Seminario Permanente de Formación en Inteligencia Artificial Aplicada a la Defensa

Machine Learning: Clasificación II

Salvador García

Instituto Andaluz de Investigación en Data Science and Computational Intelligence (DaSCI)

Dpto. Ciencias de la Computación e I.A.

Universidad de Granada

salvagl@decsai.ugr.es

http://sci2s.ugr.es

DaSCI

UNIVERSIDAD DE GRANADA

Clasificación con árboles

Árboles de decisión

1. Definición de árboles de decisión
2. Construcción de árboles de decisión
3. Criterios de selección de variables
4. Particionamiento del espacio con un árbol de decisión
5. Ventajas e inconvenientes del uso de árboles de decisión en clasificación
6. Algunos algoritmos de minería de datos basados en árboles de decisión

Definición de árboles de decisión

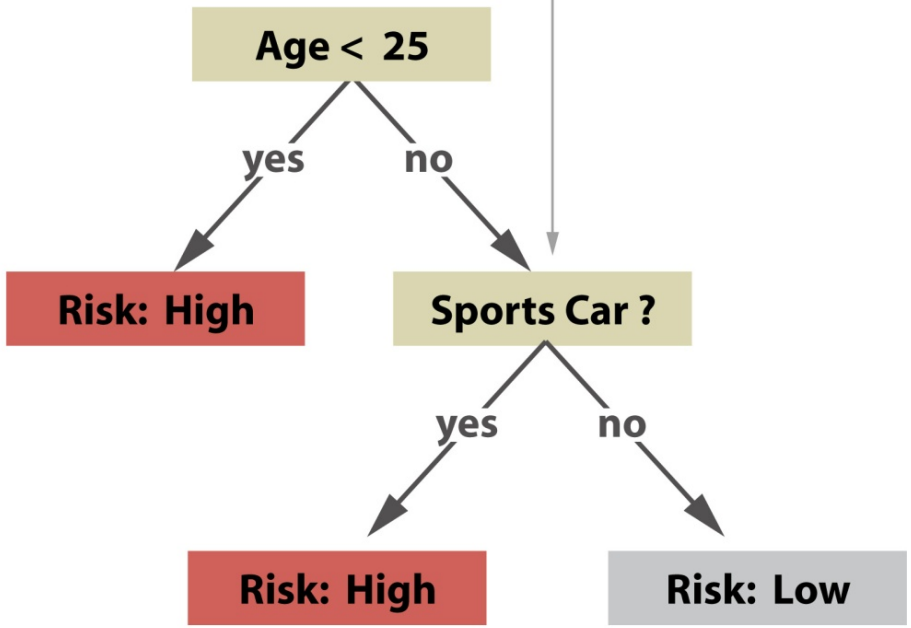
- Un árbol de decisión es un clasificador que en función de un conjunto de atributos permite determinar a que clase pertenece el caso objeto de estudio

- La estructura de un árbol de decisión es:
  - Cada hoja es una categoría (clase) de la variable objeto de la clasificación
  - Cada nodo es un nodo de decisión que especifica una prueba simple a realizar
  - Los descendientes de cada nodo son los posibles resultados de la prueba del nodo

Definición de árboles de decisión

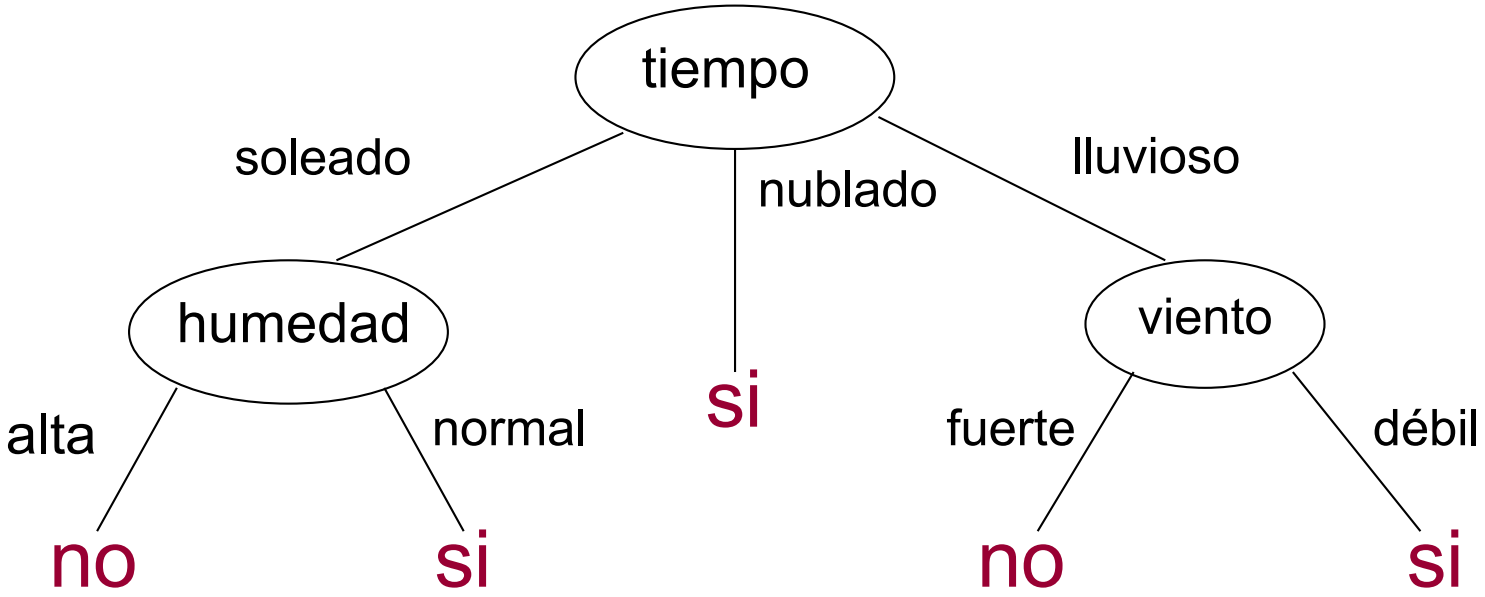
Insurance Risk Assessment

Age	Car Type	Risk
23	family	High
17	sports	High
43	sports	High
68	family	Low
32	truck	Low
20	family	High



Definición de árboles de decisión

- Ejemplo: Decidir si se puede jugar al tenis dependiendo del tiempo que hace, de la humedad y del viento. Un posible árbol de decisión es



Para clasificar se comienza por la raíz del árbol y se van haciendo las pruebas necesarias hasta llegar a una hoja (clase). P.e.

(tiempo=s, viento=d, temperatura=72, humedad=n) → si

Definición de árboles de decisión

- Otra opción es generar el conjunto de reglas asociado al árbol y utilizar un sistema basado en reglas (SBR). Del árbol anterior:

```
1) Si (tiempo soleado) y (humedad alta) entonces
   (jugar no)
2) Si (tiempo soleado) y (humedad normal) entonces
   (jugar si)
3) Si (tiempo nublado) entonces (jugar si)
4) Si (tiempo lluvioso) y (viento fuerte) entonces
   (jugar no)
5) Si (tiempo lluvioso) y (viento débil) entonces
   (jugar si)
```

Construcción de árboles de decisión

- El proceso de generación de un árbol de decisión consiste en dos fases
  - Construcción del árbol
    - Al principio todos los ejemplos de entrenamiento están en el nodo raíz
    - Se van dividiendo recursivamente los ejemplos en base a los atributos seleccionados
  - Poda del árbol
    - Identificar y quitar ramas que describen ruido o datos anómalos
- Uso del árbol de decisión: Clasificar un ejemplo desconocido
  - Comprobar los valores de los atributos del ejemplo contra el árbol de decisión

Construcción de árboles de decisión

- Algoritmo básico (un algoritmo voraz)
  - Se construye el árbol mediante la técnica divide y vencerás aplicada de forma recursiva
  - Al principio todos los ejemplos de entrenamiento están en el nodo raíz
  - Los atributos son categóricos (si son continuos, se discretizan previamente)
  - Los ejemplos se dividen recursivamente basándose en atributos seleccionados
  - Los atributos de test se seleccionan en base a una medida heurística o estadística (por ejemplo, la ganancia de información)

- Condiciones para terminar el particionamiento
  - Todos los ejemplos para un nodo dado pertenecen a la misma clase
  - No quedan más algoritmos para seguir particionando. En este caso se utiliza el voto de la mayoría para clasificar en el nodo hoja
  - NO quedan ejemplos

Construcción de árboles de decisión

- Entrada: Sea T el conjunto de ejemplos, $A=\{A_1, \ldots, A_n\}$ el conjunto de atributos y $C=\{C_1, \ldots, C_k\}$ el conjunto de valores que puede tomar la clase

ConstruirArbol (T,C,A)

1. Crear un nodo RAIZ para el árbol
2. Si todos los ejemplos en T pertenecen a la misma clase C_i, devolver el nodo RAIZ con etiqueta C_i
3. Si $A=\Phi$ devolver el nodo RAIZ con etiqueta C_i donde C_i es la clase mayoritaria en T
4. a ← el atributo de A que mejor clasifica T
5. Etiquetar RAIZ con a
6. Para cada valor v_i de a hacer
   1. Añadir una nueva rama debajo de RAIZ con el test $a=v_i$
   2. Sea T_i el subconjunto de T en el que $a = v_i$
   3. ConstruirArbol (T_i,C,A-a)
7. Devolver RAIZ

Construcción de árboles de decisión

Problema de asignación de crédito

crédito	ingresos	propietario	Gastos-mensuales
N	Bajos	N	Altos
N	Bajos	S	Altos
N	Medios	S	Altos
N	Medios	N	Altos
N	Altos	N	Altos
S	Altos	S	Altos
N	Bajos	N	Bajos
N	Medios	N	Bajos
N	Altos	N	Bajos
s	Medios	S	Bajos

Construcción de árboles de decisión

1. Se llama al algoritmo sobre el nodo raiz

crédito	ingresos	propietario	Gastos-mensuales
N	Bajos	N	Altos
N	Bajos	S	Altos
N	Medios	S	Altos
N	Medios	N	Altos
N	Altos	N	Altos
S	Altos	S	Altos
N	Bajos	N	Bajos
N	Medios	N	Bajos
N	Altos	N	Bajos
s	Medios	S	Bajos

Construcción de árboles de decisión

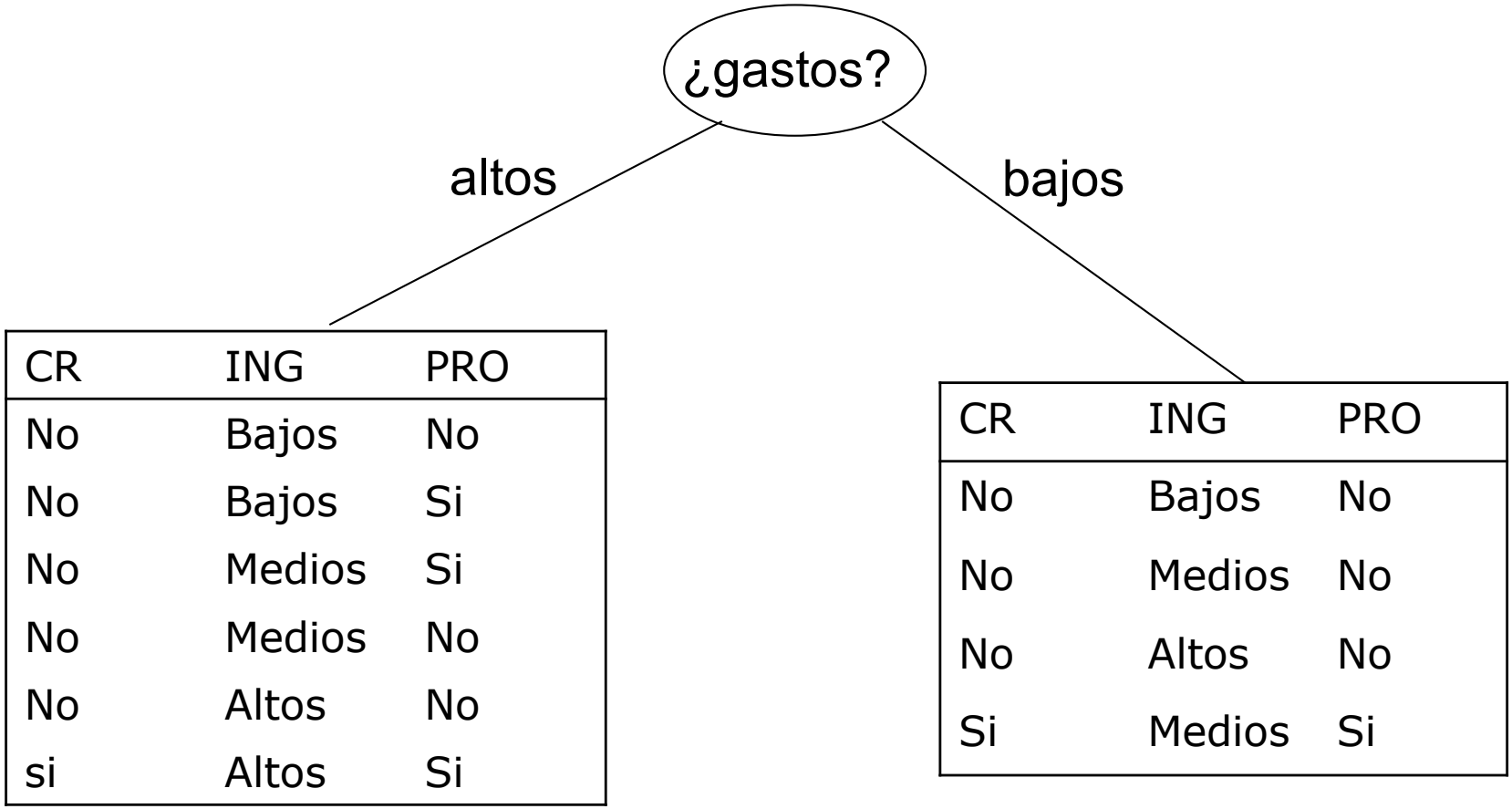
2. Seleccionamos gastos como test

CR	ING	PRO	gastos
No	Bajos	No	Altos
No	Bajos	Si	Altos
No	Medios	Si	Altos
No	Medios	No	Altos
No	Altos	No	Altos
si	Altos	Si	Altos

CR	ING	PRO	gastos
No	Bajos	No	Bajos
No	Medios	No	Bajos
No	Altos	No	Bajos
Si	Medios	Si	Bajos

Construcción de árboles de decisión

3. Preparamos los nodos para las llamadas recursivas

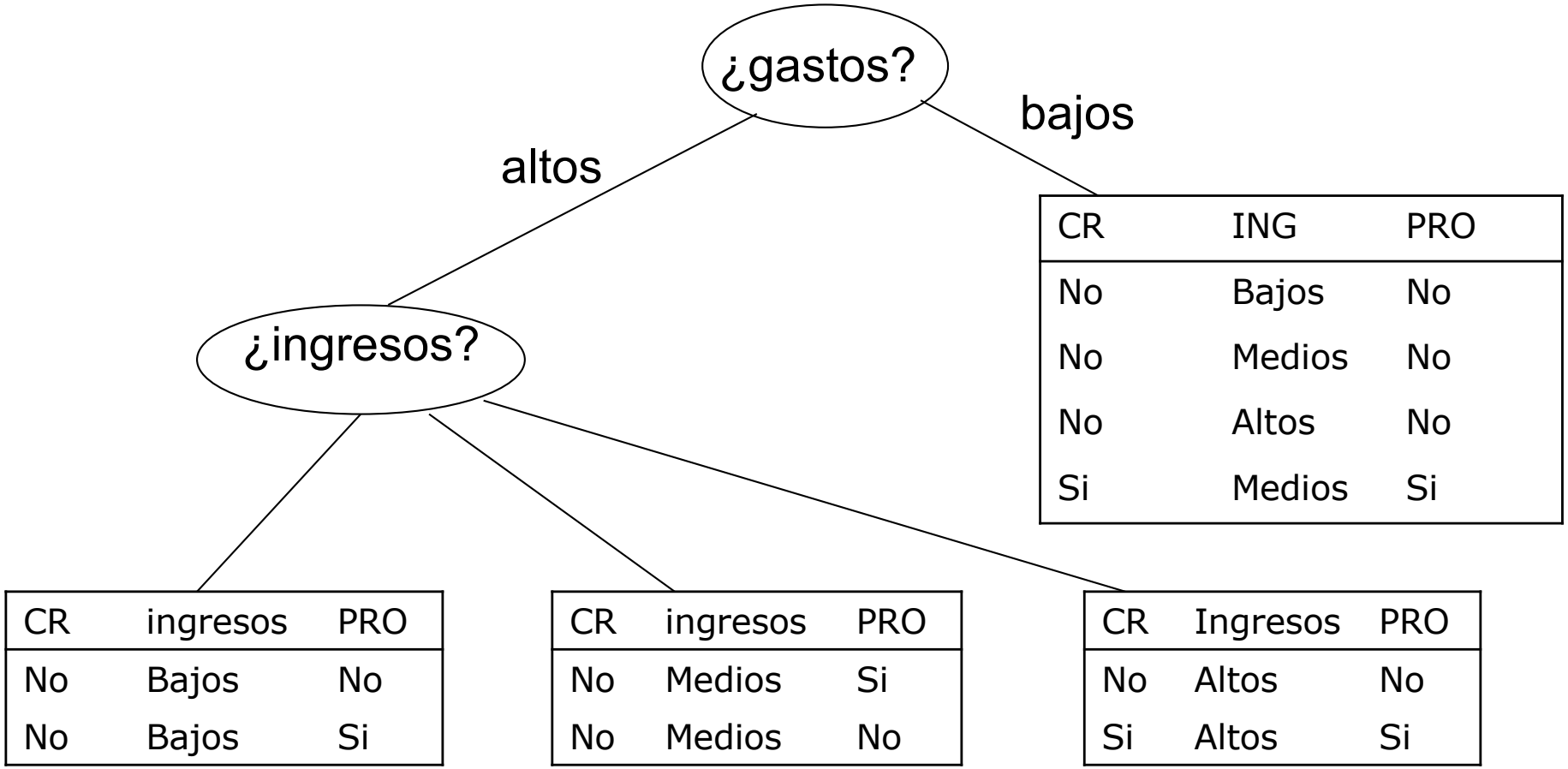


CR	ING	PRO
No	Bajos	No
No	Bajos	Si
No	Medios	Si
No	Medios	No
No	Altos	No
si	Altos	Si

CR	ING	PRO
No	Bajos	No
No	Medios	No
No	Altos	No
Si	Medios	Si

Construcción de árboles de decisión

4. Seleccionamos ingresos como test en gastos = altos



CR	ING	PRO
No	Bajos	No
No	Medios	No
No	Altos	No
Si	Medios	Si

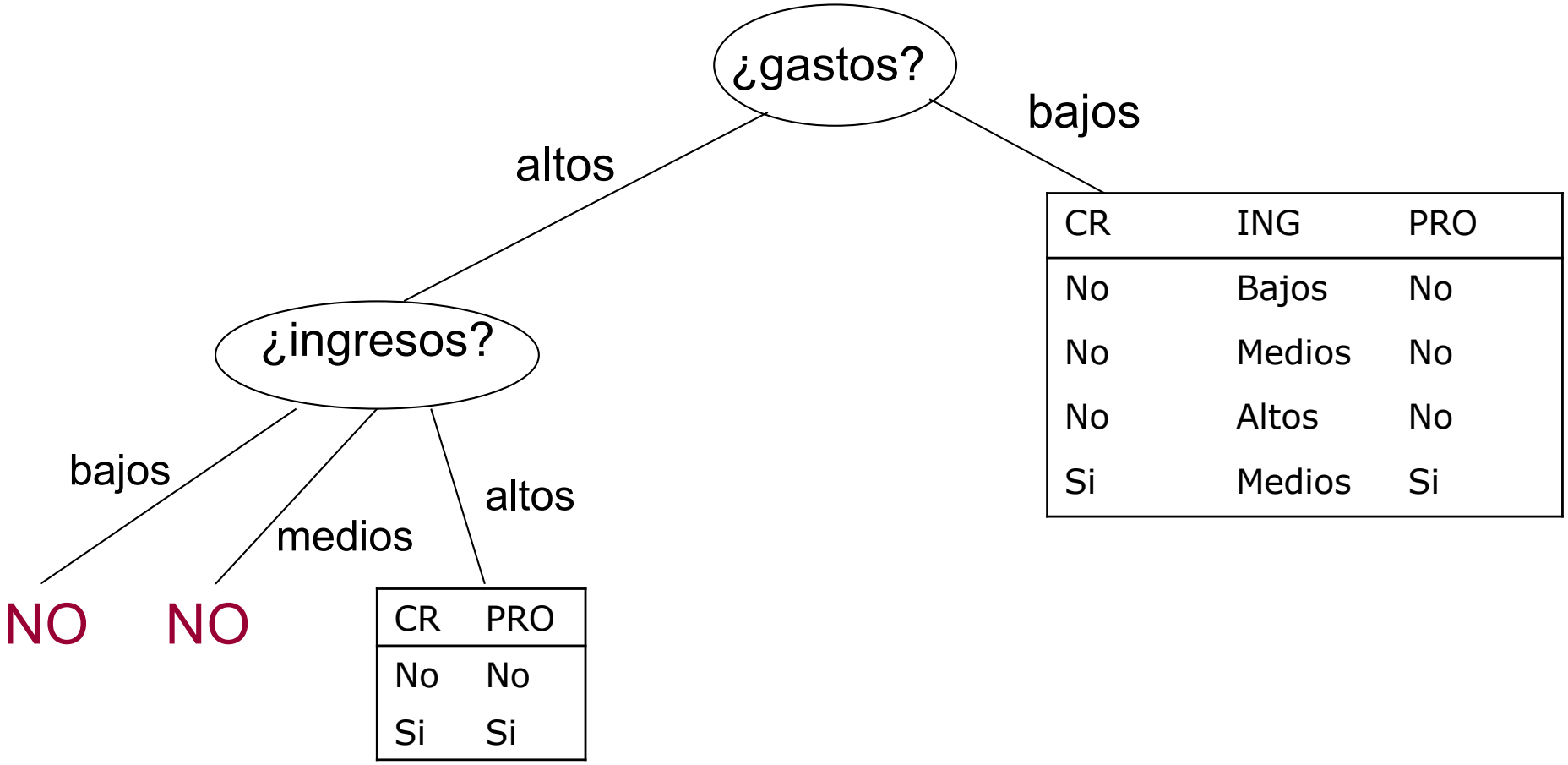
CR	ingresos	PRO
No	Bajos	No
No	Bajos	Si

CR	ingresos	PRO
No	Medios	Si
No	Medios	No

CR	Ingresos	PRO
No	Altos	No
Si	Altos	Si

Construcción de árboles de decisión

5. Creamos nodos hoja

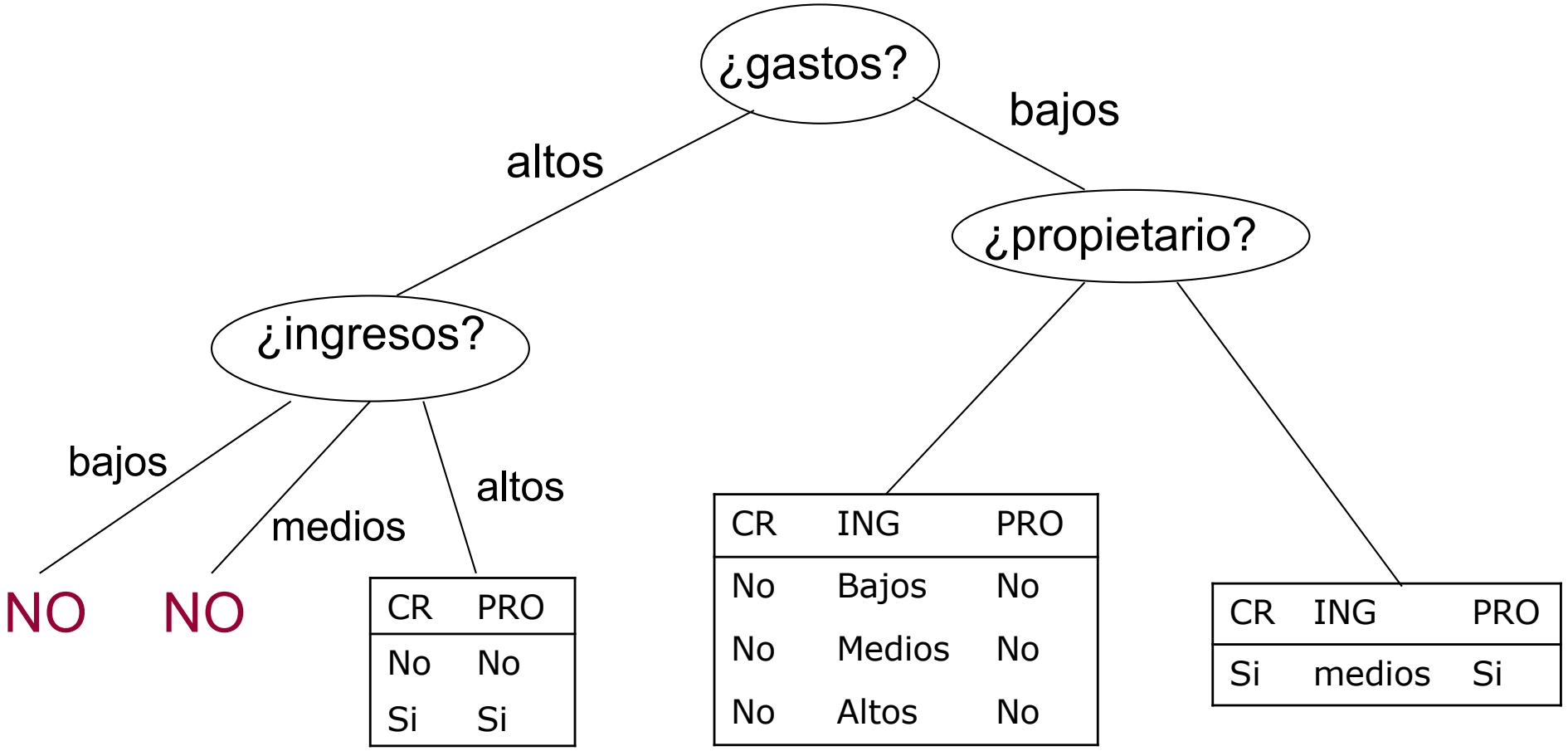


CR	ING	PRO
No	Bajos	No
No	Medios	No
No	Altos	No
Si	Medios	Si

CR	PRO
No	No
Si	Si

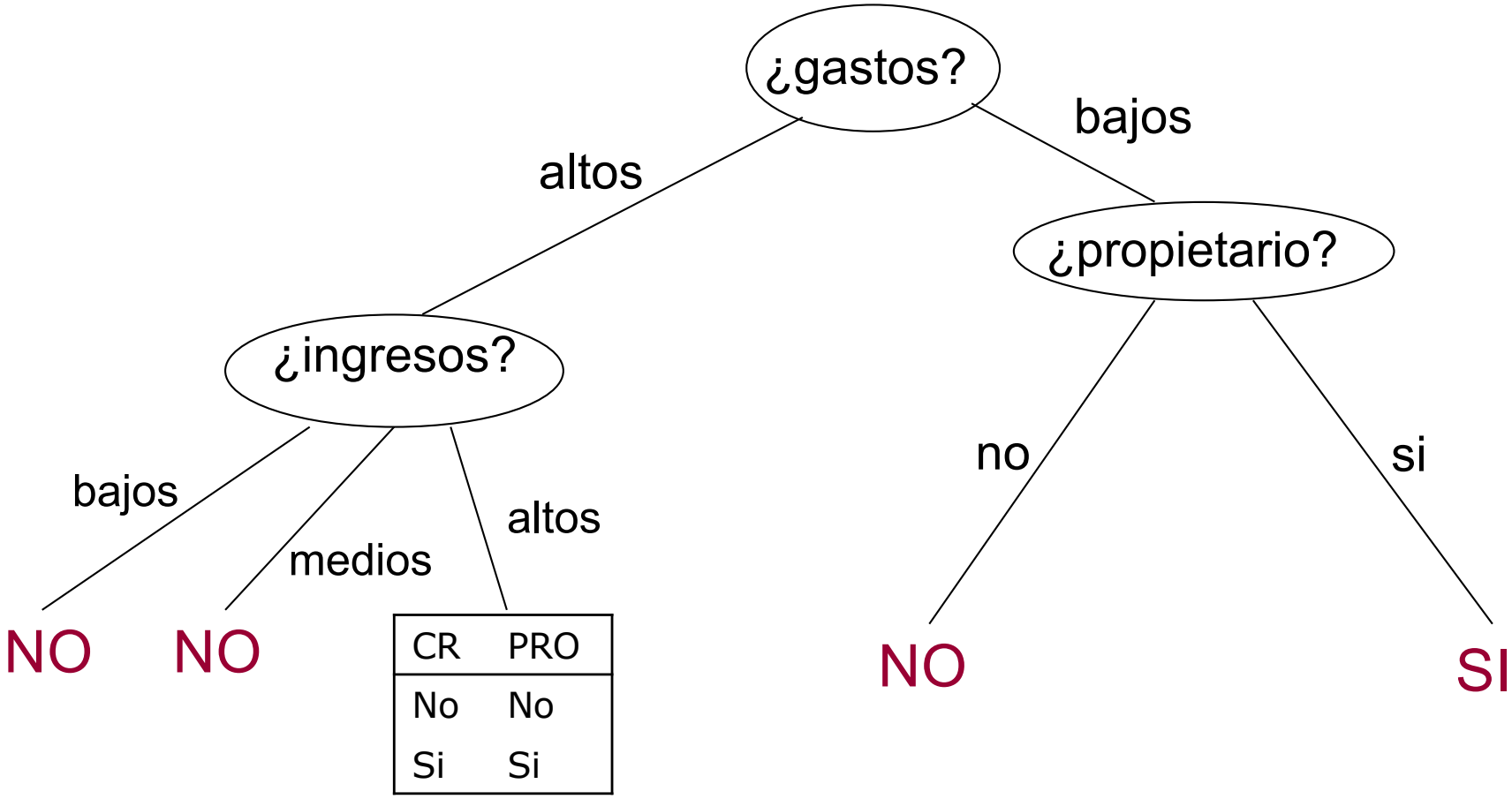
Construcción de árboles de decisión

6. Seleccionamos propietario como test en gastos = bajos



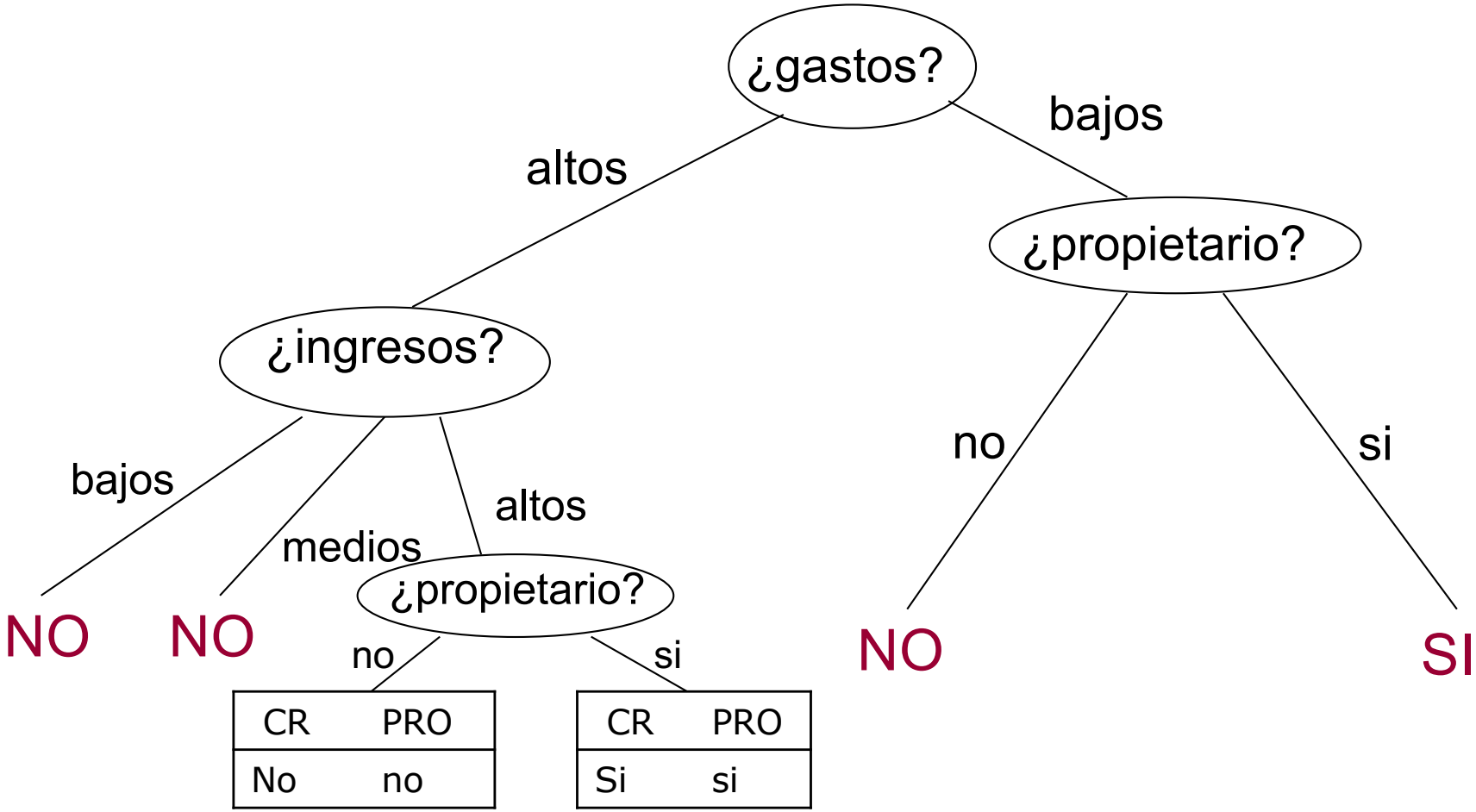
Construcción de árboles de decisión

7. Creamos nodos hoja



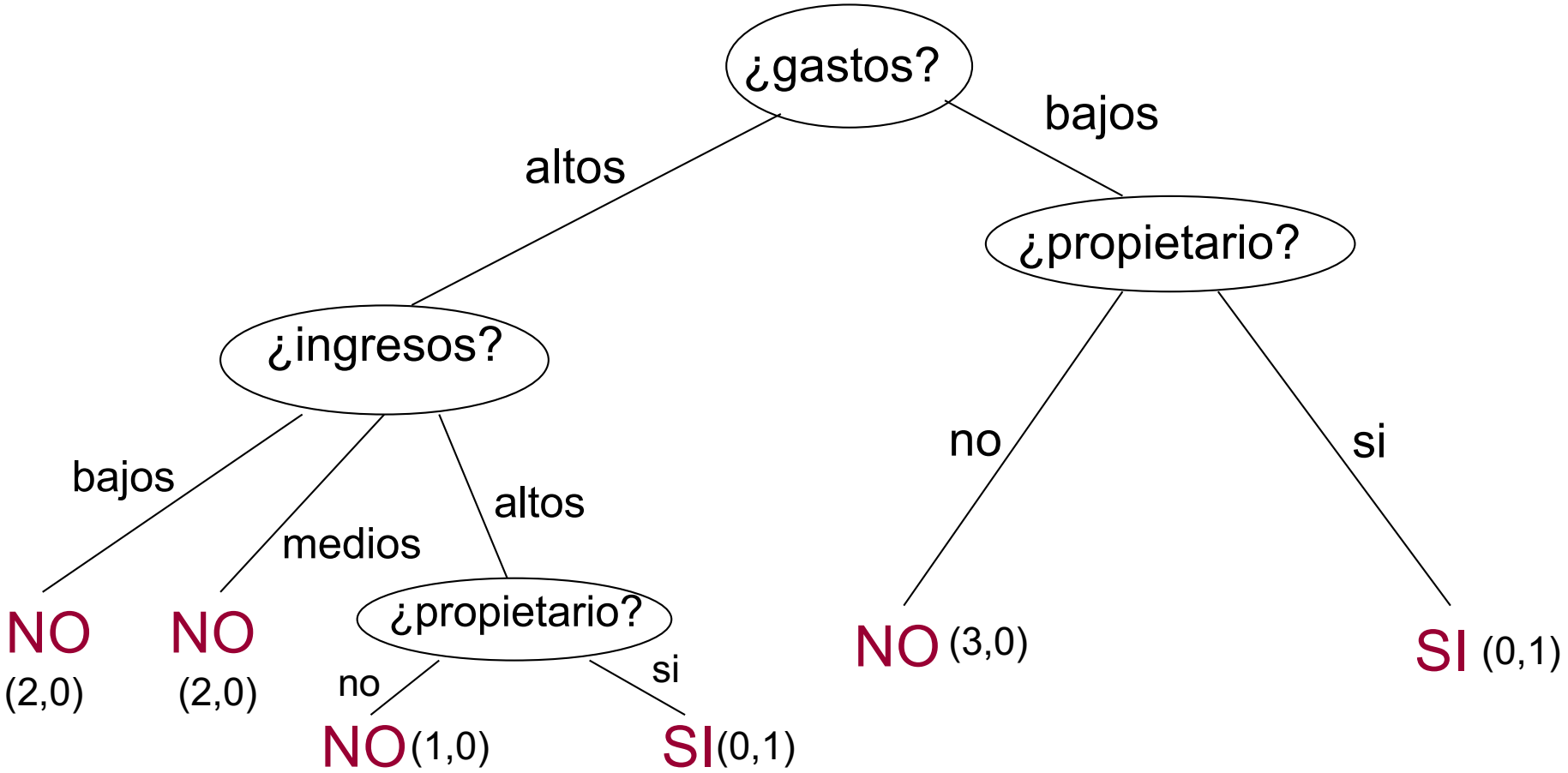
Construcción de árboles de decisión

8. Seleccionamos propietario como test en gastos=altos, ingresos=altos



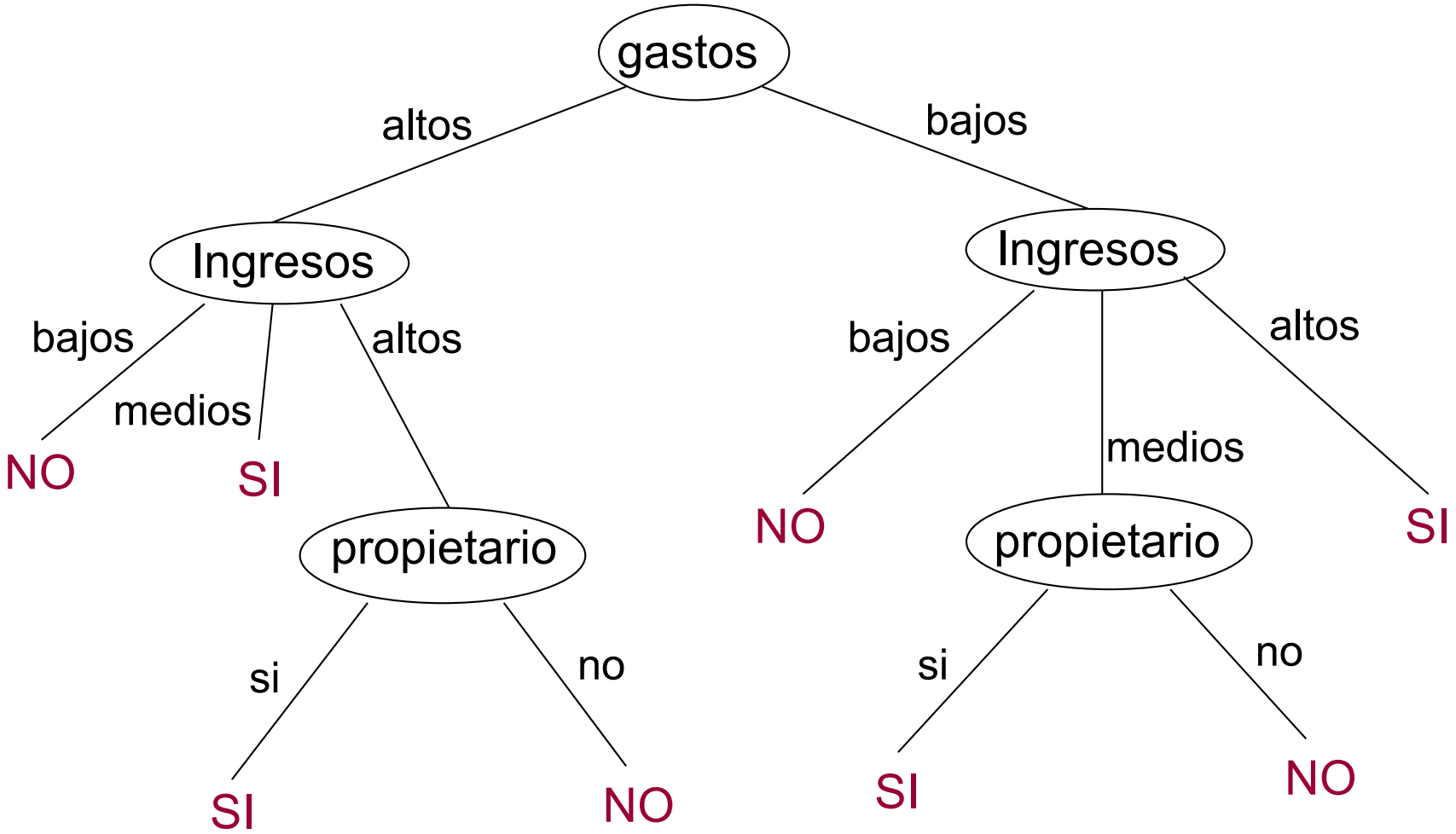
Construcción de árboles de decisión

9. Árbol de decisión (#no, #si)



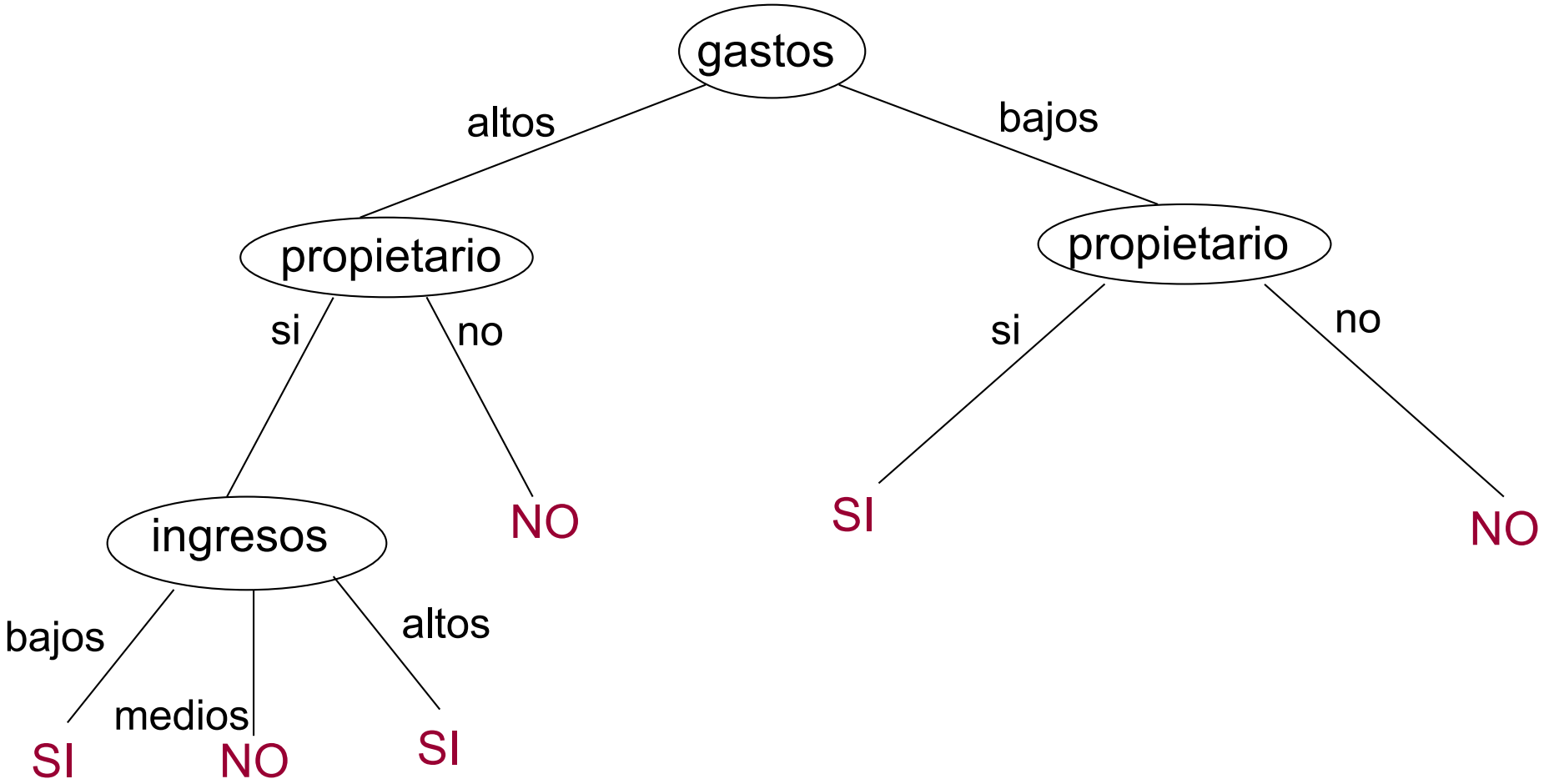
Construcción de árboles de decisión

Otro árbol de decisión para crédito



Construcción de árboles de decisión

Otro árbol de decisión para crédito



Construcción de árboles de decisión

- 1er. árbol: 6 reglas (2.33 premisas por regla)
- 2do. árbol: 8 reglas (2.5 premisas por regla)
- 3er. árbol: 6 reglas (2.5 premisas por regla)

- Dependiendo del orden en el que se van tomando los atributos obtenemos clasificadores de distinta complejidad

- Lo ideal sería tomar en todo momento el atributo que mejor clasifica

¿Cómo decidir qué atributo es el mejor?

Criterios de selección de variables

Existen distintos criterios para seleccionar la variable X* de test en cada momento. Algunos de los más conocidos son:

- *InfoGain*: Ganancia de información (ID3)

$$X^* = \max_X (H(C) - H(C|X))$$

- *GainRatio*: Ganancia de información modificada (C4.5)

$$X^* = \max_X \frac{H(C) - H(C|X)}{H(X)}$$

- GINI(CART) $X^* = \max_X (G(C) - G(C|X)) \quad con \quad G = 1 - \sum_{i=1}^{n} p_i^2$

P.e.: G(gastos)≈0.008;G(crédito)=0.32; G(credito|gastos)≈0.312

Criterios de selección de variables

Ganancia de información

- Objetivo: Seleccionar el atributo con la mayor ganancia de información
- Consideremos un problema con dos clases: *P* y *N*
  - Sea *S* el conjunto de ejemplos que contiene *p* elementos de la clase *P* y *n* elementos de la clase *N*
  - La cantidad de información necesaria para decidir si un ejemplo arbitrario dentro de *S* pertenece a *P* o a *N* viene definido por

$$I(p,n) = -\frac{p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n}$$

La entropía mide la aleatoriedad o sorpresa o incertidumbre al predecir una clase

Criterios de selección de variables

- Consideremos que utilizando un atributo *A*, el conjunto *S* se puede dividir en *v* conjuntos $\{S_1, S_2, \ldots, S_v\}$
  - Si S_i contiene p_i ejemplos de *P* y n_i ejemplos de *N*, la entropía o la información que se espera sea necesaria para clasificar los objetos en todos los subárboles S_i es

$$E(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p + n} I(p_i, n_i)$$

- La información que se podría ganar con una rama que considere *A* es

$$Gain(A) = I(p, n) - E(A)$$

Es la reducción en la entropía al considerar el atributo *A*

Criterios de selección de variables

Conjunto de entrenamiento

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Criterios de selección de variables

- Class P: buys_computer = "yes"
- Class N: buys_computer = "no"
- I(p, n) = I(9, 5) =0.940
- Cálculo de la entropía para *age*:

age	p_i	n_i	$I(p_i, n_i)$
<=30	2	3	0.971
30…40	4	0	0
>40	3	2	0.971

$$E(age) = \frac{5}{14} I(2,3) + \frac{4}{14} I(4,0) + \frac{5}{14} I(3,2) = 0.69$$

Por tanto

$$Gain(age) = I(p,n) - E(age)$$

De forma similar

$$Gain(income) = 0.029$$
$$Gain(student) = 0.151$$
$$Gain(credit\_rating) = 0.048$$

Criterios de selección de variables

GINI

- Si un conjunto de datos *T* tiene ejemplos pertenecientes a *n* classes, el índice gini, *gini*(*T*) se define como

$$gini(T) = 1 - \sum_{j=1}^{n} p_j^2$$

donde p_j es la frecuencia relativa de la clase *j* in *T*

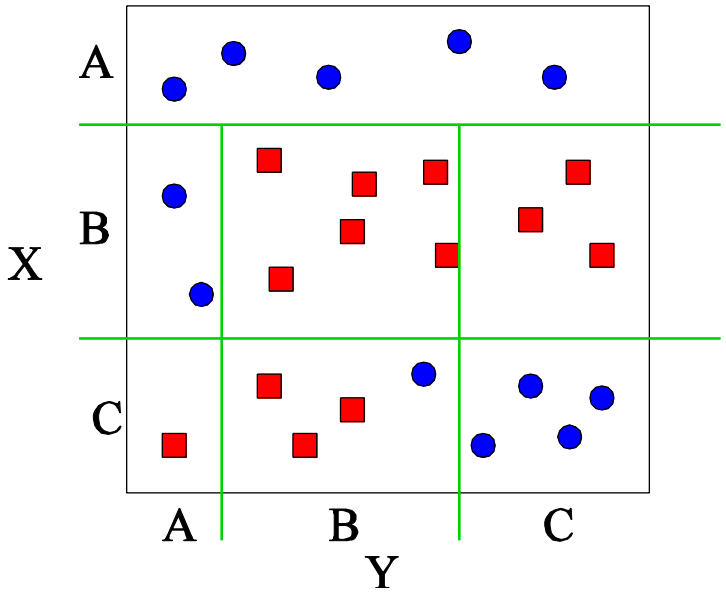
- Si se divide un conjunto de datos *T* en dos subconjuntos T_1 y T_2 de tamaños N_1 y N_2 respectivamente, el indice *gini* de los datos separados conteniendo ejemplos de las *n* clases se define como

$$gini_{split}(T) = \frac{N_1}{N} gini(T_1) + \frac{N_2}{N} gini(T_2)$$

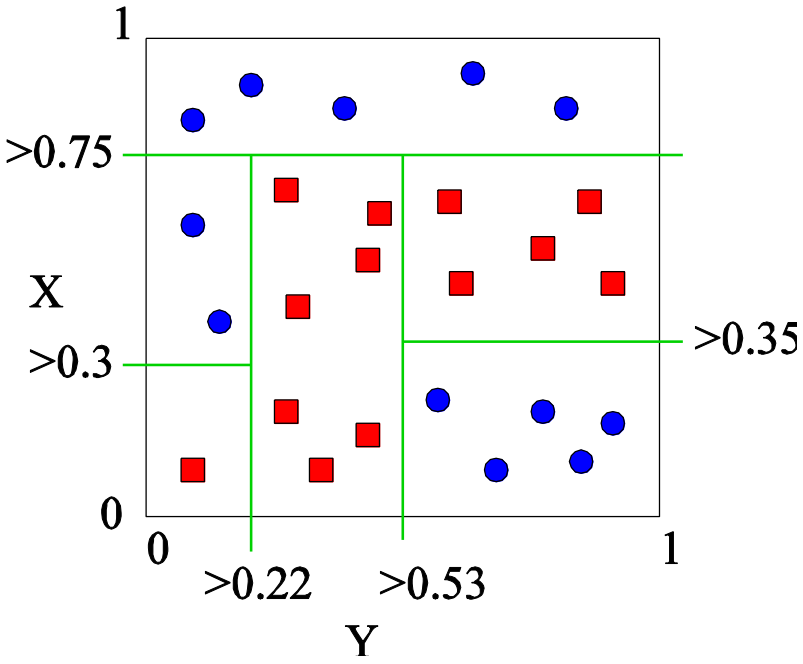
- Se elige para dividir el nodo, el atributo que proporciona el índice $gini_{split}(T)$ más pequeño (es necesario enumerar todos los posibles puntos de división para cada atributo)

Particionamiento del espacio de un árbol de decisión

Los árboles particionan el espacio de soluciones de forma exhaustiva



Variables X e Y discretas



Variables X e Y contínuas

Ventajas e inconvenientes del uso de arboles de decisión en clasificación

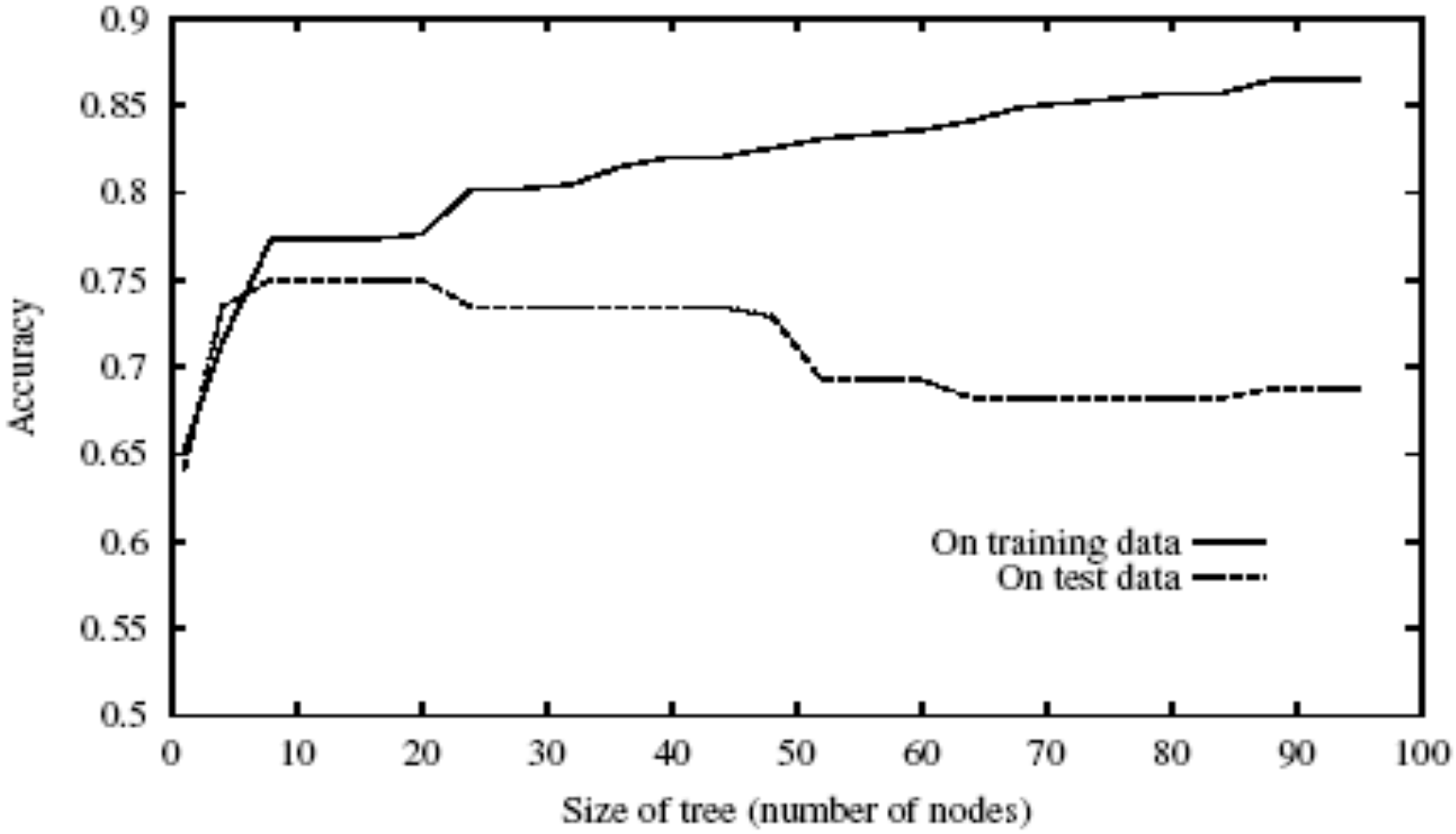
Ventajas:

- Los árboles de decisión son fáciles de utilizar y eficientes
- Las reglas que generan son fáciles de interpretar
- Escalan mejor que otros tipos de técnicas
- Tratan bien los datos con ruido

Inconvenientes:

- No manejan de forma fácil los atributos continuos
- Tratan de dividir el dominio de los atributos en regiones rectangulares y no todos los problemas son de ese tipo
- Tienen dificultad para trabajar con valores perdidos
- Pueden tener problemas de sobreaprendizaje
- No detectan correlaciones entre atributos

Ventajas e inconvenientes del uso de arboles de decisión en clasificación

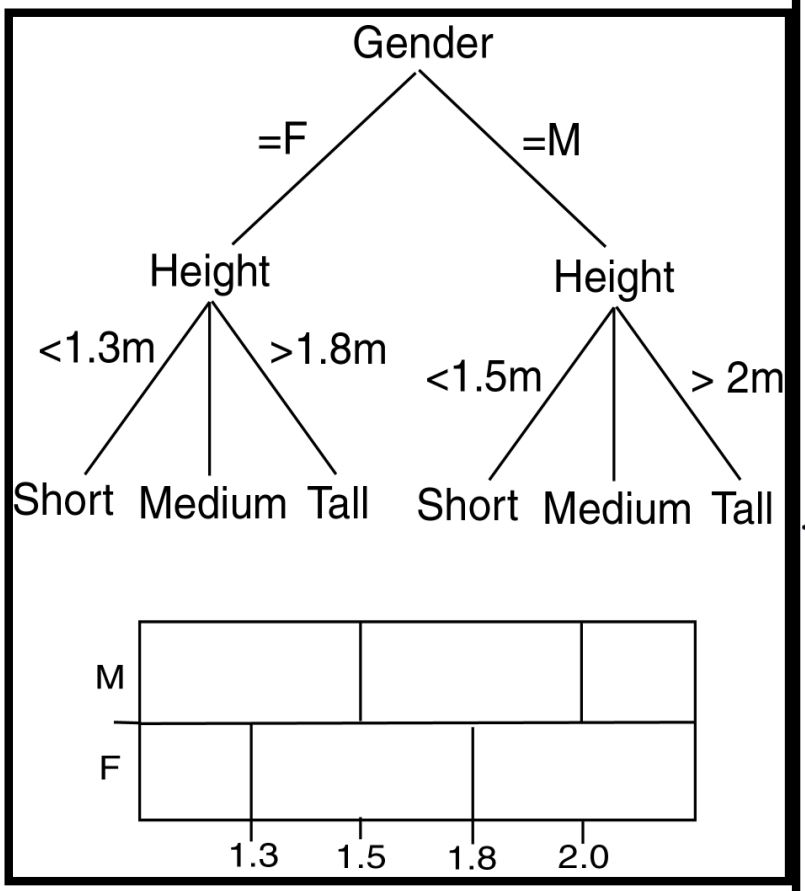


Algunos algoritmos de minería de datos basados en árboles de decisión

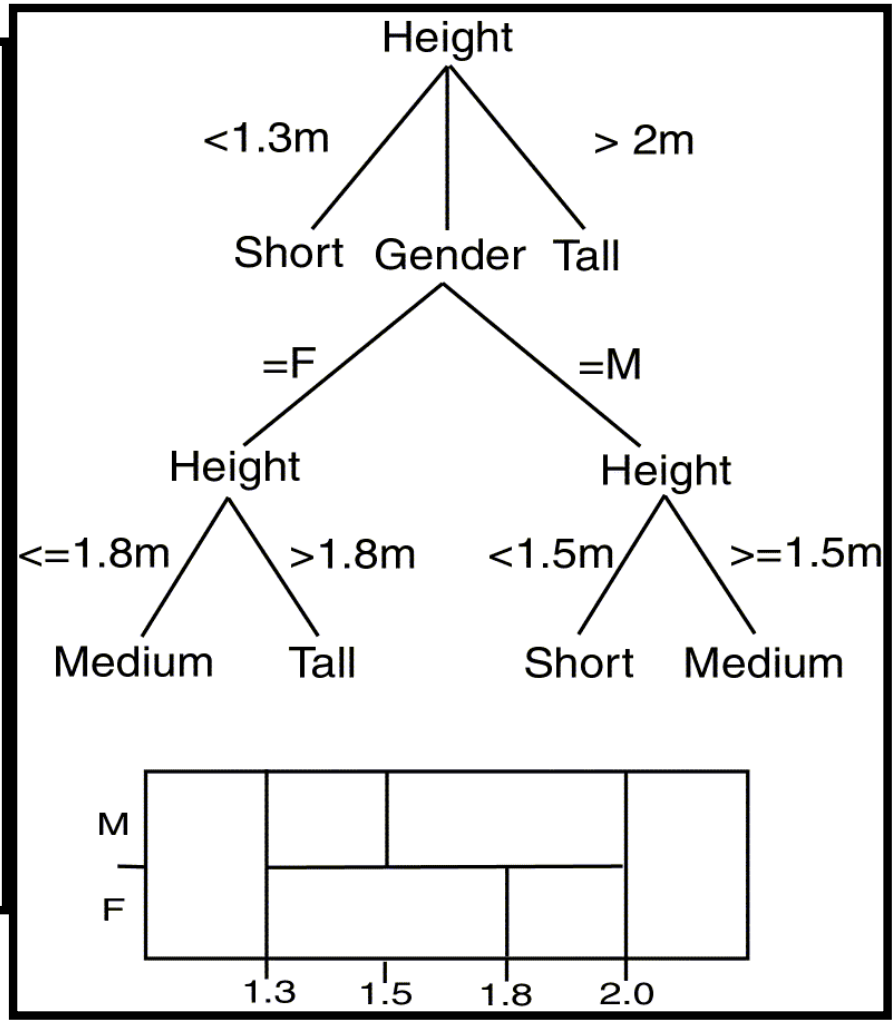
Aspectos importantes en cualquier algoritmo de MD basado en árboles de decisión:

- Elección de los atributos test: La forma en que se eligen las variables test tiene gran influencia en el rendimiento del árbol de decisión
  - En ocasiones la elección no sólo implica la revisión de los datos de entrenamiento sino también el uso de la información de expertos
- Ordenación de los atributos test. El orden en que se eligen los atributos test es importante para evitar comparaciones innecesarias
- Divisiones: Para algunos atributos y dominios las divisiones son obvias mientras que para otros es díficil de determinar
- Estructura del árbol: Para mejorar el rendimiento al aplicar el árbol de decisión es preferible un árbol balanceado
- Criterio de parada. En muchos problemas puede ser necesario parar antes para prevenir árboles muy grandes. Es un equilibrio entre precisión y rendimiento
- Ejemplos de entrenamiento. La estructura del árbol de decisión depende de los ejemplos de entrenamiento
- Poda: En ocasiones es necesario realizar un proceso de poda para mejorar el rendimiento del árbol durante la clasificación

Algunos algoritmos de minería de datos basados en árboles de decisión



Balanceado



Profundidad

Algunos algoritmos de minería de datos basados en árboles de decisión

ID3

J.R. Quinlan. Induction of Decision Trees. Machine Learning, vol. 1, pp 81-106, 1986

- Crea el árbol utilizando conceptos de teoría de información
- Intenta reducir el número de comparaciones
- ID3 elige el atributo test con máxima ganancia de información
  - Basada en la entropía que se utiliza como una medida de la cantidad de incertidumbre o sorpresa en un conjunto de datos

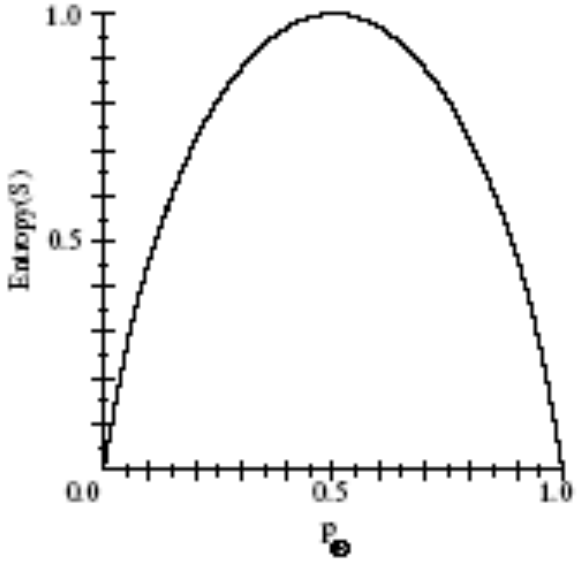
Algunos algoritmos de minería de datos basados en árboles de decisión: ID3

Definición de entropía. Dados:

- Un problema con dos clases (positiva y negativa)
- S, un conjunto de ejemplos

La entropía mide la incertidumbre en S

$$Entropia(S) = H(S) = -p_+ \cdot \log_2 p_+ - p_- \cdot \log_2 p_- =$$
$$p_+ \cdot \log_2(1/p_+) + p_- \cdot \log_2(1/p_-)$$



- Con k clases:

$$Entropia(S) = \sum_{i=1}^{k} p_i \cdot \log_2(1/p_i)$$

1.6. Algunos algoritmos de minería de datos basados en árboles de decisión: ID3

- ID3 elige el atributo con mayor ganancia de información

  - Ganancia de información: diferencia entre la cantidad de información que se necesita para hacer una clasificación antes de hacer la división y después

  - Se calcula determinando la diferencias entre la entropía del conjunto de datos de partida y la suma ponderada de las entropías una vez dividido el conjunto de ejemplos

$$Gain(S, A) = Entropia(S) - \sum_{v \in valores(A)} \frac{|S_v|}{|S|} Entropia(S_v)$$

Algunos algoritmos de minería de datos basados en árboles de decisión: ID3

Esquema algoritmo ID3

1. Seleccionar el atributo A que maximice la ganancia G(S,A)
2. Crear un nodo para ese atributo con tantos sucesores como valores tenga
3. Introducir los ejemplos en los sucesores según el valor que tenga el atributo A
4. Por cada sucesor:

   Si sólo hay ejemplos de una clase, C_k

   Entonces etiquetarlo con C_k

   Si no, llamar a ID3 con un conjunto de ejemplos formado por los ejemplos de ese nodo, eliminando la columna del atributo A

- Termina cuando todos los datos del nodo son de la misma clase y la entropía es cero

Algunos algoritmos de minería de datos basados en árboles de decisión: ID3

- El espacio de hipótesis es completo, la función objetivo está incluida en el
- Selecciona una única hipótesis
  - No es capaz de determinar todos los árboles compatibles con los ejemplos de entrenamiento
  - No puede proponer ejemplos que reduzcan el espacio de búsqueda
- No hay vueltas atrás
  - Encuentra un óptimo local que puede no ser el óptimo global (hill-climbing)
- En cada paso utiliza información estadística de todos los ejemplos en lugar de considerar los ejemplos uno a uno
  - Permite ruido en los datos de entrenamiento
- Por la ganancia de información, tiene tendencia a elegir atributos con muchos valores

Algunos algoritmos de minería de datos basados en árboles de decisión: ID3

De todos los árboles compatibles con los ejemplos de entrenamiento ¿Cuál elige ID3 ?

- Se prefieren árboles cortos frente a largos, con los atributos que producen una mayor ganancia de información cerca de la raíz

Refinamiento de ID3

- Cuándo parar en la construcción del árbol
- Atributos continuos
- Otras medidas de selección de atributos
- Atributos con coste diferente
- Ejemplos incompletos

Algunos algoritmos de minería de datos basados en árboles de decisión: C4.5

C4.5

J.R. Quinlan. C4.5: Programs for Machine Learning. San Frnacisco: Morgan Kaufmann, 1993

Mejora a ID3 en los siguientes aspectos:

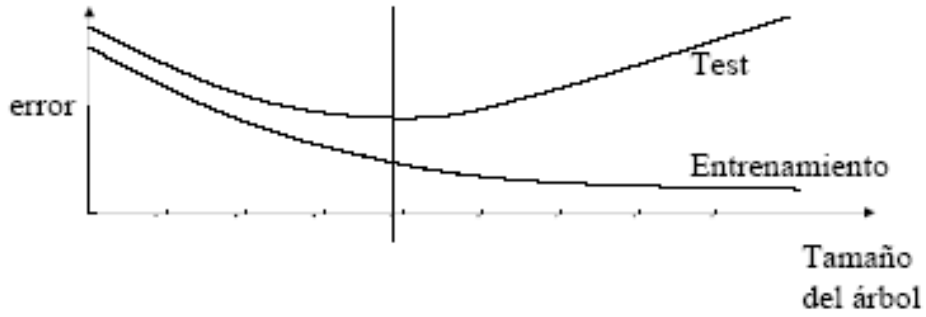
- Datos perdidos:
  - Cuando se construye el árbol, los datos perdidos se ignoran (el árbol de decisión se construye mirando sólo los registros que tienen valor para ese atributo)
  - Para clasificar un ejemplo con valor perdido, éste se predice en base a lo que se sabe sobre los valores del atributo para otros registros
- Datos contínuos: Se divide en rangos en base a los valores encontrados en el conjunto de entrenamiento
- Propone soluciones para el sobreaprendizaje. Posibilidades
  - pre-poda: se decide cuándo dejar de subdivir el árbol
  - post-poda: se construye el árbol y después se poda

Algunos algoritmos de minería de datos basados en árboles de decisión: C4.5

Pre-poda:

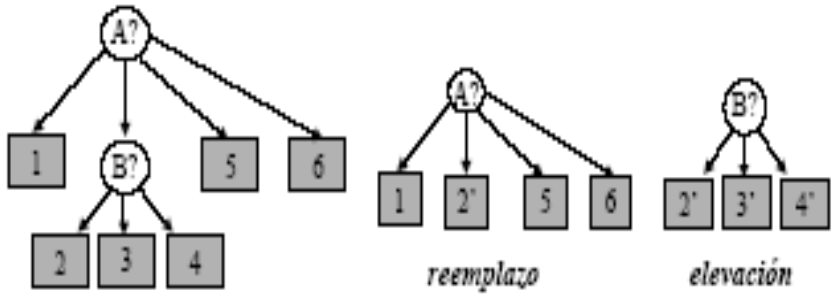
• no se divide un nodo si se tiene poca confianza en él (no es significativa la diferencia de clases), o

• se valida con un conjunto de test independiente y se para cuando la curva del conjunto de test empieza a subir



Hay dos estrategias de post- poda en C4.5

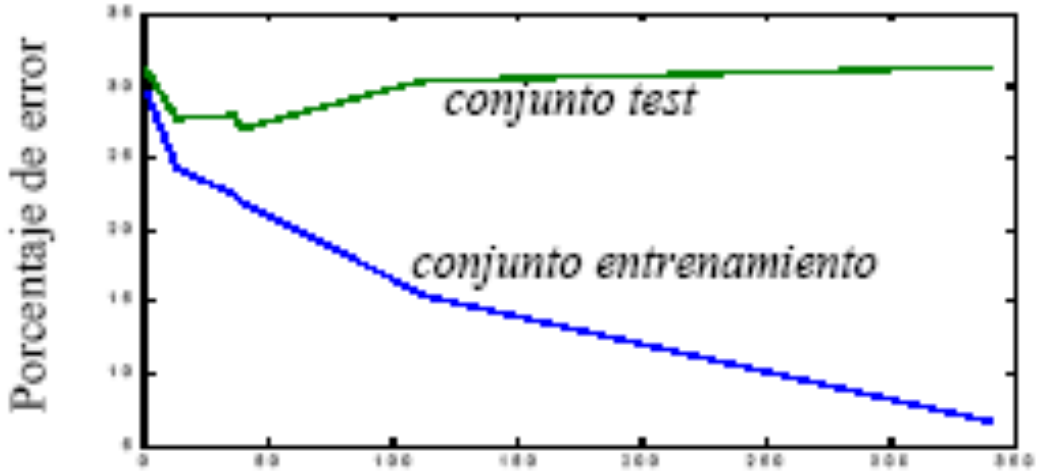
- Reemplazamiento de subárboles: Se reemplaza un subárbol por una hoja si al hacerlo el error es similar al original
- Elevación de subárbol: Reemplaza un subárbol por su subárbol más utilizado (un subárbol se mueve de su localización a un nodo superior en el árbol)



Algunos algoritmos de minería de datos basados en árboles de decisión: C4.5

Efecto del tamaño del árbol

Tamaño del árbol	Conjunto de entrenamiento		Conjunto de test	
	instancias incorrectas	porcentaje de error	instancias incorrectas	porcentaje de error
1	207	29.57 %	93	31%
13	170	24.29 %	83	27.67 %
36	157	22.43%	84	28%
39	154	22%	81	27%
95	119	17%	89	29.67%
113	108	15.43%	91	30.3%
340	47	6.71%	94	31.3%



Algunos algoritmos de minería de datos basados en árboles de decisión: C4.5

- Reglas: C4.5 permite clasificación mediante el árbol o a través de las reglas que se generan a partir de él
  - Además, incluye técnicas para simplificar las reglas
- Selección de atributos:
  - ID3 favorece atributos con muchas divisiones
  - En C4.5 se utiliza como criterio de selección el ratio de ganancia de información que tiene en cuenta la cardinalidad de cada división

$$GainRatio(S,A) = \frac{Gain(S,A)}{H\left(\frac{|S_1|}{|S|},\ldots,\frac{|S_v|}{|S|}\right)}$$

$$v \in Valores\ de\ A$$

Algunos algoritmos de minería de datos basados en árboles de decisión: C4.5

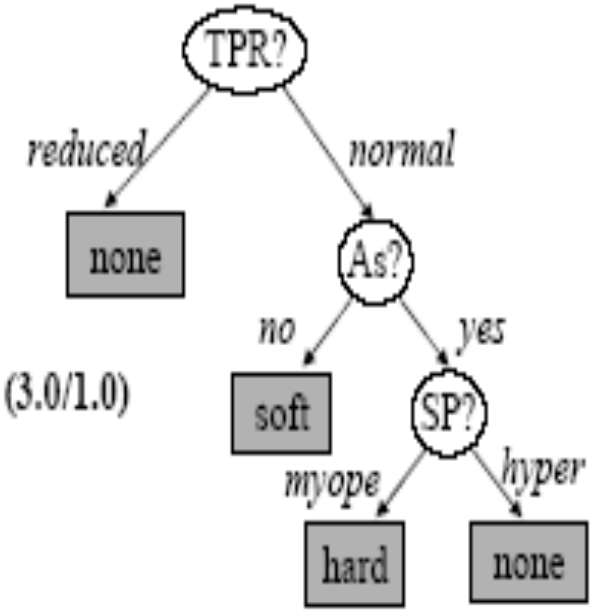
- Ejemplo de poda por número de ejemplos. Árbol original

```
tear-prod-rate = reduced: none (12.0)
tear-prod-rate = normal
|  astigmatism = no
|  |  age = young: soft (2.0)
|  |  age = pre-presbyopic: soft (2.0)
|  |  age = presbyopic
|  |  |  spectacle-prescrip = myope: none (1.0)
|  |  |  spectacle-prescrip = hypermetrope: soft (1.0)
|  astigmatism = yes
|  |  spectacle-prescrip = myope: hard (3.0)
|  |  spectacle-prescrip = hypermetrope
|  |  |  age = young: hard (1.0)
|  |  |  age = pre-presbyopic: none (1.0)
|  |  |  age = presbyopic: none (1.0)
```

Algunos algoritmos de minería de datos basados en árboles de decisión: C4.5

2 ejemplos por hoja

```
tear-prod-rate = reduced: none (12.0)
tear-prod-rate = normal
|  astigmatism = no: soft (6.0/1.0)
|  astigmatism = yes
|  |  spectacle-prescrip = myope: hard (3.0)
|  |  spectacle-prescrip = hypermetrope: none (3.0/1.0)
```



Clasificadores basados en métodos bayesianos

- Los métodos probabilísticos/bayesianos representan la incertidumbre asociada a los procesos de forma natural

  → Una gran ventaja sobre otros métodos

  Ejemplo: Supongamos que consultamos a un sistema de recomendaciones (SR) para invertir en bolsa sobre los productos P1 y P2

  - Si el modelo utilizado por el SR no trata la incertidumbre (p.e. un árbol de decisión), podríamos obtener:

    (P1, invertir) (P2, invertir)

    por lo que podríamos diversificar la cantidad a invertir entre los productos

  - Si el modelo utilizado por el SR trata la incertidumbre (p.e. una red bayesiana) podríamos obtener:

    (P1, invertir, prob=0.9) (P2, invertir, prob=0.52)

    (P1, no invertir, prob=0.1) (P2, no invertir, prob=0.48)

    por lo que repartir (al menos a partes iguales) la inversión entre ambos productos no parece lo más razonable

Teorema de Bayes e hipótesis MAP

- El teorema de Bayes orientado a un problema de clasificación con n variables tiene la siguiente expresión

$$P(C|A_1,\ldots,A_n) = \frac{P(A_1,\ldots,A_n|C)P(C)}{P(A_1,\ldots,A_n)}$$

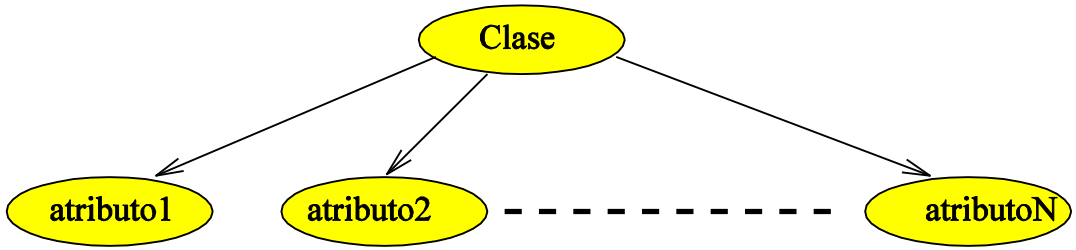
- Hipótesis MAP (máxima a posteriori): Si queremos clasificar un nuevo caso $(a_1,\ldots a_n)$ y la variable clase C tiene k posibles categorías $\Omega_C=\{c_1,\ldots,c_k\}$, lo que nos interesa es identificar la más probable y devolverla como clasificación

$$\begin{aligned} c_{MAP} &= \arg\max_{c\in\Omega_C} P(c|a_1,\ldots,a_n) \\ &= \arg\max_{c\in\Omega_C} \frac{P(a_1,\ldots,a_n|c)P(c)}{P(a_1,\ldots,a_n)} \\ &= \arg\max_{c\in\Omega_C} P(a_1,\ldots,a_n|c)P(c) \end{aligned}$$

- Problema: Hay que trabajar con la distribución conjunta y eso normalmente es inmanejable

Clasificador Naïve Bayes

- Es el modelo de red bayesiana orientada a clasificación más simple
- Supone que todos los atributos son independientes conocida la variable clase. Gráficamente



- En un Naïve BAyes (NB) la hipótesis MAP queda como:

$$c_{MAP} = \arg\max_{c \in \Omega_C} P(c|a_1,\ldots,a_n) = \arg\max_{c \in \Omega_C} P(c)\prod_{i=1}^{n} P(a_i|c)$$

- A pesar de la suposición poco realista realizada en el NB, este algoritmo se considera un estándar y sus resultados son competitivos con la mayoría de los clasificadores

Clasificador Naïve Bayes

¿Cómo se estiman estas probabilidades? Estimación de párametros

- Variables discretas
  - $P(x|c_i)$ se estima como la frecuencia relativa de ejemplos que teniendo un determinado valor de x pertenecen a la clase c_i
  - Estimación por máxima verisimilitud (EMV). Sea $n(x_i)$ el nº de veces que aparece $X_i=x_i$ en la BD y $n(x_i,x_j)$ el nº de veces que aparece el par $(X_i=x_i,X_j=x_j)$ en la BD, entonces

$$p(x_i|x_j) = \frac{n(x_i,x_j)}{n(x_j)}$$

  - Suavizando por la corrección de Laplace:

$$p(x_i|x_j) = \frac{n(x_i,x_j)+1}{n(x_j)+|\Omega_{X_i}|}$$

Cuando hay muchos casos, tiende a la EMV, si hay pocos casos tiende a la uniforme

Clasificador Naïve Bayes

Ejemplo:

Day	Outlook	Temperature	Humidity	Wind	*Play Tennis?*
1	Sunny	85	85	Light	No
2	Sunny	80	90	Strong	No
3	Overcast	83	86	Light	Yes
4	Rain	70	96	Light	Yes
5	Rain	68	80	Light	Yes
6	Rain	65	70	Strong	No
7	Overcast	64	65	Strong	Yes
8	Sunny	72	95	Light	No
9	Sunny	69	70	Light	Yes
10	Rain	75	80	Light	Yes
11	Sunny	75	70	Strong	Yes
12	Overcast	72	90	Strong	Yes
13	Overcast	81	75	Light	Yes
14	Rain	71	91	Strong	No

- Estimación para *PLAY, outlook y windy*

PLAY	EMV	Lapl.
no	5/14	6/16
yes	9/14	10/16

outlook	EMV		Lapl.	
	no	yes	no	Yes
sunny	3/5	2/9	4/8	3/12
overcast	0	4/9	1/8	5/12
rainy	2/5	3/9	3/8	4/12

windy	EMV		Lapl.	
	no	yes	no	Yes
true	3/5	3/9	4/7	4/11
false	2/5	6/9	3/7	7/11

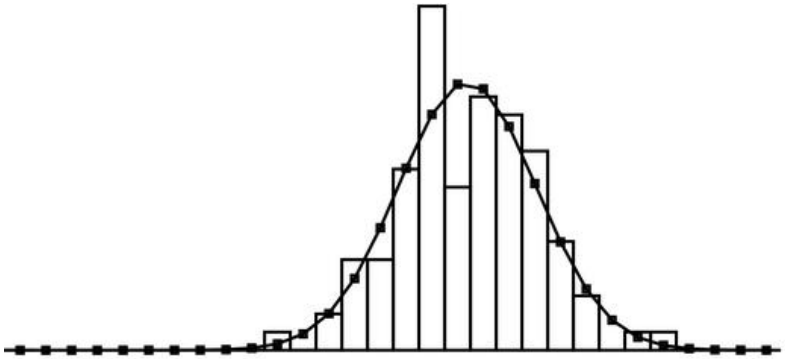
Clasificador Naïve Bayes

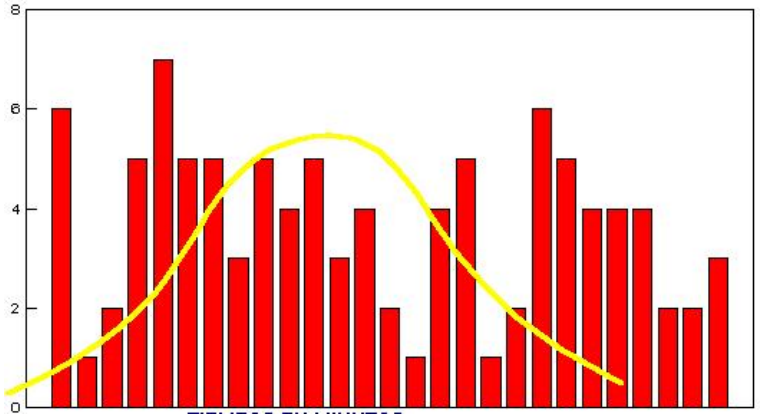
- Variables numéricas
  - $P(x|c_i)$ se estima mediante una función de densidad gaussiana. Es decir, se asume que los valores numéricos siguen una distribución normal

$$P(x|c_i) = N(x;\mu,\sigma) = \frac{1}{\sqrt{2\pi\sigma}} e^{\frac{(x-\mu)^2}{2\sigma^2}}$$

    es decir, para cada categoría de la variable clase se estima una distribución normal (de media μ y desviación estándar σ)

  - Evidentemente, en unos casos la aproximación realizada será mejor que en otros





Clasificador Naïve Bayes

Ejemplo:

Day	Outlook	Temperature	Humidity	Wind	*Play Tennis?*
1	Sunny	85	85	Light	No
2	Sunny	80	90	Strong	No
3	Overcast	83	86	Light	Yes
4	Rain	70	96	Light	Yes
5	Rain	68	80	Light	Yes
6	Rain	65	70	Strong	No
7	Overcast	64	65	Strong	Yes
8	Sunny	72	95	Light	No
9	Sunny	69	70	Light	Yes
10	Rain	75	80	Light	Yes
11	Sunny	75	70	Strong	Yes
12	Overcast	72	90	Strong	Yes
13	Overcast	81	75	Light	Yes
14	Rain	71	91	Strong	No

- Estimación para *temperature* y *humidity*
  - temperature=(PLAY=no): $\mu=74.6$; $\sigma=7.89$
  - temperature=(PLAY=yes): $\mu=73$; $\sigma=6.16$
  - humidity=(PLAY=no): $\mu=86.6$; $\sigma=9.73$
  - humidity=(PLAY=yes): $\mu=79.11$; $\sigma=10.21$

Clasificador Naïve Bayes

- Supongamos que queremos clasificar un nuevo caso:
  x=(outlook=sunny, temp=87, hum=90, windy =false)
- Suponemos la estimación por máxima verisimilitud. Aplicamos para ambas categorías de la variable clase:
  - PLAY=no

$P(x|NO) = P(NO)\cdot P_O(sunny|NO)\cdot P_w(false|NO)\cdot P_t(87|NO)\cdot P_h(90|NO)$
$= 0.36\cdot 0.6\cdot 0.4\cdot N_{temp}(87;74.6,7.89)\cdot N_{hum}(90;86.6,9.73)$
$= 0.36\cdot 0.6\cdot 0.4\cdot 0.014706\cdot 0.038573 = 4.08e\text{-}05$

  - PLAY=yes

$P(x|YES) = P(NO)\cdot P_O(sunny|YES)\cdot P_w(false|YES)\cdot P_t(87|YES)\cdot P_h(90|YES)$
$= 0.64\cdot 0.22\cdot 0.67\cdot Ntemp(87;73,6.16)\cdot Nhum(90;79.11,10.21)$
$=0.64\cdot 0.22\cdot 0.67\cdot 0.004894\cdot 0.022123 = 1.02e\text{-}05$

- Normalizando:

P(NO)=0.8 P(YES) = 0.2

Clasificador Naïve Bayes

Ejemplo de Naïve Bayes en WEKA

- Base de datos iris.arff (150 registros)
  - 4 atributos: sepalLength(numérico), sepalWidth (numérico), petalLength(numérico), petalWidth
  - Objetivo: a partir de las medidas anteriores, clasificar el objeto como un tipo de flor: iris-setosa, iris-versicolour, iris-virginica

  - Utilizamos Naïve Bayes (con opciones por defecto) en WEKA (10cv)

    Class Iris-setosa: Prior probability = 0.33
    sepallength: Normal Distribution. Mean = 4.9913 StandardDev = 0.355
    sepalwidth: Normal Distribution. Mean = 3.4015 StandardDev = 0.3925
    petallength: Normal Distribution. Mean = 1.4694 StandardDev = 0.1782
    petalwidth: Normal Distribution. Mean = 0.2743 StandardDev = 0.1096

    Class Iris-versicolor: Prior probability = 0.33
    sepallength: Normal Distribution. Mean = 5.9379 StandardDev = 0.5042
    sepalwidth: Normal Distribution. Mean = 2.7687 StandardDev = 0.3038
    petallength: Normal Distribution. Mean = 4.2452 StandardDev = 0.4712
    petalwidth: Normal Distribution. Mean = 1.3097 StandardDev = 0.1915

    Class Iris-virginica: Prior probability = 0.33
    sepallength: Normal Distribution. Mean = 6.5795 StandardDev = 0.6353
    sepalwidth: Normal Distribution. Mean = 2.9629 StandardDev = 0.3088
    petallength: Normal Distribution. Mean = 5.5516 StandardDev = 0.5529
    petalwidth: Normal Distribution. Mean = 2.0343 StandardDev = 0.2646
    Correctly Classified Instances 144 96%

Clasificador Naïve Bayes

- Supongamos que discretizamos la BD anterior, utilizando la discretización en 2 intervalos de igual anchura

- Utilizando Naïve Bayes (opciones por defecto) en WEKA (con 10cv)

Class Iris-setosa: Prior probability = 0.33
sepallength: Discrete Estimator. Counts = 51 1 (Total = 52)
sepalwidth: Discrete Estimator. Counts = 19 33 (Total = 52)
petallength: Discrete Estimator. Counts = 51 1 (Total = 52)
petalwidth: Discrete Estimator. Counts = 51 1 (Total = 52)

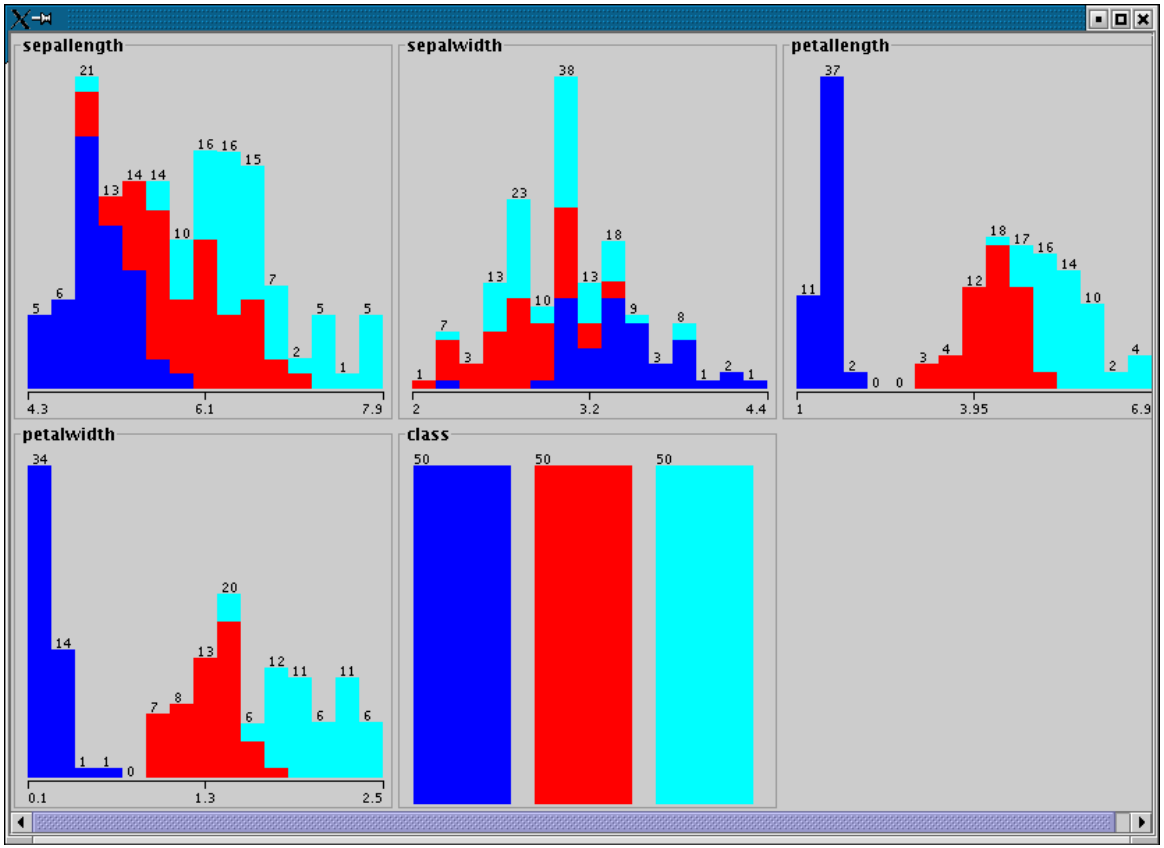
Class Iris-versicolor: Prior probability = 0.33
sepallength: Discrete Estimator. Counts = 35 17 (Total = 52)
sepalwidth: Discrete Estimator. Counts = 49 3 (Total = 52)
petallength: Discrete Estimator. Counts = 12 40 (Total = 52)
petalwidth: Discrete Estimator. Counts = 29 23 (Total = 52)

Class Iris-virginica: Prior probability = 0.33
sepallength: Discrete Estimator. Counts = 12 40 (Total = 52)
sepalwidth: Discrete Estimator. Counts = 43 9 (Total = 52)
petallength: Discrete Estimator. Counts = 1 51 (Total = 52)
petalwidth: Discrete Estimator. Counts = 1 51 (Total = 52)

Correctly Classified Instances 116 77.3333%

Clasificador Naïve Bayes

- Vamos a poner un poco más de atención en la discretización.
- Incluso utilizando la misma técnica, ¿es dos el número adecuado de intervalos?
- Para verlo, realizamos un análisis exploratorio:



Clasificador Naïve Bayes

- Parece que 3 es un número más adecuado de intervalos
- Realizamos la discretización y clasificación utilizando Naïve Bayes

Class Iris-setosa: Prior probability = 0.33
sepallength: Discrete Estimator. Counts = 48 4 1 (Total = 53)
sepalwidth: Discrete Estimator. Counts = 3 19 31 (Total = 53)
petallength: Discrete Estimator. Counts = 51 1 1 (Total = 53)
petalwidth: Discrete Estimator. Counts = 51 1 1 (Total = 53)

Class Iris-versicolor: Prior probability = 0.33
sepallength: Discrete Estimator. Counts = 12 24 17 (Total = 53)
sepalwidth: Discrete Estimator. Counts = 35 16 2 (Total = 53)
petallength: Discrete Estimator. Counts = 1 45 7 (Total = 53)
petalwidth: Discrete Estimator. Counts = 1 50 2 (Total = 53)

Class Iris-virginica: Prior probability = 0.33
sepallength: Discrete Estimator. Counts = 2 11 40 (Total = 53)
sepalwidth: Discrete Estimator. Counts = 22 25 6 (Total = 53)
petallength: Discrete Estimator. Counts = 1 2 50 (Total = 53)
petalwidth: Discrete Estimator. Counts = 1 6 46 (Total = 53)

Correctly Classified Instances 141 94 %

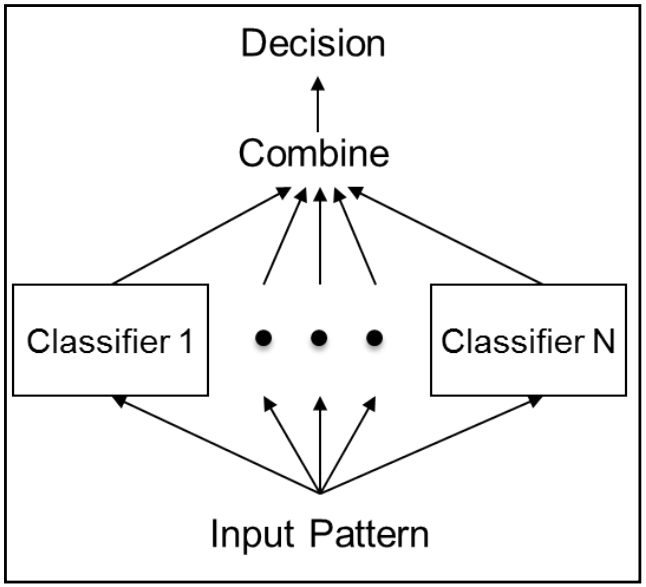
Clasificador Naïve Bayes

- Ventajas:
  - Es fácil de implementar
  - Obtiene buenos resultados en gran parte de los casos

- Desventajas:
  - Asumir que las variables tienen independencia condicional respecto a la clase lleva a una falta de precisión
  - En la práctica, existen dependencias entre las variables. P.e.: en datos hospitalarios:
    - Perfil: edad, historia familiar, etc.
    - Síntomas: fiebre, tos, etc.
    - Enfermedad: cáncer de pulmón, diabetes, etc.
  - Con un clasificador Naïve Bayes no se pueden modelar estas dependencias
  - Solución: Redes de creencia bayesianas, que combinan razonamiento bayesiano con relaciones causales entre los atributos

Ensemble Learning

Aprendizaje de múltiples clasificadores para reforzar el proceso de toma de decisión: Bagging, Boosting y Binarización



Rationale for Ensemble Learning

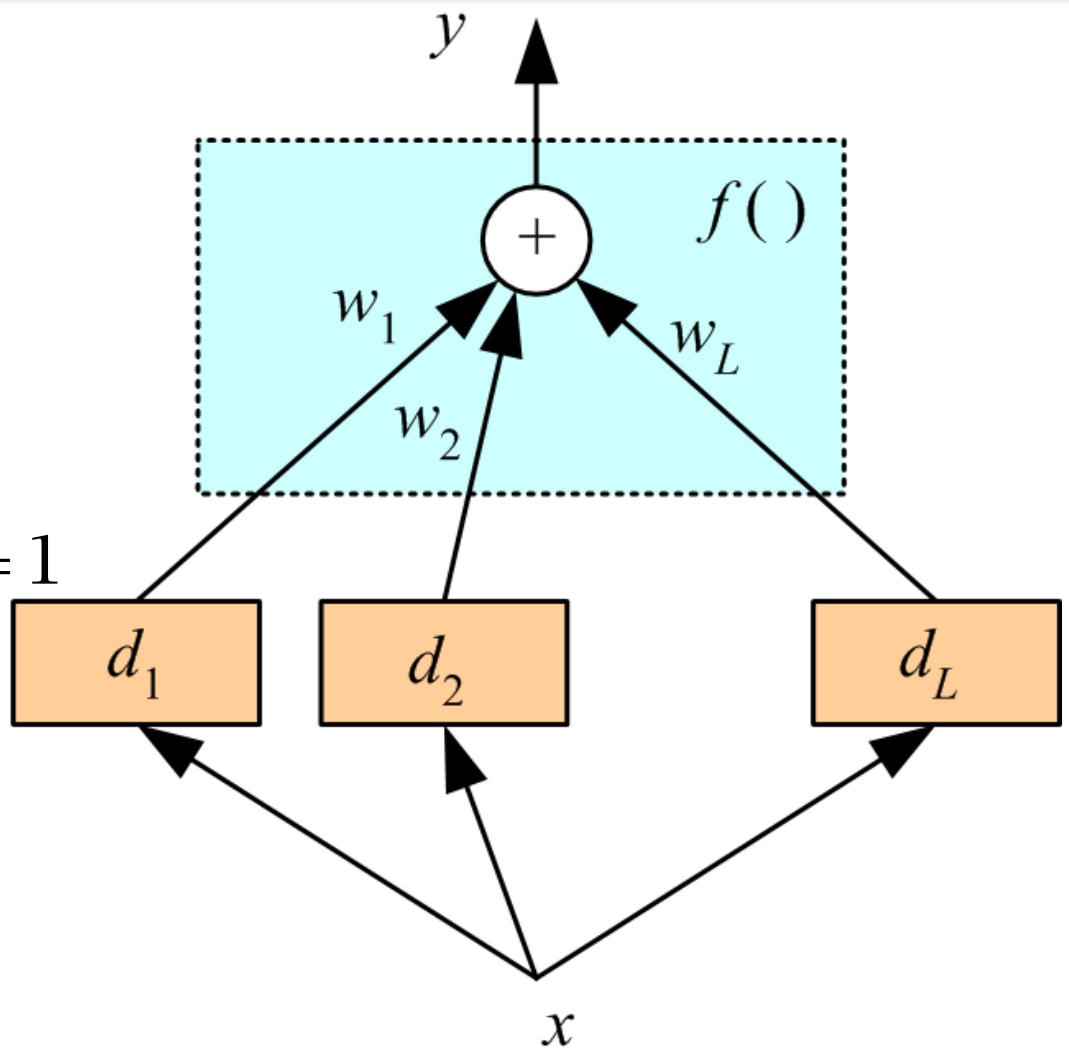
- No Free Lunch theorem: There is no algorithm that is always the most accurate
- Generate a group of base-learners which when combined have higher accuracy
- **Diversity** and **Accuracy** among classifiers are the key points for the success of ensembles.
- Different learners use different
  - Algorithms / Parameters
  - Training sets / Subproblems

Voting

- Linear combination

$$y = \sum_{j=1}^{L} w_j d_j$$

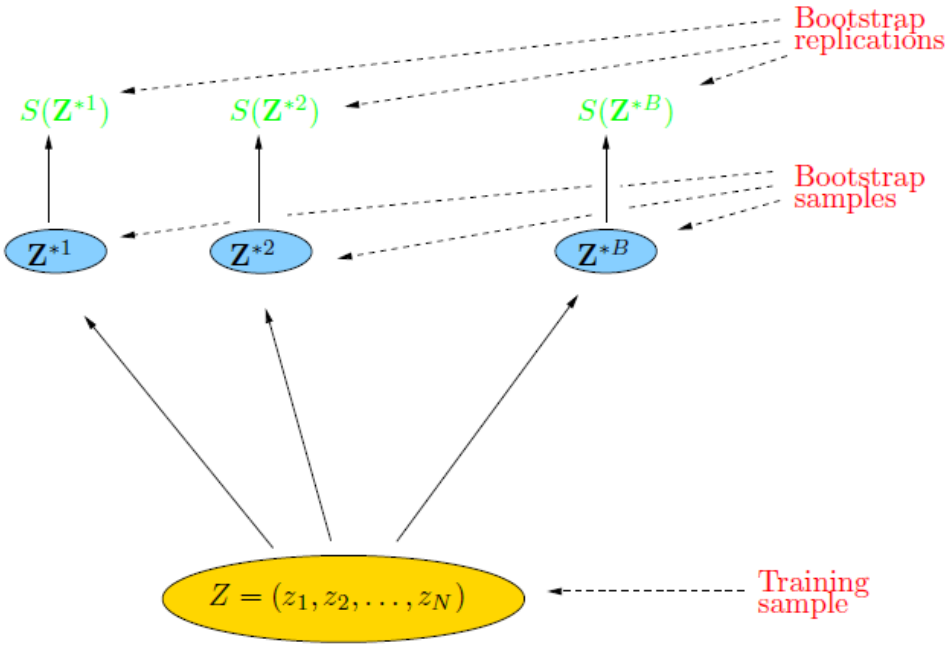
$$w_j \geq 0 \text{ and } \sum_{j=1}^{L} w_j = 1$$



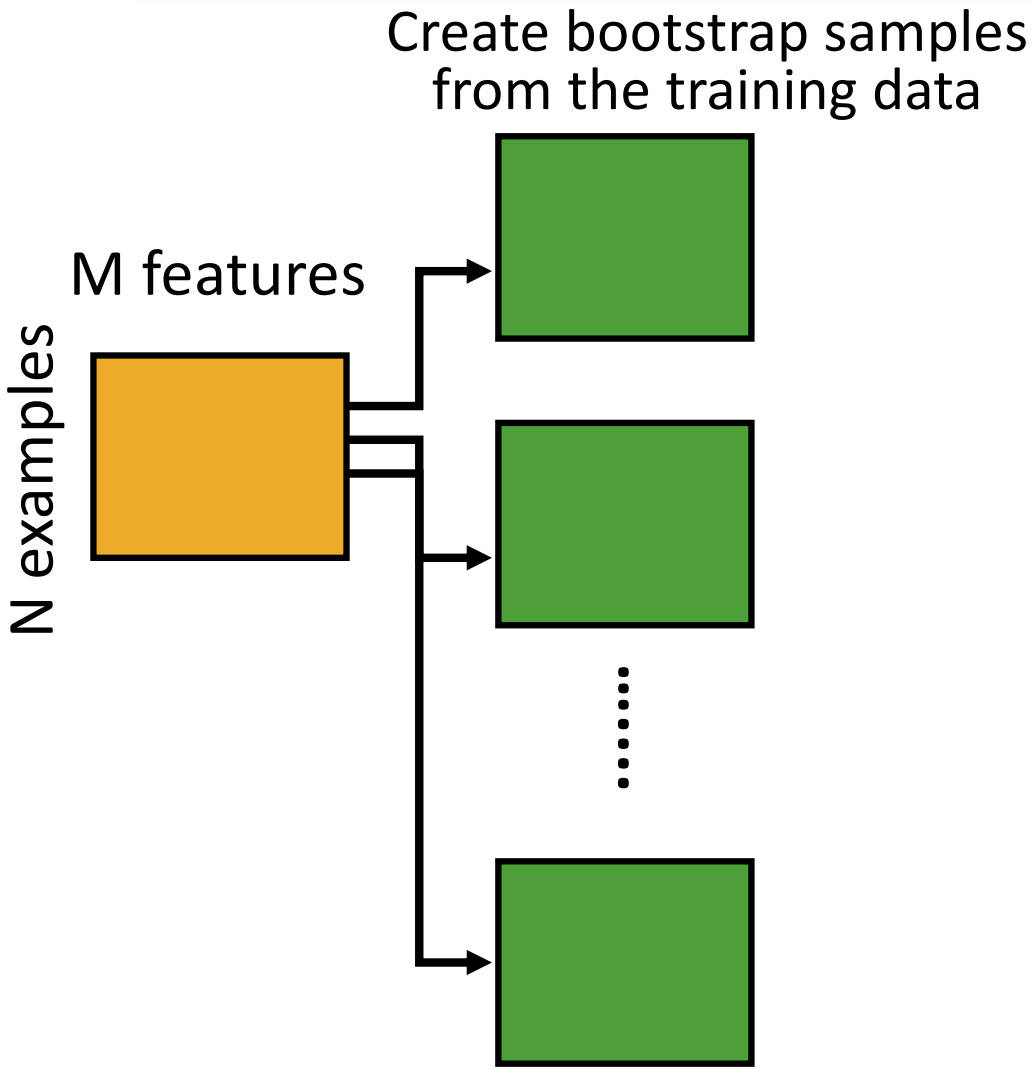
Bootstrap

The basic idea:

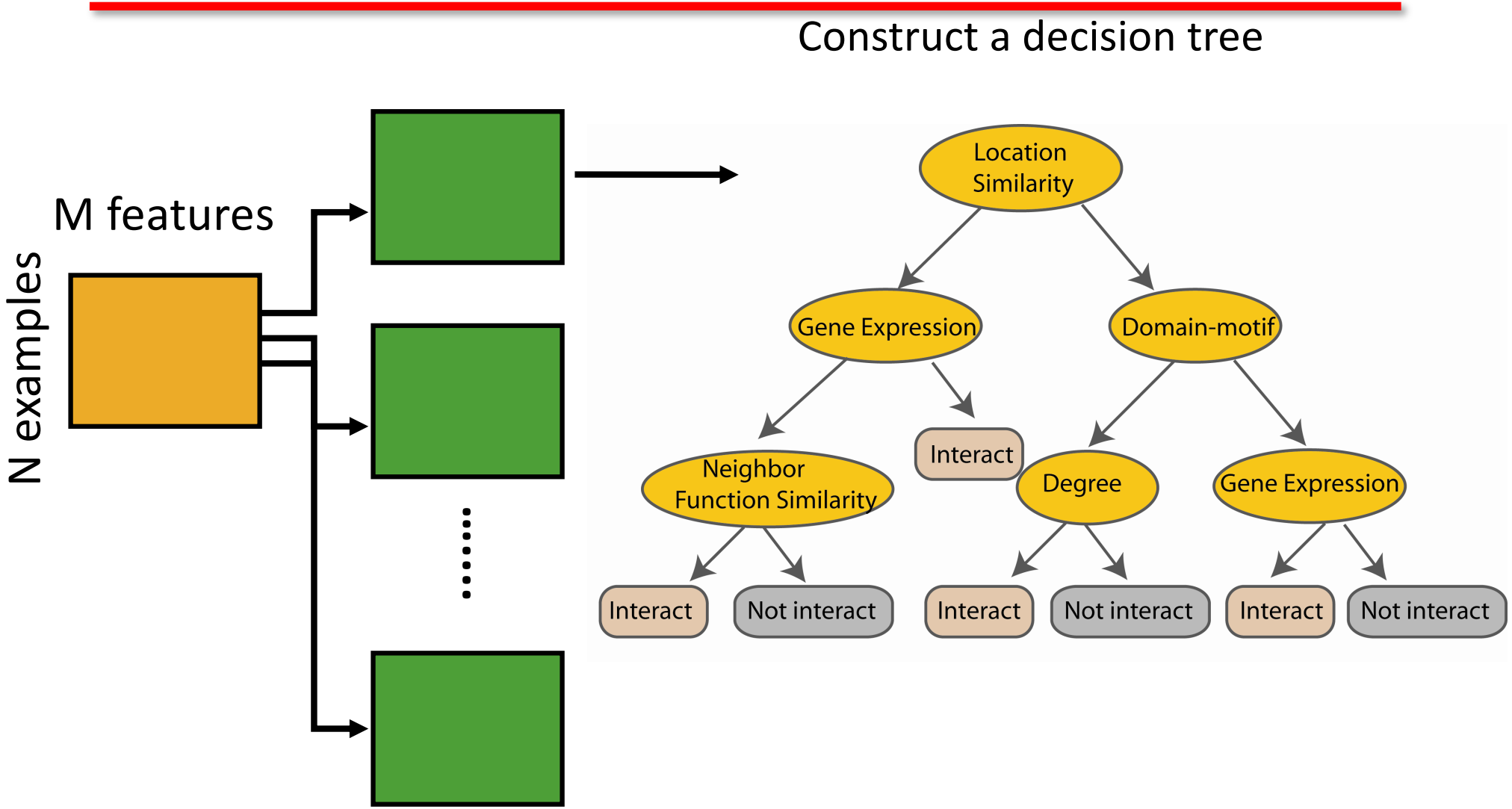
randomly draw datasets *with replacement* from the
training data, each sample *the same size as the original training set*



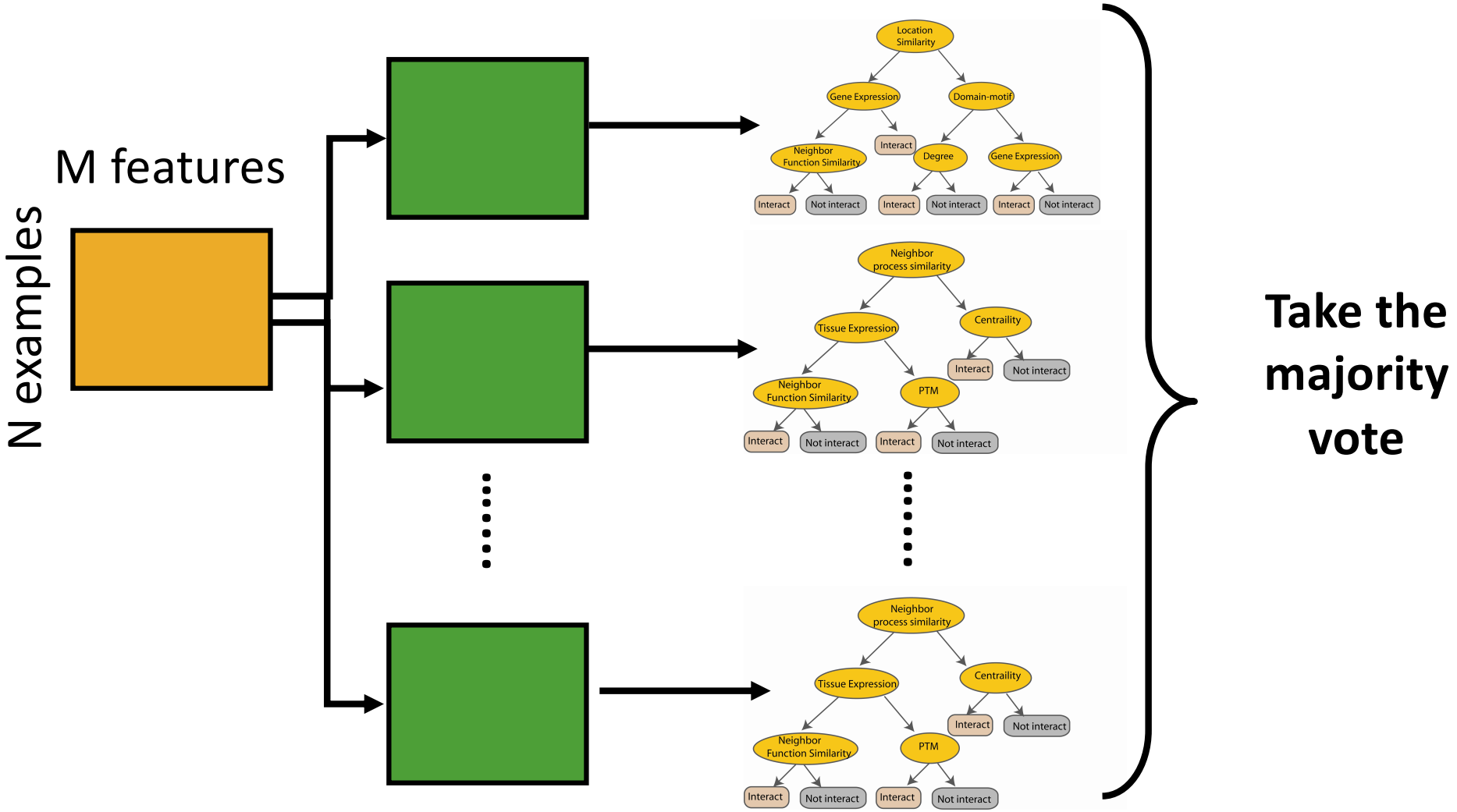
Bagging

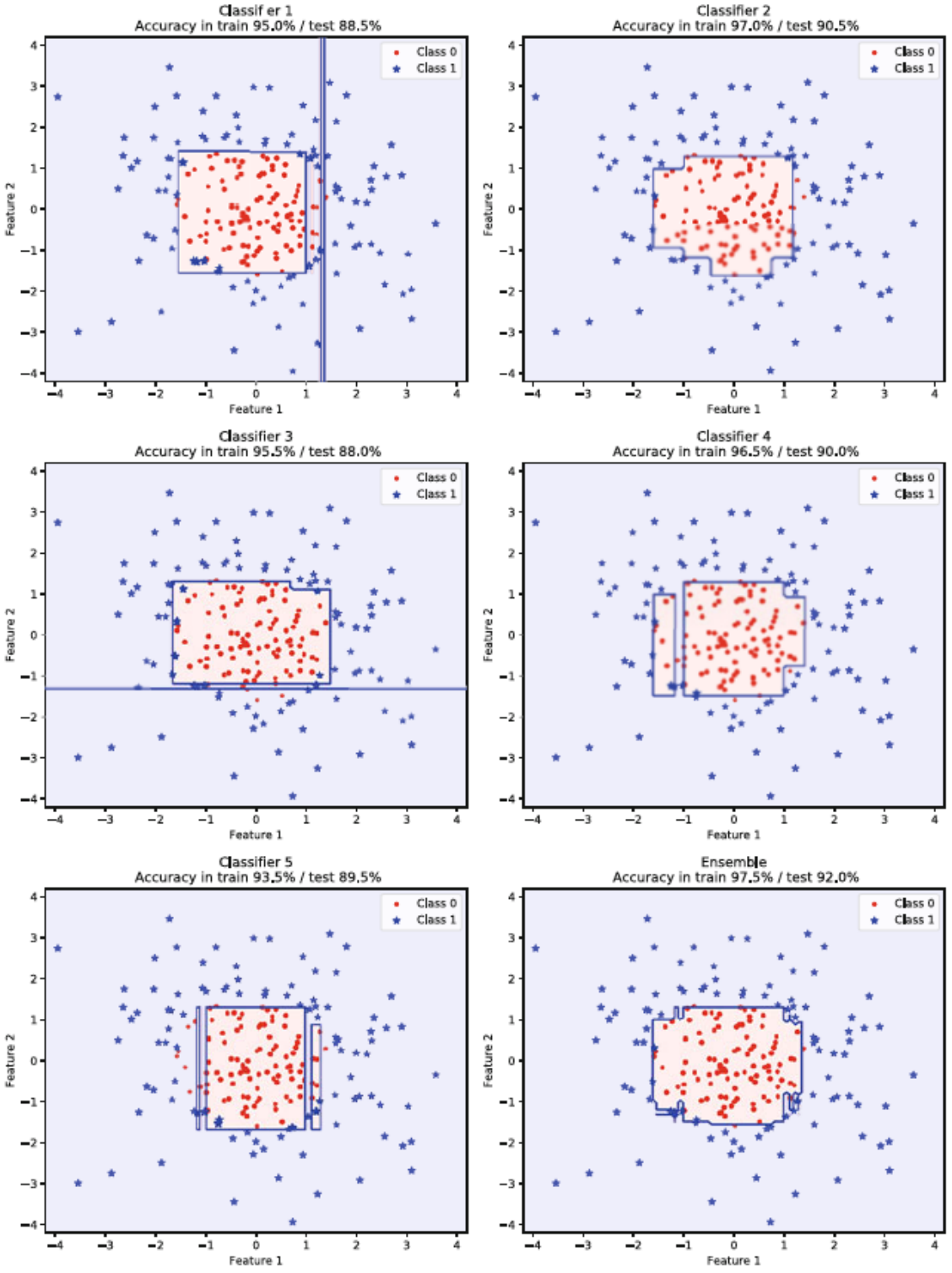

Create bootstrap samples
from the training data
M features
N examples

Random Forest Classifier



Random Forest Classifier

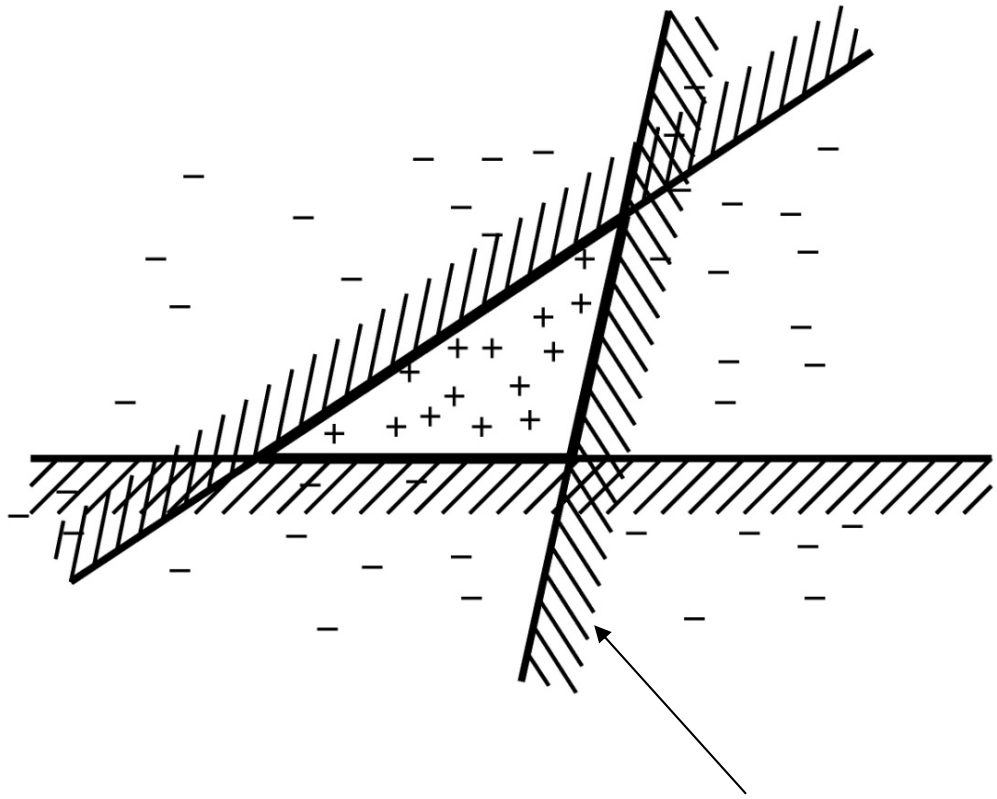



Classif er 1
Accuracy in train 95.0% / test 88.5%
Classifier 2
Accuracy in train 97.0% / test 90.5%
Classifier 3
Accuracy in train 95.5% / test 88.0%
Classifier 4
Accuracy in train 96.5% / test 90.0%
Classifier 5
Accuracy in train 93.5% / test 89.5%
Ensemble
Accuracy in train 97.5% / test 92.0%
Class 0
Class 1
Feature 1
Feature 2

Boosting

- Train classifiers (e.g. decision trees) in a sequence.
- A new classifier should focus on those cases which were incorrectly classified in the last round.
- Combine the classifiers by letting them vote on the final prediction (like bagging).
- Each classifier is “weak” but the ensemble is “strong.”
- AdaBoost is a specific boosting method.

Example

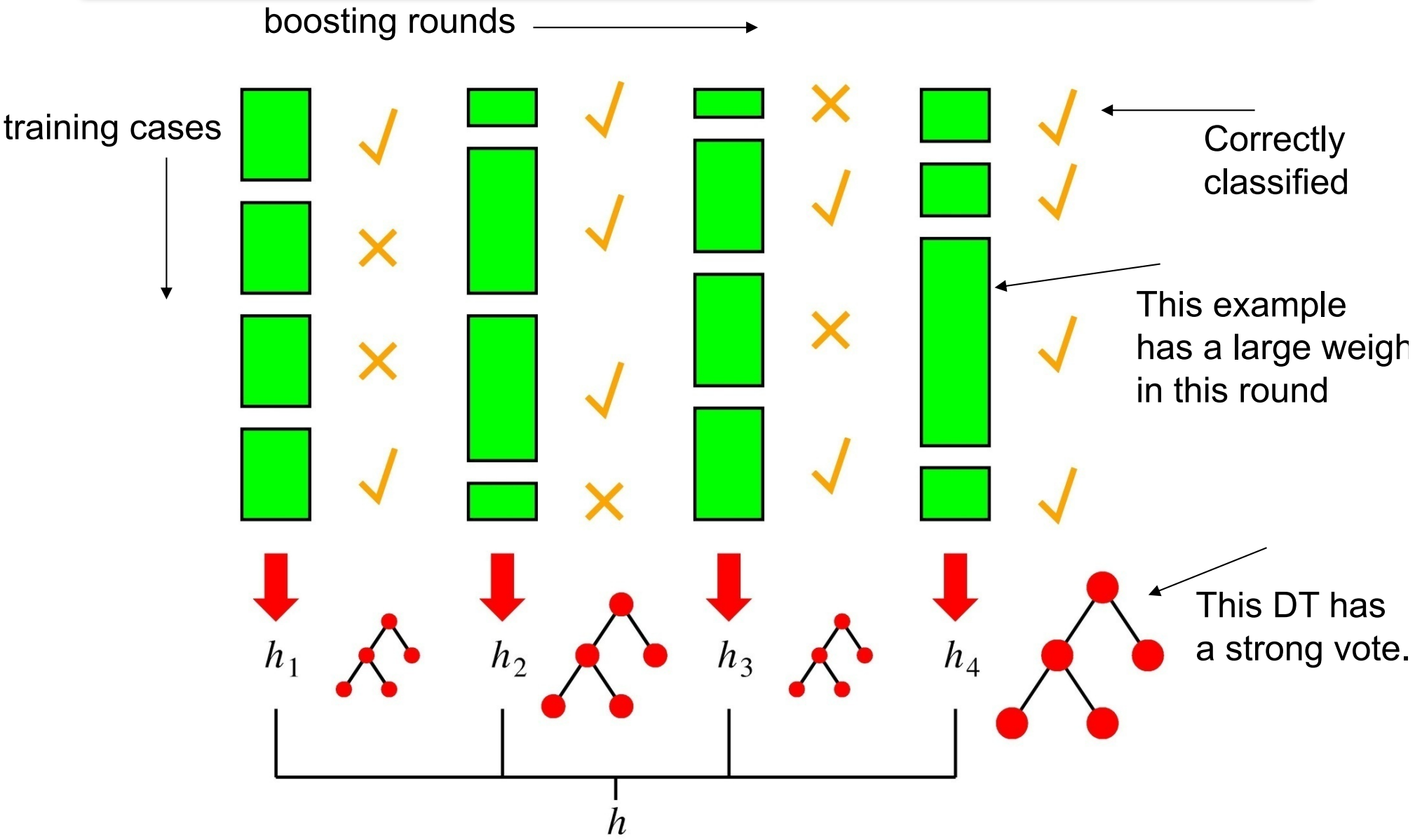


This line is one simple classifier saying that everything to the left + and everything to the right is -

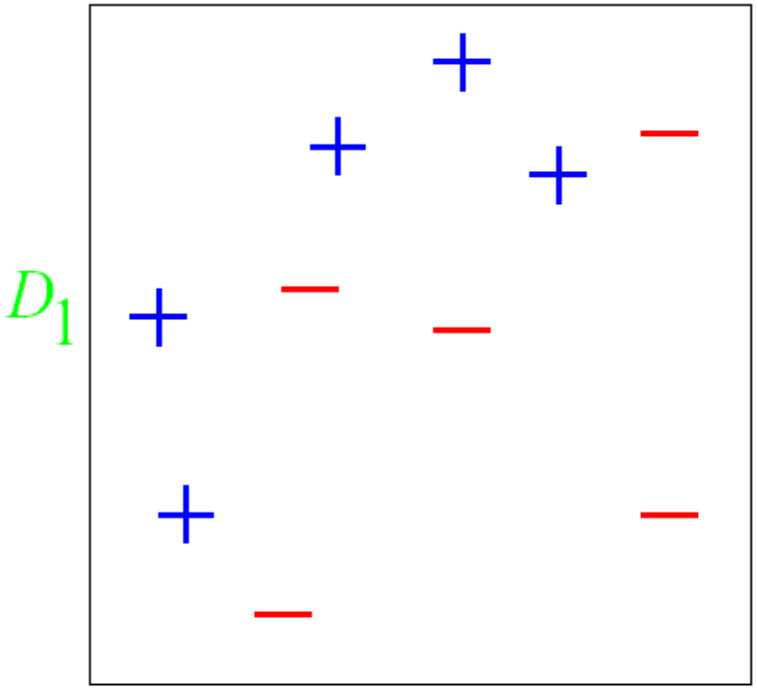
Boosting Intuition

- We adaptively weight each data case.
- Data cases which are wrongly classified get high weight (the algorithm will focus on them)
- Each boosting round learns a new (simple) classifier on the weighed dataset.
- These classifiers are weighed to combine them into a single powerful classifier.
- Classifiers that obtain low training error rate have high weight.

Boosting in a Picture



And in animation



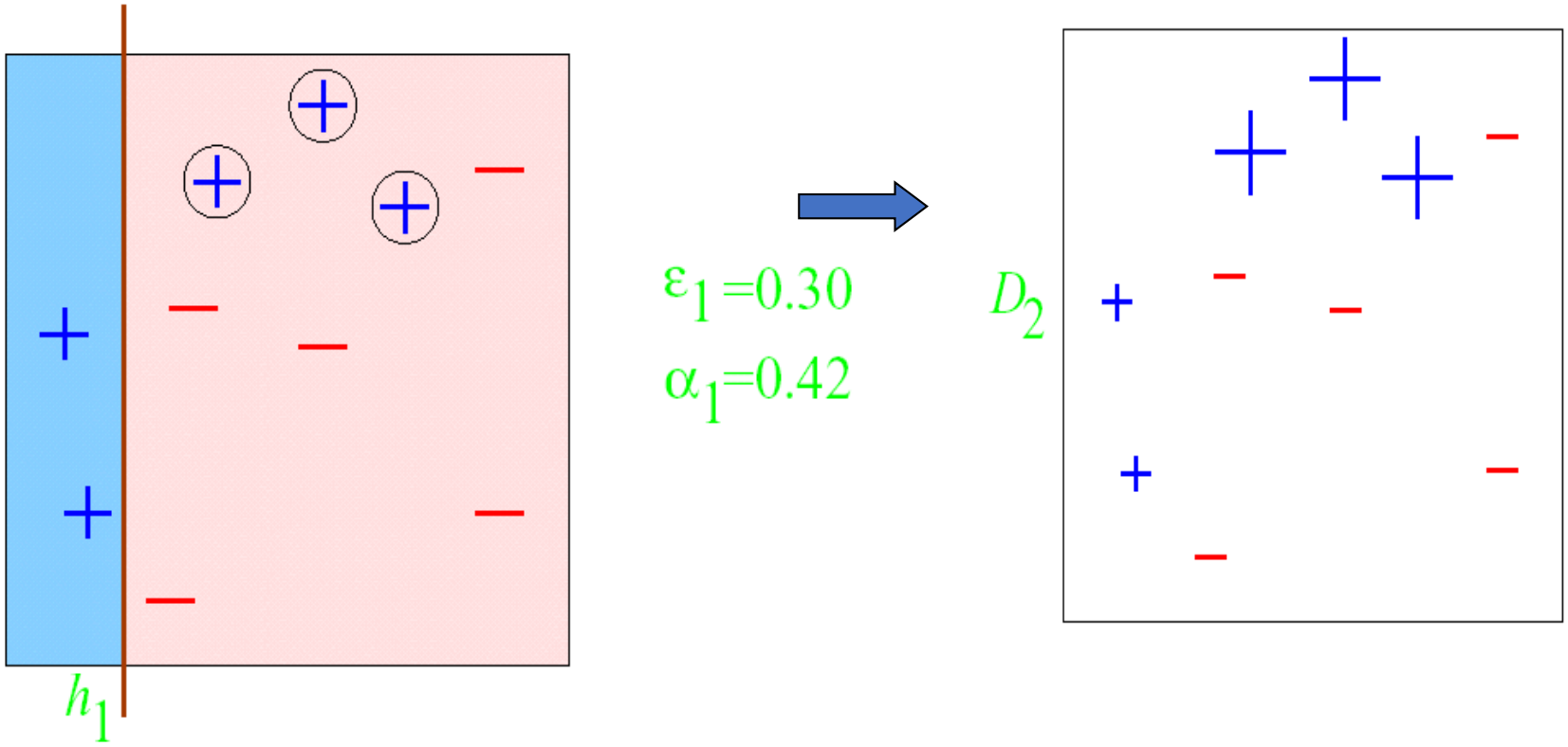
Original training set: equal weights to all training samples

Taken from "**A Tutorial on Boosting**" **by** Yoav Freund and Rob Schapire

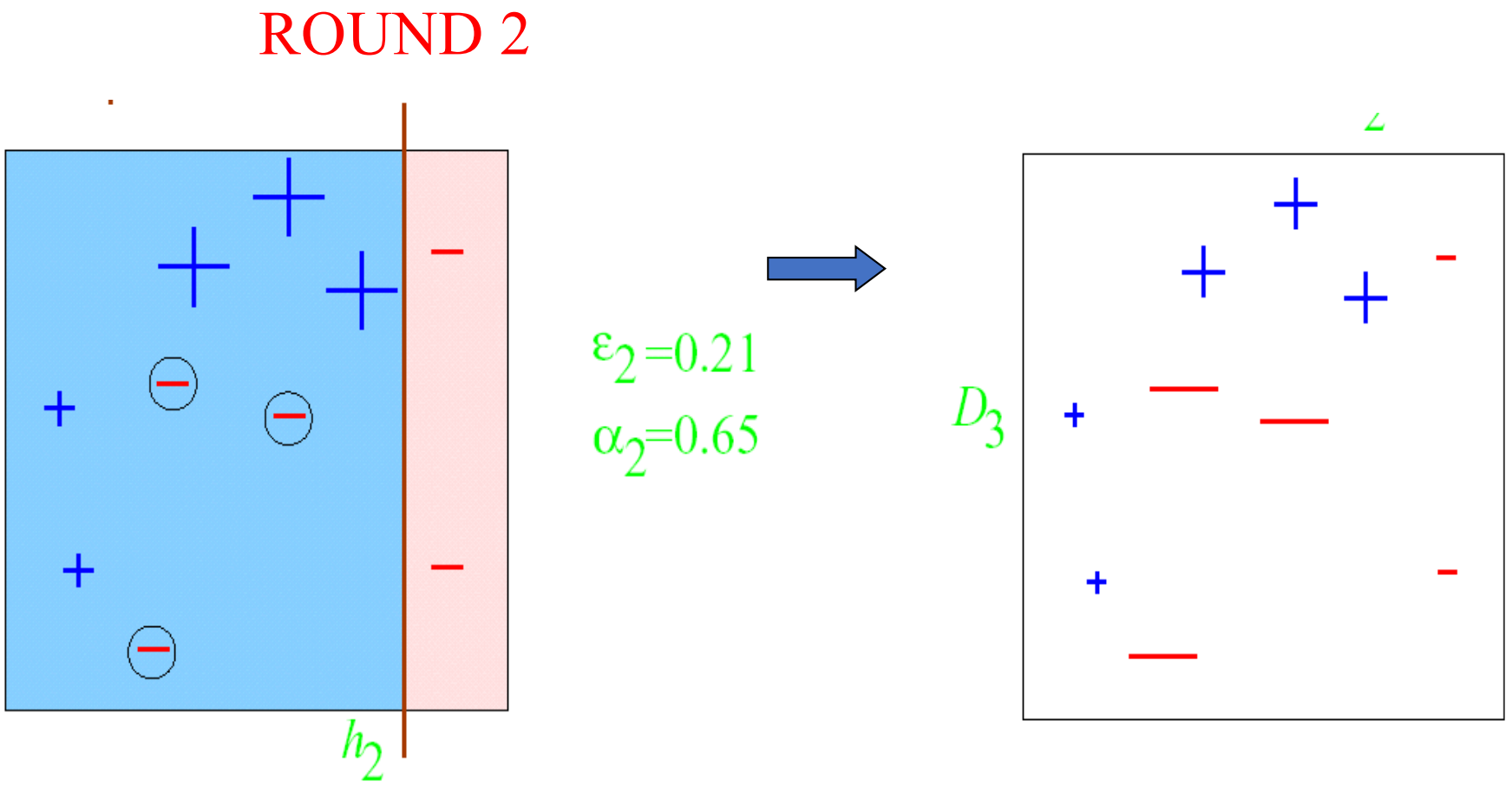
AdaBoost example

ε = error rate of classifier
α = weight of classifier

ROUND 1

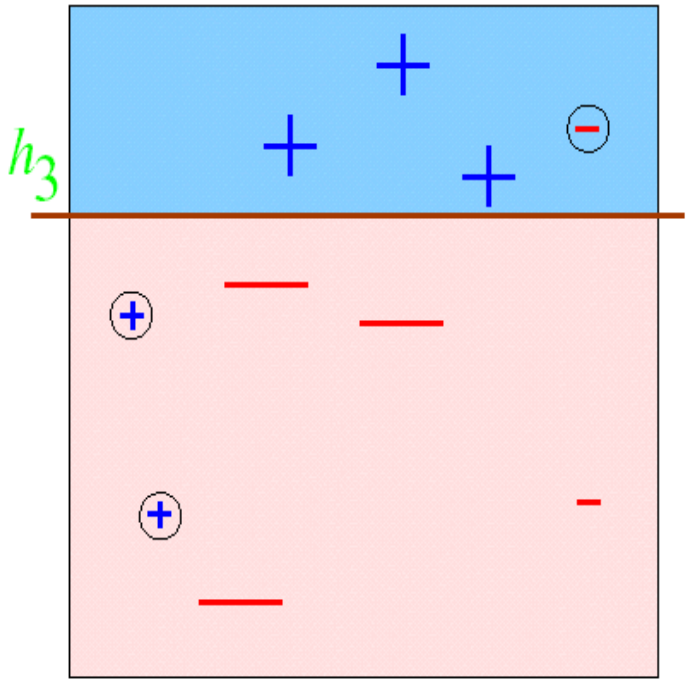


AdaBoost example



AdaBoost example

ROUND 3

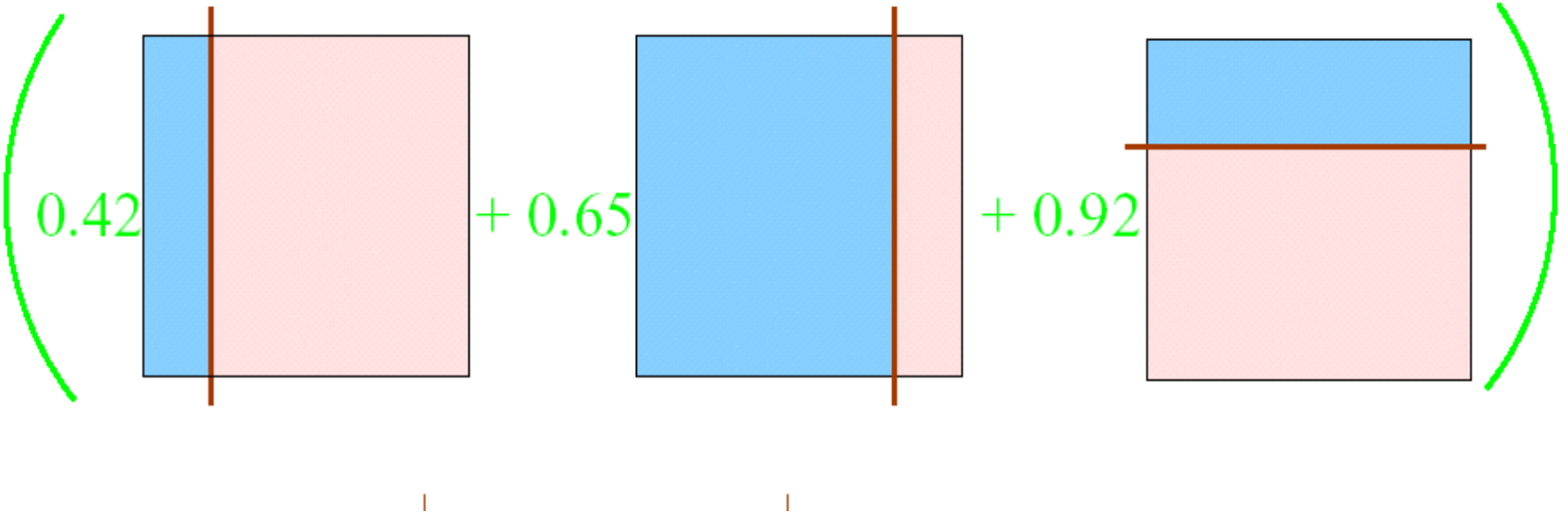


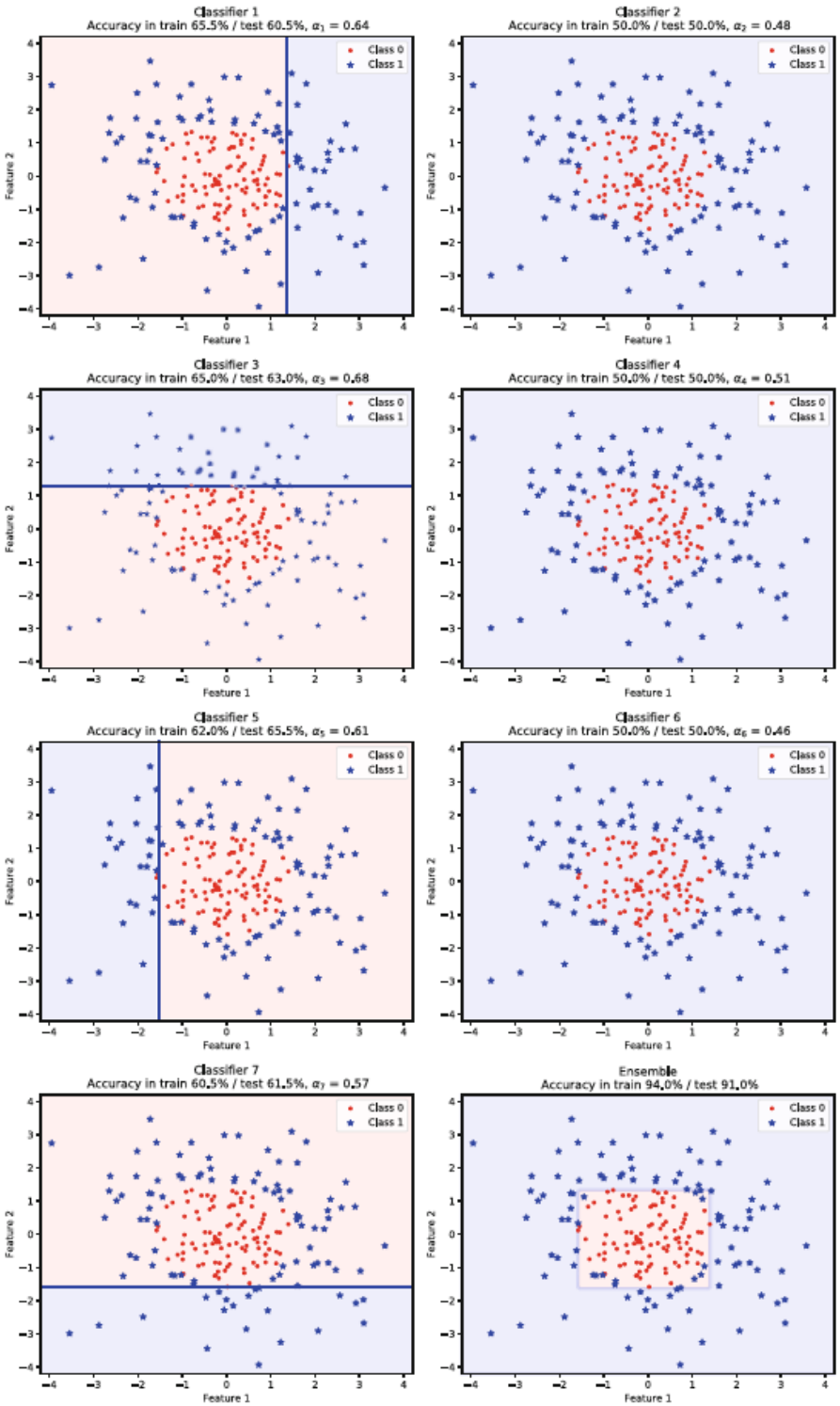
$\varepsilon_3 = 0.14$

$\alpha_3 = 0.92$

AdaBoost example

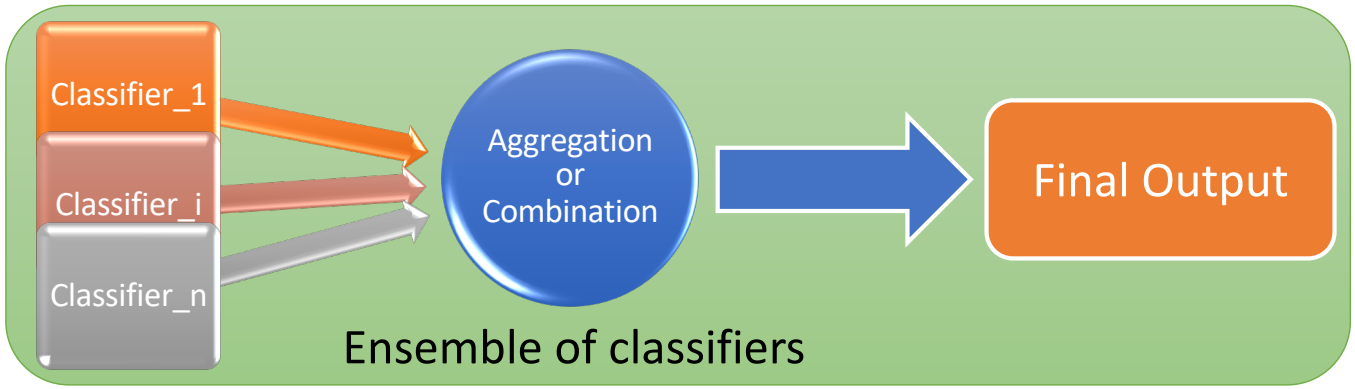
$$H_{\text{final}} = \text{sign}\left(0.42 \; + \; 0.65 \; + \; 0.92 \right)$$




Classifier 1
Accuracy in train 65.5% / test 60.5%, α_1 = 0.64
Classifier 2
Accuracy in train 50.0% / test 50.0%, α_2 = 0.48
Classifier 3
Accuracy in train 65.0% / test 63.0%, α_3 = 0.68
Classifier 4
Accuracy in train 50.0% / test 50.0%, α_4 = 0.51
Classifier 5
Accuracy in train 62.0% / test 65.5%, α_5 = 0.61
Classifier 6
Accuracy in train 50.0% / test 50.0%, α_6 = 0.46
Classifier 7
Accuracy in train 60.5% / test 61.5%, α_7 = 0.57
Ensemble
Accuracy in train 94.0% / test 91.0%
Class 0
Class 1
Feature 1
Feature 2

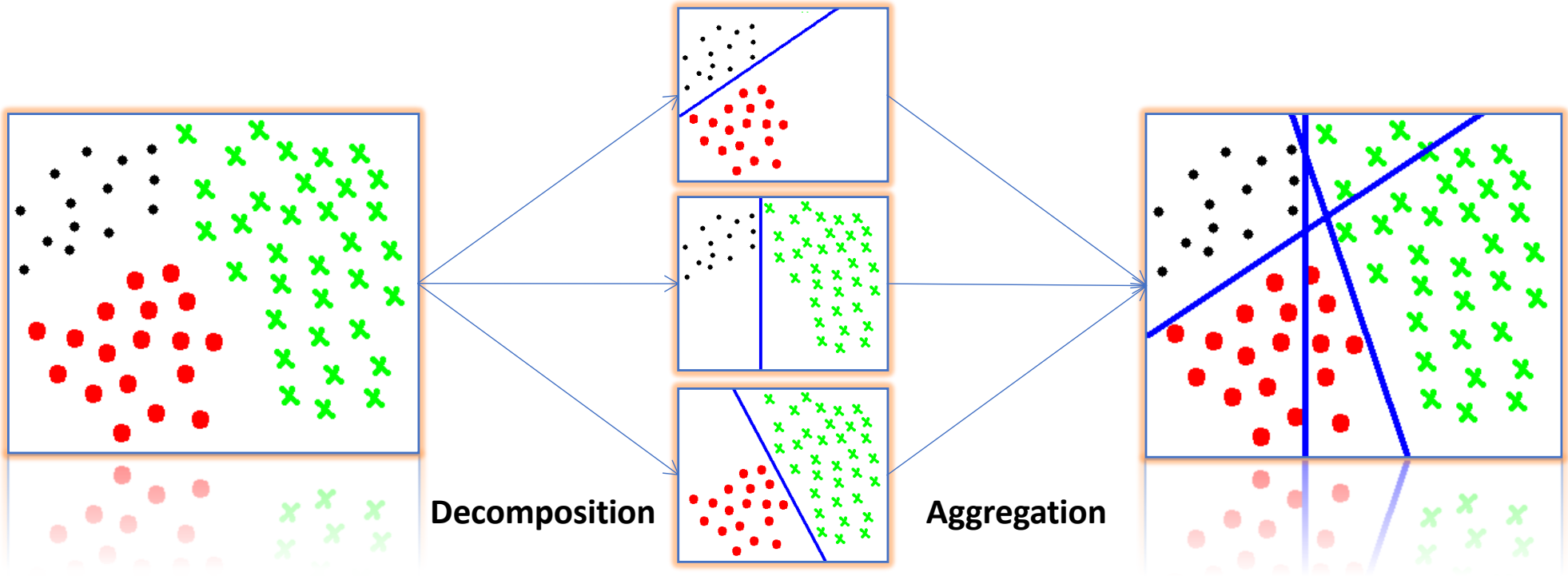
Binarization

- Decomposition of the multi-class problem
  - Divide and conquer strategy
  - Multi-class → Multiple easier to solve binary problems
    - For each binary problem
      - 1 binary classifier = base classifier
    - Problem
      - How we should make the decomposition?
      - How we should aggregate the outputs?



Decomposition Strategies

- "One-vs-One" (OVO)
  - 1 binary problem for each pair of classes
    - Pairwise Learning, Round Robin, All-vs-All…
    - *Total = m(m-1) / 2 classifiers*



One-vs-One

- Advantages
  - Smaller (number of instances)
  - Simpler decision boundaries
    - Digit recognition problem by pairwise learning
      - linearly separable (first proposal)
  - Parallelizable
  - …

One-vs-One

- Disadvantages
  - Higher testing times (more classifiers)
  - Non-competent examples
- Many different aggregation proposals
  - From the score Matrix:

$$R = \begin{pmatrix} - & r_{12} & \cdots & r_{1m} \\ r_{21} & - & \cdots & r_{2m} \\ \vdots & & & \vdots \\ r_{m1} & r_{m2} & \cdots & - \end{pmatrix}$$

State-of-the-art on aggregation for OVO

- Voting strategy (VOTE)
  - Each classifier gives a vote for the predicted class
  - The class with the largest number of votes is predicted

$$Class = arg \max_{i=1,\ldots,m} \sum_{1 \le j \ne i \le m} s_{ij}$$

  - where s_{ij} is 1 if $r_{ij} > r_{ji}$ and 0 otherwise.
- Weighted voting strategy (WV)
  - WV = VOTE but weight = confidence

$$Class = arg \max_{i=1,\ldots,m} \sum_{1 \le j \ne i \le m} r_{ij}$$

Example of prediction

- Classify x, whose real class is c_1

- $$R(x) = \begin{pmatrix} & c1 & c2 & c3 & c4 & c5 \\ c1 & - & 0{,}55 & 0{,}6 & 0{,}75 & 0{,}7 \\ c2 & 0{,}45 & - & 0{,}4 & 1 & 0{,}8 \\ c3 & 0{,}4 & 0{,}6 & - & 0{,}5 & 0{,}4 \\ c4 & 0{,}25 & 0{,}0 & 0{,}5 & - & 0{,}1 \\ c5 & 0{,}30 & 0{,}2 & 0{,}6 & 0{,}9 & - \end{pmatrix}$$

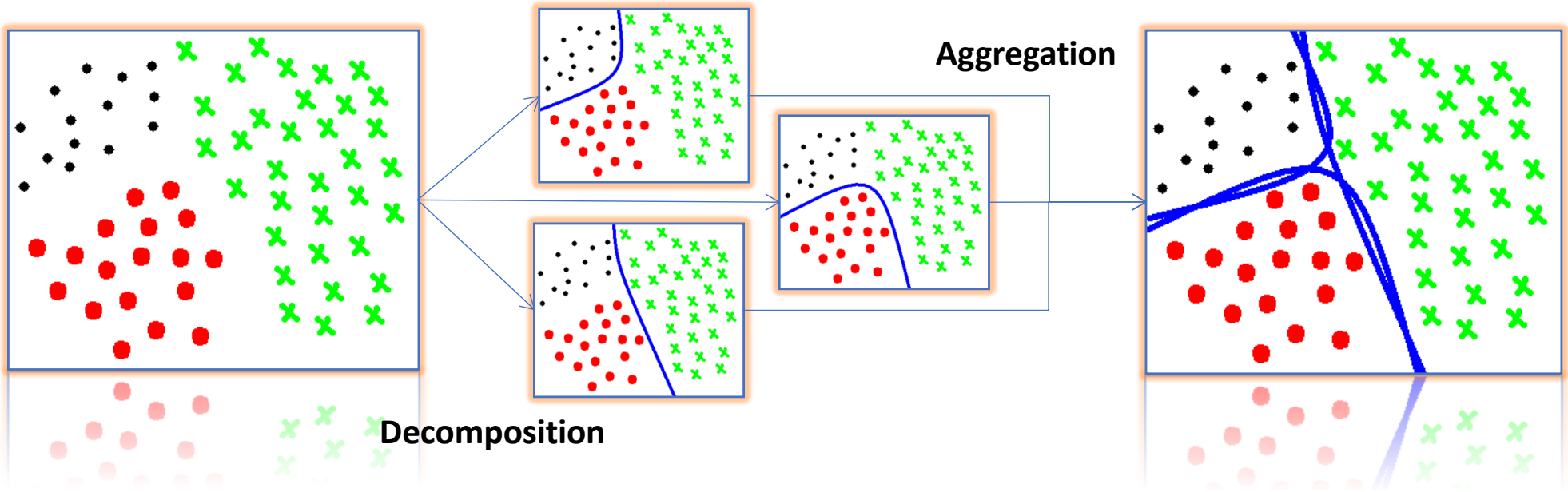
Example of prediction

- Beware of **Non-Competent** Classifiers:
  - Consider WV aggregation, c_2 is predicted
  - **None** of the classifiers **considering c_1 failed**
  - **Non-competent** classifiers **strongly voted for** c_2

$$R(x) = \left(\begin{array}{c|cccccc} & c1 & c2 & c3 & c4 & c5 & WV \\ c1 & - & 0{,}55 & 0{,}6 & 0{,}75 & 0{,}7 & 2{,}6 \\ c2 & 0{,}45 & - & 0{,}4 & 1 & 0{,}8 & \mathbf{2{,}65} \\ c3 & 0{,}4 & 0{,}6 & - & 0{,}5 & 0{,}4 & 1{,}9 \\ c4 & 0{,}25 & 0{,}0 & 0{,}5 & - & 0{,}1 & 0{,}85 \\ c5 & 0{,}30 & 0{,}2 & 0{,}6 & 0{,}9 & - & 2{,}1 \end{array}\right)$$

Decomposition Strategies

- "One-vs-All" (OVA)
  - 1 binary problem for each class
    - All instances in each problem
      - Positive class: instances from the class considered
      - Negative class: instances from all other classes
    - *Total = m classifiers*



One-vs-All

Advantages

- Less nº of classifiers
- All examples are "competent"

Disadvantages

- Less studied in the literature
  - low nº of aggregations
- More complex problems
- Imbalance training sets

State-of-the-art on aggregation for OVA

- Starting from the score-vector

$$R = (r_1, r_2, \ldots, r_i, \ldots, r_m)$$

  - r_i = confidence of classifier in favor of class i
    - Respect to all other classes
  - Usually more than 1 classifier predicts the positive class
    - Tie-breaking techniques
- Maximum confidence strategy (MAX)
  - Predicts the class with the largest confidence

$$Class = arg \max_{i=1,\ldots,m} r_i$$