Aprendizaje Automático y Data Mining

Bloque III

MÉTODOS DE APRENDIZAJE INDUCTIVO

Índice

- Clasificación de métodos:
  - Lazy
  - Eager
- Árboles de decisión.
- Listas de reglas.
- Aprendizaje Bayesiano.
- Redes neuronales

CLASIFICACIÓN DE MÉTODOS

Clasificación de métodos (I)

- Los métodos de aprendizaje inductivo se clasifican en dos grupos:
  - Lazy: no construyen un modelo; todo el 'trabajo' se pospone hasta el momento de clasificar una nueva instancia (todo el procesamiento se hace *on-line*).
    Ejemplo: vecino más cercano.
  - Eager: construyen un modelo; parte del 'trabajo' se realiza off-line, nada más recopilar todos los ejemplos de entrenamiento.
    Ejemplos: árboles de decisión, redes neuronales, etc.

Clasificación de métodos (II)

- Algoritmos que estudiaremos:

LAZY	EAGER
Vecino más cercano	Árboles de decisión
	Listas de reglas
	Métodos bayesianos
	Redes neuronales

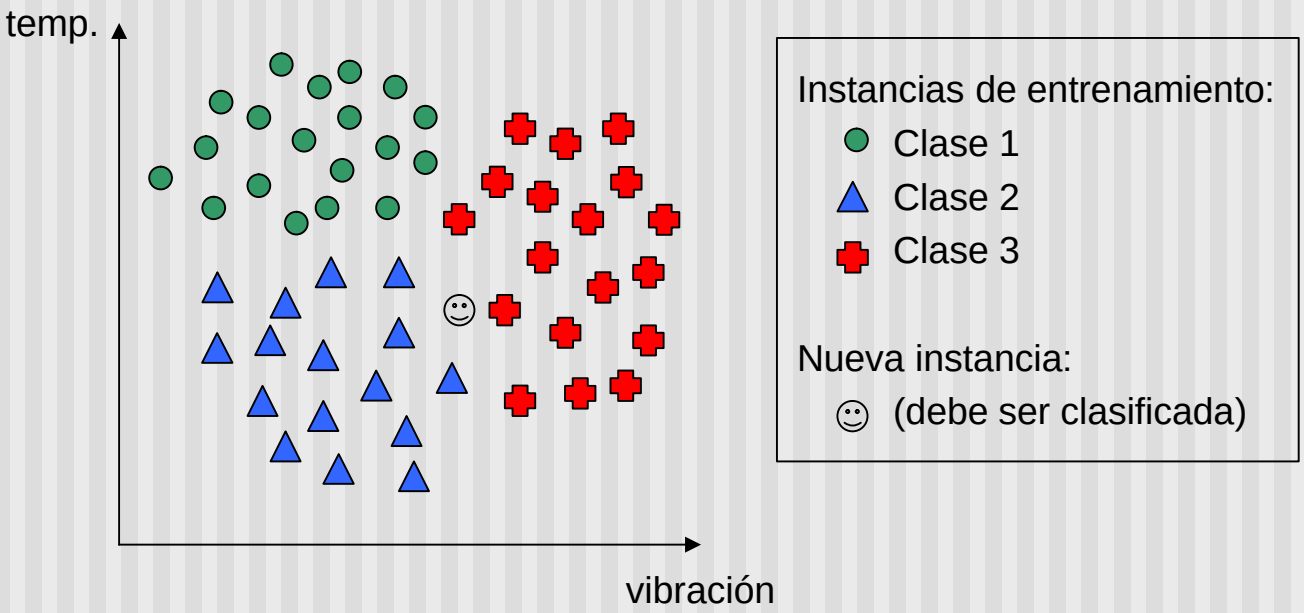
VECINO MÁS CERCANO

Vecino más cercano (I)

- Método lazy: no se crea modelo.
  - Entrenamiento (off-line): simplemente se guardan todas las instancias.
  - Clasificación (on-line): una nueva instancia se clasifica en función de la clase de las instancias almacenadas más cercanas.
    - La distancia entre dos instancias (entre dos ejemplos) se calcula a partir del valor de sus atributos.

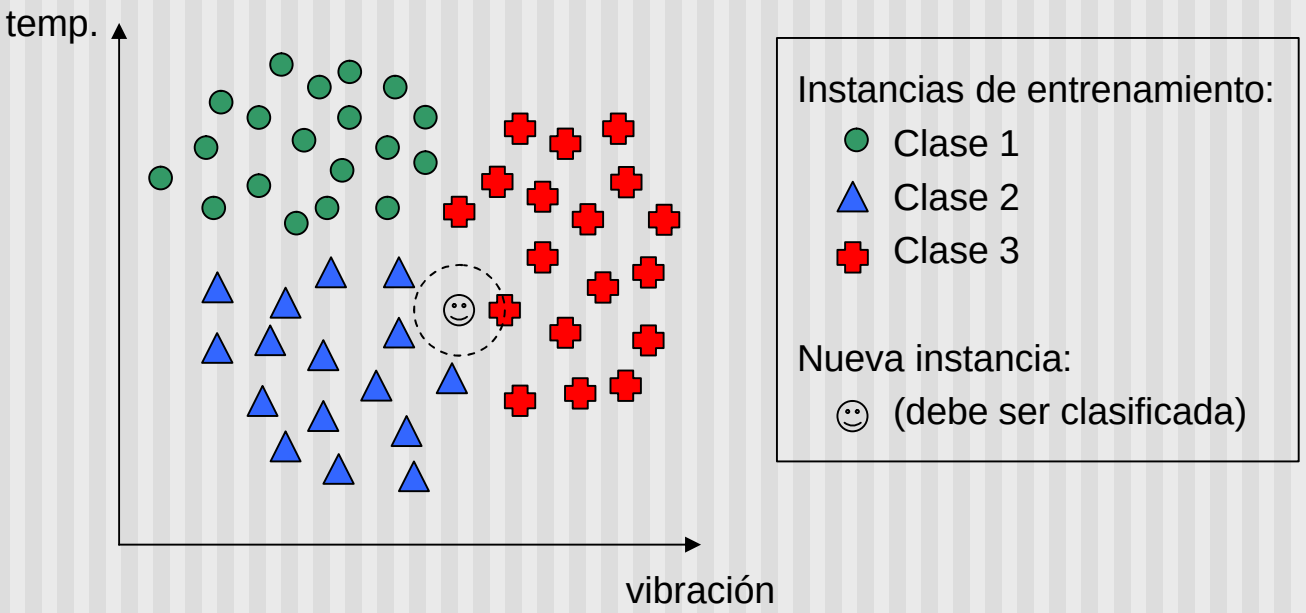
Vecino más cercano (II)

- Método básico: sólo un vecino (1-NN).



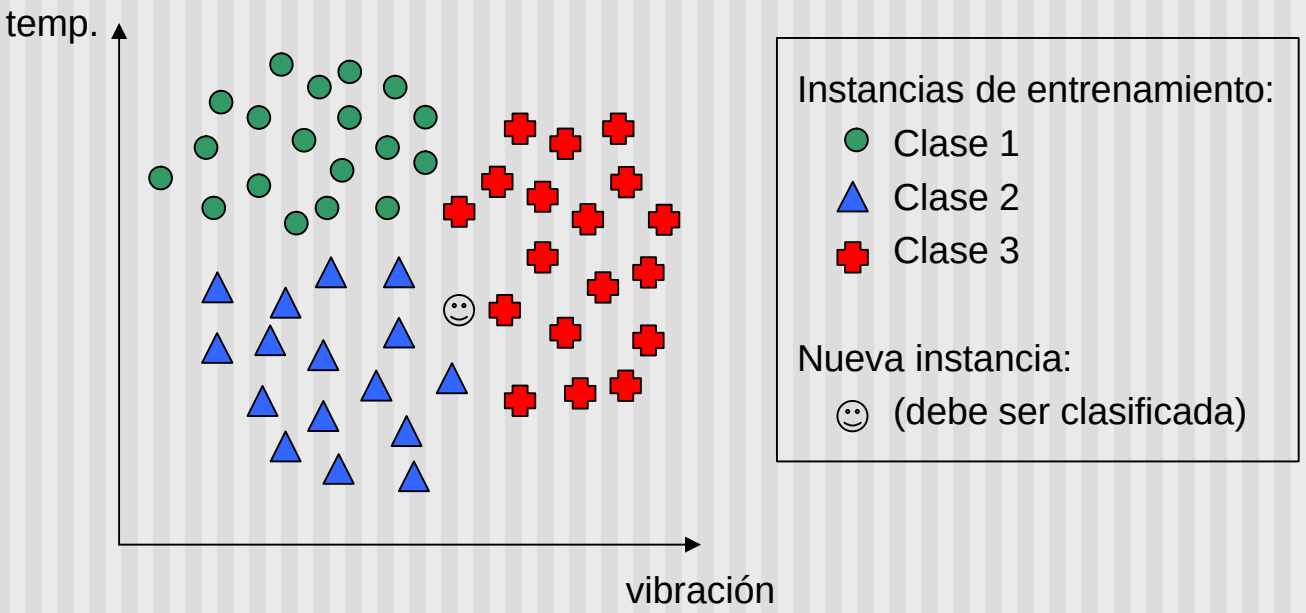
Vecino más cercano (III)

- Class assigned = class of closest element.



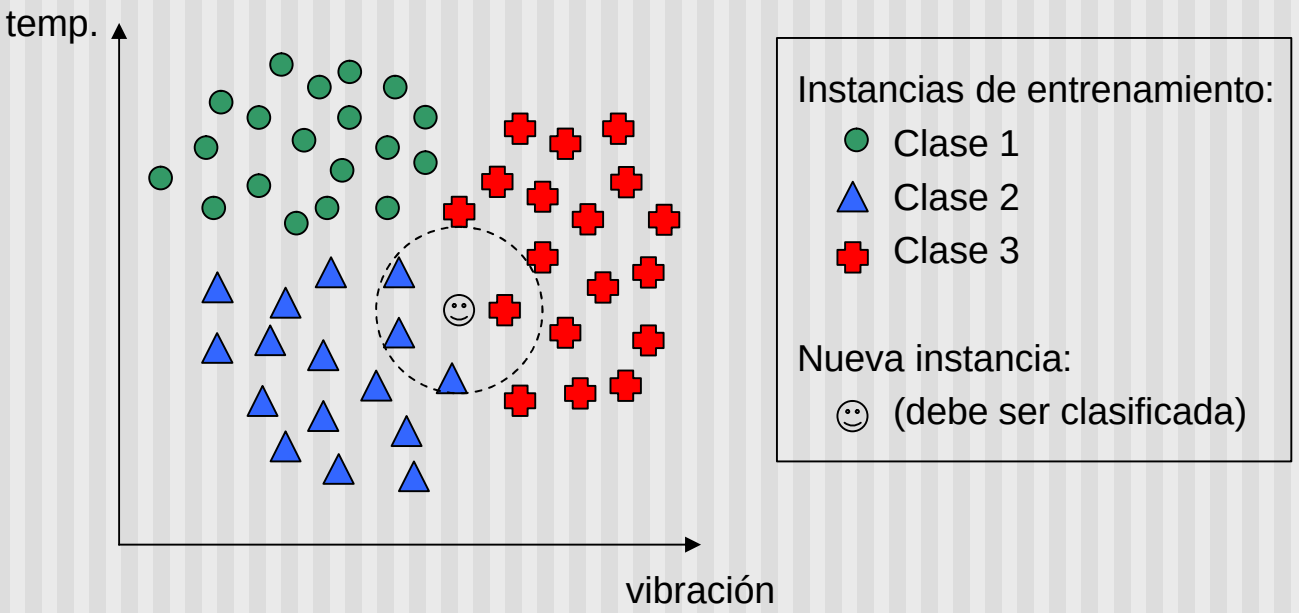
Vecino más cercano (IV)

- Más de un vecino (k-NN o k vecinos).



Vecino más cercano (V)

- Clase elegida = clase mayoritaria en las k instancias más cercanas (ejemplo con k = 4).



Vecino más cercano (VI)

- Procedimiento para la clasificación de una nueva instancia:
  1. Se mide la distancia entre la instancia a clasificar y todas las instancias de entrenamiento almacenadas.
     - Las distancias se miden en el espacio de los atributos.
     - Tantas dimensiones como número de atributos.
     - Distancia euclídea. Ejemplo con *n* atributos:

$$d(x_1, x_2) = \sqrt{(x_{11} - x_{21})^2 + (x_{12} - x_{22})^2 + \ldots + (x_{1n} - x_{2n})^2}$$

  2. Se eligen las k instancias más próximas.
  3. Se asigna como clase la clase mayoritaria entre las k instancias.
- Gran coste computacional on-line.

Vecino más cercano (VII)

- Es necesaria la normalización de atributos.
- Ejemplo: préstamo bancario.
  - Datos de entrenamiento:

Nº ejemplo	Salario	Edad	Devuelve préstamo
1	100.000	55	SI
2	90.000	30	NO
3	15.000	60	SI
4	20.000	25	NO

Aparentemente, sólo importa la edad.

Buscamos la clase de una nueva instancia (nuevo cliente del banco):

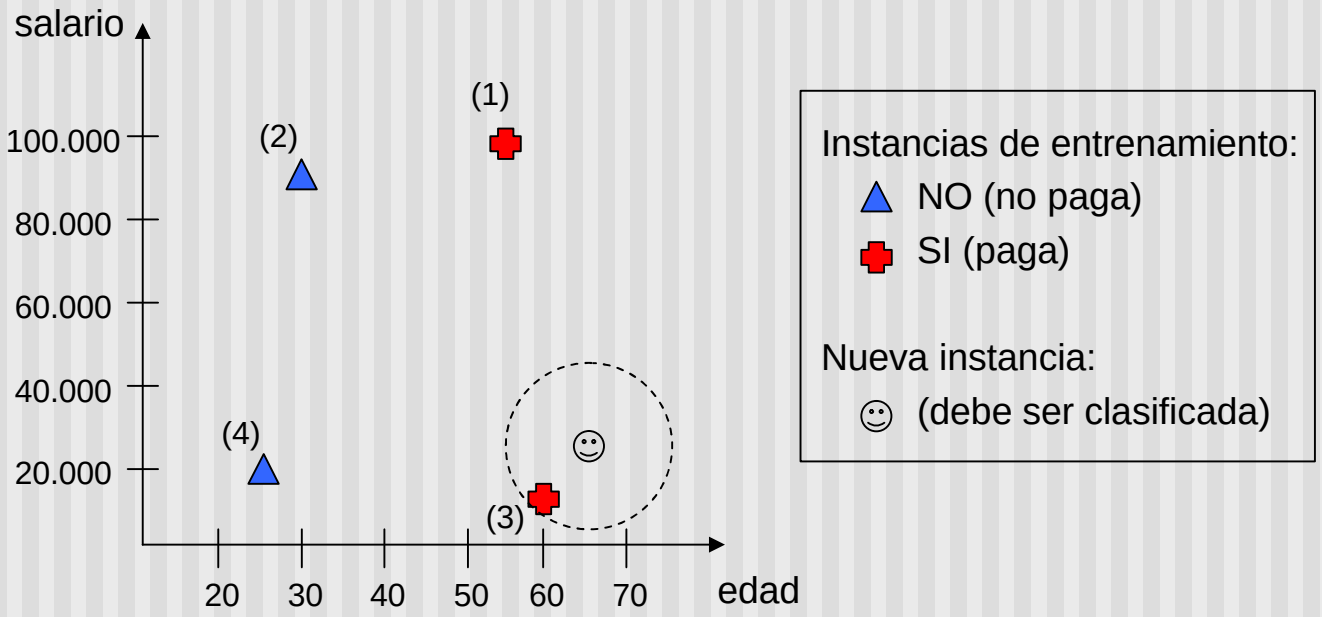
Salario = 25.000

Edad = 65

… de acuerdo con la edad, la clase debe ser SI (devuelve préstamo)

Vecino más cercano (VIII)

- Una representación gráfica confirma que sólo importa la edad.
- El vecino más cercano es el ejemplo 3 (1-NN).
- La clase de la nueva instancia debe ser SI (devuelve prestamo):



Vecino más cercano (IX)

- Si realizamos el cálculo numérico:

$$d_1 = \sqrt{75000^2 + 10^2} = 75000,00067$$

$$d_2 = \sqrt{65000^2 + 35^2} = 65000,00942$$

$$d_3 = \sqrt{10000^2 + 10^2} = 10000,00125$$

$$d_4 = \sqrt{5000^2 + 40^2} = 5000,1599$$

- … el ejemplo más cercano es el #4!
- … la clase debería ser NO (no devuelve el préstamo)!

Vecino más cercano (X)

- ¿Dónde se encuentra el problema?
  - LOS ATRIBUTOS HAN DE SER NORMALIZADOS.
  - En otro caso, los atributos con mayor valor absoluto siempre prevalecen sobre los demás.
  - Por eso parece que sólo importa el salario.
- Posibles normalizaciones:
  - Todos los atributos normalizados a media cero y varianza unidad.

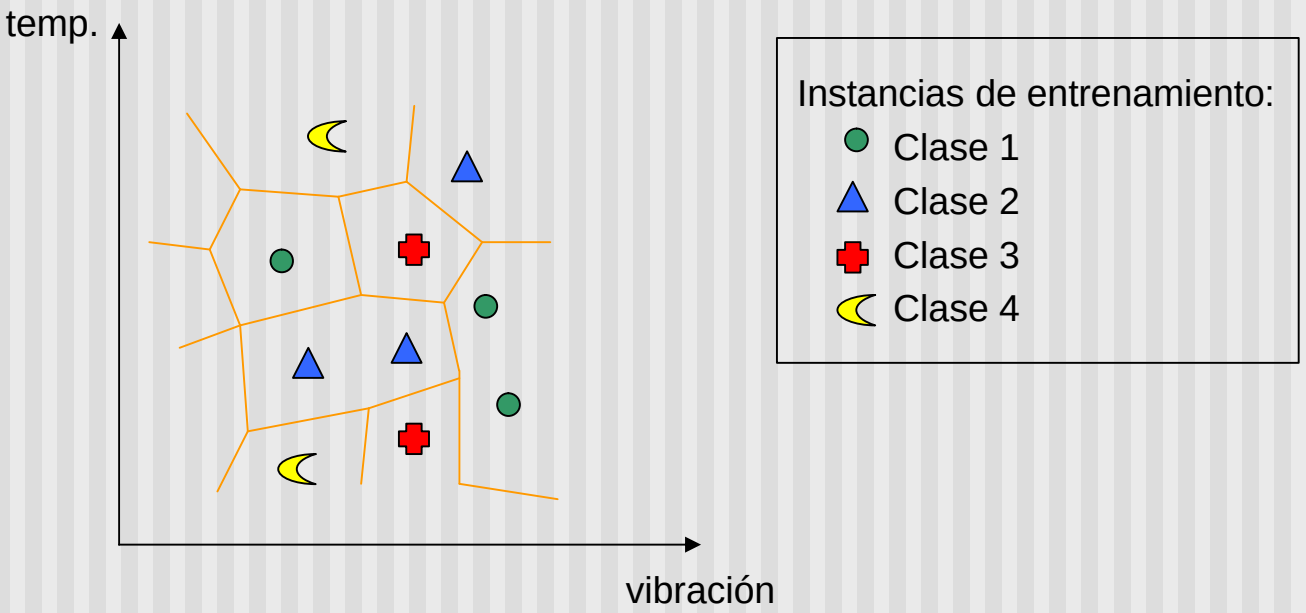
$$\hat{x}_i = \frac{x_i - m_i}{s_i}$$

  - Todos los atributos normalizados al rango 0 - 1.

$$\hat{x}_i = \frac{x_i - min_i}{max_i - min_i}$$

Vecino más cercano. Resumen (I)

1. Capacidad de representación:
   - Muy elevada, fronteras de decisión complejas (superficies de Voronoi).



Vecino más cercano. Resumen (II)

2. Legibilidad:
   - Ninguna, no se crea modelo.
3. Tiempo de cómputo on-line:
   - Lento, es necesario calcular distancias a todos los ejemplos de entrenamiento.
4. Tiempo de cómputo off-line:
   - Rápido, sólo el tiempo necesario para almacenar todas las instancias de entrenamiento.
5. Parámetros a ajustar:
   - Sólo el número de vecinos (k-NN).
6. Robustez ante instancias de entrenamiento ruidosas:
   - 1-NN no es robusto; k-NN es más robusto a medida que aumenta k.
7. Sobreajuste (overfitting):
   - Es difícil que ocurra; más difícil cuanto mayor es k.

ÁRBOLES DE DECISIÓN

Árboles de decisión (I)

- Funcionamiento básico ya estudiado.
- Método Eager: crea un modelo.
- Estudiaremos dos algoritmos:
  - ID3 para atributos discretos.
  - C4.5 para atributos continuos y discretos.
- Creación del árbol: selección de atributos comenzando en el nodo raíz.
- El criterio para elegir los atributos en ID3 y C4.5 es el 'incremento de información' o 'information gain'.
- Information gain = reducción de entropía como consecuencia de realizar una división de los datos.

Árboles de decisión (II)

- Algoritmo ID3.
  - Entropía de un nodo del árbol:

$$E = -\sum_{i=1}^{k} r_i \cdot log_2(r_i)$$

    - r_i = porcentaje de instancias pertenecientes a la clase *i*.
    - k = número de clases.
  - Entropía = falta de homogeneidad.
  - Entropía mínima: todas las instancias pertenecen a la misma clase:

$$E = -1 \cdot log_2(1) = 0$$

  - Entropía máxima: todas las clases representadas por igual.

$$E = -k \cdot \frac{1}{k} log_2\left(\frac{1}{k}\right) = log_2(k) \underset{k=2}{=} 1$$

Árboles de decisión (III)

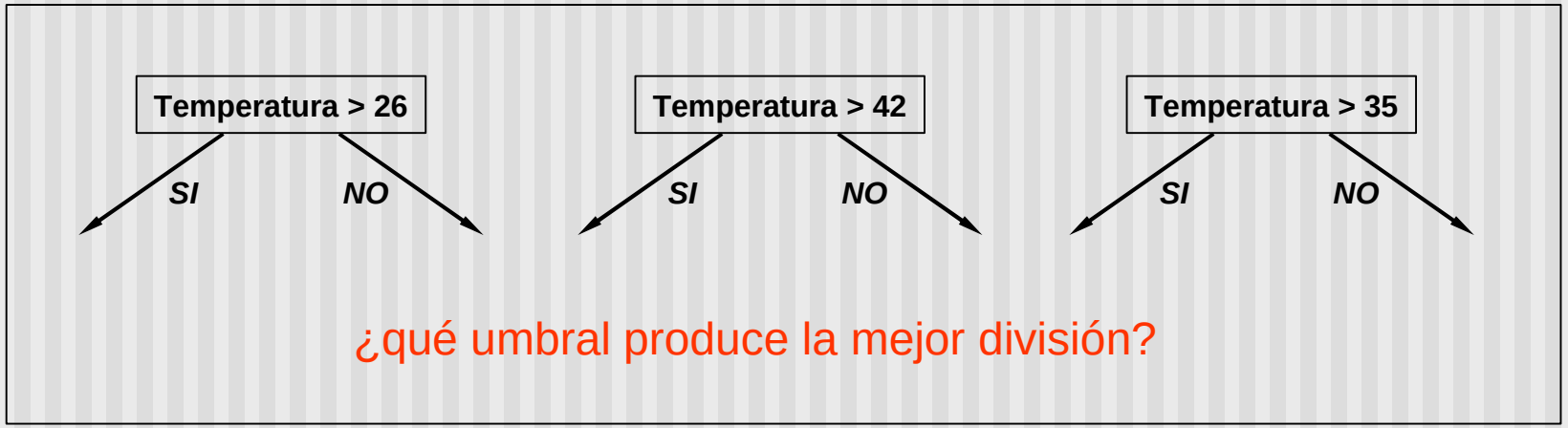
- Algoritmo ID3.
  - Ganancia de información como resultado de dividir un nodo:

$$IG = E_0 - \sum_{i=1}^{NP} \left(\frac{n_i}{n_0} \cdot E_i \right)$$

    - E_0 = entropía del nodo antes de la división.
    - E_i = entropía de cada partición (fórmula general para dos o más particiones).
    - NP = número de particiones.
    - n_i = número de instancias pertenecientes a la partición *i*.
    - n_0 = número total de instancias del nodo.
  - Hay ganancia de información cuando la división envía instancias con clases distintas a los distintos nodos.
  - El atributo que permite obtener la mayor ganancia de información es el seleccionado para dividir el nodo.
    - Criterio similar al utilizado para construir el árbol de decisión a mano en apartados anteriores.

Árboles de decisión (IV)

- Algoritmo C4.5.
  - Principal diferencia con ID3: permite usar tanto atributos discretos como atributos continuos.
  - Problema: es necesario elegir tanto un atributo como un umbral para realizar la división de un nodo.



Árboles de decisión (V)

- Algoritmo C4.5.
  - No es necesario probar todos los posibles umbrales (serían infinitas posibilidades, dado que los atributos son valores reales).
  - Los únicos umbrales razonables son aquellos que se encuentran en la frontera entre dos clases, de acuerdo con los ejemplos de entrenamiento.
  - Se oredenan los atributos, y los umbrales candidatos son aquellos que quedan en el punto medio entre dos clases.
  - Ejemplo:

Temp.	25	34	42	42	51	54	66	67	72	75
Clase	NF	NF	F	F	F	NF	F	NF	NF	NF

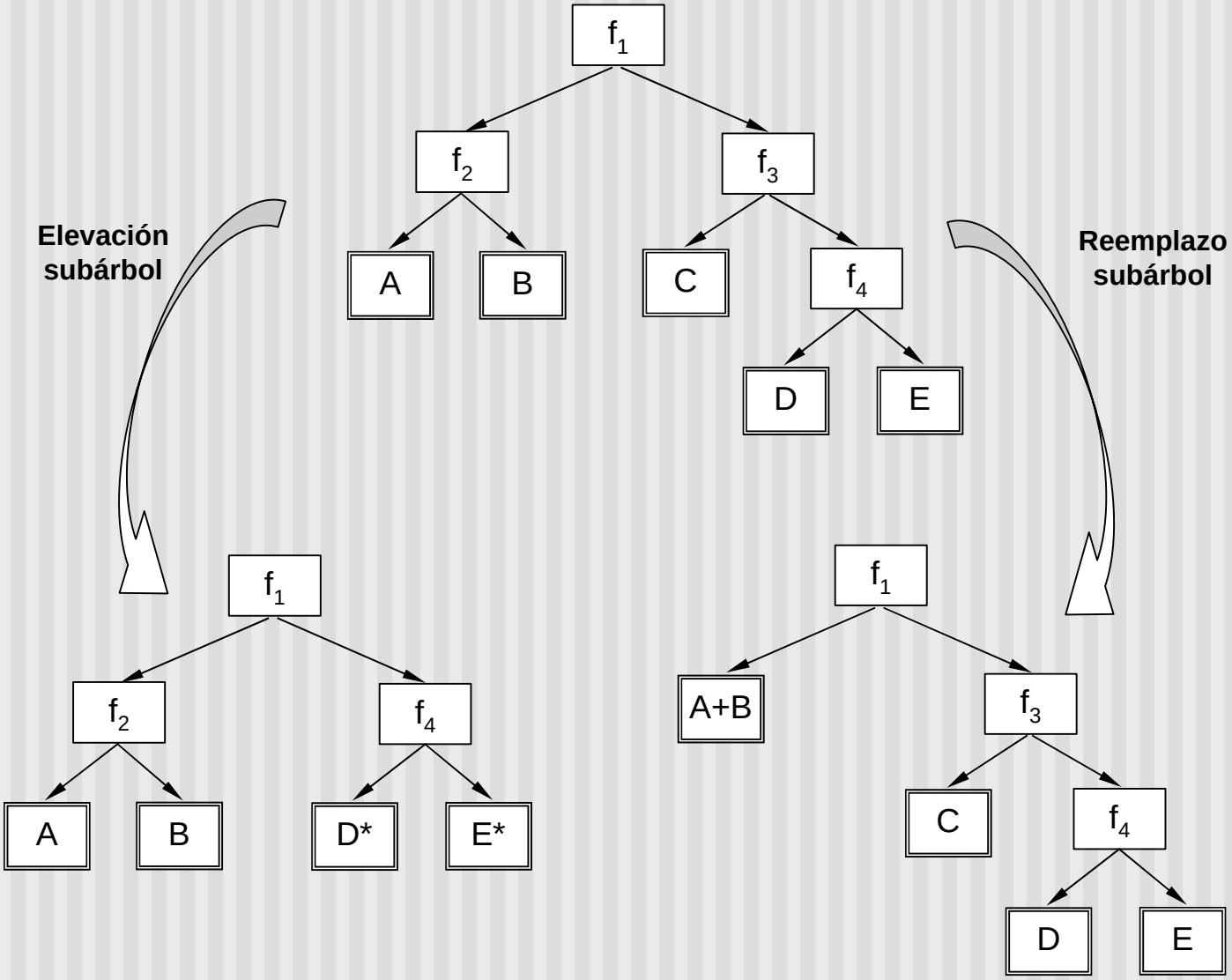
38 52.5 60 66.5

  - Se calcula la ganancia de información para cada atributo y para cada umbral y se eligen atributo y umbral óptimos.

Árboles de decisión (VI)

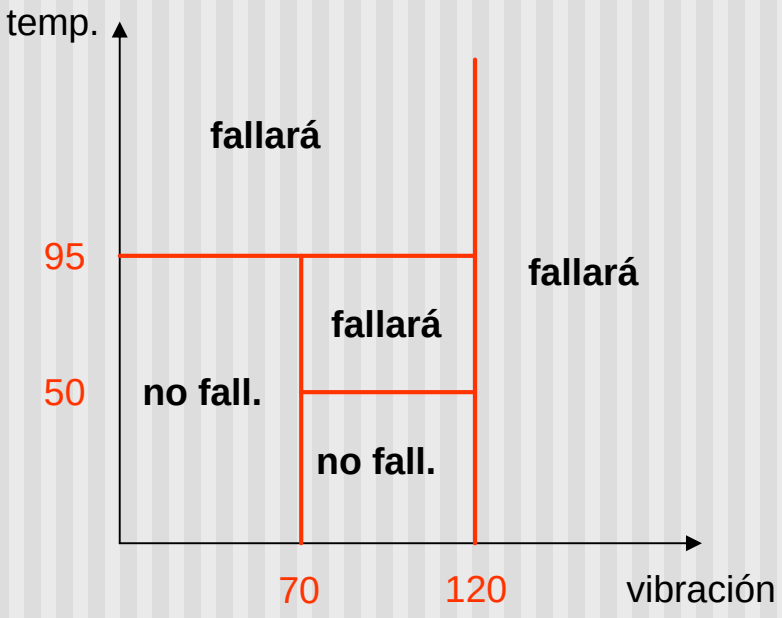
- Algoritmo C4.5: poda de los árboles.
  - ID3 nunca produce árboles demasiado grandes, pero C4.5 sí (puede repetir atributos en el árbol).
  - Un árbol demasiado grande puede producir sobreajuste.
  - Es necesario podar los árboles (pruning).
  - Se realiza un test estadístico que indica si el árbol podado funcionará previsiblemente mejor o peor que el árbol sin podar.
    - Se considera el peor caso posible (peor situación posible dentro del rango previsible).
    - El rango es mayor o menor en función de un parámetro (nivel de confianza) que es ajustable.
  - Hay dos posibles tipos de poda:
    - Reemplazo de un subárbol (un subárbol se sustituye por una hoja).
    - Elevación de un subárbol (un subárbol se eleva en el árbol principal).
  - Se muestra un ejemplo:

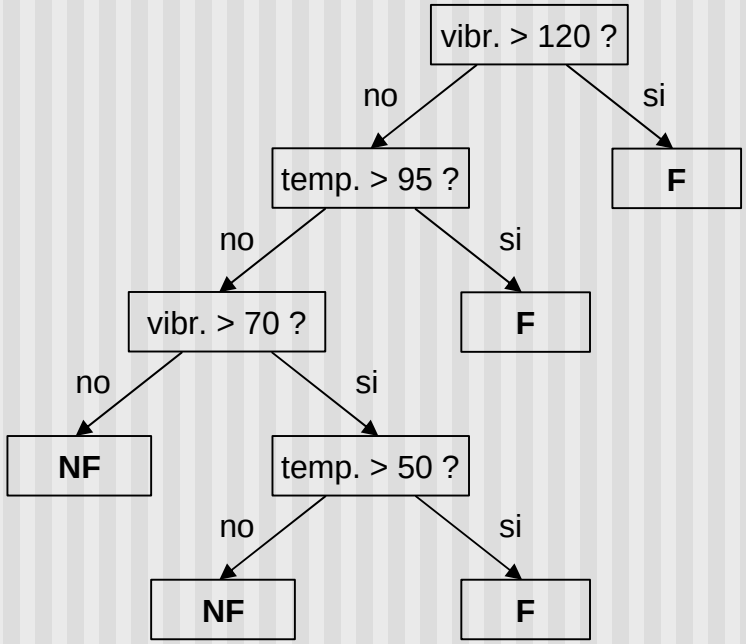
Árboles de decisión (VII)



Árboles de decisión. Resumen (I)

1. Capacidad de representación:
   - No muy elevada, las superficies de decisión son siempre perpendiculares a los ejes:





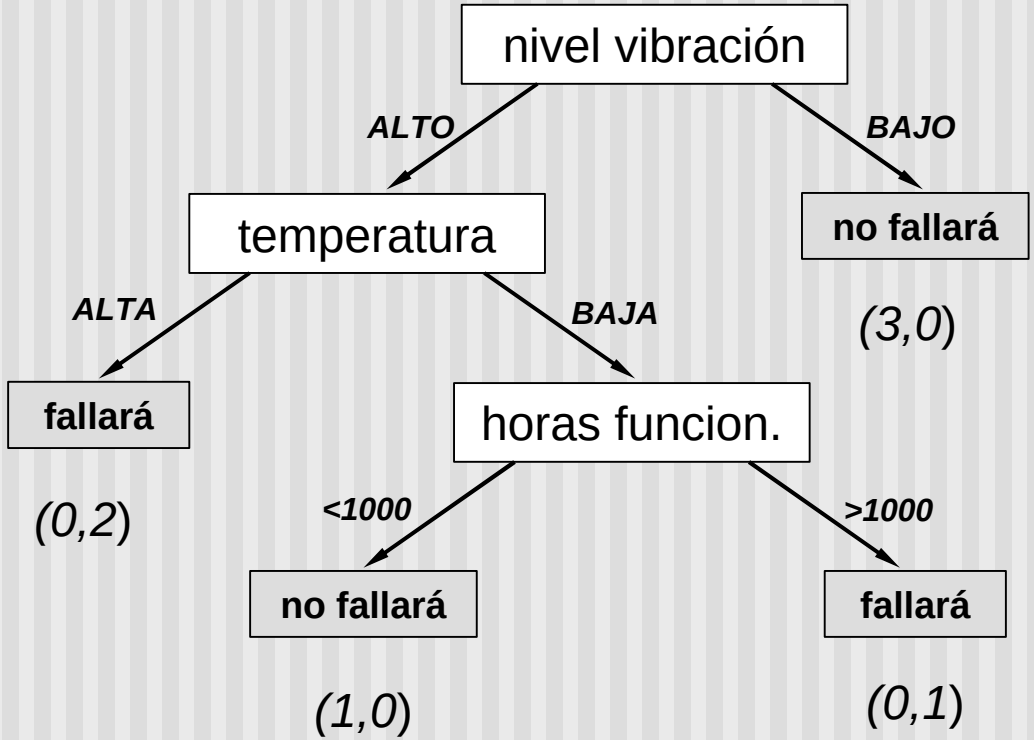
Árboles de decisión. Resumen (II)

2. Legibilidad:
   - Muy alta. Uno de los mejores modelos en este sentido.
3. Tiempo de cómputo on-line:
   - Muy rápido, clasificar un nuevo ejemplo es tan sencillo como recorrer el árbol hasta alcanzar un nodo hoja.
4. Tiempo de cómputo off-line:
   - Rápido, tanto ID3 como C4.5 son algoritmos simples.
5. Parámetros a ajustar:
   - Fundamentalmente el nivel de confianza para la poda. Fácil de ajustar: el valor por defecto (25%) da buenos resultados.
6. Robustez ante instancias de entrenamiento ruidosas:
   - Robusto.
7. Sobreajuste (overfitting):
   - No se produce sobreajuste siempre que se realice una poda del árbol (C4.5).

LISTAS DE REGLAS

Listas de reglas (I)

- Modelo muy similar a un árbol de decisión.
- Es posible obtener una lista de reglas a partir de un árbol:



Listas de reglas (II)

- Cada ruta desde el nodo raíz hasta una de las hojas se puede convertir en una regla:
  - if (nivel_vibración = BAJO) then no fallará
  - if (nivel_vibración = ALTO) and (temperatura = ALTA) then fallará
  - if (nivel_vibración = ALTO) and (temperatura = BAJA) and (horas_funcionamiento < 1000) then no fallará
  - else fallará
- Importante: éste no es el procedimiento más eficiente para generar una lista de reglas.

Listas de reglas (III)

- Tres tipos diferentes de listas de reglas:
  - Reglas de clasificación.
  - Reglas de asociación.
  - Reglas relacionales.

Rule lists (IV)

- Reglas de clasificación.
  - Una combinación lógica de los valores de los atributos sirve para predecir la clase.
  - El ejemplo anterior está formado por reglas de clasificación:
    - Atributos: nivel_vibración, temperatura, horas_funcionamiento.
    - Clase: fallará / no fallará.
- Reglas de asociación.
  - Una combinación lógica de los valores de los atributos sirve para predecir el valor de otro atributo.
  - Ejemplo:
    - If (nivel_vibración = ALTO) then (temperatura = ALTA)
  - Útiles como una forma de explorar todas las posibles relaciones entre los datos.

Listas de reglas (V)

- Reglas relacionales:
  - Una condición puede combinar los valores de varios atributos.
  - Permiten expresar relaciones más complejas.
  - Ej.: control de calidad basado en medidas de una pieza:

Alto	Ancho	Profundo	Peso	VALIDA
-	-	-	-	Si
-	-	-	-	No
-	-	-	-	Si
-	-	-	-	…

  - Podría inferirse una regla como la siguiente:
    - If (alto = ancho) then VALIDA (sólo son válidas las piezas cuadradas)
  - No podría expresarse con reglas de clasificación.

Listas de reglas. Resumen

1. Capacidad de representacion:
   - Similar a los árboles de decisión. Las reglas relacionales tienen mayor capacidad de representación pero no se utilizan normalmente (son difíciles de inferir).
2. Legibilidad:
   - Similar a los árboles de decisión (depende de las preferencias del usuario).
3. Tiempo de cómputo on-line:
   - Muy rápido, basta evaluar un conjunto de funciones lógicas.
4. Tiempo de cómputo off-line:
   - Rápido, la inferencia de listas de reglas es sencilla.
5. Parámetros a ajustar:
   - Pocos o ninguno. Fáciles de ajustar.
6. Robustez ante ejemplos de entrenamiento ruidosos:
   - Similar a los árboles de decisión (robustas).
7. Sobreajuste (overfitting):
   - No suele ocurrir. Las listas de reglas se pueden podar como los árboles.

APRENDIZAJE BAYESIANO

Aprendizaje Bayesiano (I)

- Se basa en la aplicación de la regla de Bayes:

$$P(h \mid D) = \frac{P(D \mid h) \cdot P(h)}{P(D)}$$

- Veamos el significado de cada término en la regla…

Aprendizaje Bayesiano (II)

$$P(h \mid D) = \frac{P(D \mid h) \cdot P(h)}{P(D)}$$

Término a calcular para clasificar un nuevo ejemplo

Término a calcular a partir de los ejemplos de entrenamiento

- h: hipótesis o clase (ej.: la máquina fallará).
- D: conjunto de valores para los atributos (ej.: nivel_vibraciones=alto, temperatura=baja, etc).

Aprendizaje Bayesiano (III)

- P(h): probabilidad a priori para una de las clases (ej.: probabilidad de que una máquina cualquiera falle). A priori significa 'sin conocer los valores de los atributos'.

- P(D): probabilidad a priori de que los atributos tengan unos ciertos valores (ej. probabilidad de encontrar un conjunto de atributos como: temperatura=alta, nivel_vibraciones = bajo, horas_funcionamiento < 2000, etc.). Estos valores corresponden a la nueva instancia ue se quiere clasificar.

Aprendizaje Bayesiano (IV)

- P(D|h): probabilidad a posteriori de que los atributos tengan unos ciertos valores, dado que la instancia pertenece a una cierta clase (ej. probabilidad de encontrar un conjunto de valores como: temp.=alta, vibr.=bajo, horas<2000, etc. en una instancia correspondiente a una máquina que falló).

- P(h|D): probabilidad a posteriori de que una instancia pertenezca a una cierta clase dado que los atributos muestran ciertos valores (ej. probabilidad de que la máquina falle dado que los valores de los atributos son: temp.=baja, vibr.=alto, etc.). Este es el dato a obtener para clasificar una nueva instancia.

Aprendizaje Bayesiano (V)

- Es fácil calcular estas probabilidades a partir de los datos?
  - P(h): muy fácil. La probabilidad a priori de una clase se puede calcular como el porcentaje de ejemplos de entrenamiento pertenecientes a esa clase (suponiendo que los ejemplos de entrenamiento se eligen aleatoriamente).
  - P(D): muy difícil. Para estimar la probabilidad de que los atributos tengan un cierto conjunto de valores es necesario disponer de un número extremadamente elevado de ejemplos de entrenamiento.

    ej. P(temp.=alta, vibr.=bajo, horas<2000, mantenimiento>1 mes)

    (Este conjunto de valores debe repetirse un cierto número de veces entre las instancias de entrenamiento para que el cálculo de la probabilidad se pueda considerar fiable)

Aprendizaje Bayesiano (VI)

- P(D) (cont.) El problema es aún más difícil cuando se utilizan atributos continuos:

  ej. P(temp.=52, vibr.=350, horas=347, mantenimiento= 1.7 meses)

  (En este caso es necesario estimar funciones de densidad de probabilidad; esto es relativamente fácil si se suponen distribuciones Gaussianas, pero casi imposible si hay que evaluar otras posibilidades)

- P(D|h): muy difícil. El problema es similar pero un poco más complejo, dado que sólo una fracción de las instancias de entrenamiento se puede utilizar para calcular la probabilidad: aquellas instancias cuya clase es *h*.

Aprendizaje Bayesiano (VII)

- Cómo predecir la clase de una nueva instancia?
  - La probabilidad de que la nueva instancia pertenezca a una cierta clase se obtiene aplicando el teorema de Bayes.
  - Ej. con dos clases (la máquina fallará / no fallará):

$$P(h_1 \mid D) = \frac{P(D \mid h_1) \cdot P(h_1)}{P(D)}$$

$$P(h_2 \mid D) = \frac{P(D \mid h_2) \cdot P(h_2)}{P(D)}$$

  - Se predice la clase con mayor probabilidad.
  - Dado que los denominadores son comunes, basta calcular:

$$P(D \mid h_1) \cdot P(h_1) > P(D \mid h_2) \cdot P(h_2) \quad ?$$

Aprendizaje Bayesiano (VIII)

- Conclusion: el aprendizaje bayesiano sólo se puede aplicar si se cumple alguno de los siguientes requisitos:
  - Se dispone de un gran número de ejemplos de entrenamiento.
  - Se dispone de algún conocimiento inicial sobre el problema a resolver (por ejemplo, las funciones de densidad de probabilidad de los atributos).
- En otro caso, es muy complicado calcular las probabilidades intervinientes en el teorema de Bayes.

Aprendizaje Bayesiano (IX)

- Ejemplo: applicación del teorema de Bayes para predecir la probabilidad de que un cierto paciente esté afectado por una enfermedad.
  - Se realiza un análisis de sangre al paciente con resultado positivo (de acuerdo con el test, la persona sufre la enfermedad).
  - Se conocen de antemano los siguientes datos:
    - Un 0.8% de la población está afectado por la enfermedad.
    - Si una persona está afectada, el test ofrece resultado positivo en un 98% de los casos (hay una tasa de error del 2%).
    - Si una persona no está afectada, el test ofrece resultado negativoen un 97% de los casos (tasa de error del 3%).

Aprendizaje Bayesiano (X)

- ¿Qué hipótesis tiene mayor probabilidad: afectado or no afectado?

$$P(afectado \mid test\ positivo) = \frac{P(test\ positivo \mid afectado) \cdot P(afectado)}{P(test\ positivo)}$$

$$P(no\ afectado \mid test\ positivo) = \frac{P(test\ positivo \mid no\ afectado) \cdot P(no\ afectado)}{P(test\ positivo)}$$

- Se comparan los numeradores de ambos terminos:

$$P(test\ positivo \mid afectado) \cdot P(afectado) = 0.98 \cdot 0.008 = 0.0078$$
$$P(test\ positivo \mid no\ afectado) \cdot P(no\ afectado) = 0.03 \cdot 0.992 = 0.0298$$

- Hipótesis más probable: no afectado por la enfermedad.

Aprendizaje Bayesiano (XI)

- Naïve Bayes:
  - Método muy utilizado en la práctica.
  - Premite aplicar el aprendizaje bayesiano incluso cuando el número de ejemplos disponible es pequeño.
  - Principal diferencia: se supone que todos los atributos son independientes (es más fácil estimar las probabilidades).
  - Si no se cumple esta condición, no debería funcionar (pero en la práctica, funciona en gran parte de los casos).
- ¿Por qué es posible aplicar Naïve Bayes cuando el número de ejemplos es pequeño?
  - Cuando dos atributos son independientes, se cumple:

$$P(at_1 = val_1, at_2 = val_2) = P(at_1 = val_1) \cdot P(at_2 = val_2)$$

  - Es mucho más fácil estimar probabilidades independientes.

Aprendizaje Bayesiano (XII)

- Ejemplo con el problema del fallo de la máquina:
  - Sin asumir independencia, la probabilidad a estimar es:

$$P(temp. = 56, vibr. = 250, horas = 55, meses = 3.7)$$

  - Asumiendo independencia, basta estimar:

$$P(temp. = 56)$$
$$P(vibr. = 250)$$
$$P(horas = 55)$$
$$P(meses = 3.7)$$

  - Una función de densidad de probabilidad univariante es mucho más fácil de estimar que una multivariante.

Aprendizaje Bayesiano. Resumen

1. Capacidad de representación:
   - Alta, las fronteras de decisión pueden tener cualquier forma.
2. Legibilidad:
   - Baja, los modelos son funciones de densidad de probabilidad.
3. Tiempo de cómputo on-line:
   - Rápido una vez que el modelo ha sido estimado.Slow, as p.d.f. have to be estimated from the training instances.
4. Tiempo de cómputo off-line:
   - Lento: es necesario estimar las funciones de densidad de probabilidad a partir de las instancias de entrenamiento.
5. Parámetros a ajustar:
   - Relacionados con el tipo de función de densidad de probabilidad de los atributos. Naïve Bayes es fácil de utilizar, pero en general el aprendizaje Bayesiano no lo es.
6. Robustez ante instancias de entrenamiento ruidosas:
   - Muy alta, dado que el método está basado en probabilidades.
7. Sobreajuste (overfitting):
   - Imposible obtener sobreajuste al trabajar con probabilidades.