Arboles de Decisión (III)

Carlos Hurtado L.

Depto de Ciencias de la Computación,
Universidad de Chile

Calidad de un árbol de decisión

- La calidad de un árbol depende de dos aspectos:
  - Error de predicción (inverso de precisión)
  - Tamaño
- Principio de la "Navaja de Occam":
  - *Lex parsimoniae*: favorecer lo sucinto
  - Si tenemos dos árboles con el mismo error de predicción, preferimos el más chico.

Precisión vs. Tamaño

- Existe una relación entre precisión y tamaño del árbol, pero…
- …de los árboles generados a partir de un mismo conjunto de datos, los más grandes no son necesariamente los más precisos
- Esto se debe a un fenómeno denominado "sobreajuste"

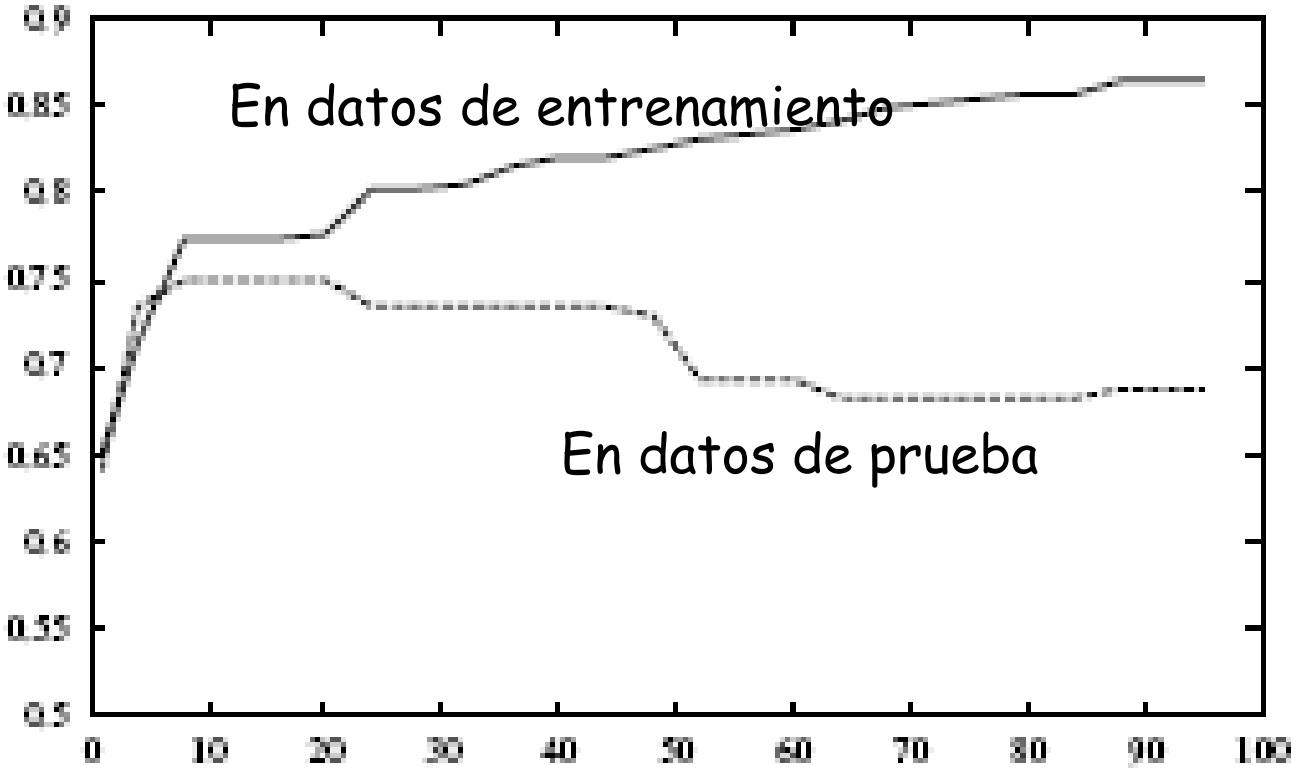
Contenido

- Sobreajuste
- Pre-poda
- Post-poda

Error de entrenamiento vs. Error de prueba

- Datos de entrenamiento
  - Se usan para entrenar el modelo
  - Error en datos de entrenamiento
- Datos de prueba
  - Se usan para estimar precisión del modelo
  - Error en datos de prueba
- Datos objetivo
  - Sobre ellos se aplica el modelo
  - Error en datos objetivos
  - Lamentablemente no podemos conocer con exactitud el error objetivo ("true error")
  - Pero podemos estimarlo

Sobreajuste



Sobreajuste

- Sobre-ajuste: el modelo está sesgado por los datos de entrenamiento
  - Bajo error en datos de entrenamiento pero alto error en datos objetivos
- Problema:
  - El modelo se generaliza mal a datos objetivos

Sobreajuste: Definición Formal

Dado un espacio de modelos M, un modelo m en M es sobreajustado si existe otro modelo m' en M tal que:

- m tiene menor error que m' en datos de entrenamiento
- m' tiene menor error que m en datos objetivo

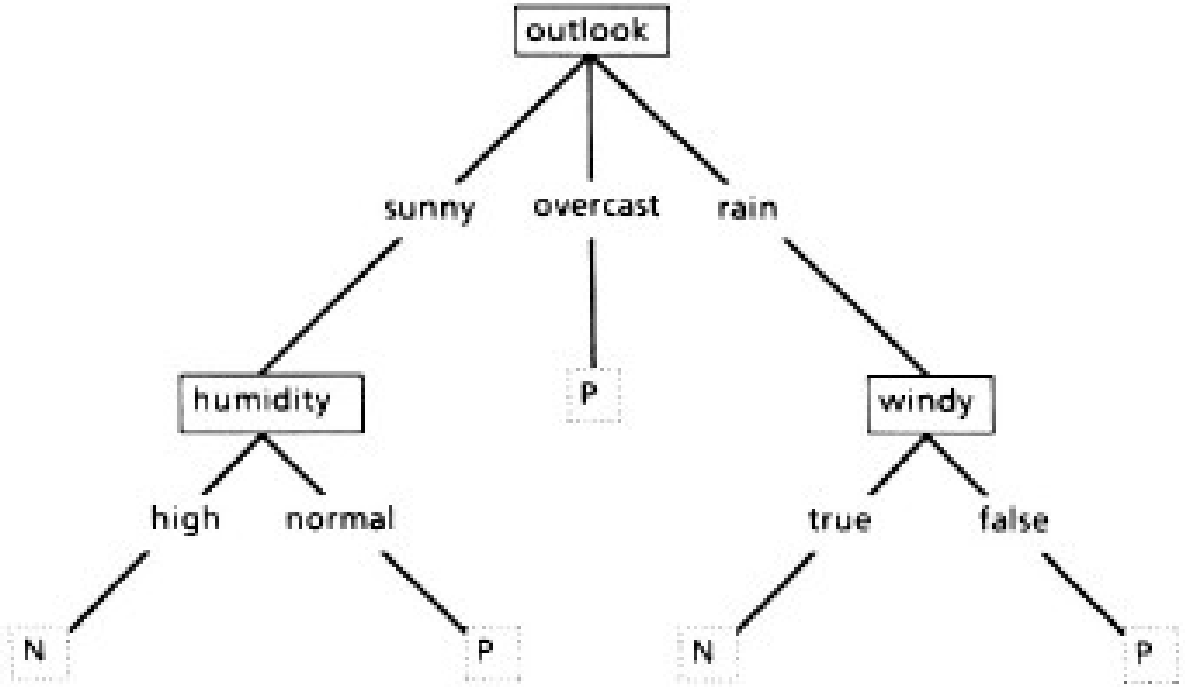
Sobreajuste: Causas

- Datos de entrenamiento no son representativos de datos objetivo
  - Por ruido o sesgo.
- Si una hoja del árbol tiene muy pocos datos la "regla" asociada tendrá baja "significancia estadística"

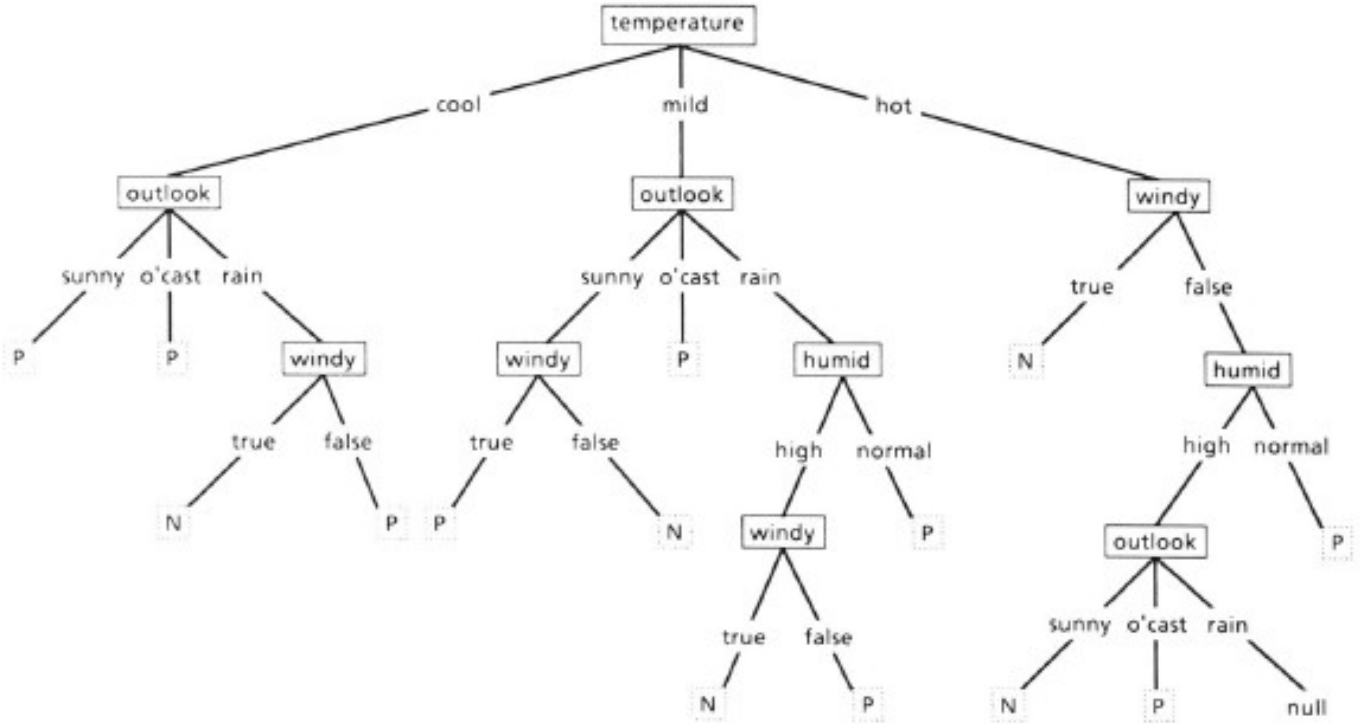
Ejemplo 1 sobreajuste
(weather.nominal)

Outlook	Temp.	Humidity	Windy	Play
Sunny	Hot	High	FALSE	No
Sunny	Hot	High	TRUE	No
Overcast	Hot	High	FALSE	Yes
Rainy	Mild	High	FALSE	Yes
Rainy	Cool	Normal	FALSE	Yes
Rainy	Cool	Normal	TRUE	No
Overcast	Cool	Normal	TRUE	Yes
Sunny	Mild	High	FALSE	No
Sunny	Cool	Normal	FALSE	Yes
Rainy	Mild	Normal	FALSE	Yes
Sunny	Mild	Normal	TRUE	Yes
Overcast	Mild	High	TRUE	Yes
Overcast	Hot	Normal	FALSE	Yes
Rainy	Mild	High	TRUE	No

Arbol Simple



Arbol Complejo



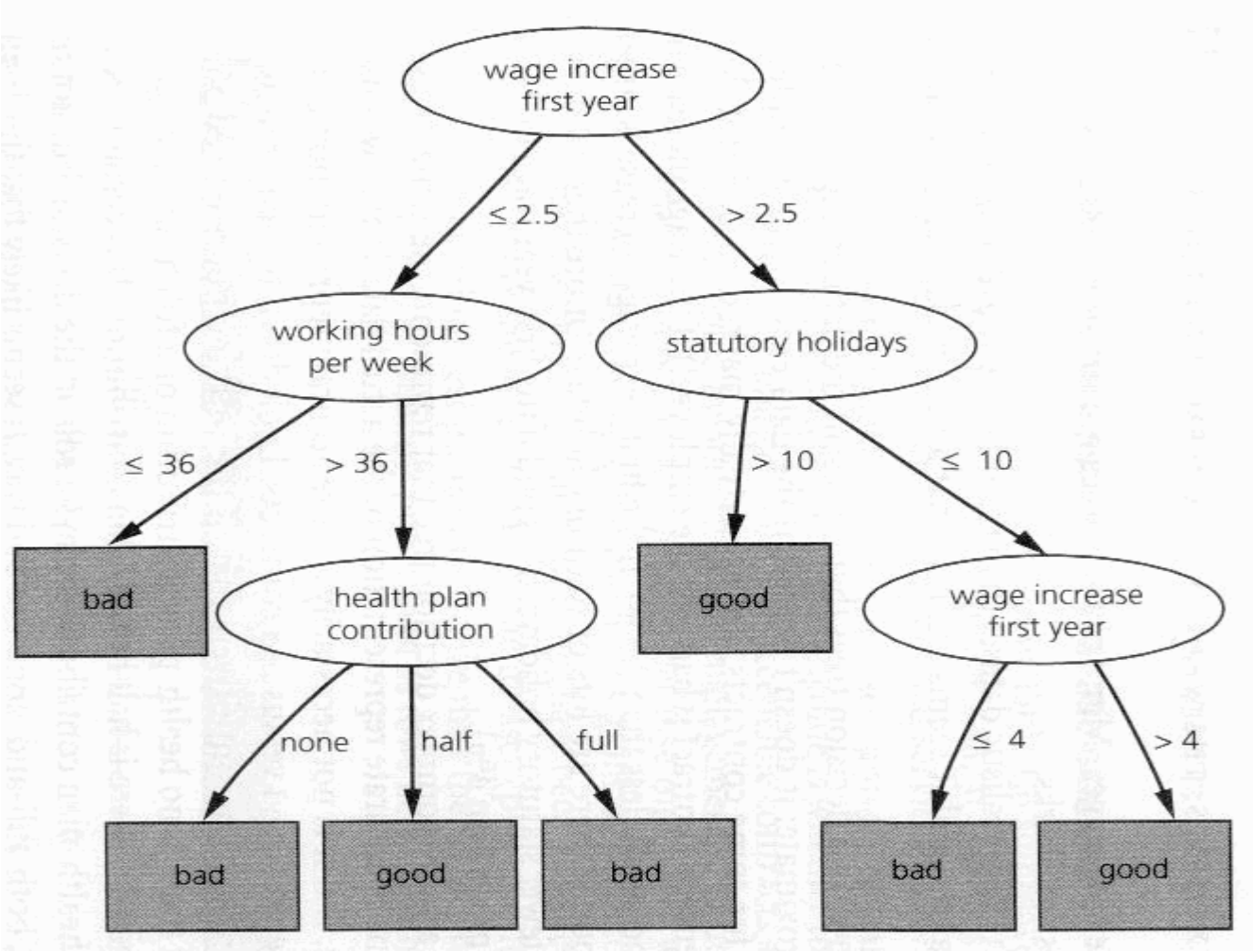
Ejemplo 2 Sobreajuste
(labor)

- Datos de resultados de negociaciones contractuales en Canadá, 1987-88
- Fuente: Weka
- Organizaciones con más de 500 miembros
  - profesores, enfermeras, policía, administrativos universitarios, etc.
- Cada contrato es clasificado bueno o malo para los empleados

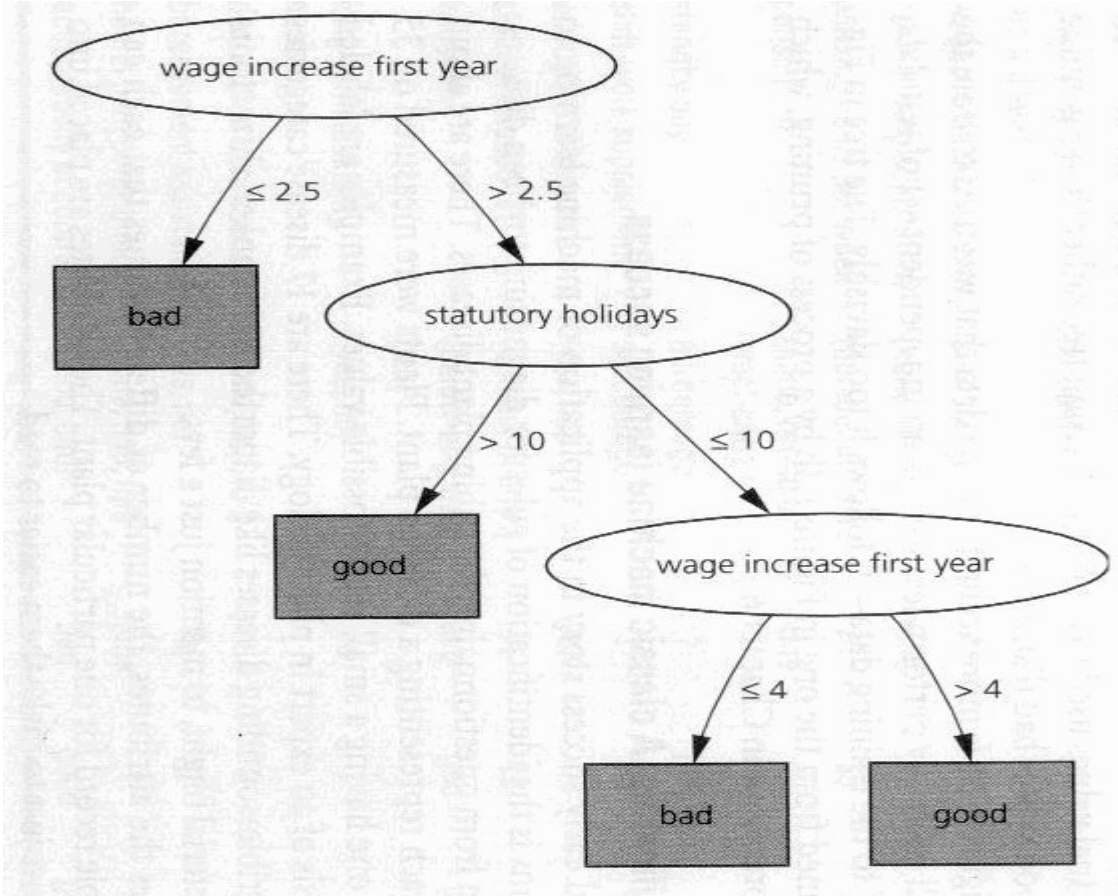
Table 1.6 The labor negotiations data.

attribute	type	1	2	3	...	40
duration	(number of years)	1	2	3		2
wage increase first year	percentage	2%	4%	4.3%		4.5
wage increase second year	percentage	?	5%	4.4%		4.0
wage increase third year	percentage	?	?	?		?
cost of living adjustment	{none, tcf, tc}	none	tcf	?		none
working hours per week	(number of hours)	28	35	38		40
pension	{none, ret-allw, empl-cntr}	none	?	?		?
standby pay	percentage	?	13%	?		?
shift-work supplement	percentage	?	5%	4%		4
education allowance	{yes, no}	yes	?	?		?
statutory holidays	(number of days)	11	15	12		12
vacation	{below-avg, avg, gen}	avg	gen	gen		avg
long-term disability assistance	{yes, no}	no	?	?		yes
dental plan contribution	{none, half, full}	none	?	full		full
bereavement assistance	{yes, no}	no	?	?		yes
health plan contribution	{none, half, full}	none	?	full		half

Arbol Sobreajustado



Arbol obtenido de podar el anterior



Poda

- Mecanismo para mejorar precisión del árbol sobre datos objetivos
- Pre-poda:
  - Parar la construcción del árbol en algunas ramas en tiempo de construcción
- Post-poda
  - Construir un árbol posiblemente sobreajustado y podarlo después.

Pre-poda

- Parar la construcción del árbol en algunas ramas.
- Se evalúa si no se expande un nodo en base a criterios. Ejemplos:
  - No expandir si *GiniSplit* < *k*
  - No expandir si el nodo tiene menos de *k* datos
  - No expandir si test de chi-cuadrado no rechaza independencia
- Al no expandir, etiquetar el nodo con la clase más frecuente

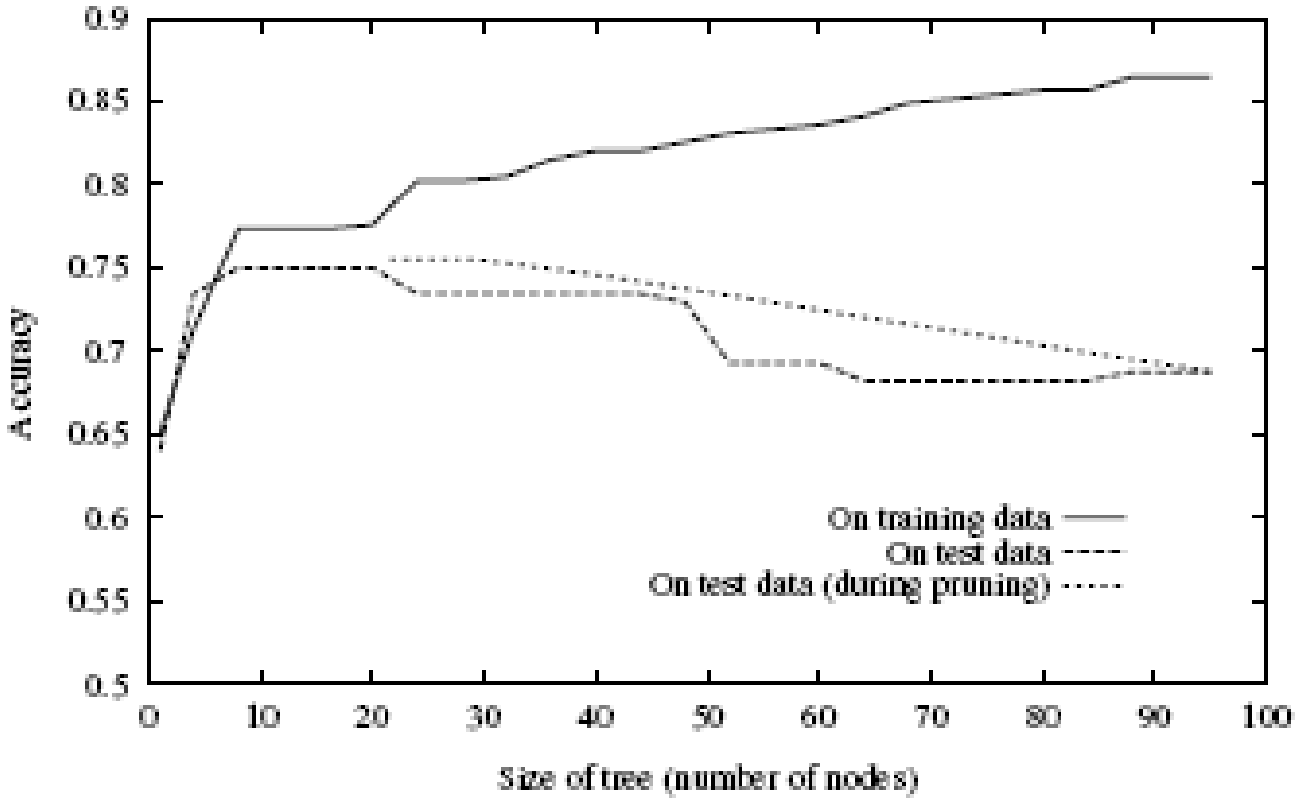
Post-poda

1. Basada en reducción del error (reduced-error post-pruning)
   - Podamos si el error en datos objetivos estimado del árbol resultante es menor.
2. Basada en reglas (rule post-pruning)
   - Variante de 1 pero primero transformamos el árbol en un conjunto de reglas y luego podamos.
3. Basada en Principio de Descripción Mínima
   - Modela el compromiso precisión vs. tamaño
   - Método: minimizar número de bits necesarios para codificar modelo y datos.

Post-Poda basada en reducción de error

- Error objetivo ("true error"):
  - Se define en datos objetivo
  - Num. datos mal clasificados / num. total de datos
- Para cada nodo:
  - A: estimación error objetivo si no se poda el nodo
    - Suma ponderada de errores estimados en nodos hijos
  - B: estimación error objetivo si se poda el nodo
    - Estimación estadística considerando que tenemos una "muestra" de los datos objetivos.
- Si B < A se poda el nodo.

Efecto de post-poda basada en reducción del error



Estimación error Objetivo

- Se estima un intervalo de confianza
  - En base a una muestra estimamos un rango para el error.
- Dos opciones:
  - Muestra son datos de entrenamiento
  - Muestra disjunta de datos de entrenamiento ("reduced-error pruning")
    - Datos de poda

Intervalo de confianza para el Error

- Supongamos que observamos *s* errores en un conjunto de $n_{muestra}$ datos
- Cada dato es un evento de Bernoulli con probabilidad de éxito (error) *p*
  - Media *p*
  - Varianza *p* (1-*p*)
- Estimamos el error como

$$R_{error} = s / n_{muestra}.$$

- R_{error} es la "media de la muestra" (sample mean)

Intervalo de confianza para el Error (cont.)

- Se tiene

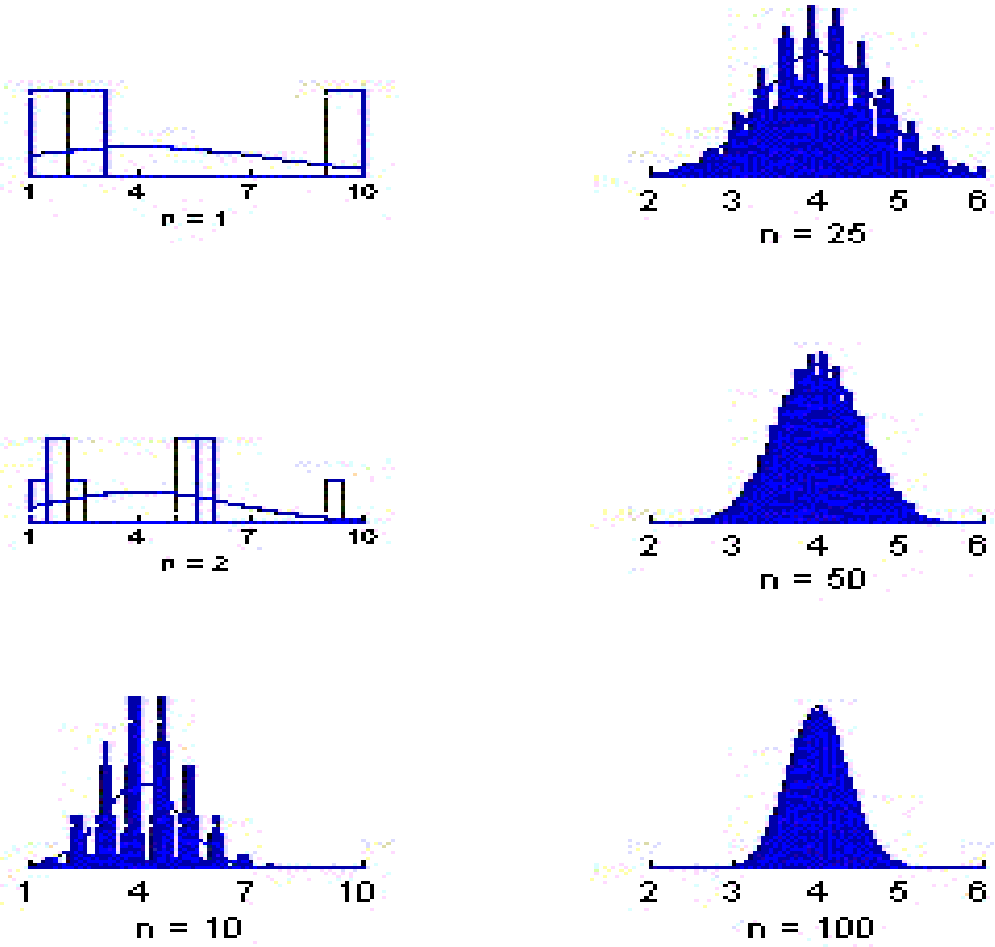
$$E[R_{error}]=p$$

$$Var[R_{error}]=p\,(1-p)/n_{muestra}$$

- Si $n_{muestra}$ es suficientemente grande por el Teorema del Límite Central tenemos

$$R_{error} \sim Normal(p, p\,(1-p)/n_{muestra})$$

Teorema del Límite Central



Intervalo de confianza (IC) para error

- IC para la normal

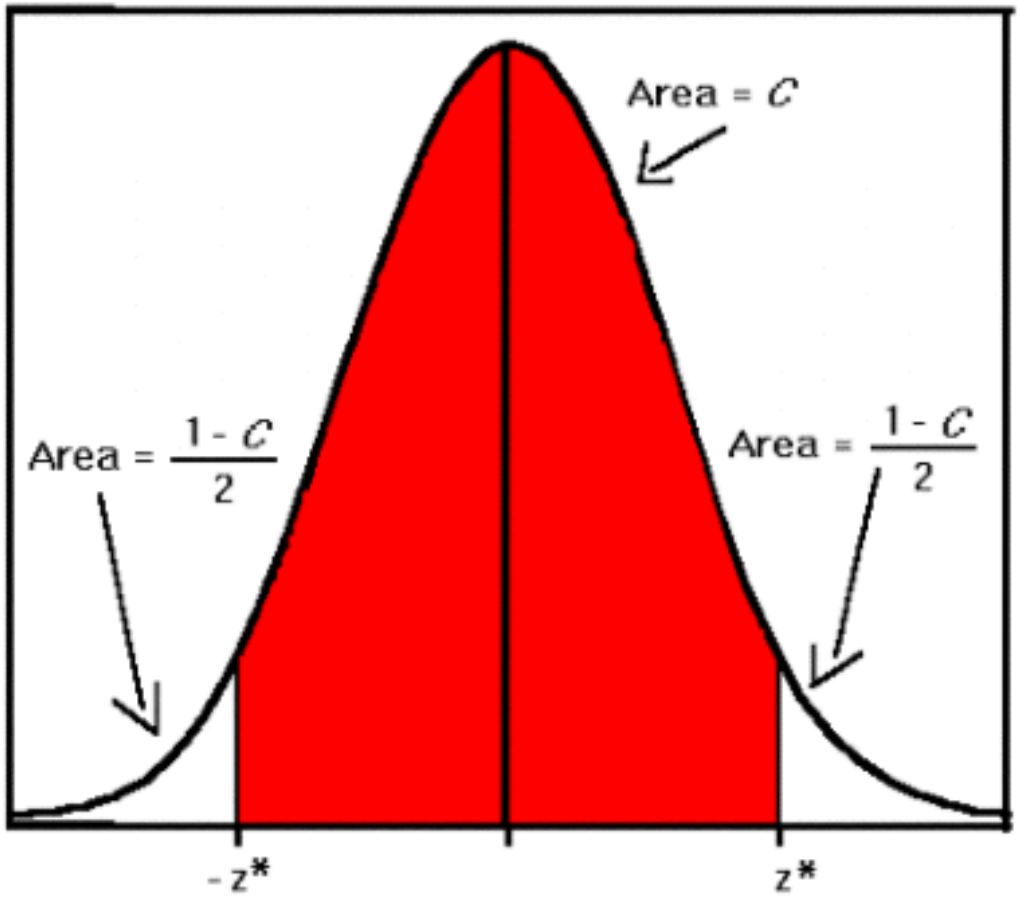
$$P\left[-z < \frac{R_{error} - p}{\sqrt{p(1-p)/n_{muestra}}} < z\right] = c$$

nivel

- IC para *p*

$$p = \frac{R_{error} + \frac{z^2}{2n_{muestra}} \pm z\sqrt{\frac{R_{error}}{n_{muestra}} - \frac{R_{error}^2}{n_{muestra}} + \frac{z^2}{4n_{muestra}^2}}}{1 + \frac{z^2}{n_{muestra}}}$$

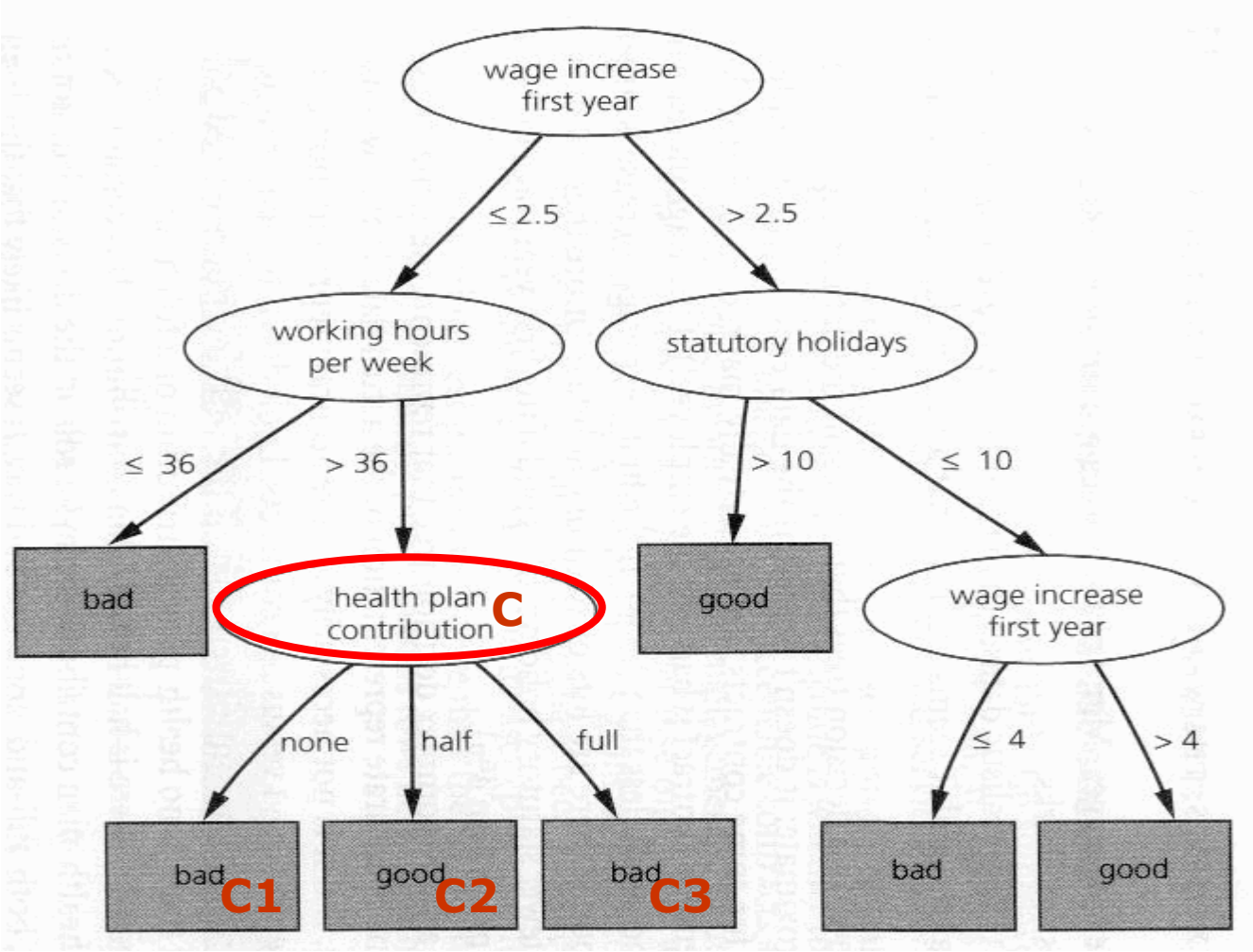
Intervalos de confianza



Estimación de error en C4.5

- Datos de poda = datos de entrenamiento
- Se usa c=50% (z=0,69)
- Bajo nivel de confianza pero el error se fija como el límite superior del intervalo de confianza

Ejemplo: poda de nodo *C*



Ejemplo: poda de nodo C

- Estimación de error A (si no se poda) en c
  - Estimación Error en c1: $s=2$, $n_{muestra.}=6$, $R_{error}=0.33$
    - Obtemos e=0.47
  - Estimación Error en c2: $s=1$, $n_{muestra.}=2$, $R_{error}=0.5$
    - Obtenemos e=0.72
  - Estimación Error en c3:
    - Obtenemos e=0.47
  - Luego, estimación error A en c = 0.51
- Estimación error B (si se poda) en c = 0.46
- B<A, luego reemplazamos c por un nodo hoja.

Post-poda basada en reglas

Pasos:

2. Construir el árbol
3. Convertir el árbol en un conjunto equivalente de reglas
   - Se crea una regla por cada nodo, conjunción de condiciones en un camino desde la raíz al nodo
4. Podar la regla: se elimina una condición de la regla si la regla resultante tiene menor error
5. Ordenar las reglas resultantes de menor a mayor error estimado, se considera este orden en la clasificación

Nota: el resultado es un conjunto de reglas y no necesariamente un árbol

Post-poda basada en reglas

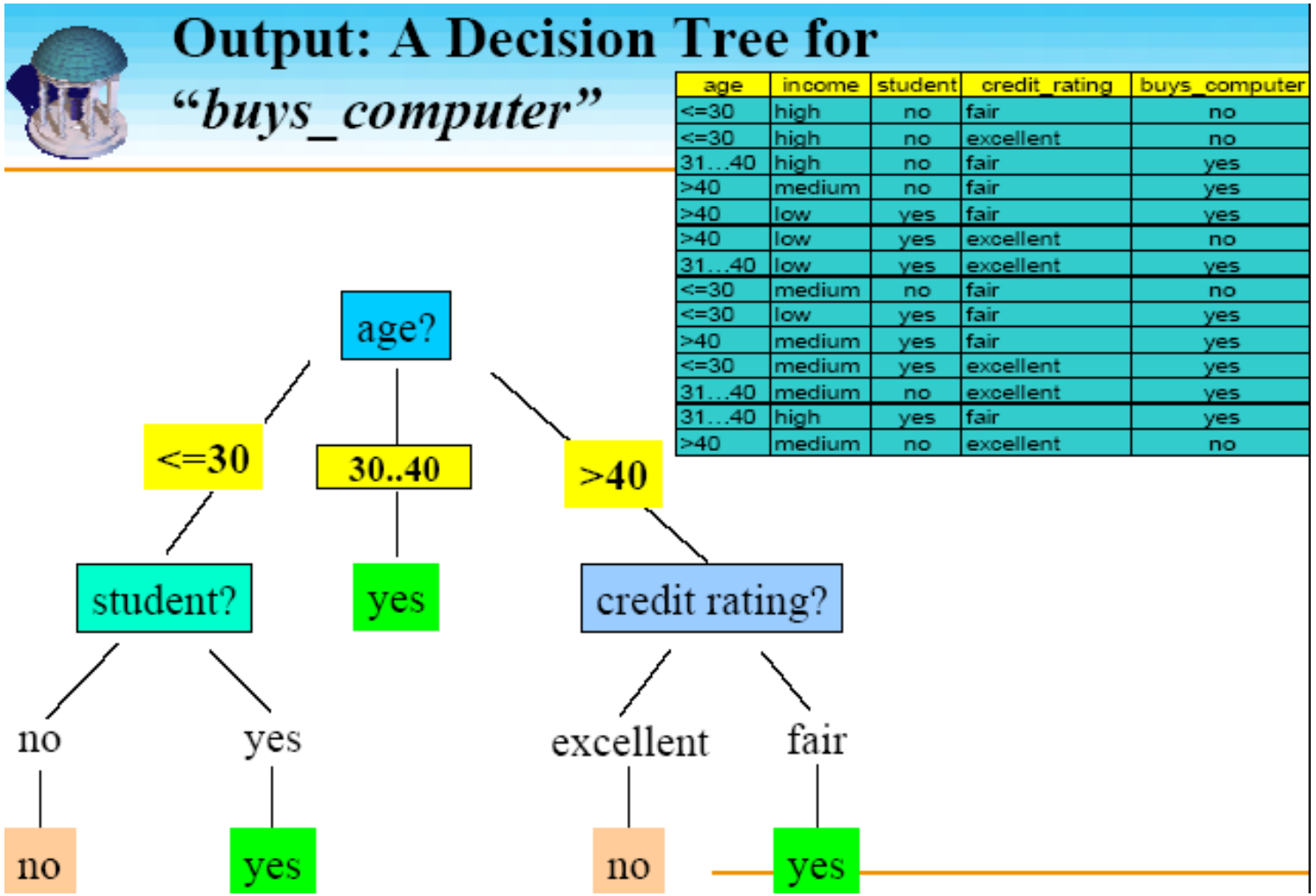
¿Por qué convertir el árbol en reglas para podar?

- El objetivo de la poda es minimizar el error estimado
- Permite más operaciones para reducir el árbol
  - Ejemplo:
  - Podemos podar un nodo interior que causa sobreajuste
  - Podemos podar sólo una condición de un split que causa sobreajuste
- La idea ahora es tratar de hacer mas generales las reglas que componen el árbol, cada una en forma independiente.

Extracción de Reglas de un Arbol

- Representación del árbol como reglas IF-THEN
- Se crea una regla por cada camino desde la raíz a una hoja del árbol.
- Podemos decir que la regla está "soportada" por los datos de la hoja.
  - "Significancia estadística" de la regla depende de la cantidad de datos en su hoja.

Reglas: ejemplo



Reglas: ejemplo (cont.)

IF *age* = "<=30" AND *student* = "*no*" THEN *buys_computer* = "*no*"

IF *age* = "<=30" AND *student* = "*yes*" THEN *buys_computer* = "*yes*"

IF *age* = "31…40" THEN *buys_computer* = "*yes*"

IF *age* = ">40" AND *credit_rating* = "*excellent*" THEN *buys_computer* = "*yes*"

IF *age* = "<=30" AND *credit_rating* = "*fair*" THEN *buys_computer* = "*no*"

Digresión: Reglas vs. Arboles

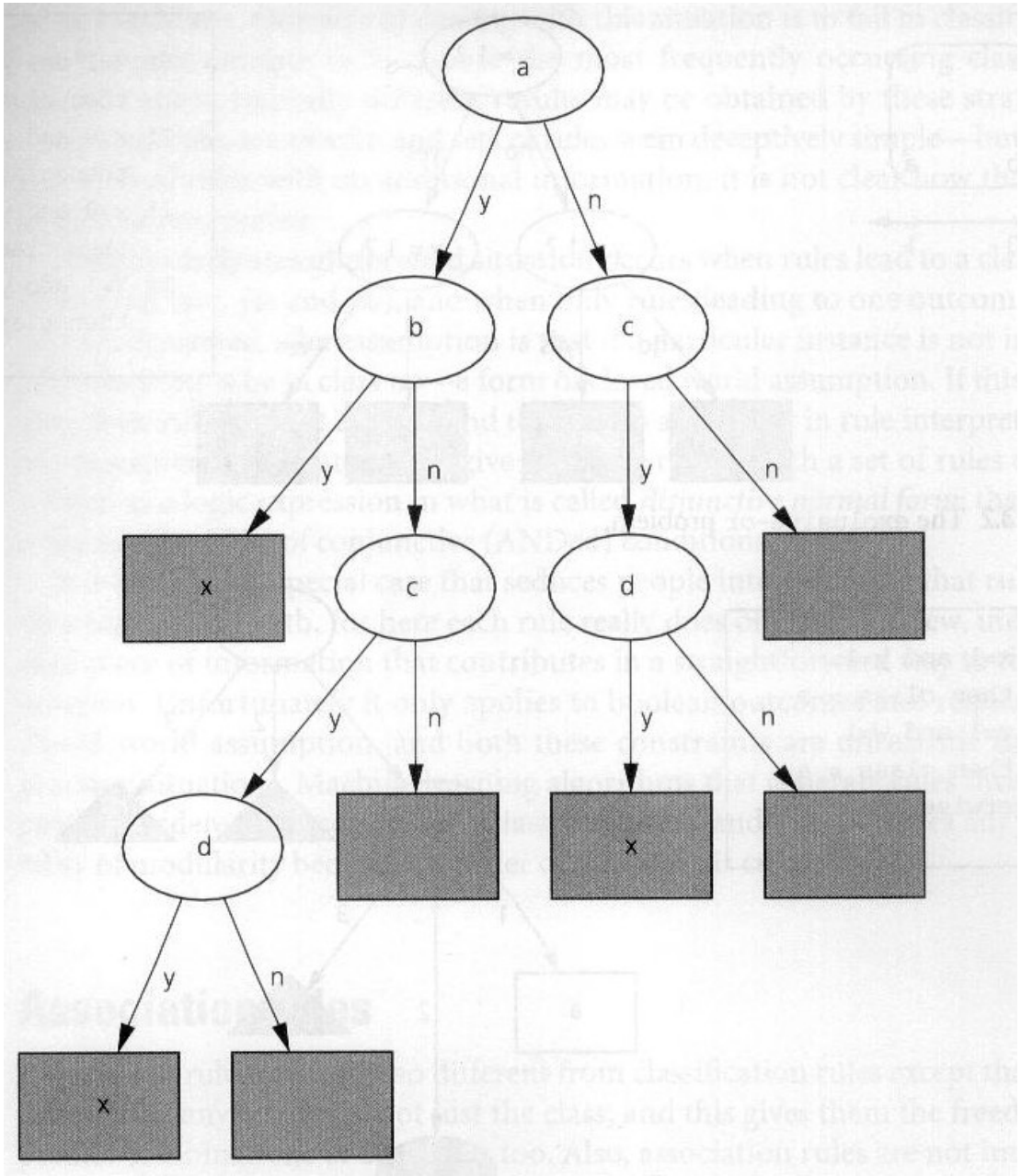
- Transformación de árbol en reglas es fácil
- El inverso puede ser mucho más complejo
- Problema: sub-árboles replicados
- Existen algoritmos que construyen reglas
  - Operan con procedimientos muy distintos a la estrategia "top-down" de Hunt.

Ejemplo

- Construir un árbol para las siguientes reglas:

If a and b then x

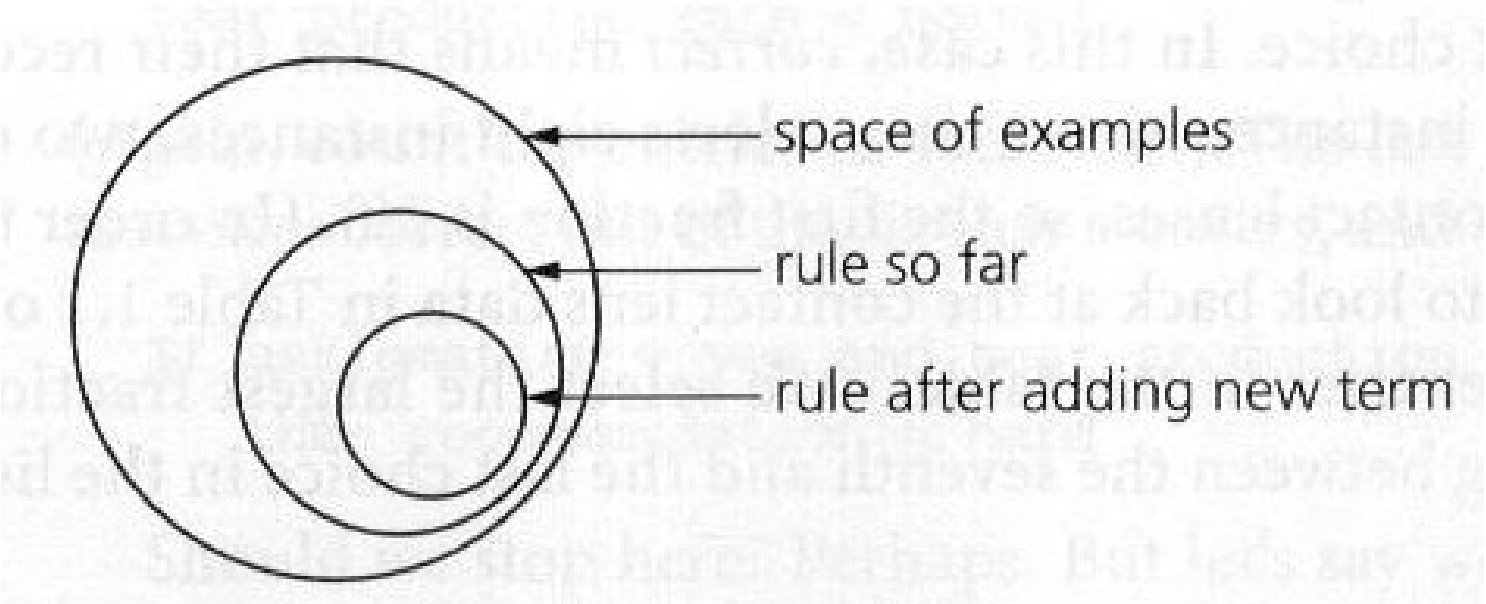
If c and d then x

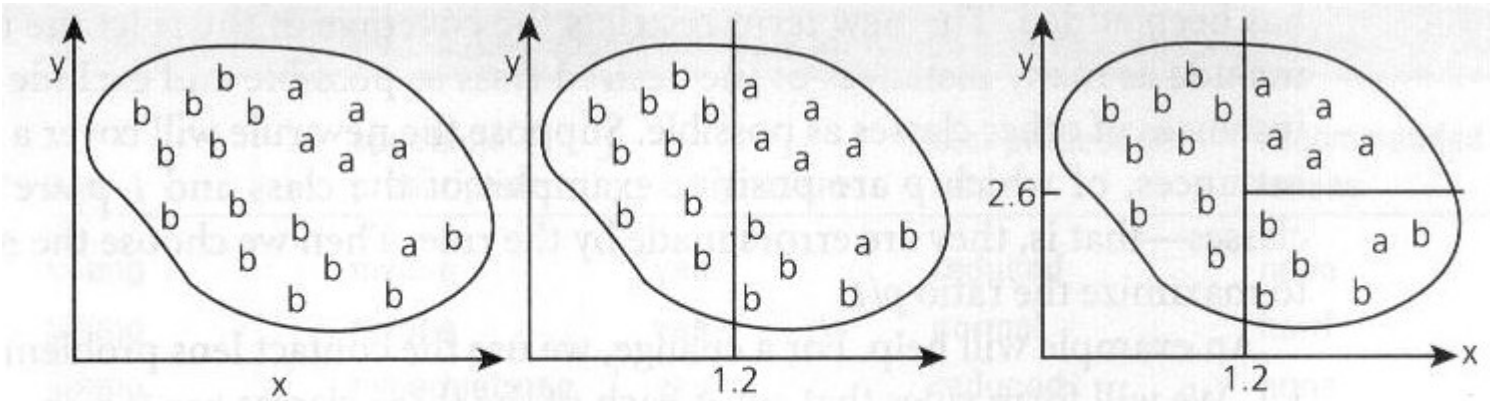

a
y
n
b
c
y
n
y
n
x
c
d
y
n
y
n
d
x
y
n
x

Digresión: Reglas vs. Arboles

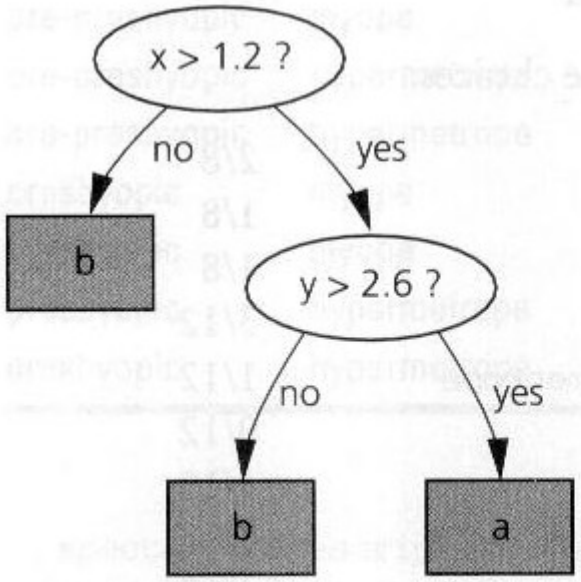
- Transformación de árbol en reglas es fácil
- El inverso puede ser mucho más complejo
- Problema: sub-árboles replicados
- Existen algoritmos que inducen reglas
  - Distintos a la estrategia "top-down" de Hunt.

Inducción de Reglas





(a)

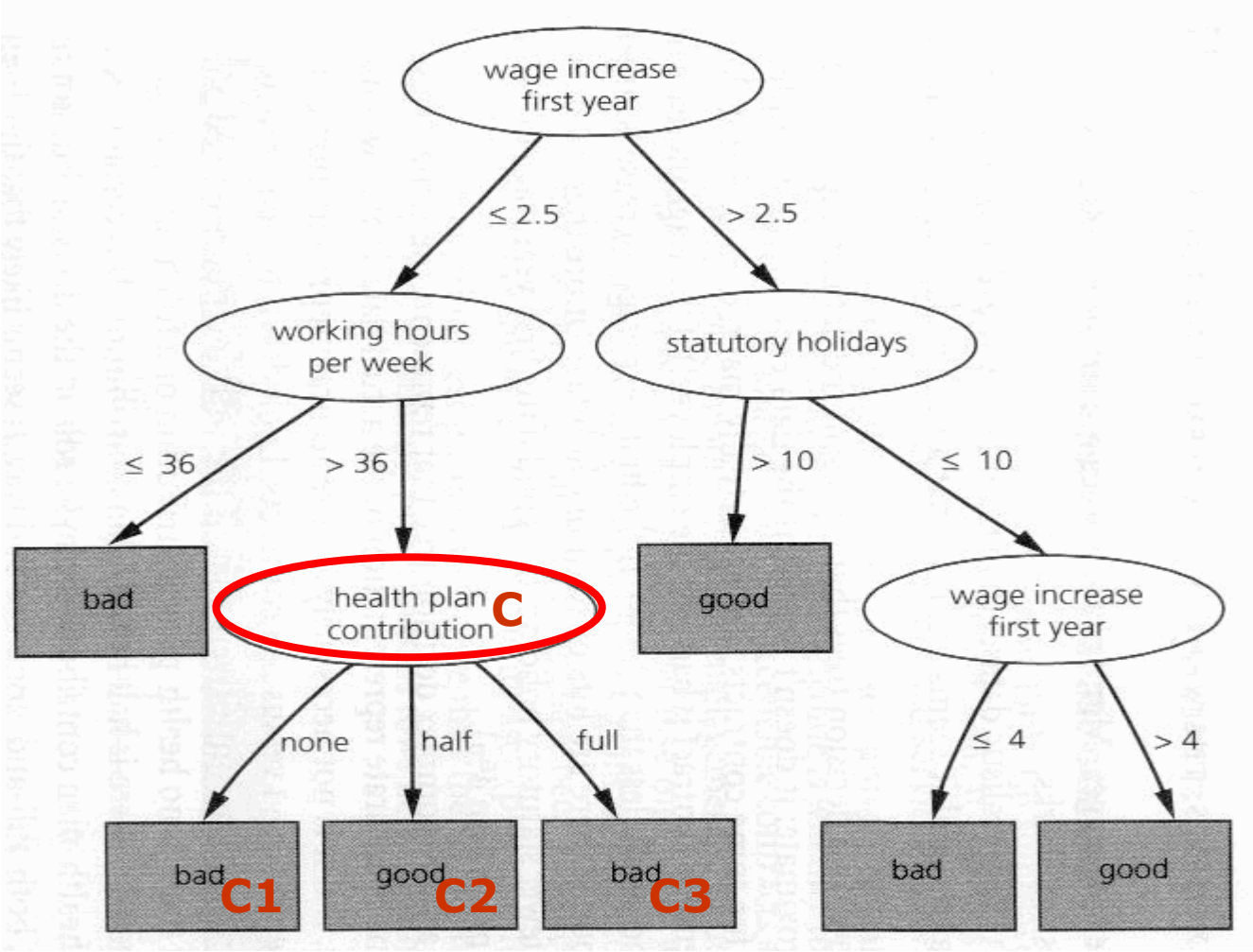


(b)

Figure 4.6 (a) Operation of a covering algorithm; (b) decision tree for the same problem.

Fin digresión sobre reglas

Ejemplo: poda de reglas



Regla C1 Regla C2 Regla C3

Ejemplo: poda basada en reglas

- Supongamos que:
  - Error regla *C1*: 0.45
  - Error regla *C2*: 0.72
  - Error regla *C3*: 0.47
  - Error regla *Ci* (i en {1,2,3}) sin condición *C*=k (k en {non,half,full}): 0.46
- Podamos condición *C*=k en regla *C2*
- Podamos condición *C*=k en regla *C3*
- Regla *C1* queda igual
- Orden de ejecución: *C1* primero que *C2* y *C3*
- Usando poda no basada en reglas podríamos o bien podar *C* completamente o no.

Poda basada en Principio de Largo de Descripción Mínima (MDL)

- Si tenemos muchos datos de entrenamiento y estos son representativos, los métodos de poda anteriores puede dar árboles grandes.
- Necesitamos un criterio de poda que considere el tamaño del árbol.
- Compromiso entre tamaño del modelo y error:
  - Si lo que disminuye el error en una rama es muy poco en comparación al tamaño de la rama, conviene podarla.
- Principio MDL: el mejor modelo para codificar los datos es aquel que minimiza el costo de codificar el modelo más el costo de codificar los datos (dado que conocemos el modelo).
- MDL integra precisión y tamaño en una medida única y en bits.

Principio MDL

- Minimizar
  - Info. necesaria para codificar modelo + info. necesaria para codificar excepciones
- Supongamos que inducimos un modelo M de un conjunto de datos D
- Queremos minimizar:

Bits que toma enviar el modelo

Bits que toma enviar excepciones

$$Cost(M, D) = Cost(M) + Cost(D \mid M)$$

Costos en Poda basada en MDL

- Enfoque simple:
  - C(M): Costo de codificar el modelo
    - Codificación del árbol (como grafo)
    - Codificación de los splits
  - Cost(D|M): Se estima como el número de datos mal clasificados.

Poda basada en Principio MDL (enfoque simple)

- Para cada nodo t, se calcula
  - Costo si podamos t :

$$CostPodado(t) = CostDatos(t)$$

  - Costo si no podamos t:

$$Cost(t) = CostModelo(t) + \sum_{s \text{ hijo } t} Cost(s)$$

- Si $CostPodado(t) < Cost(t)$ podar

Costos en Poda basada en MDL (enfoque basado en Entropía)

- Mensajes a transmitir son las clases de los datos.
- Emisor tiene los datos con las clases y el receptor sin ellas.
- El código (i.e., el costo de enviarlo) de cada mensaje depende del nodo donde se clasifica el dato
- Para hacer esto, debemos trasmitir inicialmente el árbol (C(M))
- C(M) es el tamaño del árbol en bits
- C(D|M) es el costo en bits de trasmitir las clases.
  - Suma de las entropías de las hojas multiplicadas por núm. de datos en las hojas.

Arboles de decisión en Weka

- ID3
  - Sólo maneja atributos nominales
  - Usa ganancia de información (entropía) para seleccionar splits
  - No implementa poda
- J48
  - Implementa algoritmo C4.5 revisión 8.
  - Considera atributos categóricos y numéricos
  - Splits binarios o no binarios.
  - Usa ganancia de información (entropía) para selección de splits
  - Post-poda basada en estimación de error
    - Incluye reduced-error pruning

Ejercicio

- Corra id3 y j48 con los datos weather y weather.nominal
- Compare el error de cada uno
- Cambie los parámetros de poda de j48 con weather y con labor
- Para datos labor haga un ranking de los atributos más relevantes para clasificar, por chi-cuadrado y ganancia de información