Introducción al Diseño de Experimentos para el Reconocimiento de Patrones

Capítulo 4: Inducción de árboles de decisión

Curso de doctorado impartido por
Dr. Quiliano Isaac Moro
Dra. Aránzazu Simón Hurtado
Marzo 2004

Capítulo 4: Inducción de árboles de decisión

Introducción

- Modelo de clasificación
  - Entrevistando a expertos
  - Inductivamente, generalizando a partir de ejemplos específicos
    - C4.5 de Quinlan (proviene del ID3)
    - Clasificación mediante la división del espacio de ejemplos

Introducción. Requisitos modelo C4.5

- Descripción atributo-valor
  - Caso -> Atributos -> Valores discretos o continuos
  - Atributos no deben variar de un caso a otro (excluye dominios en que los objetos tienen estructura variable)
- Clases predefinidas
  - Las categorías se deben establecer de antemano (aprendizaje supervisado)
- Clases discretas
  - Un caso o pertenece o no pertenece a una clase
  - Debe haber muchos más casos que clases
  - Clases continuas se pueden dividir en categorías difusas

Introducción. Requisitos modelo C4.5

- Datos suficientes
  - Se aplican tests estadísticos
  - La cantidad de datos depende de:
    - Nº de propiedades
    - Nº de clases
    - Complejidad del modelo de clasificación
- Modelos de clasificación lógicos
  - Descripción de la clase mediante una expresión lógica
  - Discriminante lineal -> aritmético, no lógico

Introducción

- Problemas apropiados
  - Los ejemplos se representan mediante pares atributo-valor
  - La función objetivo tiene valor de salida discreto
  - Los datos de entrenamiento pueden contener valores desconocidos de atributos

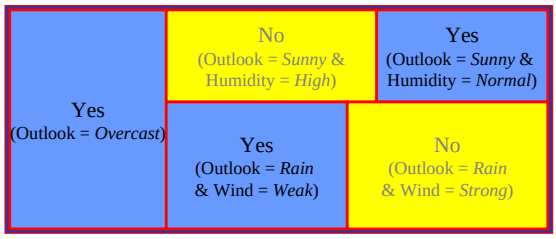
Introducción. Ejemplo

- Problema ejemplo: Jugar al tenis

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Introducción. Ejemplo

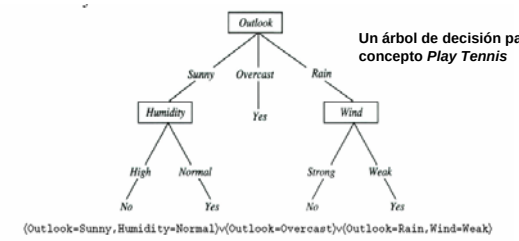
- Espacio de ejemplos



8

Introducción

- C4.5 genera un clasificador en forma de árbol de decisión:
  - Nodo terminal u hoja -> clase
  - Nodo decisión -> test sobre el valor de un atributo simple



Un árbol de decisión para el concepto *Play Tennis*

9

Introducción

- Cada camino desde la raíz hasta una hoja corresponde a una conjunción de tests de atributos y se etiqueta con un valor objetivo.
- Los árboles de decisión representan una disyunción de conjunciones de restricciones sobre los valores de los atributos.

10

Introducción

- Métodos heurísticos para simplificar árboles
- Propósito de modelos de clasificación
  - Desarrollo de predictores precisos y
  - Modelos más inteligibles para el hombre
    - Dividir un árbol en una jerarquía de árboles pequeños
    - C4.5: reexpresar el modelo de clasificación mediante reglas de producción

11

Construcción de árboles de decisión

- Divide y vencerás (Hoveland y Hunt)
  - Tenemos T casos de entrenamiento. Sean las clases $\{C_1, C_2, \ldots, C_k\}$:
    - T contiene 1 ó más casos, todos de una misma clase:
      - El árbol para T es una hoja con la clase C_j
    - T no contiene casos:
      - El árbol es una hoja con la clase mayoritaria (según conocimiento del dominio)
    - T contiene casos de varias clases:
      - Se elige un test basado en un solo atributo que tiene uno o más resultados mutuamente excluyentes $\{O_1, O_2, \ldots, O_n\}$
      - T se divide en subconjuntos $T_1, T_2, \ldots, T_n$, donde T_i contiene todos los casos de T con el resultado O_i del test elegido
      - El árbol es un nodo de decisión que identifica el test y una rama para cada posible resultado
      - Esto se aplica recursivamente a cada subconjunto de casos de entrenamiento hasta que todos los casos de los subconjuntos sean de la misma clase

12

Construcción de árboles de decisión

- Evaluación de test
  - El árbol debe revelar la estructura del dominio para tener potencia predictiva
  - Necesitamos un número significativo de casos en cada hoja, o la partición debe tener tan pocos bloques como sea posible
  - Elegir un test en cada paso tal que el árbol final sea pequeño
  - Explorar todos los árboles posibles y seleccionar el más simple es NP-completo
  - Se usan algoritmos voraces (greedy), no backtracking
  - Importante hacer bien la elección del test en cada nodo

Construcción de árboles de decisión
Evaluación de test

- Criterio de ganancia
  - Sup. un test con n resultados que dividen T (casos de entrenamiento) en subconjuntos T_1, T_2, ... , T_n. Si este test va a ser evaluado sin explorar divisiones subsiguientes de los T_i, sólo conocemos la distribución de clases en T y sus subconjuntos.
  - ID3 usó un criterio de ganancia basado en la Teoría de la Información
  - H(T) mide la cantidad media de información necesaria para identificar la clase de un caso en T (entropía)

Construcción de árboles de decisión
Evaluación de test

- Notación: Si S es cualquier conjunto de casos, frec(C_i, S) es el número de casos de S que pertenecen a la clase C_i, |S| es el número de casos en el conj. S

$$I(C_j) = -\log_2\left(\frac{frec(C_j,S)}{|S|}\right) bits$$

$$H(S) = -\sum_{j=1}^{k} \frac{frec(C_j,S)}{|S|} \times \log_2\left(\frac{frec(C_j,S)}{|S|}\right) bits$$

Construcción de árboles de decisión
Evaluación de test

- Consideremos una medida similar después de que T ha sido dividido según los n resultados de un test X

$$H_X(T) = \sum_{i=1}^{n} \frac{|T_i|}{|T|} \times H(T_i)$$

- La información que se gana dividiendo T de acuerdo al test X

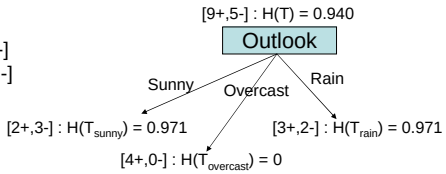
$$G(X) = H(T) - H_X(T)$$

- El criterio de ganancia selecciona el test que maximice esta ganancia de información (información mutua entre el test X y la clase)

Construcción de árboles de decisión
Ejemplo

T = [9+,5-] $H(T) = -\frac{9}{14}\log_2\left(\frac{9}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right) = 0.940\ bits$

- Atributo *Outlook*
  - T_{sunny} = [2+,3-]
    $T_{overcast}$=[4+,0-]
  - T_{rain}=[3+,2-]



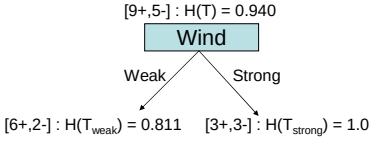
$$G(Outlook) = H(T) - \sum_{i\in\{sunny,overcast,rain\}} \frac{|Ti|}{|T|} \times H(Ti) = H(T) - \frac{5}{14}H(T_{sunny}) - \frac{4}{14}H(T_{overcast}) - \frac{5}{14}H(T_{rain})$$

$$= 0.940 - \frac{5}{14}\left(-\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5}\right) - \frac{4}{14}\left(-\frac{4}{4}\log_2\frac{4}{4} - \frac{0}{4}\log_2\frac{0}{4}\right) - \frac{5}{14}\left(-\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5}\right)$$

$$= 0.940 - \frac{5}{14}0.971 - \frac{4}{14}0.0 - \frac{5}{14}0.971 = 0.246$$

Construcción de árboles de decisión
Ejemplo

- Atributo *Wind*
  - T_{weak} = [6+,2-]
  - T_{strong}=[3+,3-]



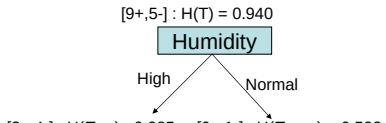
$$G(Wind) = H(T) - \sum_{i\in\{weak,strong\}} \frac{|Ti|}{|T|} \times H(Ti) = H(T) - \frac{8}{14}H(T_{weak}) - \frac{6}{14}H(T_{strong})$$

$$= 0.940 - \frac{8}{14}\left(-\frac{6}{8}\log_2\frac{6}{8} - \frac{2}{8}\log_2\frac{2}{8}\right) - \frac{6}{14}\left(-\frac{3}{6}\log_2\frac{3}{6} - \frac{3}{6}\log_2\frac{3}{6}\right)$$

$$= 0.940 - \frac{8}{14}0.811 - \frac{6}{14}1.00 = 0.048$$

Construcción de árboles de decisión
Ejemplo

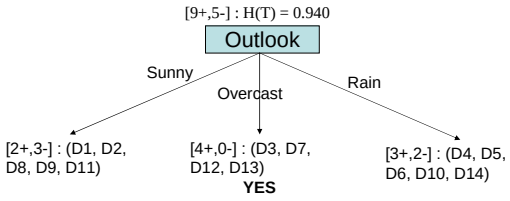
- Atributo *Humidity*
  - T_{high} = [3+,4-]
  - T_{normal}=[6+,1-]



$$G(Humidity) = H(T) - \sum_{i\in\{high,normal\}} \frac{|Ti|}{|T|} \times H(Ti) = H(T) - \frac{7}{14} H(T_{high}) - \frac{7}{14} H(T_{normal})$$
$$= 0.940 - \frac{7}{14}\left(-\frac{3}{7}\log_2\frac{3}{7} - \frac{4}{7}\log_2\frac{4}{7}\right) - \frac{7}{14}\left(-\frac{6}{7}\log_2\frac{6}{7} - \frac{1}{7}\log_2\frac{1}{7}\right)$$
$$= 0.940 - \frac{7}{14}0.985 - \frac{7}{14}0.592 = 0.151$$

Construcción de árboles de decisión
Ejemplo

- Mejor atributo?
  - ***G*(*Outlook*) = 0.246**
  - *G*(*Humidity*) = 0.151
  - *G*(*Wind*) = 0.048
  - *G*(*Temperature*) = 0.029



Construcción de árboles de decisión
Ejemplo

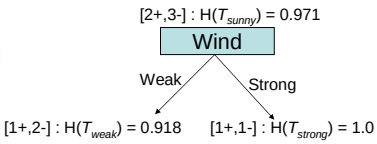
- Entropía T_{sunny}

$$H(T_{sunny}) = H([2+,3-]) = -\frac{2}{5}\log_2\left(\frac{2}{5}\right) - \frac{3}{5}\log_2\left(\frac{3}{5}\right) = 0.971\ bits$$

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes

Construcción de árboles de decisión
Ejemplo

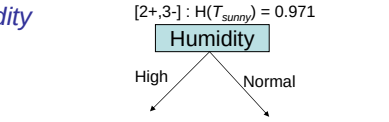
- Atributo *Wind*
  - T_{weak} = [1+,2-]
  - T_{strong}=[1+,1-]



$$G(Wind) = H(T_{sunny}) - \sum_{i\in\{weak,strong\}} \frac{|Ti|}{|T|} \times H(Ti) = H(T_{sunny}) - \frac{3}{5} H(T_{weak}) - \frac{2}{5} H(T_{strong})$$
$$= 0.971 - \frac{3}{5}\left(-\frac{1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3}\right) - \frac{2}{5}\left(-\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2}\right)$$
$$= 0.971 - \frac{3}{5}0.918 - \frac{2}{5}1.00 = 0.020$$

Construcción de árboles de decisión
Ejemplo

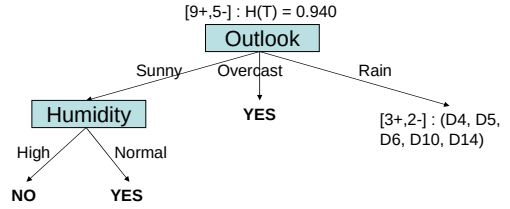
- Atributo *Humidity*
  - T_{high} = [0+,3-]
  - T_{normal}=[2+,0-]



$$G(Humidity) = H(T_{sunny}) - \sum_{i\in\{high,normal\}} \frac{|Ti|}{|T|} \times H(Ti) = H(T_{sunny}) - \frac{3}{5} H(T_{high}) - \frac{2}{5} H(T_{normal}) =$$
$$= 0.971 - \frac{3}{5}0.00 - \frac{2}{5}0.00 = 0.971$$

Construcción de árboles de decisión
Ejemplo

- Mejor atributo?
  - ***G*(*Humidity*) = 0.971**
  - *G*(*Wind*) = 0.020
  - *G*(*Temperature*) = 0.571



Construcción de árboles de decisión
Ejemplo

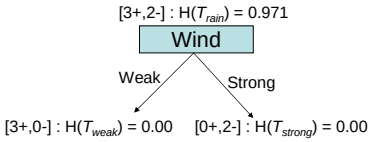
- Entropía T_{rain}

$$H(T_{rain}) = H([3+,2-]) = -\frac{3}{5}\log_2\left(\frac{3}{5}\right) - \frac{2}{5}\log_2\left(\frac{2}{5}\right) = 0.971\ bits$$

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D10	Rain	Mild	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Construcción de árboles de decisión
Ejemplo

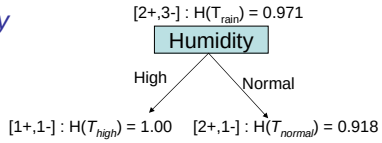
- Atributo *Wind*
  - T_{weak} = [3+,0-]
  - T_{strong}=[0+,2-]



$$G(Wind) = H(T_{rain}) - \sum_{i\in\{weak,strong\}} \frac{|Ti|}{|T|} \times H(Ti)$$
$$= H(T_{rain}) - \frac{3}{5}H(T_{weak}) - \frac{2}{5}H(T_{strong})$$
$$= 0.971 - \frac{3}{5}0.00 - \frac{2}{5}0.00 = 0.971$$

Construcción de árboles de decisión
Ejemplo

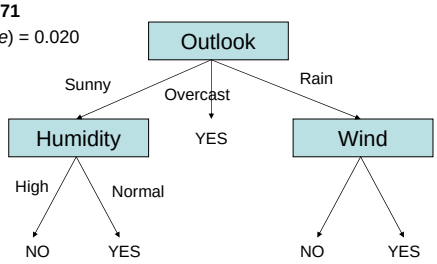
- Atributo *Humidity*
  - T_{high} = [1+,1-]
  - T_{normal}=[2+,1-]



$$G(Humidity) = H(T_{rain}) - \sum_{i\in\{high,normal\}} \frac{|Ti|}{|T|} \times H(Ti)$$
$$= H(T_{rain}) - \frac{3}{5}H(T_{high}) - \frac{2}{5}H(T_{normal})$$
$$= 0.971 - \frac{2}{5}1.00 - \frac{3}{5}0.918 = 0.020$$

Construcción de árboles de decisión
Ejemplo

- Mejor Atributo?
  - *G*(*Humidity*) = 0.020
  - ***G*(*Wind*) = 0.971**
  - *G*(*Temperature*) = 0.020



Construcción de árboles de decisión
Evaluación de test

- Criterio de razón de ganancia
  - El criterio de ganancia tiene un fuerte sesgo a favor de tests con muchos resultados. (Ej.: diagnóstico médico, atributo: ID del paciente, $H_X(T)$=0 y G(X) max, división inútil predictivamente)
  - Rectificación del sesgo mediante un tipo de normalización

$$Hd(X) = -\sum_{i=1}^{n} \frac{|T_i|}{|T|} \times \log_2\left(\frac{|T_i|}{|T|}\right)$$

  - Proporción de información generada por la división que es útil para clasificación

$$RG(X) = G(X) / Hd(X)$$

Construcción de árboles de decisión
Evaluación de test

- El criterio de razón de ganancia selecciona el test que maximice esta razón
- En el ej. de la división por ID de paciente, si hay k clases, G(X) es como mucho $\log_2(k)$, Hd(X) es $\log_2(n)$, donde n es el número de casos de entrenamiento. Es lógico suponer que n >> k, por tanto RG(X) es pequeña
- Ventaja: conduce a árboles pequeños
- Inconveniente: tiende a favorecer divisiones no balanceadas en las que un subconjunto T_i es mucho más pequeño que los otros

Construcción de árboles de decisión Ejemplo

- Criterio de razón de ganancia. Ejemplo

- Atributo *Outlook*
  - T_{sunny} = [2+,3-] = 5
  - $T_{overcast}$=[4+,0-] = 4
  - T_{rain}=[3+,2-] = 5

$$Hd(Outlook) = -\sum_{i\in\{sunny,overcast,rain\}} \frac{|T_i|}{|T|}\times\log_2\left(\frac{|T_i|}{|T|}\right)$$

$$= -\frac{5}{14}\log_2\frac{5}{14} - \frac{4}{14}\log_2\frac{4}{14} - \frac{5}{14}\log_2\frac{5}{14} = 1.557$$

$$RG(Outlook) = G(Outlook)/Hd(Outlook) = 0.246/1.577 = 0.156$$

31

Construcción de árboles de decisión

- Posibles tests considerados
  - C4.5: tres tipos de tests
    - Test estándar sobre un atributo discreto con un resultado y rama para cada posible valor de ese atributo
    - Test más complejo basado en un atributo discreto, en el que los valores posibles son asignados a un número variable de grupos con un resultado para cada grupo
    - Test binario sobre un atributo con valores numéricos continuos, con resultados A ≤ Z y A > Z, basado en comparar el valor de A contra un valor umbral Z
  - Estos tests se evalúan mediante RG (o G) producida a partir de la división de los casos de entrenamiento que originan
  - Restricción: para cualquier división, al menos dos de los subconjuntos T_i deben contener un número de casos razonable

32

Construcción de árboles de decisión

- Test sobre atributos continuos
  - Algoritmo para encontrar umbrales apropiados contra los que comparar los valores de atributos continuos
    - Los casos de entrenamiento T se ordenan primero sobre los valores del atributo A, $\{v_1, v_2, \ldots, v_m\}$
    - Cualquier valor umbral que caiga entre v_i y v_{i+1} dividirá los casos en aquellos cuyo valor del atributo A cae en $\{v_1, v_2, \ldots, v_i\}$ y aquellos cuyo valor está en $\{v_{i+1}, v_{i+2}, \ldots, v_m\}$
    - Se examinan las m-1 posibles divisiones sobre A
    - Es usual elegir el umbral como el punto medio de cada intervalo $(v_i + v_{i+1})/2$
    - C4.5 elige el valor mayor de A en el conjunto de entrenamiento que no excede el punto medio; así todos los valores umbrales que aparecen en los árboles están en los datos

33

Valores desconocidos de atributos

- Cuando los datos tienen valores de atributos perdidos:
  - Descartar una proporción de datos significativa y señalar como no clasificables algunos casos de test (inaceptable por la dificultad para encontrar datos)
  - Modificar los algoritmos para funcionar con valores de atributos perdidos

34

Valores desconocidos de atributos

- Para llevar a cabo la 2ª alternativa hay que plantearse tres cuestiones:
  - Selección de un test para dividir T. Si dos tests usan atributos con número diferente de valores desconocidos, ¿hay que tenerlo en cuenta?
  - Los casos de entrenamiento con valores desconocidos de atributos relevantes no pueden asociarse con un resultado particular del test, y por tanto no pueden asignarse a un conjunto particular T_i. ¿Se deben tratar esos casos al hacer la partición?
  - Cuando el árbol de decisión se usa para clasificar un caso no visto, ¿qué hacer si el caso tiene un valor desconocido para el atributo examinado en el nodo de decisión actual?

35

Valores desconocidos de atributos

- Adopción de algoritmos previos
  - Evaluación de tests
    - Sea T un conjunto de casos de entrenamiento y X un test sobre un atributo A, y sup. que conocemos el valor de A en una fracción F de los casos de T
    - Sean H(T) e H_X(T) calculadas sólo con los casos de T que tienen valores conocidos de A.
    - Podemos modificar la definición de G(X)

$$G(X) = F\times(H(T) - H_X(T))$$

    - Hd(X) se puede modificar tomando los casos con valores desconocidos como un grupo adicional. Si un test tiene n resultados, su información de división se calcula como si el test dividió los casos en n+1 subconjuntos
    - A partir de G(X) o RG(X) puede seleccionarse el test

36

Valores desconocidos de atributos

- Partición del conjunto de entrenamiento
  - C4.5 utiliza un enfoque probabilístico: supone que los resultados desconocidos del test se distribuyen probabilísticamente de acuerdo a la frecuencia relativa de resultados conocidos
    - Cuando un caso de T con resultado conocido O_i es asignado al subconjunto T_i, la probabilidad de que el caso pertenezca al subconjunto T_i es 1 y a todos los demás es 0
    - Cuando el resultado no se conoce, la declaración probabilística es más débil. Se asocia con cada caso en cada subconjunto T_i un peso que es la probabilidad de que el caso pertenezca a cada subconjunto
    - Cada subconjunto T_i es ahora una colección de posibles casos fraccionales, por lo que $|T_i|$ debería interpretarse como la suma de los pesos fraccionales de los casos en el conjunto

Valores desconocidos de atributos

- Los casos de entrenamiento de T podrían tener pesos no unitarios inicialmente, ya que T podría ser un subconjunto de una partición anterior
- Un caso de T con peso ω cuyo resultado no es conocido se asigna a cada subconjunto T_i con peso

$$\omega \times \text{probabilidad del resultado } O_i$$

$$\omega \times \frac{\text{casos de T con resultado conocido } O_i}{\text{casos de T con resultado conocido}}$$

Valores desconocidos de atributos

- Clasificación de un caso no visto
  - Si se encuentra un nodo de decisión en el que el valor del atributo relevante es desconocido, de forma que el valor del test no se puede determinar, el sistema explora todos los posibles resultados y combina las clasificaciones resultantes aritméticamente
  - Como puede haber múltiples caminos desde la raíz a las hojas, una clasificación es una distribución de clases, no una única clase
  - La clase con la probabilidad más alta se asigna como la clase predicha

Valores desconocidos de atributos
Ejemplo

- Ejemplo
  - Sup. que el valor del atributo *outlook* para el caso `outlook=overcast, temperature=mild, humidity=high, windy=strong` fuera desconocido en lugar de `overcast`
  - Nos fijamos en los 13 casos restantes

	Play	Don't play	Total
Outlook=sunny	2	3	5
overcast	3	0	3
rain	3	2	5
Total	8	5	13

$$H(T) = -\frac{8}{13}\log_2\left(\frac{8}{13}\right) - \frac{5}{13}\log_2\left(\frac{5}{13}\right) = 0.961\,bits$$

$$G(Outlook) = F\left(H(T) - \sum_{i=\{sunny,overcast,rain\}} \frac{|Ti|}{|T|} \times H(Ti)\right) = F\left(H(T) - \frac{5}{13}H(T_{sunny}) - \frac{3}{13}H(T_{overcast}) - \frac{5}{13}H(T_{rain})\right)$$

$$= \frac{13}{14}\left(0.961 - \frac{5}{13}\left(-\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5}\right) - \frac{3}{13}\left(-\frac{3}{3}\log_2\frac{3}{3} - \frac{0}{3}\log_2\frac{0}{3}\right) - \frac{5}{13}\left(-\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5}\right)\right)$$

$$= \frac{13}{14}(0.961 - 0.747) = 0.199 \text{ bits}$$

Valores desconocidos de atributos
Ejemplo

- Razón de ganancia cuando hay valores desconocidos

$$Hd(Outlook) = -\sum_{i=\{sunny,overcast,rain,?\}} \frac{|T_i|}{|T|} \times \log_2\left(\frac{|T_i|}{|T|}\right)$$

$$= -\frac{5}{14}\log_2\frac{5}{14} - \frac{3}{14}\log_2\frac{3}{14} - \frac{5}{14}\log_2\frac{5}{14} - \frac{1}{14}\log_2\frac{1}{14} = 1.809 \text{ bits}$$

$$RG(Outlook) = G(Outlook) / Hd(Outlook) = 0.199 / 1.809 = 0.110$$

Valores desconocidos de atributos
Ejemplo

- Cuando los 14 casos se dividen por este test (outlook), los 13 casos con valor conocido no presentan problema
- El caso restante se asigna a todos los bloques de la partición, sunny, overcast y rain, con pesos 5/13, 3/13 y 5/13 respectivamente

Outlook	Temperature	Humidity	Wind	PlayTennis	Weight
Sunny	Hot	High	Weak	No	1
Sunny	Hot	High	Strong	No	1
Sunny	Mild	High	Weak	No	1
Sunny	Cool	Normal	Weak	Yes	1
Sunny	Mild	Normal	Strong	Yes	1
?	Mild	High	Strong	Yes	5/13

- Si este subconjunto se divide más adelante por el mismo test sobre humidity, la distribución de clases en los subconjuntos son
  - Humidity=normal 2 Yes, 0 No
  - Humidity=high 5/13 Yes, 3 No

Valores desconocidos de atributos Ejemplo

- En cada hoja del árbol hay unos números de la forma (N) o (N/E)
- N es la suma de los casos fraccionales que alcanzan la hoja
- E es el número de casos que pertenecen a otra clase

Outlook = sunny:
  humidity = normal: Play (2.0)
  humidity = high: Don't play (3.4/0.4)
Outlook = overcast: Play (3.2)
Outlook = rain:
  windy = strong: Don't play (2.4/0.4)
  windy = false: Play (3.0)

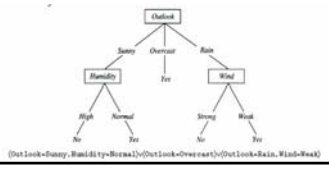
- Ahora suponemos que este árbol se usa para clasificar el siguiente caso:
  Outlook=sunny, temperature=mild, humidity=?, y windy=weak
  - Si humidity = normal el caso sería clasificado como *Play*
  - Si humidity = high el caso sería clasificado como *Don't play* con probabilidad 3/3.4 (88%) y Play 0.4/3.4 (12%)

  Al construir el árbol las particiones para estos resultados contuvieron 2.0 y 3.4 casos respectivamente:
  - *Play*: 2.0/5.4 x 100% + 3.4/5.4 x 12% = 44%
  - *Don't play*: 3.4/5.4 x 88% = 56%

43

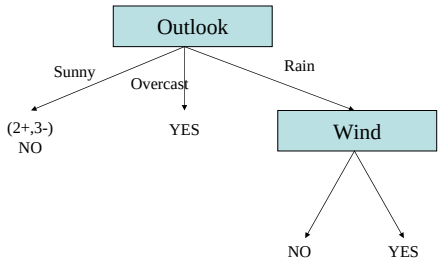
Poda de árboles de decisión

- Evitar sobreentrenamiento: simplificar
  - Consideremos lo siguiente:
    (Outlook = *Sunny* & Humidity = *Normal* & PlayTennis = *No*)
  - La predicción del árbol de decisión es incorrecta
    - (Outlook = *Sunny* & Humidity = *Normal*) ⇒ *Yes*
  - ¿Qué pasa si podamos el nodo "humidity"?
    - Cuando (outlook = *Sunny*), PlayTennis ⇒ *No*
    - Puede predecirse correctamente



44

Poda de árboles de decisión



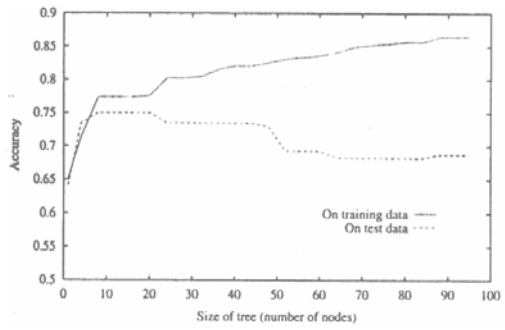
45

Poda de árboles de decisión

- Definición
  - Dado un espacio de hipótesis *H*, una hipótesis $h \in H$ se dice que hay **sobreentrenamiento** de los datos si existe alguna hipótesis alternativa $h' \in H$, tal que *h* tiene un error menor que *h*' sobre los ejemplos de entrenamientro, pero *h*' tiene un error menor que *h* sobre la distribución entera de ejemplos
- Navaja de Occam
  - Se prefiere la hipótesis más simple que ajuste los datos

46

Poda de árboles de decisión



47

Poda de árboles de decisión

- Soluciones
  - Dividir los ejemplos en tres conjuntos: entrenamiento, validación y evaluación
  - Usar todos los datos para entrenamiento, pero aplicar un test estadístico para estimar si expandir o podar un determinado nodo va a producir probablemente una mejora sobre el conjunto de entrenamiento (stopping, prepruning)
  - Eliminar retrospectivamente alguna estructura ya construida

48

Poda de árboles de decisión

- Stopping
  - Ventaja: ahorro de tiempo
  - Enfoque:
    - Buscar la mejor forma de dividir un subconjunto
    - Evaluar la división desde el punto de vista de significación estadística, ganancia de inf., reducción del error, etc.
    - Si es menor que un umbral, se rechaza la división. (Es difícil elegir un umbral apropiado)
  - Depende bastante del dominio
- C4.5 y CART siguen el último camino
  - Método más lento pero más confiable
    - Desarrollar el árbol mediante "Divide y vencerás"
    - Poda

Poda de árboles de decisión

- Normalmente peor clasificación de los casos de entrenamiento
- Las hojas del árbol podado tendrán una distribución de clases (prob. de pertenecer a cada clase)
- Esto puede alterar la determinación de la clase de un caso no visto

Poda de árboles de decisión

- Poda basada en el error
  - Sup. que fuera posible predecir la razón de error de un árbol y de sus subárboles
    - Comenzar desde abajo del árbol y examinar cada subárbol no hoja.
      - Si el reemplazamiento de este subárbol por una hoja o por su rama más frecuentemente usada, condujera a una razón de error predicha más baja, entonces podar
      - La razón de error predicha para todos los árboles que incluyen éste se ve afectada
      - La razón de error para el árbol completo disminuye a medida que la razón de error de todos sus subárboles se reduce. Este proceso conduce a un árbol con razón de error predicha mínima

Poda de árboles de decisión

  - La razón de error sobre el conjunto de casos de entrenamiento a partir de los que fue construido el árbol no proporciona una estimación adecuada ya que aumenta a medida que el árbol se poda
  - Técnicas para predecir la razón de error
    - Usar un conjunto de casos de test para predecir la razón de error:
      - **Poda coste-complejidad**: la razón de error predicha de un árbol es la suma ponderada de su complejidad y su error sobre los casos de entrenamiento, con los casos de test usados principalmente para determinar una ponderación adecuada
      - **Poda de error reducido**: evalúa la razón de error del árbol y sus componentes directamente sobre el conjunto de casos de test
    - Usar sólo el conjunto de entrenamiento (C4.5)

Poda de árboles de decisión

- Inconveniente de la primera familia de técnicas:
  - Menor número de casos de entrenamiento para construir el árbol (no es problema si hay datos abundantes)
    - Si no hay muchos datos, usar validación cruzada

Poda de árboles de decisión

- C4.5 usa un método de poda pesimista
  - Calcular para un nivel de confianza CF, el límite superior de la razón de error predicha, $U_{CF}(E,N)$
  - Para simplificar los cálculos, la estimación del error para las hojas y subárboles se computan suponiendo que se usa para clasificación un conjunto de casos no vistos del mismo tamaño que el conjunto de entrenamiento
  - Así una hoja que cubra N casos de entrenamiento con una razón de error predicha de $U_{CF}(E,N)$ tendría N x $U_{CF}(E,N)$ errores predichos
  - El número de errores predichos asociados con un (sub)árbol es la suma de los errores predichos de sus ramas

Poda de árboles de decisión
Ejemplo

```
physician fee freeze = n:
    adoption of the budget resolution = y: democrat (151)
    adoption of the budget resolution = u: democrat (1)
    adoption of the budget resolution = n:
        education spending = n: democrat (6)
        education spending = y: democrat (9)
        education spending = u: republican (1)
physician fee freeze = y:
    synfuels corporation cutback = n: republican (97/3)
    synfuels corporation cutback = u: republican (4)
    synfuels corporation cutback = y:
        duty free exports = y: democrat (2)
        duty free exports = u: republican (1)
        duty free exports = n:
            education spending = n: democrat (5/2)
            education spending = y: republican (13/2)
            education spending = u: democrat (1)
physician fee freeze = u:
    water project cost sharing = n: democrat (0)
    water project cost sharing = y: democrat (4)
    water project cost sharing = u:
        mx missile = n: republican (0)
        mx missile = y: republican (3/1)
        mx missile = u: republican (2)
```

Después de podar

```
physician fee freeze = n: democrat (168/2.6)
physician fee freeze = y: republican (123/13.9)
physician fee freeze = u:
    mx missile = n: republican (3/1.1)
    mx missile = y: republican (4/2.2)
    mx missile = u: republican (2/1)
```

(N/E)
N = número de casos de entrenamiento cubiertos por esa hoja
E = número de errores predichos si un conjunto de N casos no vistos fuera clasificado por el árbol

Poda de árboles de decisión
Ejemplo

- Tomamos el siguiente subárbol que no tiene errores sobre el conjunto de entrenamiento

```
education spending = n: democrat (6)
education spending = y: democrat (9)
education spending = u: republican (1)
```

- Tomando el nivel de confianza por defecto de C4.5 (25%),

$U_{25\%}(0,6) = 0.206$, número de errores predichos= 6 x 0.206
$U_{25\%}(0,9) = 0.143$, número de errores predichos= 9 x 0.143
$U_{25\%}(0,1) = 0.750$, número de errores predichos= 1 x 0.750
nº errores predichos por este subárbol=6 x 0.206 + 9 x 0.143 + 1 x 0.750=3.273

- Si el subárbol fuera reemplazado por la hoja **democrat**

Nº de errores predichos = 16 x $U_{25\%}(1,16)$ = 16 x 0.157 = 2.512

Extracción de reglas

- Los árboles grandes son difíciles de entender porque cada nodo tiene un contexto específico establecido por los resultados de los tests de los nodos antecesores
- La estructura del árbol puede hacer que subconceptos individuales sean fragmentados

```
F = 0:
    J = 0: no
    J = 1:
        K = 0: no
        K = 1: yes
F = 1:
    G = 1: yes
    G = 0:
        J = 0: no
        J = 1:
            K = 0: no
            K = 1: yes
```


Extracción de reglas

- Soluciones:
  - Definir nuevos atributos más específicos como F=G=1
  - Pasar a otro tipo de representación de clasificadores a partir del árbol de decisión
    - Transformar el camino a cada hoja en una regla de producción.
    - La colección resultante de reglas clasificaría los casos exactamente como lo hace el árbol
    - Las partes **si** de las reglas son mutuamente exclusivas y exhaustivas, por tanto el orden de las reglas no importa
    - Una regla cubre un caso si el caso satisface las condiciones del antecedente de la regla

```
Si F=1
   G=0
   J=1
   K=1
entonces clase yes
```


Extracción de reglas

- Generalizar reglas simples
  - Rescribir el árbol mediante una colección de reglas, una por cada, no resultaría más simple que el árbol
  - Los antecedentes de algunas reglas pueden contener condiciones irrelevantes
  - Ej.: en la regla anterior la conclusión no está afectada por los valores de F y G y las podemos eliminar

```
Si J=1
   K=1
entonces clase yes
```

  - Podemos decidir cuándo se debe eliminar una condición

Extracción de reglas

- Sea R de la forma Si A entonces clase C
- Y R^- una regla más general Si A^- entonces clase C donde A^- se obtiene eliminando una condición X de las condiciones de A
- La evidencia de la importancia de la condición X se encuentra en los casos de entrenamiento usados para construir el árbol de decisión
- Cada caso que satisface el antecedente A^- pertenece o no a la clase C y satisface o no la condición X

	Clase C	Otra clase
Satisface X	Y_1	E_1
No satisface X	Y_2	E_2

Extracción de reglas

- R cubre $Y_1 + E_1$ casos, E_1 mal clasificados
- R^- cubre $Y_1 + Y_2 + E_1 + E_2$ casos con $E_1 + E_2$ errores
- Para decidir si la condición X se elimina se usa una estimación pesimista de la precisión de las reglas R y R^- sobre casos no vistos, como en la poda

Poda
- Una hoja que cubre N casos de entrenamiento con E errores es muy poco probable que tenga una razón de error tan baja como E/N (estimación de resustitución) cuando clasifica casos no vistos
- Estimamos la razón de error verdadera de la hoja como el límite superior $U_{CF}(E,N)$ del intervalo de confianza para esta razón para un nivel de confianza CF

Regla
- La estimación de la razón de error de la regla R es $U_{CF}(E_1,Y_1+E_1)$ y la de R^- es $U_{CF}(E_1+E_2,Y_1+Y_2+E_1+E_2)$
- Si la razón de error pesimista de la regla R^- no es mayor que la de la regla R entonces tiene sentido eliminar la condición X

Extracción de reglas

- Puede que haya que eliminar más de una condición de una regla.
- En lugar de comprobar todos los posibles subconjuntos de condiciones que pueden eliminarse, C4.5 lleva a cabo una eliminación voraz directa:
  - Cuando una o más condiciones cumplan que la razón de error pesimista sea menor bajo la regla generalizada se eliminan
  - Como con las búsquedas voraces, no hay garantía de que minimizar la razón de error pesimista en cada paso conduzca a un mínimo global
  - El sistema podría mejorarse llevando a cabo una búsqueda exhaustiva cuando el número de condiciones es pequeño
  - La búsqueda voraz funciona razonablemente bien en la práctica y es relativamente rápida

Extracción de reglas. Ejemplo

if TSH > 6
FTI ≤ 64
TSH measured = t
T4U measured = t
thyroid surgery = t
then class negative

Cubre 3 de los casos de entrenamiento, 2 de ellos pertenecen a la clase negative, es decir, $Y_1=2$ y $E_1=1$
$U_{25\%}(1,3)=69\%$

- Si eliminamos cada una de las condiciones

Condición eliminada	$Y_1 + Y_2$	$E_1 + E_2$	Razón de error pesimista (%)
TSH > 6	3	1	55
FTI ≤ 64	6	1	34
TSH measured = t	2	1	69
T4U measured = t	2	1	69
thyroid surgery = t	3	59	97

Extracción de reglas. Ejemplo

- Eliminar cualquiera de las cuatro primeras condiciones de una regla cuya razón de error pesimista no es más alta que la de la regla original (69%)
- Eliminamos la segunda condición (34%)
- Se sigue el proceso para ver si se pueden eliminar otras condiciones sin aumentar la razón de error pesimista de la regla

Condición eliminada	$Y_1 + Y_2$	$E_1 + E_2$	Razón de error pesimista (%)
TSH > 6	31	1	8
TSH measured = t	6	1	34
T4U measured = t	7	1	30
thyroid surgery = t	44	179	82

Extracción de reglas. Ejemplo

- Se elimina la primera (8%) y el proceso continúa, la regla final es

If thyroid surgery = t
then class negative

- Esta regla mucho más simplificada cubre 35 de los casos de entrenamiento con solamente un error y tiene una razón de error pesimista de 7%

Extracción de reglas

- Conjuntos de reglas de clases
  - El proceso de generalización de reglas se repite para cada camino del árbol de decisión no simplificado. Normalmente se producen menos reglas que hojas
  - Problema: las reglas dejan de ser mutuamente excluyentes y exhaustivas. Habrá casos que satisfagan las condiciones de más de una regla, o de ninguna si las reglas imprecisas han sido eliminadas
  - Un intérprete de reglas de producción completo debe especificar cómo se van a clasificar estos casos
  - Los casos que no satisfagan ninguna regla se pueden clasificar definiendo una regla por defecto
  - Resolución de conflictos:
    - Se ordenan las reglas y la primera que cubre un caso se toma como la operativa
    - Se agrupan las reglas por clases y se ordenan estos subconjuntos. Así los conjuntos de reglas son más inteligibles y el orden para una clase particular no es importante
      - Se selecciona un subconjunto de las reglas para representar cada clase

Extracción de reglas

- Ordenación de clases y elección de una por defecto
  - El subconjunto de reglas seleccionadas para cada clase normalmente producirá FP
  - Retrasar las clases que produzcan muchos FP. Subconjuntos anteriores podrían cubrir correctamente algunos de estos casos
  - Se coloca la primera clase aquella cuyo subconjunto de reglas tenga menos FP. Y así sucesivamente
  - Clase por defecto: la que contiene el mayor número de casos de entrenamiento no cubiertos por ninguna regla

Extracción de reglas

- Proceso de refinamiento final
  - Si hay una o más reglas cuya eliminación reduciría el número de errores de clasificación sobre los casos de entrenamiento, la primera de ellas se descarta, y se examina el conjunto otra vez, y así sucesivamente
  - En el ejemplo anterior:

Clase	Reglas generalizadas	Reglas seleccionadas	Casos cubiertos	FP	FN
Negative	6	5	2319	2	3
Primary	3	2	66	2	3
Compensated	2	1	120	0	9

  - Las reglas para la clase 3 se sitúan las primeras, seguidas de las de la clase 2 y finalmente las de la 1
  - Hay 8 casos de la clase 3 que no son cubiertos por ninguna regla. Ésta es la clase por defecto

Extracción de reglas

if on thyroxine = f
  thyroid surgery = f
  TSH > 6
  TT4 ≤ 150
  FTI > 64
then class compensated hypothyroid [98.9%]

if thyroid surgery = f
  TSH > 6
  FTI ≤ 64
then class primary hypothyroid [95.6%]

if on thyroxine = f
  TT4 measured = f
  TSH > 6
then class primary hypothyroid [45.3%]

if TSH ≤ 6
then class negative [99.9%]

if on thyroxine = t
  FTI > 64
then class negative [99.5%]

if TSH measured = f
then class negative [99.5%]

if TT4 > 150
then class negative [99.4%]

if thyroid surgery = t
then class negative [92.7%]

if none of the above
then class compensated hypothyroid

- Entre corchetes aparece la precisión predicha de cada regla
- Estas reglas representan una teoría más amigable que el árbol original y son tan precisas como éste

Extracción de reglas. Resumen

- Cada camino desde la raíz a una hoja en un árbol no podado da lugar a una regla
- Cada regla se simplifica (generaliza) eliminando las condiciones no útiles para discriminar la clase en cuestión de otras usando una estimación pesimista de la precisión de la regla
- Para cada clase, todas las reglas simplificadas se examinan para eliminar reglas que no contribuyen a la precisión del conjunto de reglas total
- Se ordenan los conjuntos de reglas de forma que se minimicen los FP y se elige una clase por defecto
- Este proceso conduce a un clasificador de reglas de producción que es normalmente tan preciso como un árbol podado, pero más fácil de entender

Aplicación de ventanas

- Surgió a finales de los 70 por las restricciones de memoria de los computadores. Inicialmente:
  - Se selecciona aleatoriamente un subconjunto de entre los casos de entrenamiento al que llamamos “ventana”, y construimos un árbol de decisión a partir de él
  - Usamos este árbol para clasificar los casos de entrenamiento que no han sido incluidos en la ventana
  - Se añaden algunos de los casos mal clasificados a la ventana inicial, y se construye un segundo árbol que se usa para clasificar los casos de fuera de la ventana
  - Esto se hace hasta que el árbol construido a partir de los casos de la ventana clasifique correctamente los casos de fuera
  - Frecuentemente la ventana contiene sólo una fracción pequeña de los casos de entrenamiento, que son todos los “interesantes”

Aplicación de ventanas

- C4.5:
  - Elige los casos de la ventana de manera que la distribución de clases sea lo más uniforme posible
  - Incluye al menos la mitad de las excepciones (casos mal clasificados de fuera de la ventana) en cada ciclo (para converger rápidamente a un árbol final)
  - Para antes de que logre clasificar correctamente los casos de fuera de la ventana si la secuencia de árboles no alcanza mayor precisión
  - En dominios con ruido o indeterminación, terminar pronto previene el que la ventana crezca hasta que incluya casi el total de casos de entrenamiento

Aplicación de ventanas. Ejemplo

- Ejemplo del hypothyroid

Ciclo	Tamaño del árbol	Objetos		Errores					
		Ventana	Resto	Ventana	Razón (%)	Resto	Razón (%)	Total	Razón (%)
1	15	502	2012	5	1.0	15	0.7	20	0.8
2	19	517	1997	8	1.5	29	1.5	37	1.5
3	21	546	1968	8	1.5	0	0.0	8	0.3

– Los 546 casos (22% del total) aparentemente contienen la misma información que los 2514

73

Aplicación de ventanas

- Si ya no hay problemas de memoria, ¿por qué seguir aplicando ventanas?
  - Construcción más rápida de árboles (a veces)
    - Ej. de la biblioteca de datos de la Univ. de California, Irvine, tiene una colección de 8124 descripciones de setas, clasificadas como venenosas o comestibles. Hay un árbol de 31 nodos que las clasifica correctamente. Mediante aplicación de ventanas se llega al árbol final en el primer ciclo. La velocidad mejora un 15%
    - En otros dominios con conjuntos de entrenamiento mayores, la aplicación de ventanas llega a reducir el tiempo requerido para construir un árbol de decisión por un factor de tres
    - Para la mayor parte de los dominios de clasificación del mundo real el uso de ventanas ralentiza el proceso de construcción de árboles. Cuando el árbol final tiene una razón de error significativa, el uso de ventanas requiere generalmente muchos ciclos para converger

74

Aplicación de ventanas

- Árboles más precisos
  - Merece la pena el uso de ventanas en dominios donde las clases están bastante equilibradas
  - Diferentes ventanas iniciales conducen generalmente a árboles diferentes iniciales, con diferentes excepciones, etc.
  - Esto permite obtener diferentes árboles finales sin cambiar el conjunto de casos de entrenamiento. Dos enfoques:
    - Desarrollar varios árboles alternativos y seleccionar el que tenga la menor razón de error predicha
    - Desarrollar varios árboles, generar reglas de producción a partir de todos ellos, y construir un clasificador de reglas de producción a partir de todas las reglas disponibles
  - Normalmente el clasificador obtenido de esta forma es más preciso, pero requiere más computación

75

Aplicación de ventanas. Ejemplo

- Multiplexor
  - Un caso viene descrito por una serie de bits, los *a* primeros constituyen una dirección (desde 0 a 2^a-1), seguido de un bit de datos para cada dirección
  - Hay dos clases, si o no, y un caso es de la clase si, si el bit de datos correspondiente a la dirección es 1
  - Si a=3, hay tres bits de dirección $a_0a_1a_2$ seguido por 8 bits de datos $d_0d_1d_2d_3d_4d_5d_6d_7$, así el caso 01001101001 pertenece a la clase si ya que 010=2 y d_2 es 1
  - Se generaron aleatoriamente (con reemplazamiento) 5 conjuntos de entrenamiento de tamaños 100, 200, ..., 500, y un conjunto de 1000 casos de test.

76

Aplicación de ventanas. Ejemplo

- Cada conjunto de entrenamiento se procesó usando C4.5 de tres formas:
  - Modo estándar: se generó un único árbol a partir de todos los casos de entrenamiento
  - Usando ventanas: se generó un único árbol
  - Usando ventanas: se desarrollaron 10 árboles y (automáticamente) se eligió uno de ellos
- Se generó un conjunto de reglas en cada caso

77

Aplicación de ventanas. Ejemplo

- Razón de error (árboles)

Casos de entr.	Sin ventanas (%)	Con ventanas, un único árbol (%)	Con ventanas, 10 árboles (%)
100	35.4	36.0	34.4
200	24.4	24.6	16.9
300	18.5	13.9	11.6
400	17.9	9.4	5.7
500	13.2	8.0	6.3

- Razón de error (reglas)

Casos de entr.	Sin ventanas (%)	Con ventanas, un único árbol (%)	Con ventanas, 10 árboles (%)
100	36.4	36.0	26.0
200	18.3	18.7	9.2
300	7.4	4.3	1.7
400	3.5	0.0	0.6
500	0.5	0.0	0.0

78

Aplicación de ventanas

- No todos los dominios experimentan esta mejora con el uso de ventanas
- Rara vez el uso de ventanas da un árbol final menos preciso, a veces lo da mejor pero normalmente lleva más tiempo
- Construir árboles múltiples siempre lleva más tiempo, pero ocasionalmente puede merecer la pena el esfuerzo, en particular si los árboles se van a reprocesar para dar un clasificador basado en reglas de producción

79