Tema 5
Aprendizaje Automático

5.1 Aprendizaje supervisado
- Vecinos más próximos (kNNs)
- Árboles de decisión

Clasificación con k vecinos más próximos

Nombre: **kNN**

Entrada: Datos de entrenamiento : $\{(\mathbf{x}_n, c_n); n = 1,2,\ldots,N\}$

Ejemplo de test : $\mathbf{x}_{test}$

No. de vecinos : k

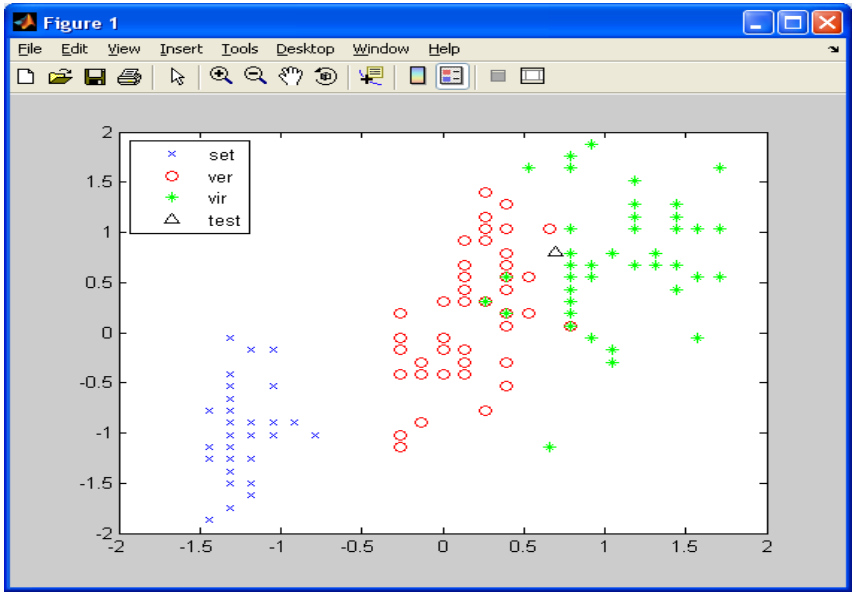
Métrica para calcular distancias : $d(\mathbf{x}_i, \mathbf{x}_j)$

Salida: Clase predicha por k-NN para el ejemplo de test $\mathbf{x}_{test}$

Código:

1. Encontrar los k vecinos más próximos a $\mathbf{x}_{test}$ en el conjunto de entrenamiento
2. Encontrar la clase mayoritaria entre esos k vecinos más próximos a $\mathbf{x}_{test}$
3. Asignar a $\mathbf{x}_{test}$ esta clase mayoritaria

Ejemplo: Datos en 2 dimensiones, k = 1



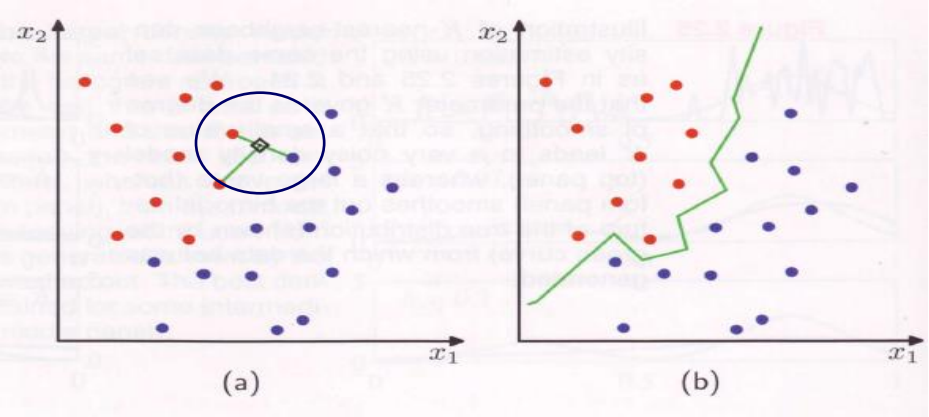
Nota: Los empates se pueden resolver de diferentes formas.
Por ejemplo, elegir la clase mayoritaria entre todos los datos de entrenamiento.
Si persiste el empate, asignar una clase al azar, de acuerdo a las frecuencias de las clases en el conjunto de entrenamiento.

k-NN y el teorema de Bayes

Consideremos la esfera más pequeña centrada en el ejemplo de test $\mathbf{x}_{test}$ que contiene exactamente k puntos del conjunto de entrenamiento.

k = 3

[Fig. 2.27, Bishop]



$$\mathrm{R}(\mathbf{x}_{test}) = \text{Elemento de volumen centrado en } \mathbf{x}_{test}$$
$$N\left(R(\mathbf{x}_{test})\right) = \text{Ejemplos contenidos en } \mathrm{R}(\mathbf{x}_{test})$$
$$N_c\left(R(\mathbf{x}_{test})\right) = \text{Ejemplos de clase } c \text{ contenidos en } \mathrm{R}(\mathbf{x}_{test})$$

$$p(c \mid \mathbf{x}_{test}) \approx \frac{N_c\left(R(\mathbf{x}_{test})\right)}{N\left(R(\mathbf{x}_{test})\right)}$$

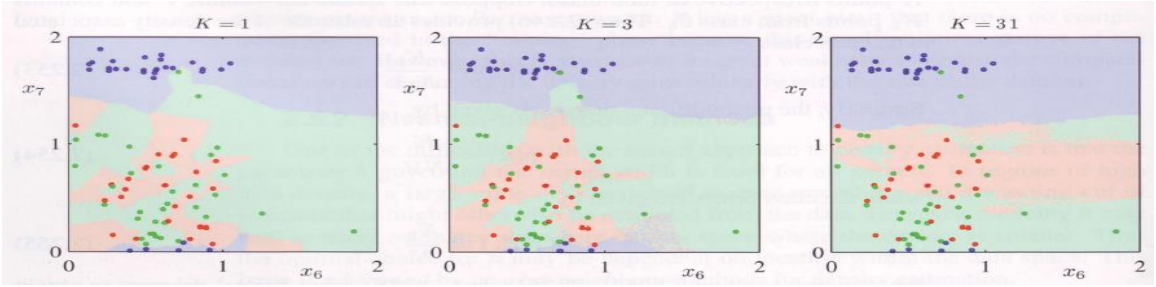
$$\text{MAP}: \qquad c^*(\mathbf{x}_{test}) = \max_c p(c \mid \mathbf{x}_{test}) \approx \max_c N_c\left(R(\mathbf{x}_{test})\right)$$

Por tanto: k-NN puede ser visto como un predictor MAP que utiliza estimaciones locales de las densidades de probabilidad relevantes.

¿Cuántos vecinos?

El número de vecinos k actúa como un "smoothing parameter"
k pequeño: muchas regiones y más pequeñas
k grande: menos regiones y más grandes

[Fig 2.28, Bishop]



- **Clasificador por vecino más próximo** (k=1) Si $N \to \infty$ $error_{NN} < 2\,error_{Bayes}$
- **k-NN, k =3** Si $N \to \infty$ $error_{3-NN} \approx error_{Bayes} + 3\left(error_{Bayes}\right)^2$
- **El k óptimo puede ser determinado con leave-one-out cross validation**

Entrada: Datos de entrenamiento : $\{(\mathbf{x}_n, c_n); n = 1,2,\mathrm{K}\ ,N\}$

Métrica para calcular distancias : $\mathrm{d}(\mathbf{x}_i, \mathbf{x}_j)$

Salida: k* (número óptimo de vecinos)

Código: Calcular $\mathrm{error}_{\mathrm{CV}}(\mathrm{n},\mathrm{k}) = 1 - \delta\left(c_n, \mathrm{kNN}\left(\mathbf{x}_n; \{(\mathbf{x}_m, c_m)\}_{\substack{m=1 \\ m\neq n}}^{N}, \mathrm{k}, d(\cdot,\cdot)\right)\right),$

$\mathrm{k} = 1,2,\ldots,\mathrm{N}-1, \qquad \mathrm{n} = 1,2,\ldots,\mathrm{N}$

$$\mathrm{k}^* := \min_{\mathrm{k}}\left\{\sum_{\mathrm{n}=1}^{\mathrm{N}} \mathrm{error}_{\mathrm{CV}}(\mathrm{n},\mathrm{k})\right\}$$

Dificultades

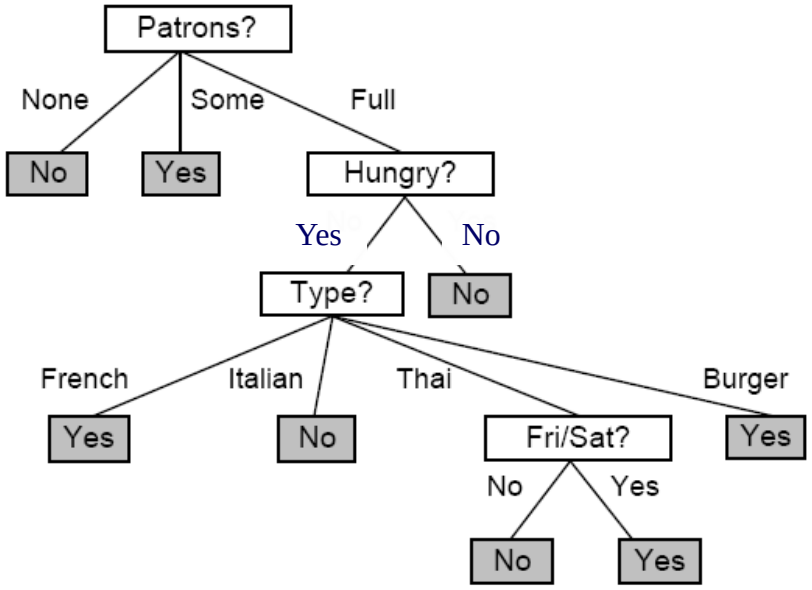
- Presencia de atributos irrelevantes.
- Selección de la función de distancia.
- ¿Distancias considerando atributos cualitativos?
- Alto coste computacional

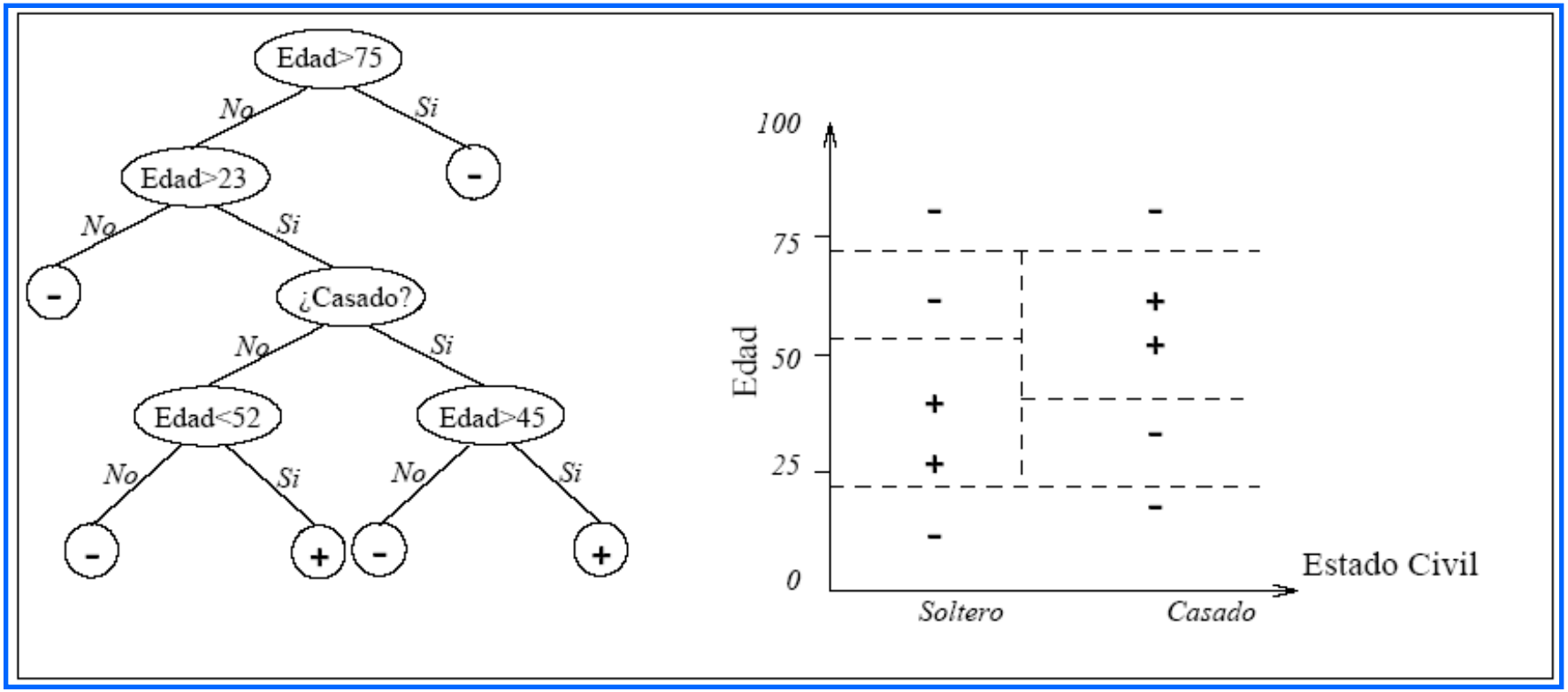
Clasificación con árboles de decisión

Datos para aprendizaje:

Example	Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	*Yes*
X_2	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	*No*
X_3	*No*	*Yes*	*No*	*No*	*Some*	*$*	*No*	*No*	*Burger*	*0–10*	*Yes*
X_4	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*10–30*	*Yes*
X_5	*Yes*	*No*	*Yes*	*No*	*Full*	*$$$*	*No*	*Yes*	*French*	*>60*	*No*
X_6	*No*	*Yes*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Italian*	*0–10*	*Yes*
X_7	*No*	*Yes*	*No*	*No*	*None*	*$*	*Yes*	*No*	*Burger*	*0–10*	*No*
X_8	*No*	*No*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Thai*	*0–10*	*Yes*
X_9	*No*	*Yes*	*Yes*	*No*	*Full*	*$*	*Yes*	*No*	*Burger*	*>60*	*No*
X_{10}	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$$$*	*No*	*Yes*	*Italian*	*10–30*	*No*
X_{11}	*No*	*No*	*No*	*No*	*None*	*$*	*No*	*No*	*Thai*	*0–10*	*No*
X_{12}	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Burger*	*30–60*	*Yes*

Árbol aprendido:





El árbol es computacionalmente atractivo: eficiente, compacto.
Además, a partir de él se pueden sacar reglas (una rama ➔ una regla):

IF (¬ (*Edad* > 75) ^ (*Edad* > 23) ^ (*Casado* = *Si*) ^ (*Edad* > 45))
THEN ((*Clase* = +))

PSEUDOCÓDIGO DEL ALGORITMO PARA GENERAR ÁRBOLES DE DECISIÓN

```
function DECISION-TREE-LEARNING(examples, attributes, default) returns a decision tree
  inputs: examples, set of examples
          attributes, set of attributes
          default, default value for the goal predicate

  if examples is empty then return default
  else if all examples have the same classification then return the classification
  else if attributes is empty then return MAJORITY-VALUE(examples)
  else
      best ← CHOOSE-ATTRIBUTE(attributes, examples)
      tree ← a new decision tree with root test best
      for each value v_i of best do
          examples_i ← {elements of examples with best = v_i}
          subtree ← DECISION-TREE-LEARNING(examples_i, attributes − best,
                                           MAJORITY-VALUE(examples))
          add a branch to tree with label v_i and subtree subtree
      end
      return tree
```

ELECCIÓN DEL MEJOR ATRIBUTO PARA PONER EN LA RAÍZ: CHOOSE-ATTRIBUTE

- Buscamos la *primera pregunta* que mejor determine la clasificación lo mejor posible
- La mejor pregunta es la que permite obtener más información sobre la clase
- ¿Podemos cuantificar la ganancia de información? **Sí.**

ENTROPÍA BINARIA: Definición *H* (p, q)

- Concepto base de la Teoría de la Información (Shannon, 1948)

$$H(p,q) = -\,p \times \log_2 p \;-\; q \times \log_2 q$$

Notas: p + q = 1.

En el caso en que o p o q sean 1, el valor de H se define como 0.

- La entropía es no negativa, como se puede observar con la expresión equivalente: $H(p,q) = p \cdot \log_2 \frac{1}{p} + q \cdot \log_2 \frac{1}{q} \geq 0$

- El resultado no es nunca negativo (tiene sentido: no hay información negativa)
- La entropía mide el grado de desconocimiento que tenemos del valor de una variable aleatoria.
- H es mínima si p o q son 1 (H=0 bits), y máxima si p=q=1/2 (H=1 bits)

ENTROPÍA BINARIA

- Consideremos una variable aleatoria *g* binaria con dos valores g1 y g2 cuyas probabilidades p y $q = 1-p$
- ¿Qué/cuánta información tenemos sobre el próximo valor que tomará *g*?

Ejemplo: una moneda no trucada; cada tirada es independiente de la anterior, y p(cara) = p(cruz) = 1/2: C, X, C, X, X, X, C, C, X, C, X, C, X, X, C, C, ...

En este caso, nuestro desconocimiento sobre el resultado del lanzamiento de la moneda es $H(p, q) = H(1/2, 1/2) = 1$ bits, que es el valor máximo de la entropía binaria.

Supongamos que ahora la moneda está trucada. p(cara) = 1/3, p(cruz) = 2/3

$H(1/3, 2/3) = 0.92$ **bits**

En este caso tenemos a priori menos desconocimiento (más información) sobre el resultado de la próxima tirada.

- Máxima información (= **mínimo desconocimiento, entropía**) si la moneda está completamente trucada: p(cara) = 1, p(cruz) = 0, ya que H(1,0) = 0 bits.

GANANCIA DE INFORMACIÓN

La ganancia de información aportada por la variable aleatoria X sobre la variable G se define como:

$$GI(X,G) = H(G) - H(G|X)$$

Donde:

- H(G) mide el desconocimiento que tenemos de la variable G antes de saber X
- H(G|X) mide el desconocimiento sobre G después de saber el valor de X

La entropía condicional H(G|X) se define como:

$$H(G|X) = \sum_i p(X = x_i) \times H(G|X = x_i)$$

Por su parte,

$$H(G|X = x_i) = H\left(p(G = g_1|X = x_i), p(G = g_2|X = x_i)\right)$$

Si X es estadísticamente independiente de G, H(G|X)=H(G) y por tanto GI(X,G)=0 bits

Para un mismo G, GI(X,G) es máxima si H(G|X) = 0 bits

GANANCIA DE INFORMACIÓN: EJEMPLO

En los datos del restaurante: $H(\textit{Esperará/NoEsperará}) = H(\frac{6}{12}, \frac{6}{12}) = 1$ bit

Supongamos que sabemos que el restaurante está *vacío*,

$$H(\textit{Esperará/NoEsperará}|\textit{vacio}) = H(\tfrac{0}{2}, \tfrac{2}{2}) = 0 \text{ bits}$$

Luego hemos ganado información tras la observación de que el restaurante está vacío

Si el restaurante tiene valor *algunos* en el atributo *Patrons?*,

$$H(\textit{Esperará/NoEsperará}|\textit{algunos}) = H(\tfrac{4}{4}, \tfrac{0}{4}) = 0 \text{ bits}$$

Y si está *lleno*,

$$H(\textit{Esperará/NoEsperará}|\textit{lleno}) = H(\tfrac{2}{6}, \tfrac{4}{6}) = 0.92 \text{ bits}$$

En general, si preguntamos por el valor de *Patrons?* la entropía restante será un promedio ponderado de esas tres:

$$\frac{2}{12} \cdot 0 + \frac{4}{12} \cdot 0 + \frac{6}{12} \cdot 0.92 = 0.46 \text{ bits}$$

Preguntando "*Patrons?*" **ganamos en promedio la información**

1 – 0.46 bits = 0.54 bits

USO DE LA GANANCIA DE INFORMACIÓN EN CHOOSE-ATTRIBUTE

- El atributo *tipo* de restaurante aporta GI = 0 bits (verificarlo)
- Luego no ganamos nada de información sobre la clase preguntando por el tipo
- Ningún otro atributo puede ser peor que él (¡ni siquiera un atributo aleatorio?)
- Esta es la función heurística que guiará al algoritmo: se calcula la **ganancia de información** de todos los atributos, y se elige el de mayor valor para colocarlo definitivamente como nodo raíz
- Una vez realizada la elección, se particiona la tabla de datos por la pregunta elegida y se resuelve el mismo problema desde cero para cada rama con su nueva tabla

SIGAMOS UN PASO EL PROCESO RECURSIVO

Por la rama *lleno* han bajado 6 de los 12 ejemplos inciales:

X_4, X_{12}, (positivos) y X_2, X_5, X_9, X_{10} (negativos)

La entropía actual es $H(\textit{Esperará/NoEsperará}) = H(\frac{2}{6}, \frac{4}{6}) = 0.92$ bits

Calculando ganancia de información de todos los atributos, el mejor es: *Hungry?*

$H(\textit{Esperará/NoEsperará}|\textit{Hungry?}) = \frac{4}{6}.H(\frac{2}{4}, \frac{2}{4}) + \frac{2}{6}.H(\frac{0}{2}, \frac{2}{2}) = 0.67$ bits

Luego, su ganancia de información es 0.25 bits (verificar que es la mejor)

El **proceso recursivo termina** con una de estas tres condiciones:

- ▷ entropía cero (todos los ejemplos de la misma clase): hoja con clase
- ▷ rama sin ejemplos: hoja con valor por defecto
- ▷ no hay más posibles atributos por los que preguntar (pero ¿es posible!)

ASPECTOS GENERALIZABLES DE ESTOS ARBOLES DE DECISION

Un aspecto sencillo y útil de generalizar es tener una **clasificación no binaria**: p.ej, Alto, Medio, Bajo.

Sólo tenemos que generalizar la noción de entropía binaria.

Entropía generalizada para variable v con n posibles valores,
cuyas probabilidades son $p(v_1), (v_2), \ldots, (v_n)$

$$H(v) = -\sum_{i=1}^{n} p(v_i).\log_2 p(v_i)$$

Todos los términos de probabilidad nula se ignoran

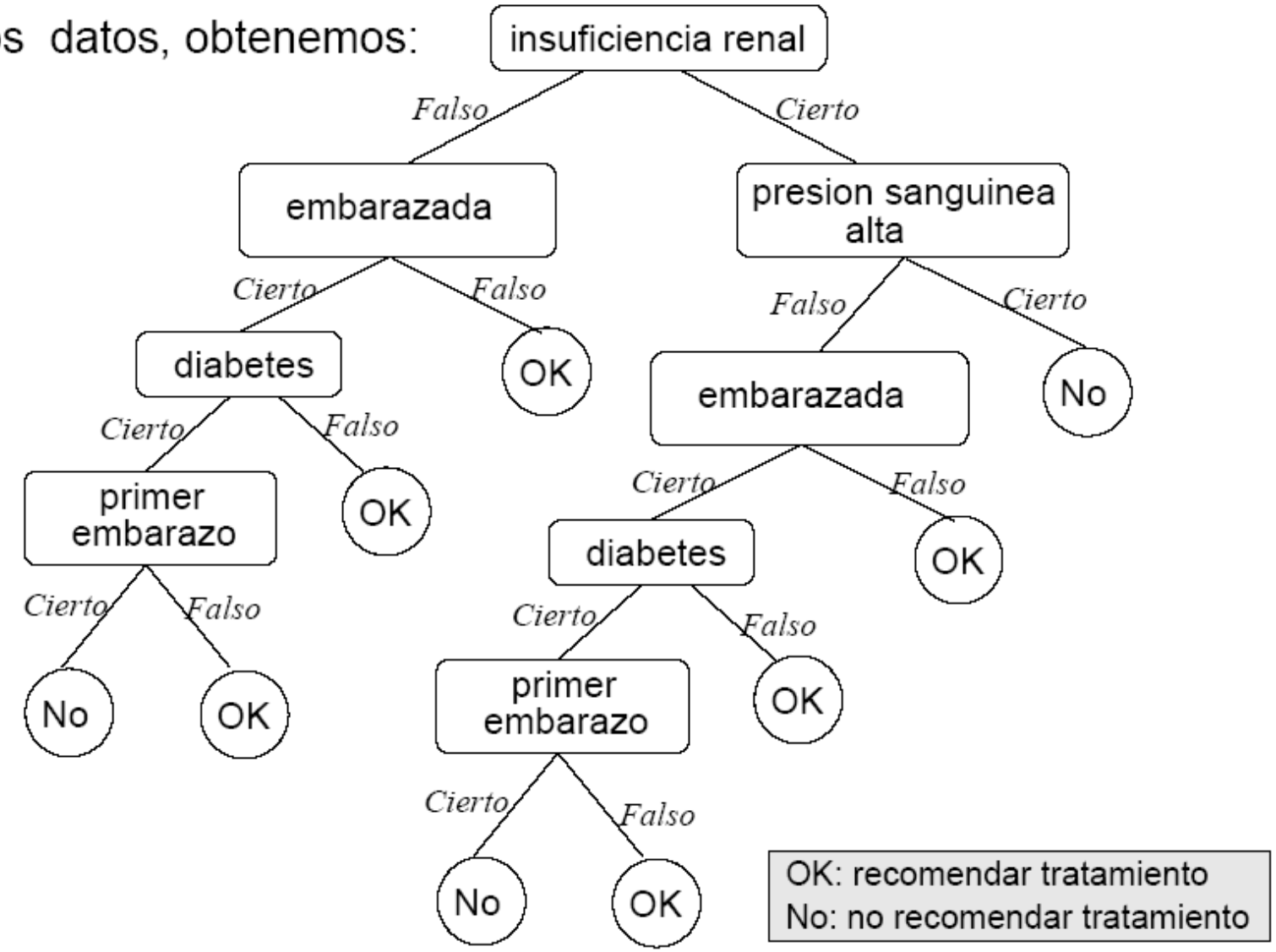
La unidades siguen siendo los **bits** (porque seguimos usando $\log_2$)

$H(\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4},) = 2$ bits

$H(0, \frac{1}{2}, \frac{1}{4}, \frac{1}{4},) = 1.5$ bits

Extracción y simplificación de reglas

Con otros datos, obtenemos:



- Nos centramos en la clase con menos nodos hoja, y extraemos las 3 reglas para el No (la otra clase queda definida por negación por fallo):

  (insuficiencia renal ^ presión alta) $\Rightarrow$ *No*

  (ins. renal ^ ¬ pres. alta ^ embarazada ^ diabetes ^ 1er embarazo) $\Rightarrow$ *No*

  (¬ ins. renal ^ embarazada ^ diabetes ^ 1er embarazo) $\Rightarrow$ *No*

- Eliminamos antecedentes o incluso reglas enteras si el acierto en los datos que nos dan se mantiene.

- En la base de datos que se ha usado para construir el árbol, las reglas siguientes tienen el mismo acierto que las originales:

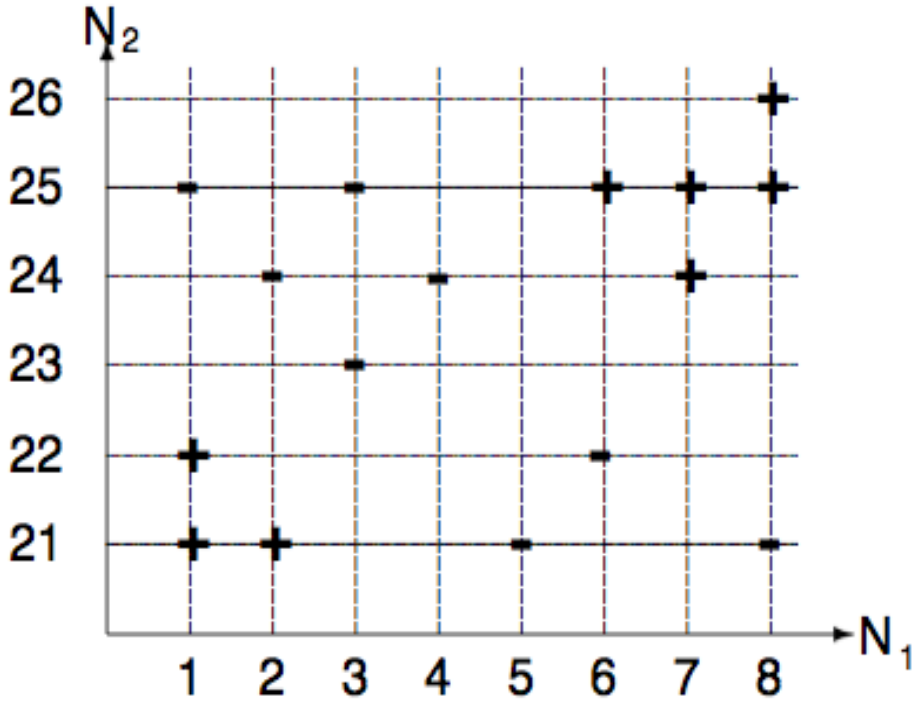
  (insuficiencia renal ^ presión alta) $\Rightarrow$ *No*
  (diabetes ^ primer embarazo) $\Rightarrow$ *No*

Árboles de decisión: recapitulación y pasos siguientes

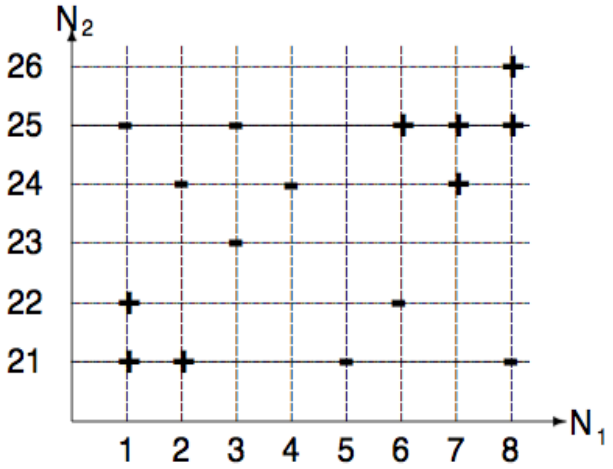
- El algoritmo estudiado es una versión simplificada de ID3 (Quinlan, 1986)
- ID3 incluye un tipo de (pre-)poda para limitar el tamaño del árbol.
- Después de diversas ampliaciones y mejoras se llega a C4.5 (Quinlan, 1992)
- Permite el uso de atributos numéricos, incluso con un decisiones "fuzzy".
- Normaliza la Ganancia de Información de los atributos de muchos valores
  Permite el uso de valores desconocidos en los datos
- Añade la obtención de reglas, muy mejoradas heurísticamente
- En 1998 surgen las herramientas comerciales (Rulequest, Quinlan)

C4.5

- Incluye la gestión de atributos numéricos



C4.5

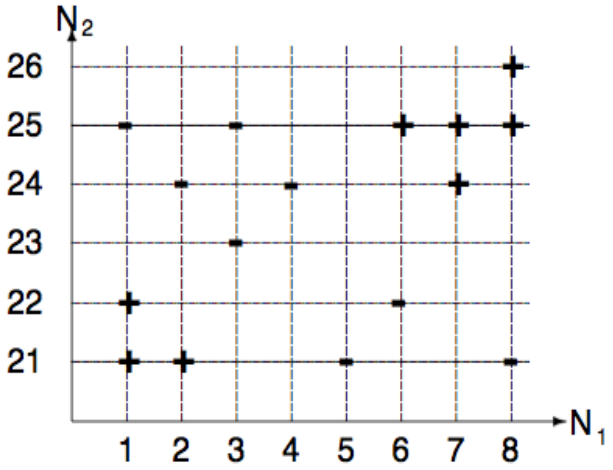


La entropía de la clase es, ya que hay 8+ y 8-,
H(8/16, 8/16) = 1 bits.

Las posibles preguntas con N1 son:

Pregunta	Rama "No"	Rama "Sí"	Entropía clase en Rama "No"	Entropía clase en Rama "Sí"	H(clase | Pre-gunta)	IG
N_1>1	2+, 1-	6+, 7-	H(2/3, 1/3) = 0.918 bits	H(6/13, 7/13) = 0.996 bits	3/16*0.918+ 13/16*0.996 = 0.981 bits	1-0.981= 0.019 bits
N_1>2	3+, 2-	5+, 6-	H(3/5, 2/5) = 0.971 bits	H(5/11, 6/11) = 0.994 bits	5/16*0.971+ 11/16*0.994 = 0.987 bits	1-0.987= 0.013 bits
N_1>3	3+, 4-	5+, 4-	H(3/7, 4/7) = 0.985 bits	H(5/9, 4/9) = 0.991 bits	7/16*0.985+ 9/16*0.991 = 0.988 bits	1-0.988= 0.012 bits
N_1>4	3+, 5-	5+, 3-	H(3/8, 5/8) = 0.954 bits	H(5/8, 3/8) = 0.954 bits	8/16*0.954+ 8/16*0.954 = 0.954 bits	1-0.954= 0.046 bits
N_1>5	3+, 6-	5+, 2-	H(3/9, 6/9) = 0.918 bits	H(5/7, 2/7) = 0.863 bits	9/16*0.918+ 7/16*0.863 = 0.894 bits	1-0.894= 0.106 bits
N_1>6	4+, 7-	4+, 1-	H(4/11, 7/11)= 0.946 bits	H(4/5, 1/5) = 0.722 bits	11/16*0.946+ 5/16*0.722 = 0.876 bits	1-0.876= 0.124 bits
N_1>7	6+, 7-	2+, 1-	H(6/13, 7/13)= 0.996 bits	H(2/3, 1/3) = 0.918 bits	13/16*0.996+ 3/16*0.918 = 0.981 bits	1-0.981= 0.019 bits
N_1>8	8+, 8-	0+, 0-	1 bit	--	16/16*1+ 0/16*-- = 0 bits	1-1 = 0 bits

C4.5



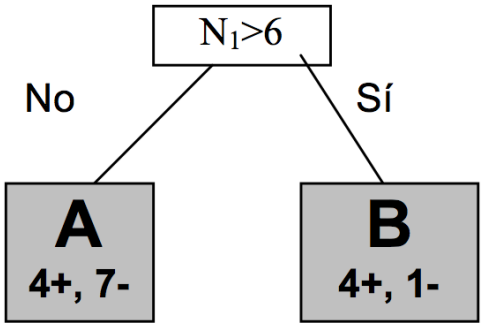
Por otra parte, las posibles preguntas con N2 son:

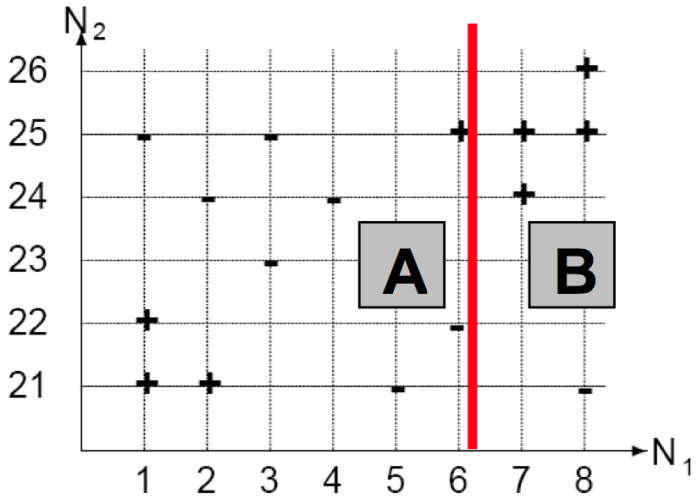
Pregunta	Rama "No"	Rama "Sí"	Entropía clase en Rama "No"	Entropía clase en Rama "Sí"	H(clase | Pre-gunta)	IG
N_2>21	2+, 2-	6+, 6-	H(2/4, 2/4) = 1 bits	H(6/12, 6/12) = 1 bits	4/16*1+ 12/16*1 = 1 bits	1-1= 0 bits
N_2>22	3+, 3-	5+, 5-	H(3/6, 3/6) = 1 bits	H(5/10, 5/10) = 1 bits	6/16*1+ 10/16*1 = 1 bits	1-1= 0 bits
N_2>23	3+, 4-	5+, 4-	H(3/7, 4/7) = 0.985 bits	H(5/9, 4/9) = 0.991 bits	7/16*0.985+ 9/16*0.991 = 0.988 bits	1-0.988= 0.012 bits
N_2>24	4+, 6-	4+, 2-	H(4/10, 6/10)= 0.971 bits	H(4/6, 2/6) = 0.918 bits	10/16*0.971+ 6/16*0.918 = 0.951 bits	1-0.951= 0.049 bits
N_2>25	7+, 8-	1+, 0-	H(7/15, 8/15)= 0.997 bits	H(1/1, 0/1) = 0 bits	15/16*0.997+ 1/16*0 = 0.935 bits	1-0.894= 0.065 bits

C4.5

Como vemos, la mejor pregunta que podemos hacer con **N2 es N2>25.**

Sin embargo, el IG de N2 > 25 es menor que el de la mejor pregunta sobre N1 (N1>6) así que esta es la pregunta que escogeremos para el nodo raíz.



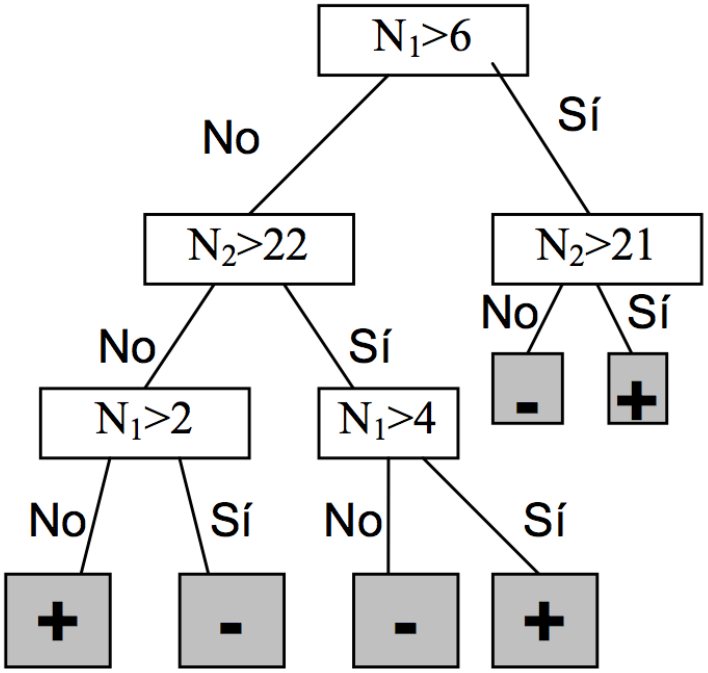


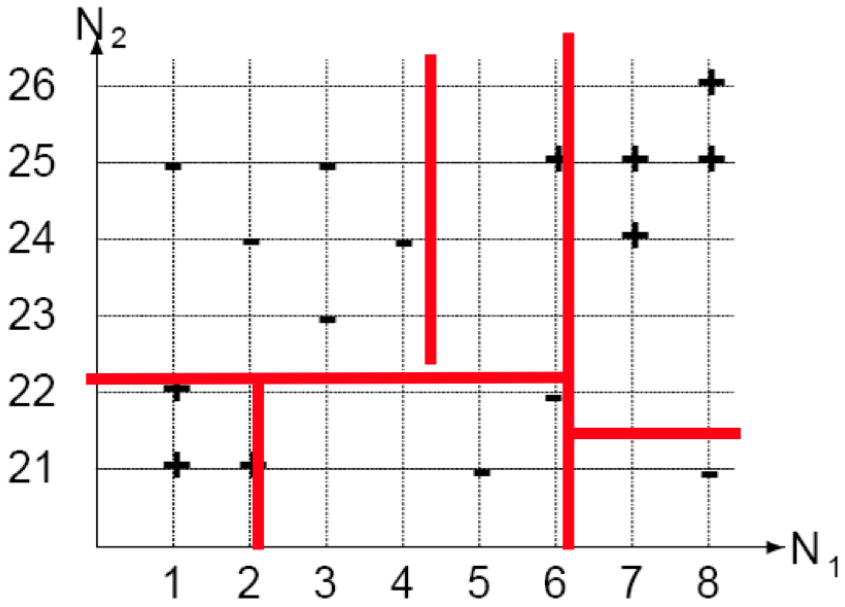
Hemos partido el espacio original en A y en B.

Ahora podemos aplicar recursivamente el mismo procedimiento en A y B por separado: estrategia **"divide y vencerás"**

C4.5

Al final de todo el proceso recursivo, llegamos a:





Generalización y complejidad de aprendizaje en árboles de decisión

- ¿Cómo de complejo es aprender? $O(\log n)$, Polinómico, NP-Hard, ... ¿Imposible?
- Es un problema de búsqueda (búsqueda en un espacio de árboles, o espacio de funciones, o de reglas, o de redes, o simplemente de hipótesis)
- Complejidad por el tamaño y estructura del espacio de búsqueda
- ¿Cuántos árboles de decisión puede haber en el problema del restaurante?
- Desconocemos la "solución correcta"
- Sólo conocemos parte de ella, los datos de entrenamiento
- ¿Podemos "inventar" el resto? Ese es el objetivo de la búsqueda
- Veamos ambos aspectos con cierto detalle

Tamaño del espacio de búsqueda

- En el espacio hay árboles intermedios (sin terminar) y finales (los que contaremos).
- Varios a árboles finales implementan la misma función de clasificación.
- Contemos entonces funciones **F** que implementen un mapeo

  Espacio de casos ➔ {*Yes*, *No*}

- Hay $2^4 \times 3^2 \times 2^2 \times 4^2 = 9216$ posibles casos, y 2 posibles clases (*Yes*, *No*), luego hay 2^{9216} posibles funciones como **F** y, por tanto, muchos más árboles.

- ¡Eso es una cantidad con más de 2770 cifras! ¿No habrá algo mal?
- Como tenemos 12 casos (de entrenamiento) preclasificados, podríamos descartar todas las **F** que no sean coherentes con esos datos (que no los clasifiquen bien).
- Entonces, hay 2^{9204} funciones **F** compatibles con los 12 casos observados.
- Aun así, ¡más de 2770 cifras!
- Y ¿con 6000 casos? 2^{3216} **F** s, ¡969 cifras! ...

Tamaño del espacio de búsqueda: Preferencia o "Bias"

- En general, para aprender una función binaria sobre atributos, el espacio de búsqueda contiene 2^(2^n) funciones (o hipótesis) ... demasiadas incluso restringiéndonos a las que son coherentes con los datos de entrenamiento

- Hay que imponer una **preferencia** (en inglés, **bias**) sobre las hipótesis/funciones

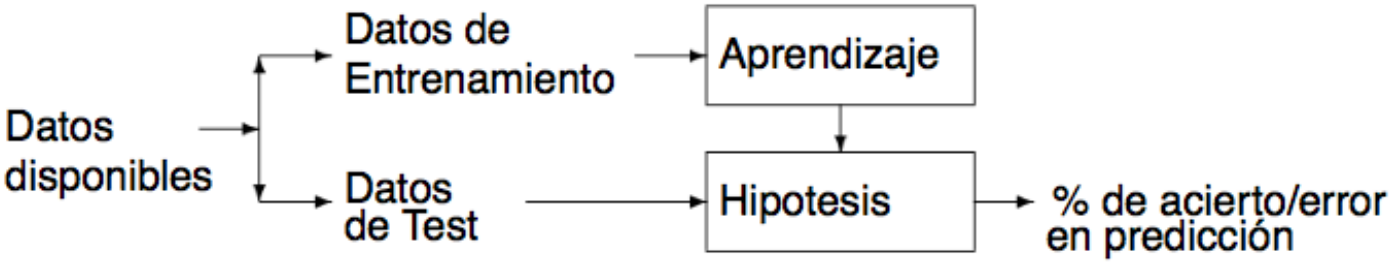
- Hay **dos tipos de bias** (y cada uno implementable de muchas formas):

  - preferencia algorítmica: ordena heurísticamente las hipótesis, para recorrer el espacio llegando antes a las que *se espera que sean mejores*
  (p.ej.: en ID3, atributo con más ganancia de información a la raíz).
  - preferencia de representación: modelos simples, lineales, nodos con un solo atributo, nodos con combinaciones de varios atributos, …

- El bias no es malo. Es imprescindible. ¿Cuál es mejor, en cada caso?

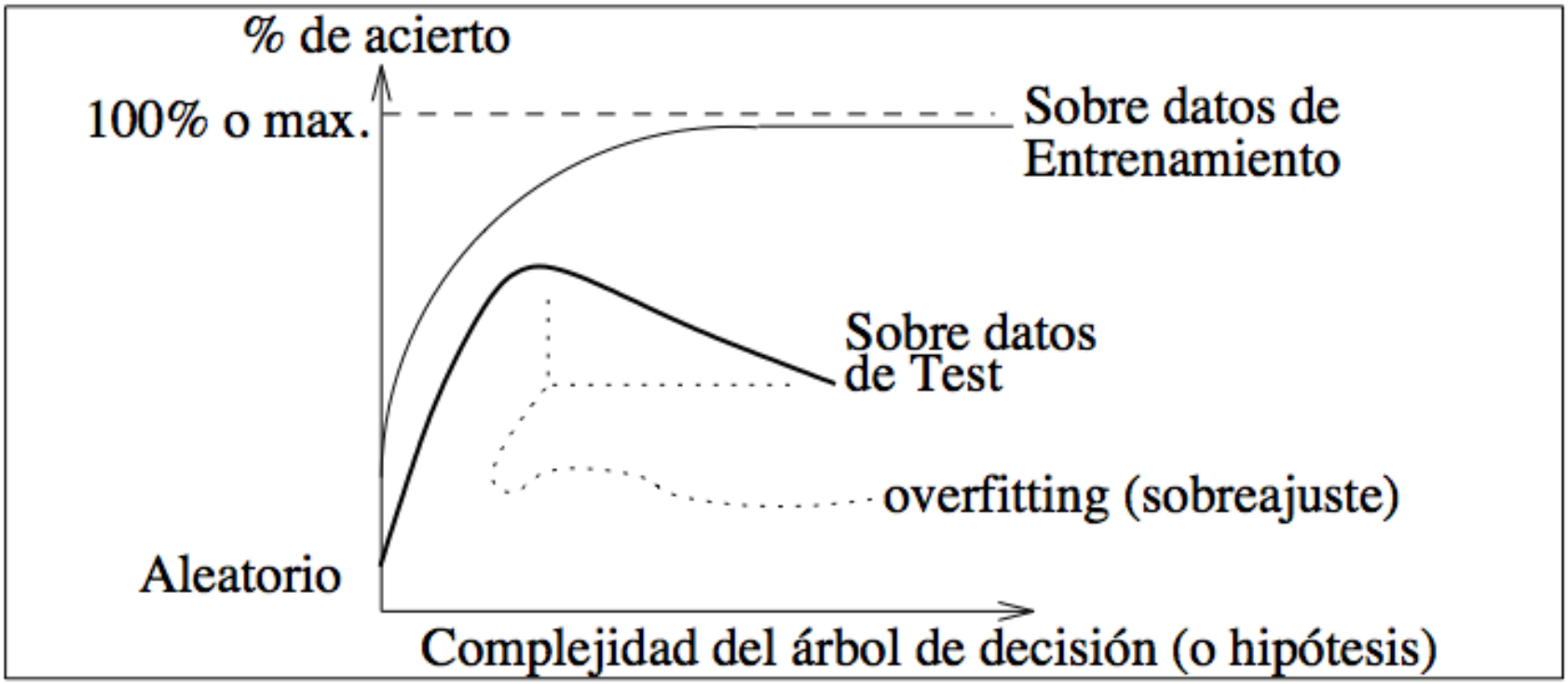
Desconocemos la solución correcta

- Ya hemos hablado de complejidad por el tamaño del espacio de búsqueda
- El problema también es complejo porque se desconoce la solución correcta
- Sólo conocemos parte de ella, los datos de entrenamiento, y tenemos que "inventar" el resto ...
- Hemos de suponer que el bias guía la búsqueda hacia el objetivo ... pero al final, ¿cómo podemos evaluar la calidad del árbol generado?
- Debemos usar **datos independientes** de los de entrenamiento para evaluar la hipótesis**:**



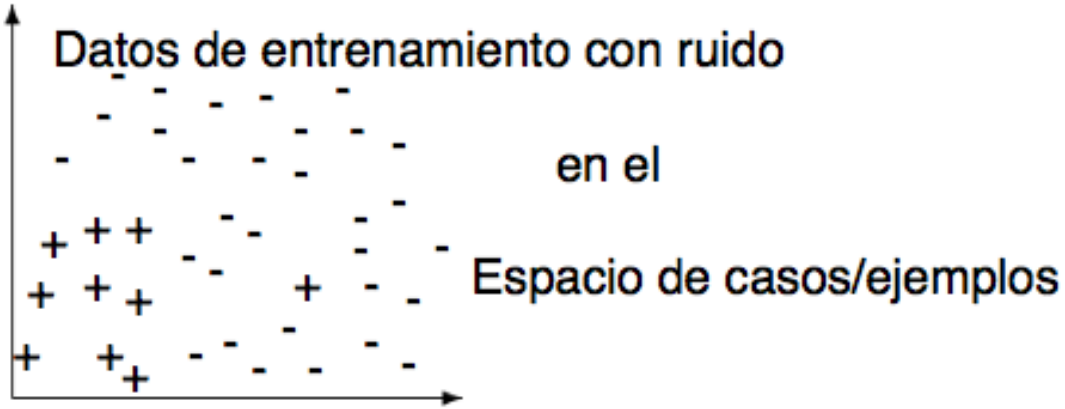
Complejidad de la hipótesis y sobreajuste (overfitting)

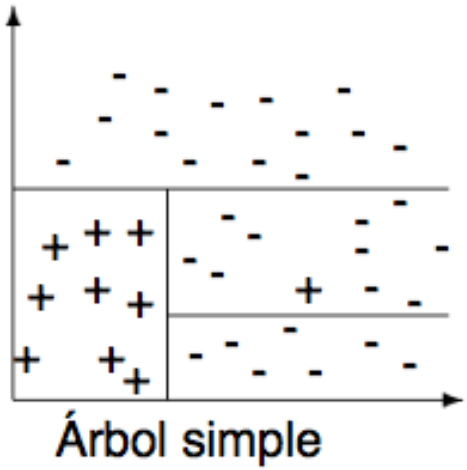
Comportamiento típico:

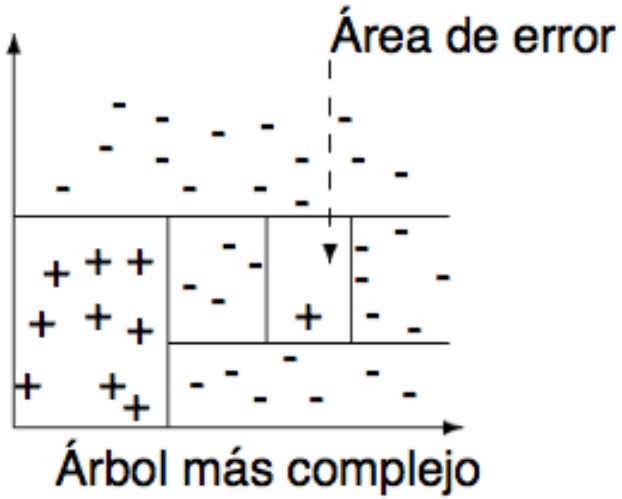


No es siempre bueno crecer el árbol: puede haber errores en los datos (ruido), puede estar creciendo ya sin suficiente base en los datos (muestras pequeñas) ...

Ilustración de sobreajuste por ruido







Control del sobreajuste en árboles

- ID3 prefiere (bias) árboles pequeños, aunque fallen en algún caso de entrenamiento

- Una forma obvia de reducir el tamaño del árbol es evitar que crezca
  IF (null examples) THEN Hoja ➔ IF (POCOS examples) THEN Hoja

- ID3 hace eso y además:
  - Estima (estadísticamente, chi-cuadrado) la ganancia de información de un atributo aleatorio
  - La **ganancia de información del atributo que se elige** como mejor debe ser la de mayor de todas las obtenidas con los atributos presentes en los datos, y **además debe ser superior a la ganancia estimada para el atributo aleatorio**
  - En caso contrario, ID3 para el crecimiento recursivo y forma una hoja.
  - Con muestras pequeñas es más probable que el "mejor" no mejore al aleatorio.

Poda de árboles en C4.5

- Decimos que ID3 hace pre-poda (*pre-pruning*): ni siquiera deja las ramas crecer.
- **C4.5 deja crecer el árbol al máximo** (pero, sin llegar a muestras muy pequeñas) **y después aplica poda** (*pruning*) a sus ramas heurísticamente:
  - Si convirtiendo todo un subárbol en una hoja no se empeora *significativamente* el% de aciertos en predicción, entonces se hace la poda, y se continúa hasta que no se pueda podar más.
  - C4.5 estima (estadística) el % de aciertos, en vez de usar los datos para ello.
- Podar (igual que crecer) el árbol, es búsqueda en el espacio de árboles, pero usando una heurística diferente
- Volvemos hacia atrás, hacia árboles más pequeños, pero no por el mismo camino

Extracción y poda de reglas en C4.5

- La extracción de reglas recorriendo caminos hasta las hojas es un paso obvio
- A partir de ahí, C4.5 generaliza las reglas eliminando condiciones una a una
- Elimina primero la condición que mejora más la calidad global del conjunto de reglas
- Medida de calidad basada en el principio MDL (*Minimum Description Length):*
  - Para cada condición *C*, sea *R´* el conjunto de reglas obtenido eliminando *C* de *R*.
  - Eliminar la que **minimiza la suma**

longitud de la codificación o óptima de *R´*

+

longitud de la codificación óptima de los casos clasificados erróneamente por *R´*

- Resultado: un % de acierto en predicción generalmente muy superior al del árbol sin podar