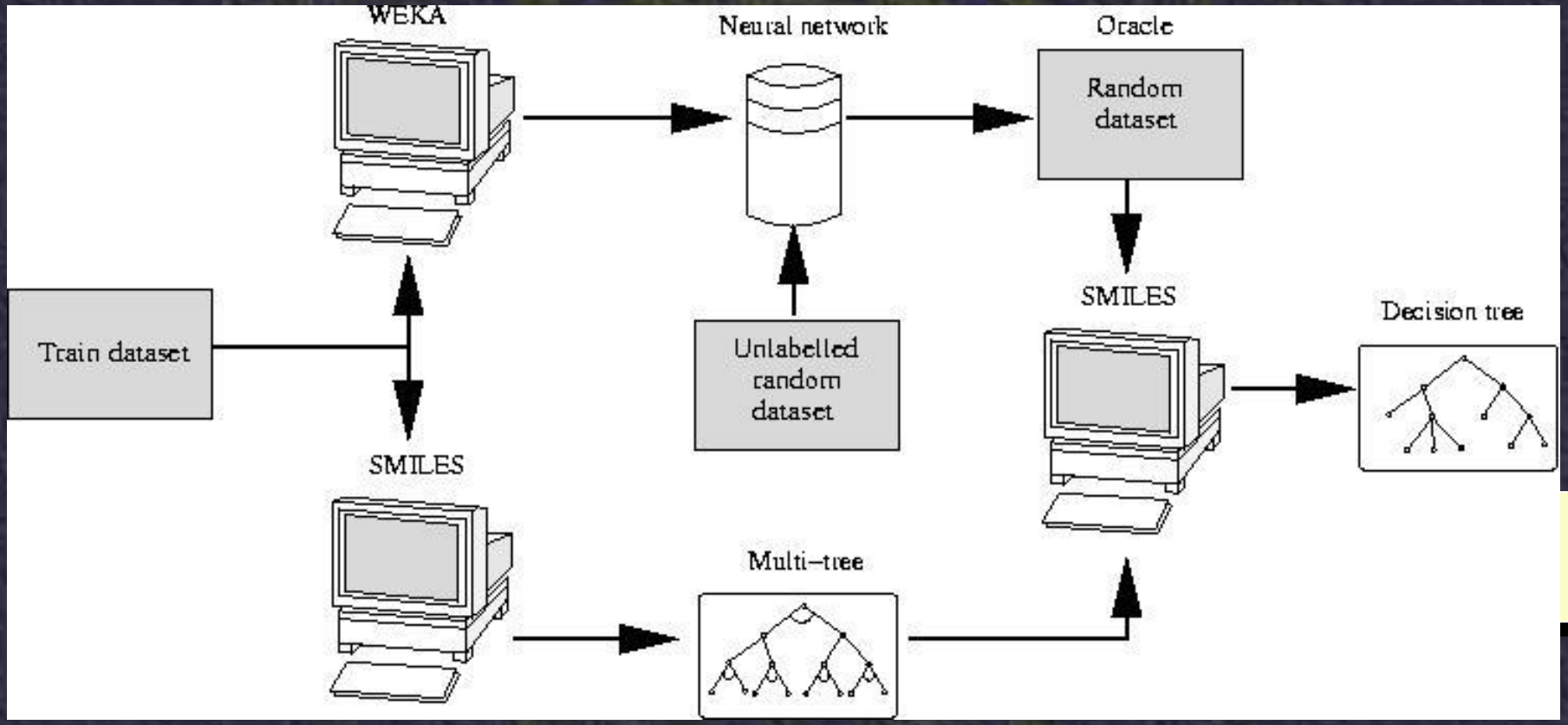
Precisión de la solución combinada y la solución arquetipo según tamaño del multi-árbol

28 datasets	1	50		100		200	
	Comb	Comb	Arc	Comb	Arc	Comb	Arc
Media Geométrica	76.73	78.33	76.99	78.95	77.52	79.26	77.61

<- Comentar volant: la utilidad es más que obvia

3. Aprendizaje de Multi-árboles de Decisión

3.4. Archetyping



3. Aprendizaje de Multi-árboles de Decisión

3.4. Archetyping

Conclusiones

- Archetyping permite la generación de una sola hipotésisi comprensible y precisa utilizando un conjunto de
- Técnica también válida para extraer modelos comprensibles a partir de otros modelos no comprensibles (redes neuronales, SVM)
- Modelo aprendido puede interpretarse como una explicación del comportamiento de la caja negra

3. Aprendizaje de Multi-árboles de Decisión

3.4. Archetyping

Conclusiones

- Archetyping permite la generación de una sola hipótesis comprensible y precisa utilizando un conjunto de
- Técnica también válida para extraer modelos comprensibles a partir de otros modelos no comprensibles (redes neuronales, SVM)
- Modelo aprendido puede interpretarse como una explicación del comportamiento de la caja negra

4. Aprendizaje Sensible al Coste

4.1. Introducción

En muchas situaciones, no todos los errores tienen las mismas consecuencias

Un modo usual de suministrar información sobre los costes de clasificación errónea es una matriz de costes, la cual indica los costes de clasificación correcta o incorrecta

Predicha \ Real	Abierta	Cerrada
Abierta	0	20 €
Cerrada	100 €	0

4. Aprendizaje Sensible al Coste

4.2. Conjunto Óptimo de Asignaciones

Uno de los métodos más efectivos para aprender árboles de decisión sensibles al coste consiste en asignar las clases de las hojas de manera que se minimice el coste

Sin embargo, dado un árbol de *n* hojas, existen 2^n posibles etiquetados, dando lugar a 2^n posibles clasificadores. Podemos utilizar el análisis ROC para determinar el subconjunto óptimo de asignaciones

Sin embargo, el conjunto de posibles asignaciones cumplen unas carácteristicas que permite descubrir el subconjunto óptimo directamente

4. Aprendizaje Sensible al Coste

4.2. Conjunto Óptimo de Asignaciones

Sin embargo, el conjunto de posibles asignaciones cumplen unas características que permite descubrir el subconjunto óptimo directamente.

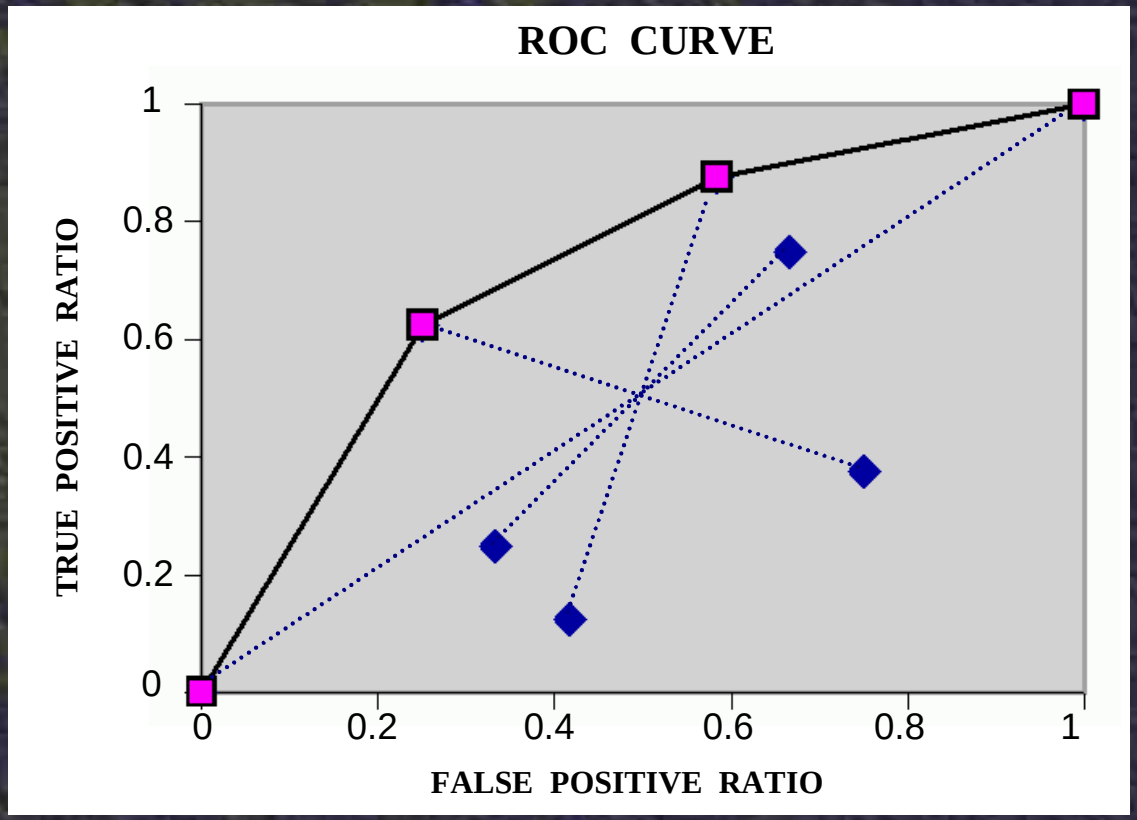
Teorema 17

Dado un árbol de decisión para un problema de 2 clases con n hojas, la superficie convexa correspondiente los 2^n etiquetados posibles está formada por y sólo por los puntos ROC correspondientes al conjunto de los etiquetados óptimos Γ.

4. Aprendizaje Sensible al Coste

4.3. Ejemplo

	T	F
Node 1	3	5
Node 2	5	1
Node 3	4	2



4. Aprendizaje Sensible al Coste

4.3. Ejemplo

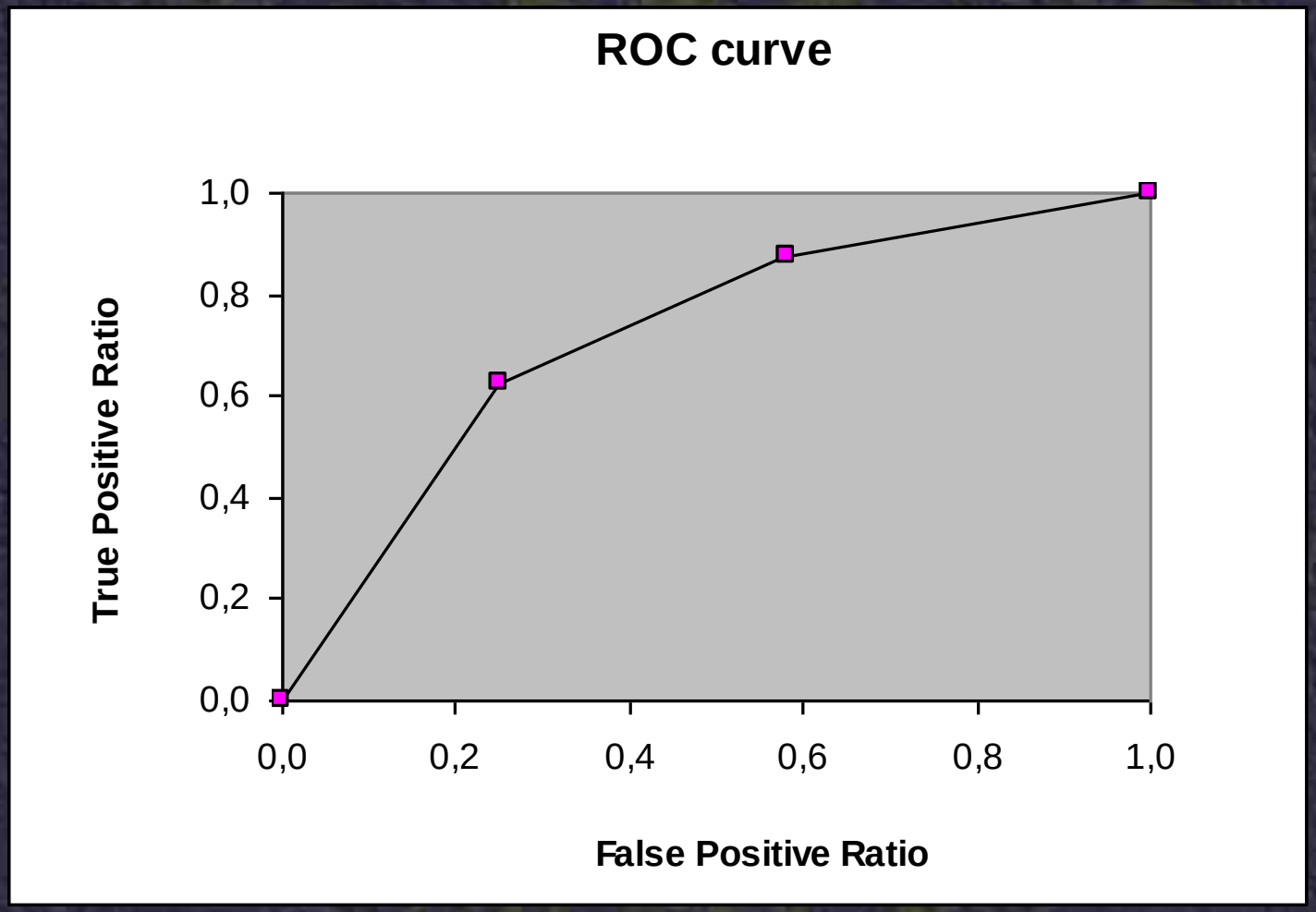
Podemos generar sólo las asignaciones optimas si ordenamos las hojas de acuerdo con la probabilidad de las clases.

Existen sólo k+1 asignaciones óptimas

	T	F				
Node 1	5	1	F	T	T	T
Node 2	4	2	F	F	T	T
Node 3	3	5	F	F	F	T

4. Aprendizaje Sensible al Coste

4.3. Ejemplo



4. Aprendizaje Sensible al Coste

4.4. Aprendiendo Árboles Decisión basados en su AUC

Los resultados anteriores permiten calcular de forma eficiente el AUC de un árbol de decisión (o cualquier otro clasificador formado por reglas disjuntas):

$$\text{Área Total} = \frac{1}{2x_t y_t} \sum_{i=1..n} E^-{}_i \left[\left(\sum_{j=1..i-1} 2E_j{}^+ \right) + E^+{}_i \right]$$

Este hecho permite que utilicemos el AUC para la definición de un nuevo criterio de partición:

$$\text{AUCsplit} = \sum_{i=1..n_j} A(P_{i-1}^j, P_i^j)$$

4. Aprendizaje Sensible al Coste

4.4. Aprendiendo Árboles Decisión basados en su AUC

El AUCSplit es aplicables sólo para problemas con 2 clases. Sin embargo podemos extender su definición con la función M (Hand&Till 2001).

$$\text{MAUC(S)} = \frac{2}{c(c-1)} \sum_{a<b} AUC_{ab}(S)$$

Siguiendo con la idea de que el AUC es una medida de "cuan bien separadas están las distribuciones estimadas de las probilidades de las clases de un clasifcador", hemos definido dos nuevos criterios de partición basados en medidas conocidas: *LogLossSplit* y *MSESplit*

4. Aprendizaje Sensible al Coste

4.5. Experimentos

AUC y precisión según el criterio de partición empleado

28 datasets	AUCs	C45	DKM	LogLoss	Mgini	MSEi
AUC Media Geométrica	0.817	0.821	0.721	0.821	0.808	0.826
Acc. Media Geométrica	75.77	76.33	67.64	76.59	75.94	76.21

4. Aprendizaje Sensible al Coste

4.6. Conclusiones

Conclusiones

- Hemos introducido una nueva manera de computar eficientemente el AUC de un clasificador formado por reglas disjuntas

- Dada la eficiencia de esta medida, puede ser empleada para la definición de nuevo criterio de partición

- Siguiendo la idea del AUC como criterio de partición, definimos nuevos criterios de partición que permiten generar árboles de decisión con mejores AUC.

4. Aportaciones Principales

- Un nuevo marco de aprendizaje basados en lenguajes multi-paradigma: IFLP
- La estructura multi-árbol como método para aprender modelos más precisos y simples
- Un nuevo método multi-clasificador que ahorra recursos compartiendo partes de sus elementos

4. Aportaciones Principales

- Una técnica que construye un modelo preciso y comprensible basandose en la similitud semánticacon respecto a otro clasificador

- Una innovadora manera de calcula el AUC de un clasificador formado por reglas disjuntas

- Nuevos criterios de partición que permiten contruir árboles de decisión con mejor comportamiento en contextos con costes variables

4. Aportaciones Principales

Publicaciones asociadas a la Tesis

- 2000: WFLP'00, Revista Iberoamericana de I.A.
- 2001: FLOPS'01, AGP'01
- 2002: ICCS'02, ICML'02, DS'02, IBERAMIA'02
- 2003: MCS'03, Decision Support for Data Mining

4. Conclusiones

Los métodos y técnicas de aprendizaje aportados permiten:

- Abordar de manera más natural problemas con estructura compleja

- Derivar modelos más simples, comprensibles y precisos de manera eficiente

- Construir hipótesis de decisión que tengan un buen comportamiento en contexto con costes asociados y/o variables