Extracción Automática de Conocimiento en Bases de Datos e Ingeniería del Software

T.1 INTRODUCCIÓN

José Hernández Orallo

- Máster en Ingeniería del Software, Métodos Formales y Sistemas de Información.

Objetivos

- Presentar el problema del análisis inteligente y automático de la información para el descubrimiento de conocimiento útil.
- Presentar las técnicas de aprendizaje automático más habituales y conocer la idoneidad de cada una para diferentes problemas, con especial interés en aquellas que generan modelos en formas de reglas o de patrones comprensibles.
- Conocer la representación de conocimiento en árboles de decisión y su flexibilidad.
- Reconocer la existencia de técnicas inductivas de alto nivel, especialmente declarativas, que permiten obtener modelos complejos (relacionales y/o recursivos) pero comprensibles, a partir de los datos y del conocimiento previo.

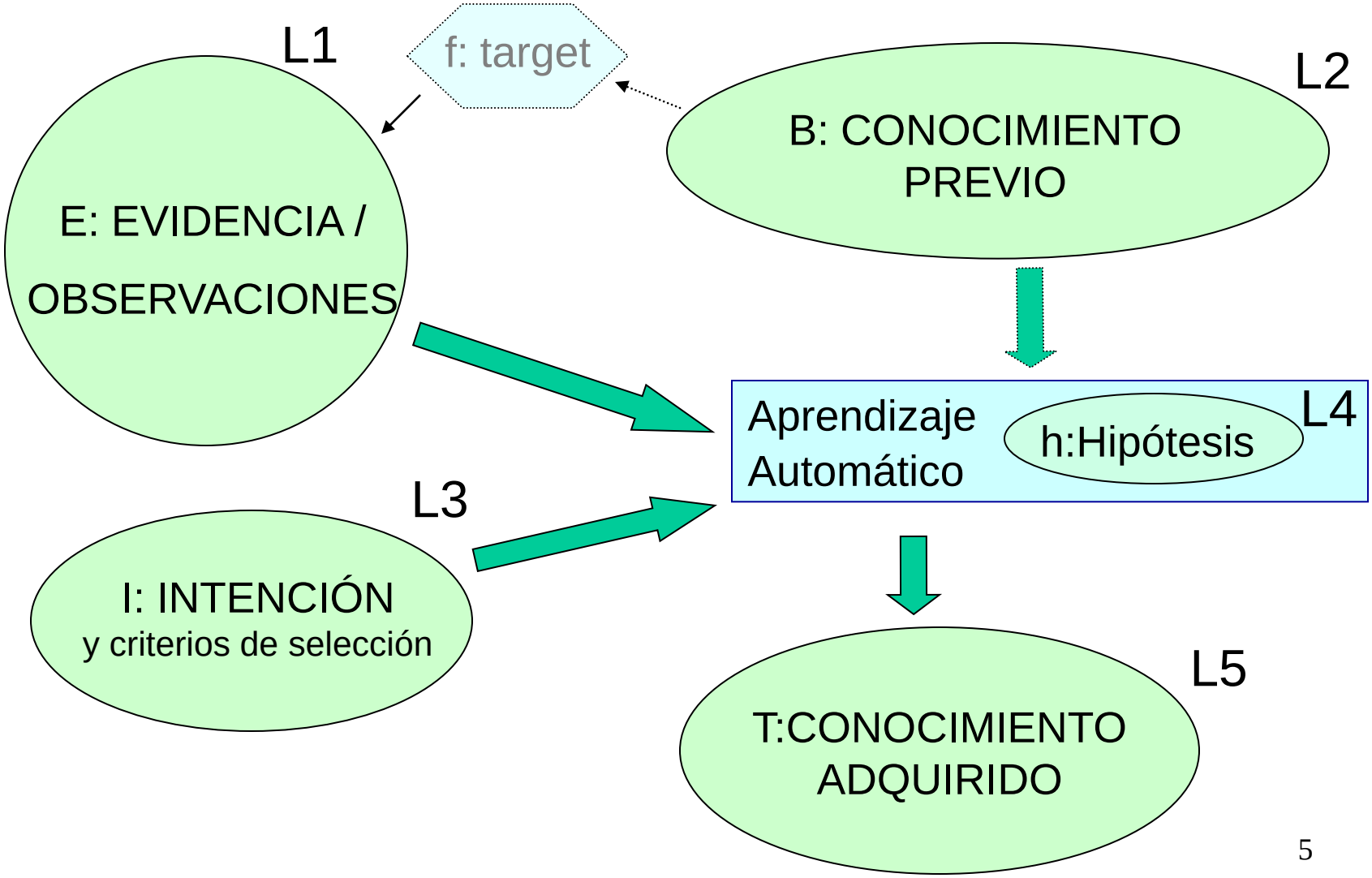
Temario

1. Introducción.

    1.1. El Problema de la Extracción Automática de Conocimiento.
    1.2. Relación de Tareas y Técnicas
    1.3. Modelos declarativos: árboles y reglas
    1.4. El caso de la Minería de Datos

1.1. El Problema de la Extracción Automática de Conocimiento.

El Problema de la Extracción Automática de Conocimiento



El Problema de la Extracción Automática de Conocimiento

Terminología:

- *E*: Evidencia, Observaciones, Ejemplos, Casos, Datos.
- *B*: Conocimiento Previo o de Base, (Background Knowledge).
- *h*: Hipótesis, *H* (*L4*): espacio de hipótesis.
- *T*: Teoría, Modelo o Conocimiento Adquirido.
- *I*: Intención, interés, objetivo o expectativas del aprendizaje.

- *f: Teoría o función objetivo (Target)*: Teoría o función a aprender

El Problema de la Extracción Automática de Conocimiento

Aspectos Fundamentales:

- ¿Qué diferencias hay entre información, datos y conocimiento?
- ¿Qué es aprendizaje?
- ¿Cómo se *representa* evidencia, conocimiento previo, intenciones, hipótesis y conocimiento adquirido (L1, L2, L3, L4 y L5)?
- ¿Cómo se *presentan* la evidencia y el conocimiento previo?
- ¿Cómo se validan/descartan las hipótesis para conformar el conocimiento adquirido?
- ¿Qué esfuerzo computacional se requiere?

El Problema de la Extracción Automática de Conocimiento

¿Qué diferencias hay entre información, datos y conocimiento?

- Generalmente “información” se pueden referir a cualquier cosa.
- Generalmente, “datos” suele referir a la “evidencia”.
- “Conocimiento” es subjetivo:
  - depende de las intenciones (objetivo del aprendizaje).
  - debe ser *inteligible* para el que aprende o el que encarga el aprendizaje (usuario).

El Problema de la Extracción Automática de Conocimiento

¿Qué es aprendizaje?

- (visión genérica, Mitchell 1997) es mejorar el comportamiento a partir de la experiencia. Aprendizaje = Inteligencia.
- (visión más estática) es la *identificación de* ***patrones***, de ***regularidades***, existentes en la evidencia.
- (visión externa) es la ***predicción*** de observaciones futuras con ***plausibilidad***.
- (visión teórico-informacional, Solomonoff 1966) es ***eliminación de redundancia*** = ***compresión de información***.

El Problema de la Extracción Automática de Conocimiento

¿Cómo se *representan* evidencia, conocimiento previo, intenciones, hipótesis y conocimiento adquirido (L1, L2, L3, L4 y L5)?

- Generalmente se usan distintos lenguajes:
  - La evidencia se suele representar extensionalmente (ejemplos positivos y/o negativos).
  - El conocimiento previo y el conocimiento adquirido suelen tener el mismo lenguaje de representación.
  - Las intenciones se suelen representar en un metalenguaje o no se representan, implícitas en el algoritmo de aprendizaje.
  - Gran variedad en la representación de hipótesis.

El Problema de la Extracción Automática de Conocimiento

¿Cómo se *presentan* la evidencia y el conocimiento previo?

- No-incremental: todos los datos se presentan de golpe, todos.
  - *Ejemplo: modelos de concesión de crédito, estudios clínicos, …*
- Incremental: los datos se presentan poco a poco.
  - No interactivo: No se puede actuar sobre la evidencia a recibir.
    - *Ejemplo: predicción meteorológica, en bolsa, …*
  - Interactivo: Se puede actuar sobre la evidencia a recibir.
    - Preguntas a un "profesor" (query-learning).
      - *Ejemplo: aprendizaje asistido por la web*
    - Interacción con un oráculo (la realidad).
      - *Ejemplo: aprendizaje de robots*

El Problema de la Extracción Automática de Conocimiento

¿Cómo se validan/descartan las hipótesis para conformar el conocimiento adquirido?

- Principio ('escándalo') de la Inducción: las hipótesis pueden ser refutadas, pero nunca confirmadas.
- Y para las que todavía no han sido refutadas, ¿cuál elegimos?
  - Necesidad de criterios de selección: simplicidad, refuerzo, ...
  - Existencia de métodos de validación: estadísticos, cross-validation, informacionales, ...
- ¿Cuánto afecta a la plausibilidad el número de ejemplos?
- Las cosas son más complejas cuando hay presencia de ruido.

El Problema de la Extracción Automática de Conocimiento

¿Qué esfuerzo computacional se requiere?

- ¿De qué depende? Número y separabilidad de ejemplos, espacio de hipótesis, conocimiento previo, nivel de error permitido, ...
  ⇨(Computational Learning Theory, COLT).
- Cuanto más expresivos son L1-L5, más difícil computacionalmente es el problema de aprendizaje.
- Para lenguajes universales, el problema de aprendizaje como compresión no es sólo intratable, sino no computable si el objetivo es la *máxima* compresión.
- ¿Qué conceptos se pueden aprender eficientemente (i.e. en tiempo polinomial)? ⇨ PAC learning

1.2. Relación de Tareas y Técnicas

Tareas y Técnicas

- Vista de pájaro sobre diferentes tareas y técnicas.
  - Más información en la bibliografía
  - En asignaturas de la ETSINF (II). y de los másters del DSIC:
    - Almacenes y minería de datos (II, este máster)
    - Aprendizaje automático (II, máster IARFID)
    - Aprendizaje y percepción (II, máster IARFID)
    - Aprendizaje y generalización (máster IARFID)
    - Clasificación basada en prototipos (máster IARFID)
    - Introducción a la computación neuronal (II, máster IARFID)
    - Redes neuronales (máster IARFID)

Tareas de aprendizaje

- **Asociaciones y dependencias (análisis exploratorio)**: Una asociación entre dos atributos categóricos ocurre cuando la frecuencia de que se den dos valores determinados de cada uno conjuntamente es relativamente alta.
  - Ejemplo, en un sitio web se analiza si las visitas a una página X y las visitas a otra página Y se realizan conjuntamente.

- **Correlaciones**: Las correlaciones analizan las relaciones bivariantes o multivariantes entre atributos numéricos.
  - Ejemplo: analizar la relación entre el número de llamadas mensuales y el importe total de las ventas mensuales.

La búsqueda de asociaciones y dependencias, junto con los análisis correlacionales se conoce a veces como análisis exploratorio.

Tareas de aprendizaje

Tipos de conocimiento (cont.):

- **Agrupamiento / Segmentación / Sumarización**: El agrupamiento (o clustering) es la detección de grupos de individuos. Se diferencia de la clasificación en el que no se conocen ni las clases ni su número (aprendizaje no supervisado), con lo que el objetivo es determinar grupos o racimos (clusters) diferenciados del resto.
  - Ejemplo: determinar qué tipos de clientes tengo atendiendo a sus patrones de compra.

Tareas de aprendizaje

Tipos de conocimiento (cont.):

- **Clasificación**: Una clasificación se puede ver como el esclarecimiento de una dependencia, en la que el atributo dependiente puede tomar un valor entre varias clases, ya conocidas.
  - Ejemplo: obtener para qué pacientes una operación de cirugía ocular es satisfactoria según los atributos edad, número de miopías y astigmatismo

Tareas de aprendizaje

Tipos de conocimiento (cont.):

- **Tendencias/Estimación/Regresión**: El objetivo es predecir los valores de una variable continua a partir de la evolución sobre otra variable continua, generalmente el tiempo.
  - Ejemplo, se intenta predecir el número de clientes o pacientes, los ingresos, llamadas, conexiones, fallos, ganancias, costes, etc. a partir de los resultados de semanas, meses o años anteriores.

Tareas de aprendizaje (resumen)

Tipos de conocimiento / Tareas (RESUMEN):

- **DESCRIPTIVOS (ningún atributo de salida):**
  - **RELACIÓN ENTRE VARIABLES (ATRIBUTOS):**
    - **Asociaciones y dependencias (si las variables son categóricas):**
    - **Correlaciones (si las variables son numéricas).**
  - **RELACIÓN ENTRE INDIVIDUOS (EJEMPLOS)**
    - **Agrupamiento**
- **PREDICTIVOS (un atributo de salida):**
    - **Clasificación (si la variable de salida es categórica)**
    - **Regresión (si la variable de salida es numérica)**

Técnicas de Aprendizaje Automático

- ***Flexibilidad debido a la presentación del problema:*** *muchas técnicas de supervisado se han adaptado a no supervisado (y viceversa).*

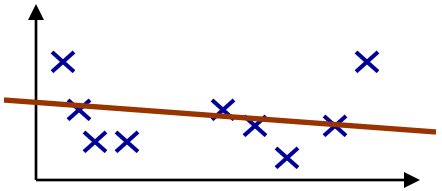
TÉCNICA	PREDICTIVO / SUPERVISADO		DESCRIPTIVO / NO SUPERVISADO		
	Clasificación	Regresión	Clustering (agrup.)	Reglas asociación	Otros (factoriales, correl, dispersión)
Redes Neuronales	✓	✓	✓ *		
Árboles de Decisión	✓ (c4.5)	✓ (CART)	✓		
Kohonen			✓		
Regresión lineal (local, global), exp..		✓			
Reg. Logística	✓				
Kmeans	✓ *		✓		
A Priori (asociaciones)				✓	
Estudios Factoriales, análisis multivariante					✓
CN2	✓				
K-NN	✓		✓		
RBF	✓				
Bayes Classifiers	✓	✓			

Regresión

Regresión Lineal Global.

Se buscan los coeficientes de una función lineal

$$\hat{f}(x) = w_0 + w_1 x_1 + \ldots + w_n x_n$$



Una manera fácil (si es *lineal simple*, sólo dos dimensiones x e y):

$$w_1 = \frac{n\sum xy - \sum x \sum y}{n\sum x^2 - \left(\sum x\right)^2} \qquad w_0 = \frac{\sum y \sum x^2 - \sum x \sum xy}{n\sum x^2 - \left(\sum x\right)^2}$$

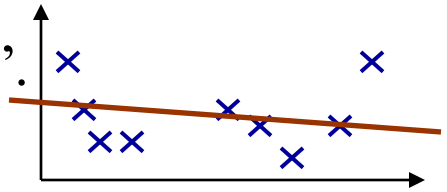
obteniendo $y = w_0 + w_1 x$

Error típico de una regresión lineal simple:

$$E_{tipico} = \sqrt{\left[\frac{1}{n(n-2)}\right]\left[n\sum y^2 - \left(\sum y\right)^2 - \frac{\left[n\sum xy - \sum x \sum y\right]^2}{n\sum x^2 - \left(\sum x\right)^2}\right]}$$

Regresión

Regresión Lineal Global por Gradient Descent.



Una manera usual es utilizando "gradient descent".
Se intenta minimizar la suma de cuadrados:

$$E = \frac{1}{2}\sum_{x \in D}(f(x) - \hat{f}(x))^2$$

Derivando,

$$\Delta w_j = r \cdot \sum_{x \in D}(f(x) - \hat{f}(x))x_j$$

Iterativamente se van ajustando los coeficientes y reduciendo el error.

Regresión

Regresión No Lineal.

Estimación Logarítmica (se sustituye la función a obtener por $y=\ln(f)$):

$$y = w_0 + w_1 x_1 + ... + w_m x_m$$

Se hace regresión lineal para calcular los coeficientes y a la hora de predecir se calcula la $f = e^y$.

Regresión Logística. *(variación que se usa para clasificación entre 0 y 1 usando la f= ln(p/(1-p)))*

Pick and Mix - Supercharging

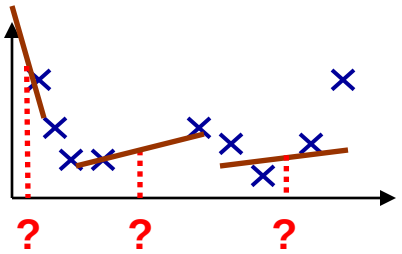
Se añaden dimensiones, combinando las dadas. P.ej. si tenemos cuatro dimensiones: x_1, x_2, x_3 (además de y) podemos definir $x_4 = x_1 \cdot x_2$, $x_5 = x_3^2$, $x_6 = x_1^{x_2}$ y obtener una función lineal de x_1, x_2, x_3, x_4, x_5, x_6

Regresión

Regresión Lineal Ponderada Localmente.

La función lineal se aproxima para cada punto x_q a interpolar:

$$\hat{f}(x) = w_0 + w_1 x_1 + \ldots + w_m x_m$$



Se intenta minimizar la suma de cuadrados de los *k* más cercanos

$$E = \frac{1}{2} \sum_{x \in \{\text{los k puntos más cercanos}\}} (f(x) - \hat{f}(x))^2 K(d(x_q, x))$$

donde $d(\cdot,\cdot)$ es una distancia y K es una función que disminuye con la distancia (una función Kernel), p.ej. $1/d^2$

Gradient Descent:

$$\Delta w_j = r \cdot \sum_{x \in \{\text{los k puntos más cercanos}\}} (f(x) - \hat{f}(x)) \cdot K(d(x_q, x)) \cdot x_j$$

A mayor k más global, a menor k más local (pero ojo con el overfitting)

Regresión

Regresión Adaptativa:

- *Son casos particulares de regresión local, en el que se supone un orden y se utiliza preferentemente para predecir futuros valores de una serie:*
- *Muy utilizada en compresión de sonido y de vídeo, en redes, etc. (se predicen las siguientes tramas)*

Algoritmos mucho más sofisticados (cadenas de Markov, VQ)

- Algoritmo MARS (Multiple Adaptive Regression Splines) (Friedman 1991).
- Series temporales: ARIMA, MARIMA, ARCH, MARCH, …

Aprendizaje Supervisado

Case-Based Reasoning (CBR) y k-NN (Nearest Neighbour):

- Se basa en la intuición de que datos similares tendrán clases similares. ¿Cómo se mide la similitud?
- DISTANCIA inversa a SIMILITUD.
- Los métodos de similitud (o de distancia) se basan en almacenar los ejemplos vistos, y calcular la similitud/distancia del nuevo caso con el resto de ejemplos.

Aprendizaje Supervisado

Case-based Reasoning (CBR) y k-NN (Nearest Neighbour):

- Muchísimas formas de calcular la distancia:
  - ***Distancia Euclídea:*** $\sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$
  - ***Distancia de Manhattan:*** $\sum_{i=1}^{n}|x_i - y_i|$
  - ***Distancia de Chebychev:*** $max_{i=1..n}|x_i - y_i|$

  } Valores Continuos (conveniente normalizar entre 0-1 antes)

  - ***Distancia del coseno:***
    cada ejemplo es un vector y
    la distancia es el coseno del ángulo que forman

  } Valores Continuos. No es necesario normalizar

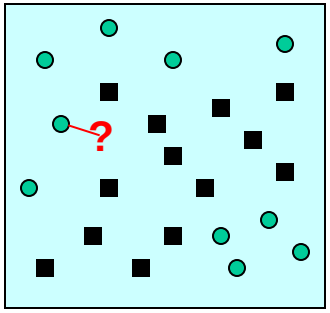
  - ***Distancias por Diferencia:***
    ejemplo: if $x=y$ then $D=0$ else $D=1$
  - ***Distancia de Edición:***
  - ***Distancias Específicas:*** *para los ejemplos complejos de CBR.*

  } Valores Discretos

Aprendizaje Supervisado

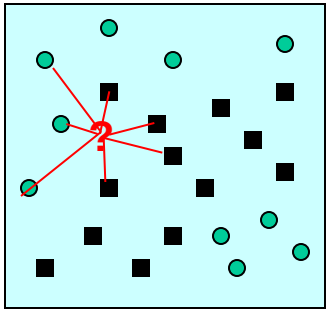
k-NN (Nearest Neighbour):

1. Se miran los *k* casos más cercanos.
2. Si todos son de la misma clase, el nuevo caso se clasifica en esa clase.
3. Si no, se calcula la distancia media por clase o se asigna a la clase con más elementos.



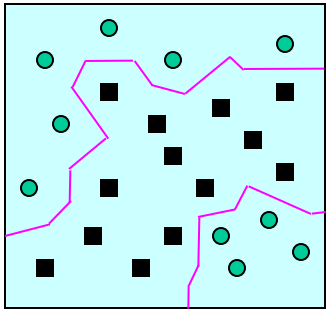
Clasifica círculo

1-nearest neighbor



Clasifica cuadrado

7-nearest neighbor



PARTICIÓN DEL 1-nearest neighbor (Poliédrica o de Voronoi)

- El valor de *k* se suele determinar heurísticamente, aunque $k = \sqrt{n}$, donde *n* es el número de ejemplos, es una opción con base teórica

Aprendizaje Supervisado

k-NN (Nearest Neighbour). Mejora (ponderar más los más cercanos):

$$Atracción(c_j, x_q) = \left|\{x_i : x_i \in c_j\}\right| \cdot krnl_i \quad \text{donde: } krnl_i = \frac{1}{d(x_q, x_i)^2}$$

Se calcula la fuerza de atracción de cada clase c_j para el nuevo punto x_q. Y se elige la clase que más atrae.

(Si el punto x_q coincide con un punto x_i, la clase es la de x_i)

(Si el punto x_q coincide con más de un punto x_i, se procede de la forma anterior)

Para valores continuos (sirve para interpolar):

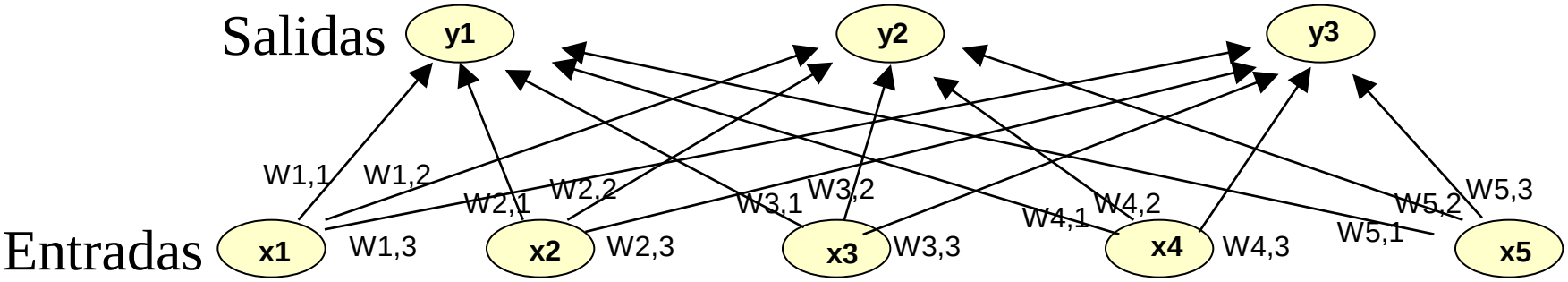
Si la clase es un valor real, el *k*-NN es fácilmente adaptable:

$$\hat{f}(x_q) = \frac{\sum_{i=1}^{k} krnl_i \, f(x_i)}{\sum_{i=1}^{k} krnl_i}$$

donde los x_i son los *k* vecinos más próximos y $f(\cdot)$ es la función que da el valor real de cada uno.

Aprendizaje Supervisado

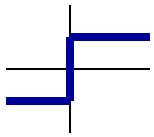
Perceptron Learning (McCulloch & Pitts 1943, Widrow & Hoff 1960, Rosenblatt 1962).



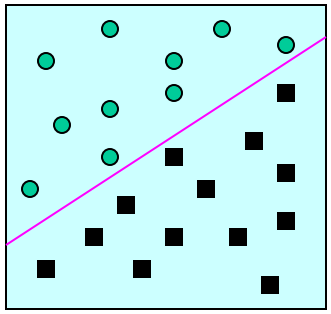
- Computan una función lineal para cada y_j es:

$$y'_j = \sum_{i=1}^{n} w_{i,j} \cdot x_i$$

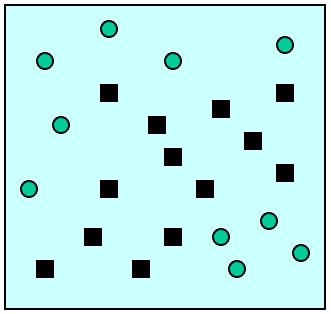
Se añade un threshold escalón: $output_j = \text{sgn}(y'_j)$



$$\text{sgn}(x) = \begin{cases} 1 \text{ si } x > 0 \\ -1 \text{ si no} \end{cases}$$



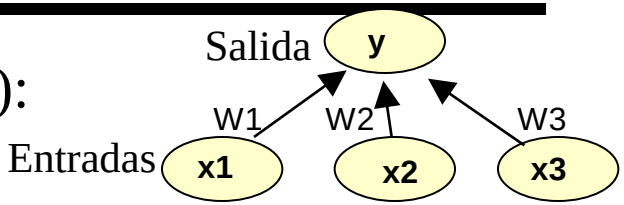
PARTICIÓN LINEAL POSIBLE



PARTICIÓN LINEAL IMPOSIBLE

Aprendizaje Supervisado

Gradient Descent (formul. para una sola salida):



- El error de Least Mean Squares de los *p* ejemplos se define como:

$$E(\vec{w}) = \frac{1}{2}\sum_{k:1..p}(e_k)^2 = \frac{1}{2}\sum_{k:1..p}(y_k - y'_k)^2$$

- Si queremos disminuir el error poco a poco. El gradiente es la derivada por cada componente del vector.

$$\frac{\partial E}{\partial w_i} = \frac{\partial}{\partial w_i}\frac{1}{2}\sum_{k:1..p}(y_k - y'_k)^2 = \frac{1}{2}\frac{\partial}{\partial w_i}\sum_{k:1..p}(y_k - y'_k)^2 = \frac{1}{2}\sum_{k:1..p}2(y_k - y'_k)\frac{\partial}{\partial w_i}(y_k - y'_k) =$$

$$= \sum_{k:1..p}(y_k - y'_k)\frac{\partial}{\partial w_i}(y_k - \vec{w}\cdot\vec{x}_k) = \sum_{k:1..p}(y_k - y'_k)(-x_{i,k})$$

- Queda:

$$\Delta w_i = \sum_{k:1..p}(y_k - y'_k)x_{i,k} = \sum_{k:1..p}x_{i,k}\cdot e_k$$

Aprendizaje Supervisado

Perceptron Learning (Gradient Descent).

- El algoritmo Gradient Descent ajusta así:
  1. Se asigna a los $w_{i,j}$ un valor pequeño aleatorio entre 0 y 1.
  2. Hasta que la condición de terminación se cumple, hacer:
  3. Para todos los p ejemplos $(\vec{x}_k,\vec{y}_k)^t$ se calcula la matriz de error $(\vec{e}^t_k = \vec{y}^t_k - \vec{y}'^t_k)$
  4. Se recalculan los pesos siguiendo Least-Mean Squares (LMS), con un learning rate (r):

$$w_{i,j}^{t+1} = w_{i,j}^{t} + \sum_{k:1..p} r \cdot x_{i,k}^{t} \cdot e_{j,k}^{t}$$

  5. t:= t+1, ir a 2.

r es un valor generalmente pequeño (0.05) y se determina heurísticamente. A mayor r converge más rápido pero puede perderse en valles locales.

Aprendizaje Supervisado

Perceptron Learning:

- El algoritmo Perceptron (*versión incremental o aproximación estocástica al gradient descent*):
  1. Se asignan aleatoriamente los $w_{i,j}$ entre 0 y 1 (o se pone .5)
  2. t= 1 (se toma el primer ejemplo).
  3. Para el ejemplo $(\vec{x},\vec{y})^t$ se calcula el vector error ($\vec{e^t}=\vec{y^t}-\vec{y'^t}$)
  4. Se recalculan los pesos siguiendo Least-Mean Squares (LMS), también llamada regla delta, Adaline o *Widrow-Hoff*:

$$w_{i,j}^{t+1} = w_{i,j}^{t} + r \cdot x_i^t \cdot e_j^t$$

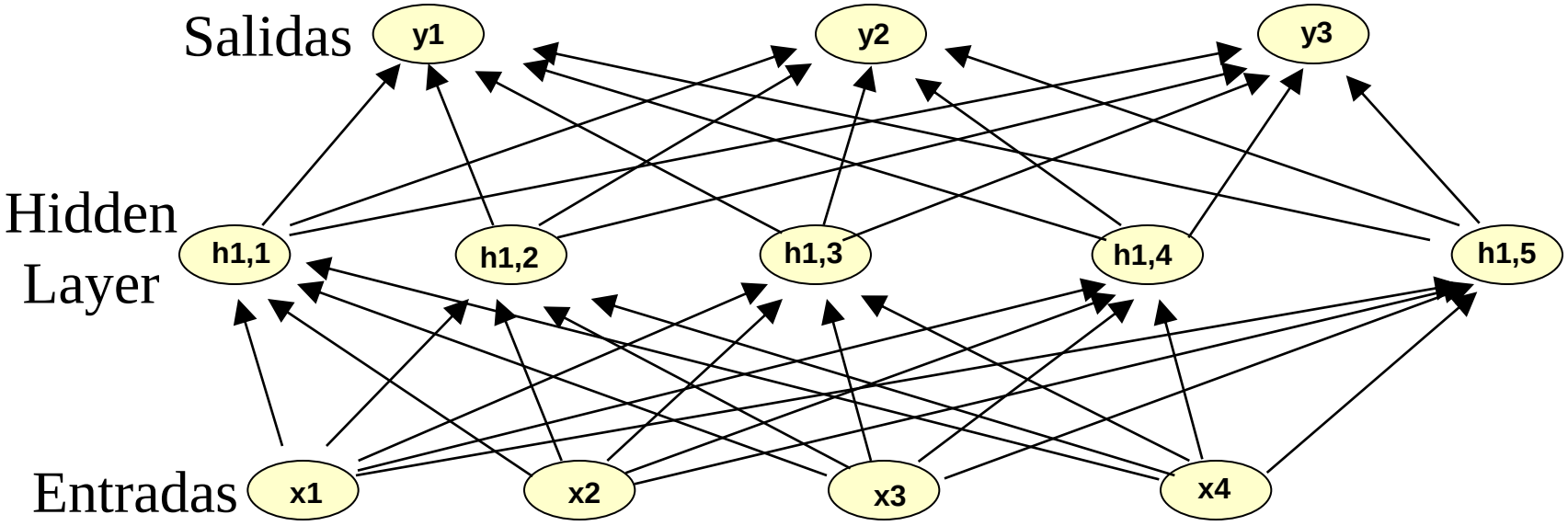
  5. t:= t+1, ir a 2 hasta que no queden ejemplos o el error medio se ha reducido a un valor deseado.

En general, esta versión es más eficiente que la anterior y evita algunos mínimos locales.

Aprendizaje Supervisado

Multilayer Perceptron (redes neuronales artificiales, ANN).

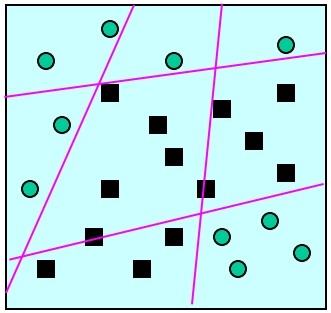
- El perceptron de una capa no es capaz de aprender las funciones más sencillas.
- Se añaden capas internas, se introducen diferentes funciones de activación e incluso recientemente se introducen bucles y retardos.



Aprendizaje Supervisado

Multilayer Perceptron (redes neuronales artificiales, ANN).

- En el caso más sencillo, con la función de activación sgn, el número de unidades internas *k* define exactamente el número de *boundaries* que la función global puede calcular por cada salida.



PARTICIÓN POLIGONAL POSIBLE CON 4 UNIDADES INTERNAS

- El valor de *k* se suele determinar heurísticamente.

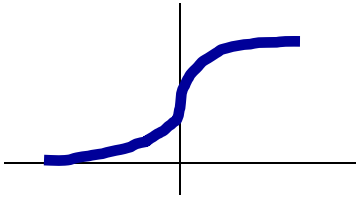
- Pero, ¿cómo entrenar este tipo de red?

Aprendizaje Supervisado

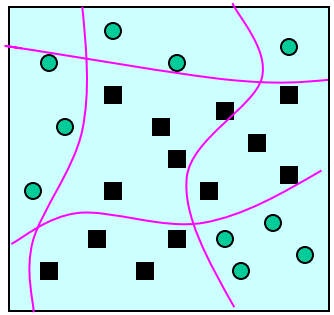
Multilayer Perceptron (redes neuronales artificiales, ANN).

- Para poder extender el gradient descent necesitamos una función de activación continua:
- La más usual es la función sigmoidal:

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



- Esto permite particiones no lineales:



PARTICIÓN NO LINEAL
MÚLTIPLE POSIBLE CON
4 UNIDADES INTERNAS

Aprendizaje Supervisado

Algoritmo Backpropagation (Rumelhart et al. 1986)

- Inicializar todos los pesos a valores pequeños aleatorios (entre -.05 y .05)
- Hasta que se cumpla la condición de terminación hacer:
  - Para cada ejemplo $(\vec{x},\vec{y})$:
    - Se calculan las salidas de todas las unidades o_u
    - Se calcula el error en cada salida k:

$$\delta_k = o_k(1-o_k)(y_k - o_k)$$

    - Para cada unidad oculta h se calcula su error:

$$\delta_h = o_h(1-o_h)\sum_{k \in outputs}(w_{k,h} \times \delta_k)$$

    - Se actualizan los pesos: $w_{j,i} = w_{j,i} + r \times \delta_j \times x_{j,i}$

Se necesitan muchos ejemplos: al menos 10 ejemplos por cada peso y output a aprender. P.ej, una red con 50 entradas y 10 nodos internos, necesita 10.220 ejemplos por lo menos.

Aprendizaje Supervisado

Variaciones:

- Si hay más de una capa oculta:

$$\delta_h = o_h(1-o_h) \sum_{k \in outputs\_of\_next\_layer} (w_{k,h} \times \delta_k)$$

- Si la red es no recursiva, pero no está organizada en capas (se trata de cualquier árbol acíclico), también se puede:

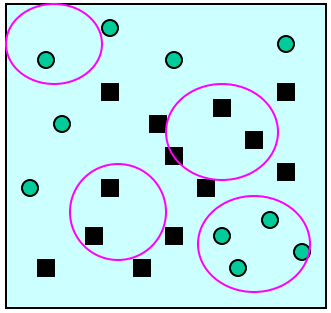
$$\delta_h = o_h(1-o_h) \sum_{k \in downstream(h)} (w_{k,h} \times \delta_k)$$

- Existe una variante que va añadiendo capas según se necesitan, denominado cascade-correlation (Fahlman and Lebiere 1990), resolviendo el problema de determinar el número de unidades ocultas.

Aprendizaje Supervisado

Radial-Basis Function (Clustering Method + LMS).

- PRIMER PASO: Algoritmo Clustering:
  1. Dividir aleatoriamente los ejemplos en *k* conjuntos y calcular la media (el punto medio) de cada conjunto.
  2. Reasignar cada ejemplo al cjto. con punto medio más cercano.
  3. Calcular los puntos medios de los *k* conjuntos.
  4. Repetir los pasos 2 y 3 hasta que los conjuntos no varíen.
- SEGUNDO PASO: Recodificar los ejemplos como distancias a los centros y normalizar (cada ejemplo pasa a ser un vector de *k* eltos).
- TERCER PASO: Con un perceptron de *k* elementos de entrada y una salida, aplicar el algoritmo visto antes.



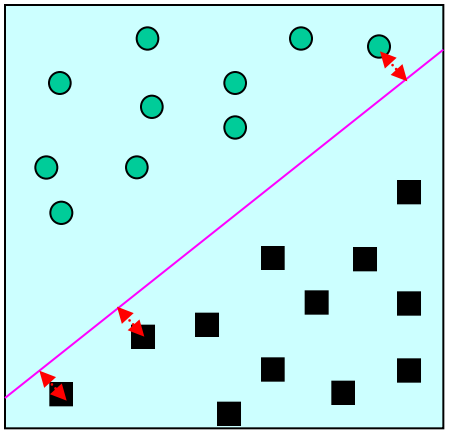
PARTICIÓN HIPERESFÉRICA CON 4 centros.

Se convierte en una partición lineal (hiperplano) en un espacio de 4 dimensiones con los ejemplos siendo las distancias a los centros.

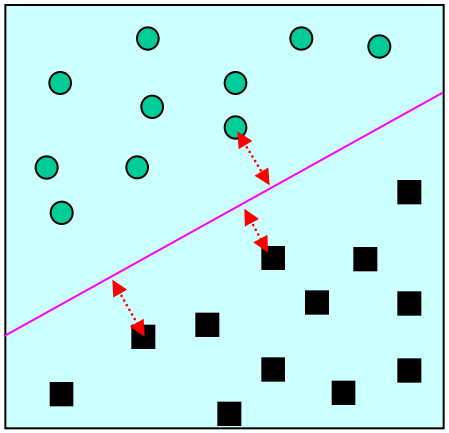
Aprendizaje Supervisado

Máquinas de vectores soporte (métodos basados en núcleo).

- Se basan en un clasificador lineal muy sencillo (el que maximiza la distancia de los tres ejemplos (vectores) soporte), precedido de una transformación de espacio (a través de un núcleo) para darle potencia expresiva.
- El clasificador lineal que se usa simplemente saca la línea (en más dimensiones, el hiperplano) que divida limpiamente las dos clases y además que los tres ejemplos más próximos a la frontera estén lo más distantes posibles.



Separa perfectamente, pero los tres ejemplos más cercanos (vectores soporte) están muy cerca de la frontera.

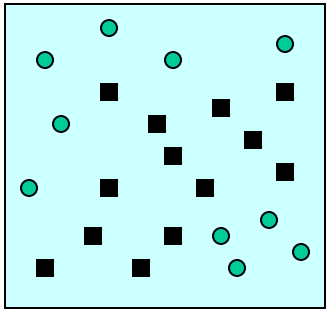


Separa perfectamente, pero además los ejemplos más cercanos (vectores soporte) están lo más lejos posible de la frontera.

Aprendizaje Supervisado

Máquinas de vectores soporte (métodos basados en núcleo).

- Son eficientes (incluso para cientos de dimensiones), pues el separador lineal sólo tiene que mirar unos pocos puntos (vectores soporte) y puede descartar muchos que estarán lejos de la frontera.
- ¿Pero qué ocurre si los datos no son separables linealmente?

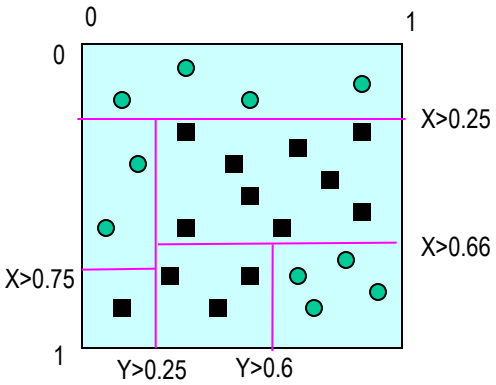


- Se aplica una función núcleo (“kernel”) que suele aumentar el número de dimensiones de tal manera que los datos sean separables.

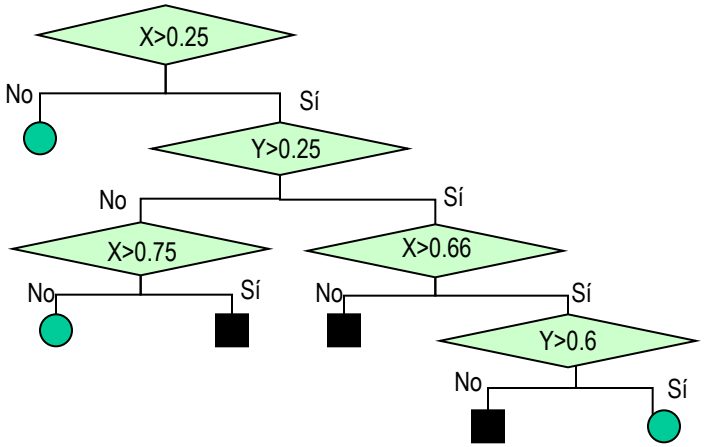
Aprendizaje Supervisado

Árboles de Decisión (ID3 (Quinlan), C4.5 (Quinlan), CART (Breiman)).

- Algoritmo Divide y Vencerás:
  1. Se crea un nodo raíz con S:= todos los ejemplos.
  2. Si todos los elementos de S son de la misma clase, el subárbol se cierra. Solución encontrada.
  3. Se elige una condición de partición siguiendo un criterio de partición (split criterion).
  4. El problema (y S) queda subdivido en dos subárboles (los que cumplen la condición y los que no) y se vuelve a 2 para cada uno de los dos subárboles.



PARTICIÓN CUADRICULAR.



Aprendizaje Supervisado

Árboles de Decisión.

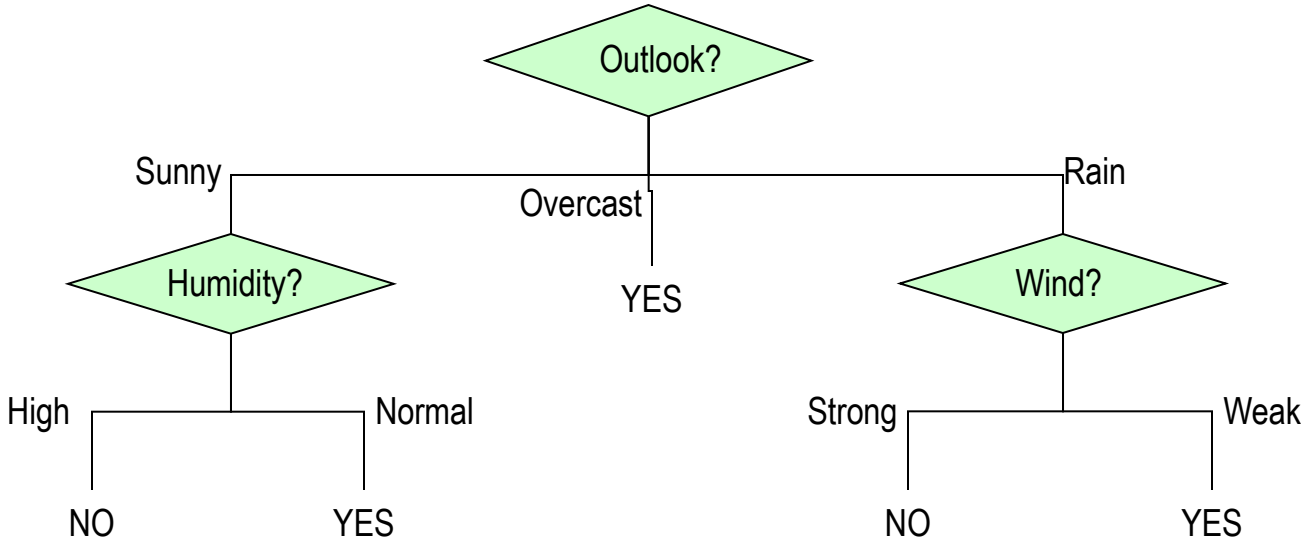
- Ejemplo C4.5 con datos discretos:

Example	Sky	Temperature	Humidity	Wind	PlayTennis
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Aprendizaje Supervisado

Árboles de Decisión.

- Ejemplo C4.5 con datos discretos:



- Representación Lógica:

  (Outlook=Sunny AND Humidity=Normal) OR (Outlook=Overcast) OR (Outlook=Rain AND Wind=Weak)

P.ej., la instancia (Outlook = sunny, Temperature = cool, Humidity = high, Wind = strong) es NO.

Aprendizaje Supervisado

Árboles de Decisión (ID3, C4.5, CART).

- El criterio GANANCIA DE INFORMACIÓN (C4.5) ha dado muy buenos resultados. Suele derivar en una preferencia en árboles pequeños (navaja de Occam).

- VENTAJAS:
  - Muy fácil de entender y de visualizar el resultado.
  - Son robustos al ruido. Existen algoritmos de *post-pruning* para podar hojas poco significativas (que sólo cubren uno o muy pocos ejemplos).

- DESVENTAJAS:
  - Suele ser muy voraz (no hace backtracking: mínimos locales).
  - Si el criterio de partición no está bien elegido, las particiones suelen ser muy ad-hoc y generalizan poco.

Aprendizaje Supervisado

Naive Bayes Classifiers.

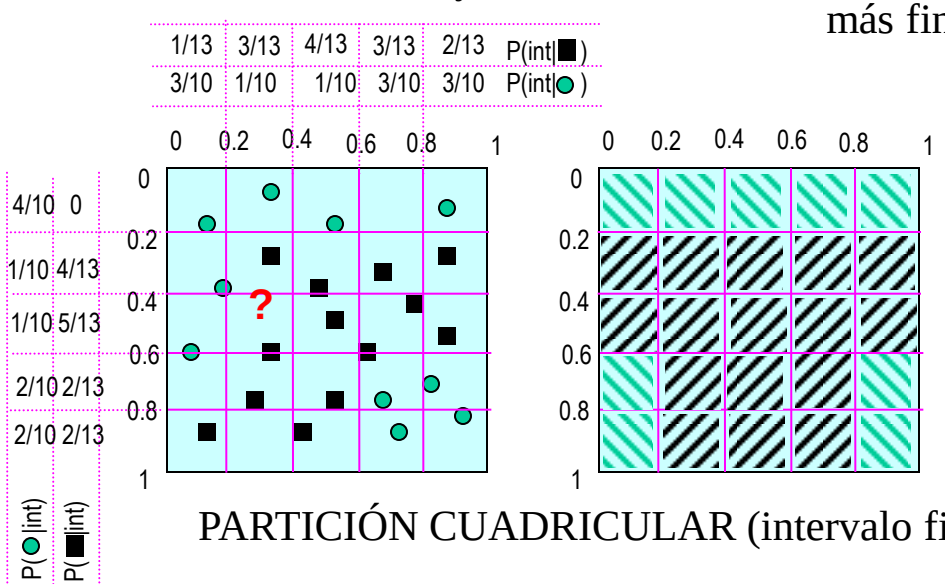
$$\arg\max_{c_i \in C} P(c_i \mid (x_1, x_2, ..., x_m)) \underset{Bayes}{=} \arg\max_{c_i \in C} \frac{P((x_1, x_2, ..., x_m) \mid c_i) \cdot P(c_i)}{P(x_1, x_2, ..., x_m)} =$$

$$= \arg\max_{c_i \in C} P((x_1, x_2, ..., x_m) \mid c_i) \cdot P(c_i)$$

- Asumiendo independencia entre los atributos, tenemos:

$$V_{NB} = \arg\max_{c_i \in C} P(c_i) \prod_j P(x_j \mid c_i)$$

Si estos x_j son continuos podemos discretizar en intervalos y calcular $P(c_i \mid t_k < x_j \leq t_{k+1})$ (cuanto más finos, más costoso será el algoritmo)



P(●)=10/23= 0.435
P(■)=13/23= 0.565

P(● |?) = P(●) · P(0.2<x<=0.4|●) · P(0.4<y<=0.6| ●)=
= 0.435 · 1/10 · 1/10 = 0.004

P(■|?) = P(■) · P(0.2<x<=0.4|■) · P(0.4<y<=0.6|■)=
= 0.565 · 3/13 · 5/13 = 0.05

PARTICIÓN CUADRICULAR (intervalo fijo, 0.2).

Aprendizaje Supervisado

Naive Bayes Classifiers.

- Otra manera es hacer los intervalos continuos y calcular la frecuencia acumulada f($c_i \mid x_j \leq t$). Tenemos f($c_i \mid s < x_j \leq t$) = f($c_i \mid x_j \leq t$) – f($c_i \mid x_j \leq s$).
- Se puede fijar un radio *r*.
- O podemos utilizar una función de densidad

$$p(x_0) = \lim_{\varepsilon \to \infty} \frac{1}{\varepsilon} P(x_0 \leq x < x_0 + \varepsilon)$$

- Así las particiones son más ajustadas.

- En el último caso (función de densidad), a partir del Maximum Likelihood obtendríamos la hipótesis Least-Squared Error:

$$h_{ML} = \arg\max_{h \in H} p(D \mid h) = \ldots = \arg\min_{h \in H} \sum_{i:1..m} (d_i - h(x_i))^2$$

donde d_i representa el dato *i*.

Aprendizaje Supervisado

Naive Bayes Classifiers.

- Se utilizan más con variables discretas. Ejemplo del playtennis:
- Queremos clasificar una nueva instancia:

  (Outlook = sunny, Temperature = cool, Humidity = high, Wind = strong)

$$V_{NB} = \arg\max_{c_i \in \{yes,no\}} P(c_i) \prod_j P(x_j \mid c_i) =$$

$$= \arg\max_{c_i \in \{yes,no\}} P(c_i) \cdot P(Outlook = sunny \mid c_i) \cdot P(Temperature = cool \mid c_i)$$

$$\cdot P(Humidity = high \mid c_i) \cdot P(Wind = strong \mid c_i)$$

- Estimando las 10 probabilidades necesarias:

  P(Playtennis=yes)=9/14=.64, P(Playtennis=no)=5/14=.36

  P(Wind=strong|Playtennis=yes)=3/9=.33 P(Wind=strong|Playtennis=no)=3/5=.60

  ...

- Tenemos que:

  P(yes)P(sunny|yes)P(cool|yes)P(high|yes)P(strong|yes)=0.0053

  P(no)P(sunny|no)P(cool|no)P(high|no)P(strong|no)=0.206

Aprendizaje Supervisado

Naive Bayes Classifiers. *m*-estimate.

- Generalmente, si hay pocos datos, es posible que alguna probabilidad condicional sea 0 (p.ej. P(water=cool| enjoysport=no)), porque no ha aparecido un determinado valor de un atributo para una cierta clase.
- Para evitar este problema se utiliza un *m*-estimado de la probabilidad:

$$\frac{n_c + mp}{n + m}$$

- donde n son los casos de la clase considerada, n_c son los casos de esta clase en los que el atributo considerado toma un cierto valor, m es una constante denominada "tamaño equivalente de muestra" y p es la probabilidad de cada atributo a priori.
- Generalmente p se escoge uniformemente, es decir, si hay k atributos posibles, $p = 1/k$.
- El valor de m se escoge lo más pequeño posible (1 a 10) para no interferir en la proporción observada (n_c/n).

Aprendizaje No Supervisado

Correlación y Asociaciones:

- **Coeficiente de correlación:**

$$Cor(\bar{x}, \bar{y}) = \frac{Cov(\bar{x}, \bar{y})}{\sigma_x \cdot \sigma_y}$$

donde

$$Cov(\bar{x}, \bar{y}) = \frac{1}{n} \sum_{i=1}^{n} (x_i - \mu_x)(y_i - \mu_y)$$

- **Asociaciones (cuando los atributos son discretos).**
  Ejemplo: tabaquismo y alcoholismo están asociados.

- **Dependencias funcionales: asociación unidireccional.**
  Ejemplo: el nivel de riesgo de enfermedades cardiovasculares depende del tabaquismo y alcoholismo (entre otras cosas).

Aprendizaje No Supervisado

Clustering (Segmentación):

Se trata de buscar agrupamientos naturales en un conjunto de datos tal que tengan semejanzas.

Métodos de Agrupamiento:

- Jerárquicos: los datos se agrupan de manera arborescente (p.ej. el reino animal).
- No jerárquicos: generar particiones a un nivel.
  - (a) Paramétricos: se asume que las densidades condicionales de los grupos tienen cierta forma paramétrica conocida (p.ej. Gaussiana), y se reduce a estimar los parámetros.
  - (b) No paramétricos: no asumen nada sobre el modo en el que se agrupan los objetos.

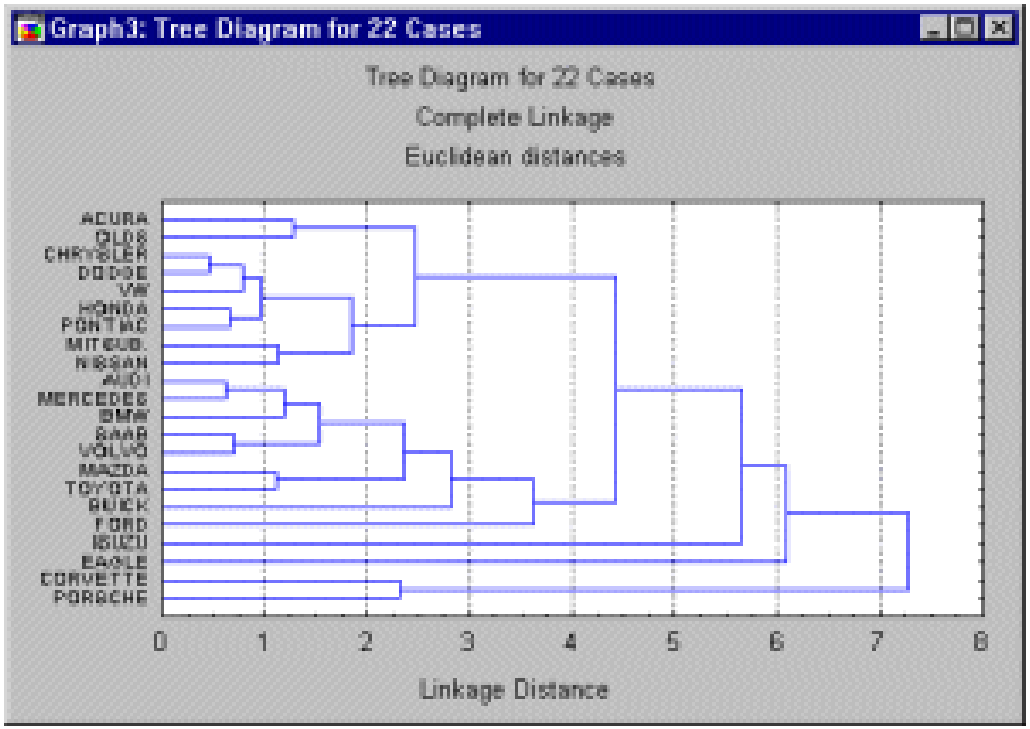
- *Si el modelo resultante es en forma de patrones se llama "conceptual clustering".*

Aprendizaje No Supervisado

Clustering (Segmentación). Métodos jerárquicos:

Un método sencillo consiste en ir separando individuos según su distancia (en concreto medidas derivadas de enlazado, *linkage*) e ir aumentando el límite de distancia para hacer grupos. Esto nos da diferentes agrupaciones a distintos niveles, de una manera jerárquica.

Se denomina *Dendograma* o *Hierarchical Tree Plot*:



Aprendizaje No Supervisado

Clustering (Segmentación). Métodos jerárquicos:

Minimal Spanning Tree Clustering Algorithm

Algoritmo (dado un número de clusters deseado C).

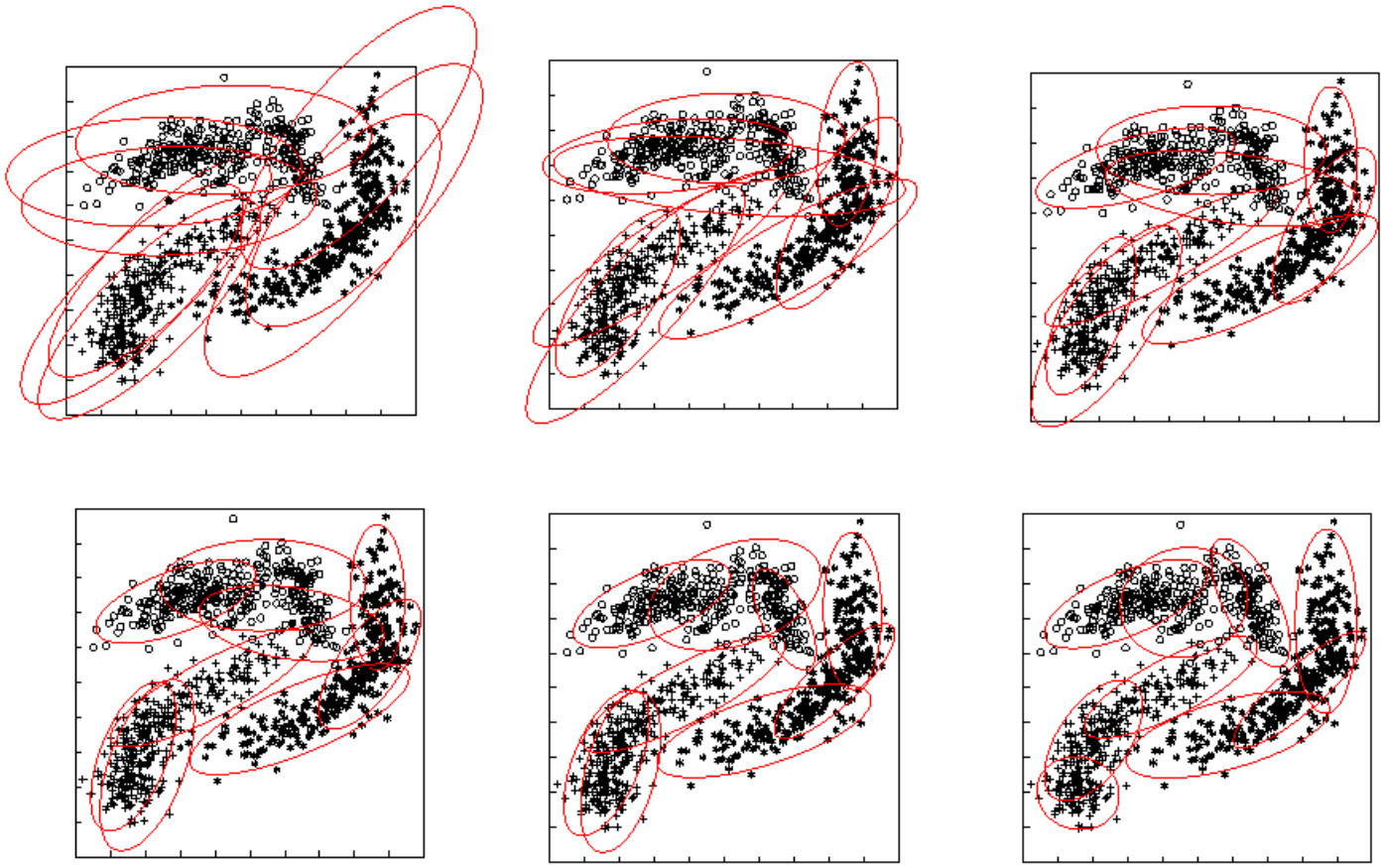
Inicialmente considera cada ejemplo como un cluster.

- Agrupa el par de clusters más cercanos para formar un nuevo cluster.
- Repite el proceso anterior hasta que el número de clusters = C.

Aprendizaje No Supervisado

Clustering (Segmentación). Métodos paramétricos:

El algoritmo EM (Expectation Maximization, Maximum Likelihood Estimate) (Dempster et al. 1977).



Gráficas: Enrique Vidal

Aprendizaje No Supervisado

Clustering (Segmentación). Métodos No Paramétricos

Métodos:

- *k*-NN
- *k*-means clustering,
- online *k*-means clustering,
- centroides
- SOM (Self-Organizing Maps) o Redes Kohonen.

Otros específicos:

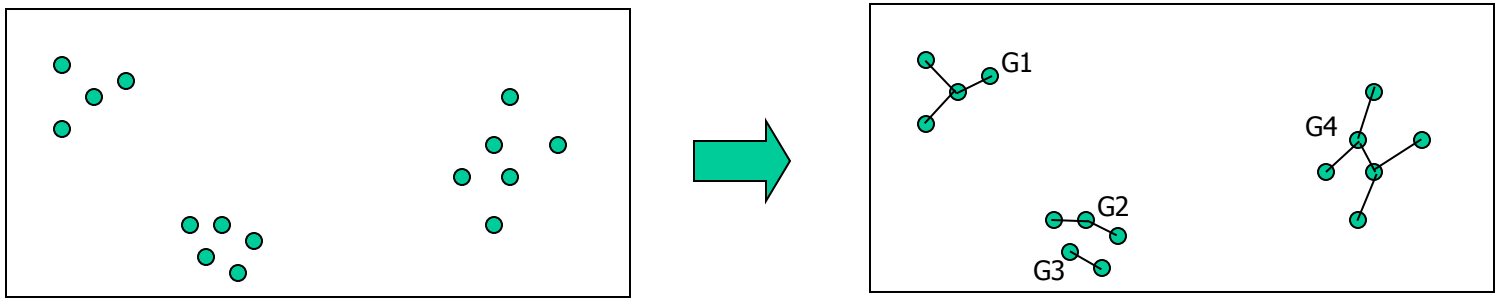
- El algoritmo Cobweb (Fisher 1987).
- El algoritmo AUTOCLASS (Cheeseman & Stutz 1996)

Aprendizaje No Supervisado

Clustering (Segmentación). Métodos No Paramétricos

1-NN (Nearest Neighbour):

Dado una serie de ejemplos en un espacio, se conecta cada punto con su punto más cercano:



La conectividad entre puntos genera los grupos.

A veces hace grupos pequeños.

Existen variantes: k-NN o como el spanning tree que para de agrupar cuando llega a un número de grupos.

Aprendizaje No Supervisado

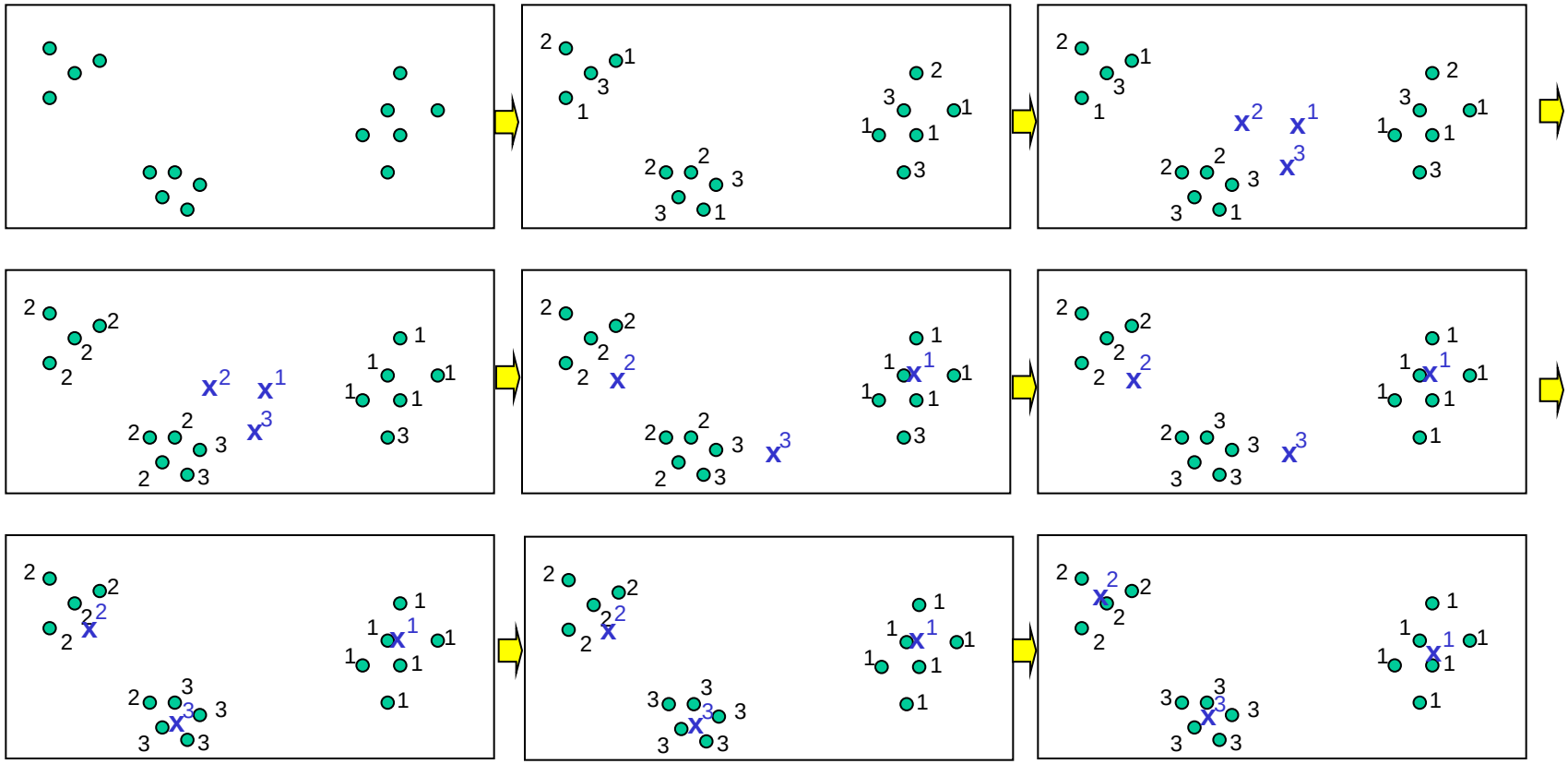
***k*-means clustering**:

- Se utiliza para encontrar los *k* puntos más densos en un conjunto arbitrario de puntos.

- Algoritmo:
  1. Dividir aleatoriamente los ejemplos en *k* conjuntos y calcular la media (el punto medio) de cada conjunto.
  2. Reasignar cada ejemplo al conjunto con el punto medio más cercano.
  3. Calcular los puntos medios de los *k* conjuntos.
  4. Repetir los pasos 2 y 3 hasta que los conjuntos no varíen.

Aprendizaje No Supervisado

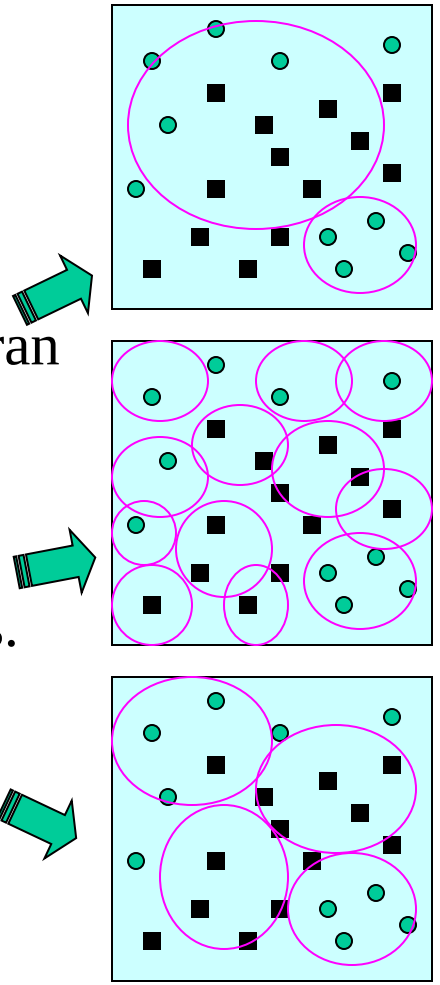
***k*-means clustering**:



Aprendizaje No Supervisado

***k*-means clustering**:

- El valor de *k* se suele determinar heurísticamente.
- Problemas:
  - Si se sabe que hay *n* clases, hacer *k*=*n* puede resultar en que partes de las clases se encuentran dispersas, y el número resulta pequeño.
  - Si *k* se elige muy grande, la generalización es pobre y las clasificaciones futuras serán malas.
  - Determinar el *k* ideal es difícil.



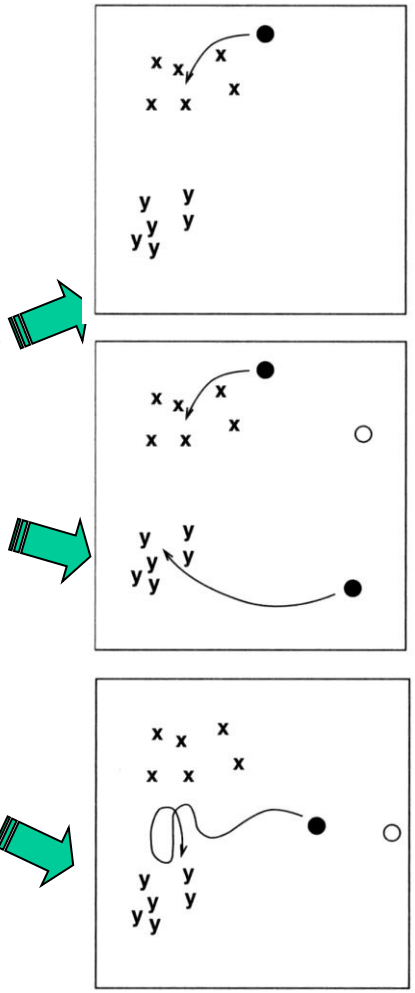
Aprendizaje No Supervisado

K-means

Versión incremental.

El valor de *k* se suele determinar heurísticamente.

- Problemas:
  - Si *k* se elige muy pequeño, hay grupos que se quedan sin centro.
  - Si *k* se elige muy grande, hay centros que se quedan huérfanos.
  - *Aunque esto es preferible a...*
  - Incluso con *k* exacto, puede haber algún centro que quede huérfano.

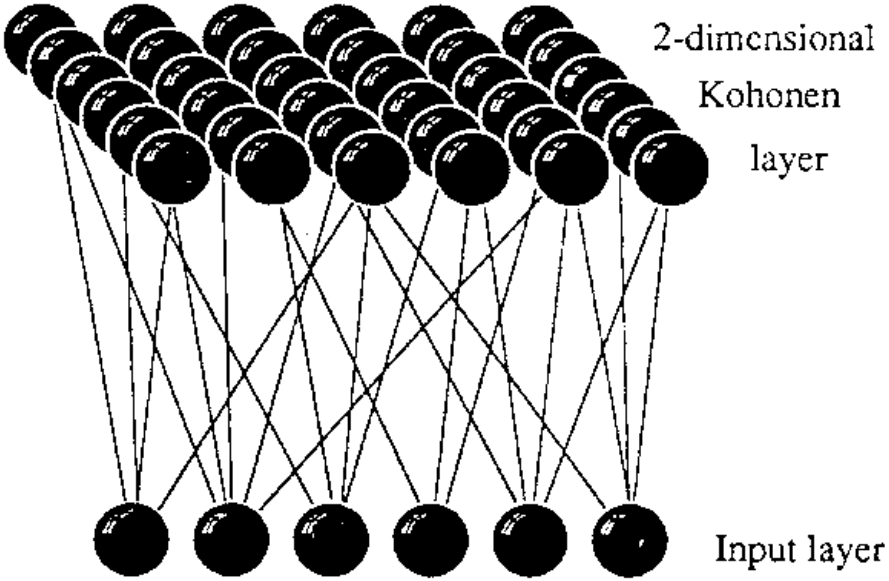


Aprendizaje No Supervisado

Clustering (Segmentación). Métodos No Paramétricos

SOM (Self-Organizing Maps) o Redes Kohonen

También conocidos como redes de memoria asociativa (Kohonen 1984).



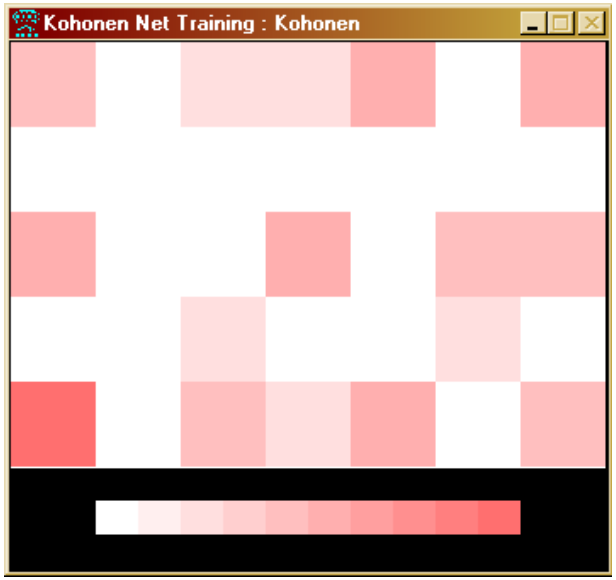
La matriz de neuronas de la última capa forma un grid bidimensional.

Aprendizaje No Supervisado

Clustering (Segmentación). Métodos No Paramétricos

SOM (Self-Organizing Maps) o Redes Kohonen

Durante el entrenamiento cada uno de los nodos de este grid compite con los demás para ganar cada uno de los ejemplos. Finalmente los nodos fuertes (representados con colores más oscuros) ganan más ejemplos que los nodos débiles. Al final del aprendizaje la red se estabiliza y sólo unas pocas combinaciones de pares (X,Y) obtienen registros. Estos son los grupos formados.



También puede verse como una red que reduce la dimensionalidad a 2.
Por eso es común realizar una representación bidimensional con el resultado de la red para buscar grupos visualmente.

Aprendizaje No Supervisado

Análisis Estadísticos:

- Estudio de la distribución de los datos.
- Estimación de densidad.
- Detección datos anómalos.
- Análisis de dispersión (p.ej. las funciones de separabilidad pueden considerarse como técnicas muy simples no supervisadas).

Muchas veces, estos análisis se pueden utilizar previamente para determinar el método más apropiado para un aprendizaje supervisado

También se utilizan mucho para la limpieza y preparación de datos para el uso de métodos supervisados.

Otras Técnicas

- **Redes bayesianas y otros métodos probabilísticos**
- **Computación Evolutiva**
- **Métodos fuzzy**
- **Aprendizaje por refuerzo (iterativo)**
- **…**

Otras Tareas

- **Derivadas de clasificación: ranking, probability estimation, preference learning, quantification, …**
- **Derivadas de regresión: ordinal regression**
- **Derivadas de agrupamiento: subgroup discovery**
- **…**

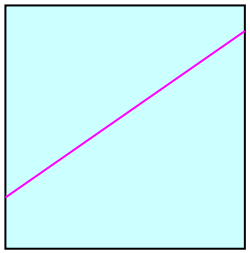
Resumen de métodos: error esperado

Comparación de *accuracy* (k-NN, C4.5 y CS) (de Thornton 2000):

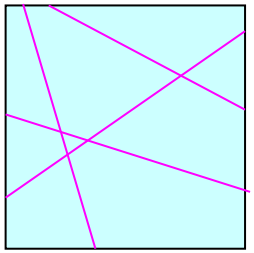
Dataset (del UCI repository)	**C4.5**	**1-NN**	**CS**
BC (Breast Cancer)	72.0	67.31	70.6
CH (chess endgames)	99.2	82.82	89.9
GL (glass)	63.2	73.6	67.19
G2 (GL con clases 1 y 3 combinadas, y 4 a 7 borradas)	74.3	81.6	78.87
HD (heart disease)	73.6	76.24	78.77
HE (hepatitis)	81.2	61.94	62.62
HO (horse colic)	83.6	76.9	77.73
HY (hypothyroid)	99.1	97.76	96.1
IR (iris)	93.8	94.0	95.76
LA (labor negotiations)	77.2	94.74	90.7
LY (lymphography)	77.5	77.03	79.4
MU (agaricus-lepiota)	100.0	100.0	100.0
SE (sick-euthyroid)	97.7	93.19	91.3
SO (soybean-small)	97.5	100.0	99.38
VO (house votes, 1984)	95.6	92.87	92.59
V1 (VO con “physician fee freeze” borrado)	89.4	87.47	89.46
Media:	**85.9**	**84.8**	**85**

Resumen de métodos: expresividad

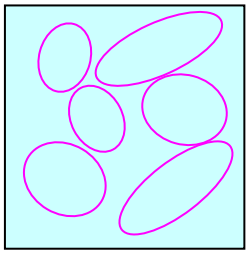
Comparación de representación:



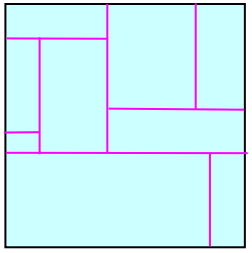
Perceptron / LMS



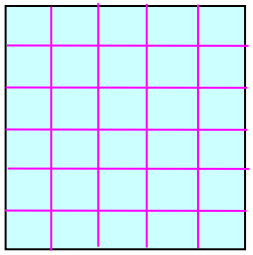
Redes Neuronales Multicapa



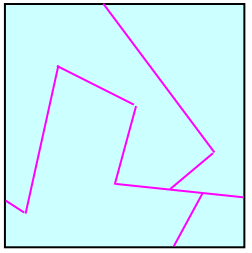
RBF



C4.5/ID3/CART



Naive Bayes Classifier

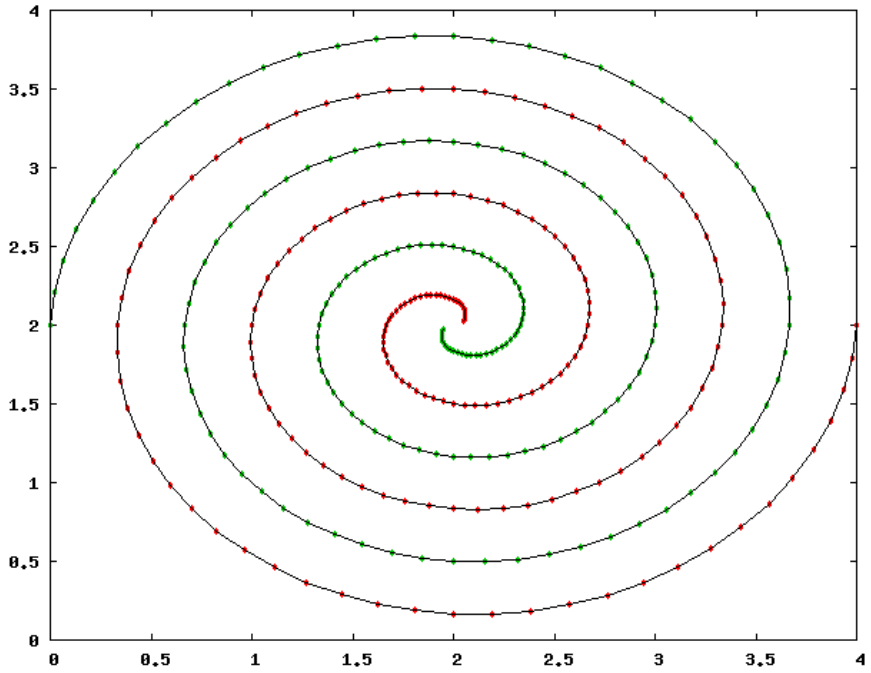


k-NN, LVQ

Resumen de métodos: expresividad

- Fawcett Gallery:

- http://home.comcast.net/~tom.fawcett/public_html/ML-gallery/pages/index.html



Resumen de métodos: expresividad

Limitaciones de los métodos Fence & Fill:

- Están basados en *distancia*, y no capturan muchos conceptos relacionales donde la clase no depende de una proximidad o similitud *espacial o geométrica*.

- Los más expresivos (redes neuronales, LVQ, etc) pueden capturar algunos conceptos relacionales. Incluso las ANN recursivas son un leng. universal. PERO...
  - Lo hacen de manera artificiosa: p.ej. la función paridad no queda definida a partir del número de atributos con un determinado valor sino como una combinación 'arbitraria' de pesos. Además, la red que calcula la paridad de un número de *n* dígitos (entradas) no es la misma que la que lo calcula para un número de *n*+1 dígitos (entradas).

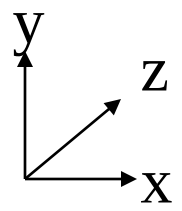
Resumen de métodos: expresividad

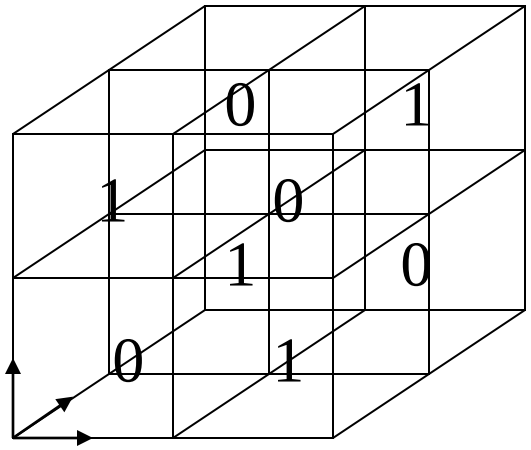
La *continuidad* hacia problemas relacionales.

- Ejemplo. Paridad

$x\ y\ z \rightarrow$ clase

$0\ 0\ 0 \rightarrow 0$
$0\ 0\ 1 \rightarrow 1$
$0\ 1\ 0 \rightarrow 1$
$0\ 1\ 1 \rightarrow 0$
$1\ 0\ 0 \rightarrow 0$
$1\ 0\ 1 \rightarrow 0$
$1\ 1\ 0 \rightarrow 0$
$1\ 1\ 1 \rightarrow 1$





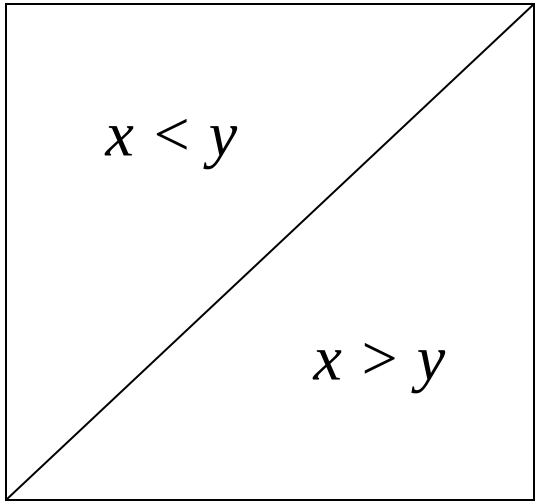
¡¡¡¡No hay manera de hacer grupos geométricos!!!!
La paridad es un problema relacional PURO.

Resumen de métodos: expresividad

La *continuidad* hacia problemas relacionales.

- Ejemplo. ($x > y$)

x	$y \rightarrow$ clase
3	4 → 0
8	2 → 1
7	6 → 1
8	9 → 0



La función '>' es un problema relacional IMPURO.

Algún método no-relacional puede funcionar bien, pero otros no.

Resumen de métodos: expresividad

¿Relacional o No? Funciones Heurísticas:

Permiten determinar el grado de continuidad o separabilidad, considerando una medida de distancia. Si la separabilidad es baja, debemos intentar métodos no basados en distancias.

- Separabilidad Lineal (Minsky and Papert 1988)

  Problema: Muy pobre. Muchos métodos no relacionales son capaces de aprender aunque los datos no sean separables linealmente.

- Separabilidad Geométrica de Thornton (1997)

$$GS(f) = \frac{\sum_{i=1}^{n} eq(f(e_i), f(nn(e_i)))}{n}$$

donde $f(\cdot)$ es la función definida por los datos, $nn(\cdot)$ es el vecino más cercano y $eq(a,b)$ =1 sii $a=b$. Si no, =0.

Problema: depende mucho del número de ejemplos.

Resumen de métodos: expresividad

Funciones Heurísticas.

- Separabilidad Geométrica Radial:

  Porcentaje medio de los ejemplos tales que sus ejemplos próximos en un radio r son de la misma clase.

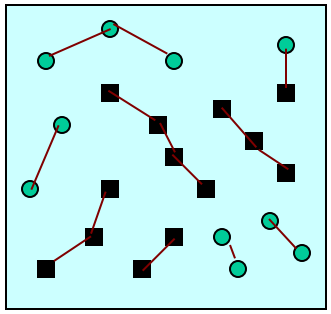
$$RGS(f) = \frac{\sum_{i=1}^{n} \frac{\sum_{e_j : dist(e_i, e_j) < r} eq(f(e_i), f(e_j))}{\left| e_j : dist(e_i, e_j) < r \right|}}{n}$$

El radio a utilizar se puede calcular a partir de la densidad de los ejemplos.

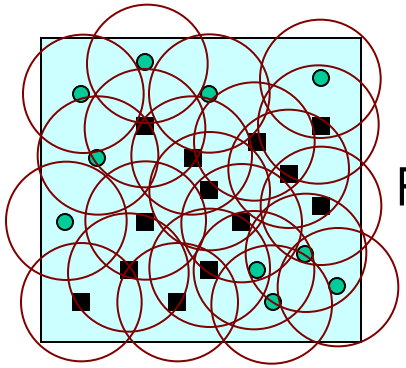
También se puede calcular una RGS' que no incluye el propio punto.

Resumen de métodos: expresividad

- Ejemplos:

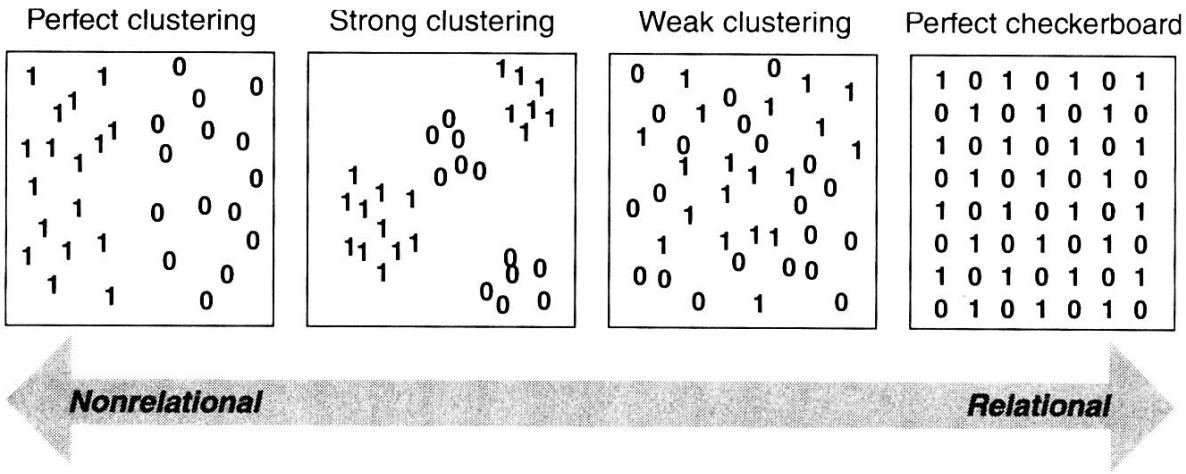


GS = 21/23 = 0.91



RGS = 18.44/23 = 0.8

- GS(Paridad) = 0
- GS(Mayor) = 1 (suponiendo infinitos ejemplos)



Resumen de métodos: expresividad

Los métodos proposicionales no son capaces de:

- detectar relaciones entre varios ejemplos o entre partes complejas del mismo ejemplo.
- aprender funciones recursivas.

¿En qué casos es necesario expresividad relacional y/o recursiva?

Veamos un ejemplo que sí y otro que no...

Resumen de métodos: expresividad

EJEMPLO. Aprender el concepto *daughter* con LINUS (Lavrac et al. 1991):

B= { parent(eve, sue). parent(ann, tom). parent(pat,ann). parent(tom,sue).
female(ann). female(sue). female(eve). }

E^+= { daughter(sue,eve). daughter(ann,pat). }

E^-= { daughter(tom,ann). daughter(eve,ann). }

LINUS transforma *B* y *E* a un problema de atributos (proposicional):

Clase	Variables		Atributos proposicionales						
	X	Y	fem(X)	fem(Y)	par(X,X)	par(X,Y)	par(Y,X)	par(Y,Y)	X=Y
+	sue	eve	true	true	false	false	true	false	false
+	ann	pat	true	false	false	false	true	false	false
–	tom	ann	false	true	false	false	true	false	false
–	eve	ann	true	true	false	false	false	false	false

Resultado del aprendizaje de atributos (p.ej. C4.5):

class = + if (female(X) = true) $\wedge$ (parent(Y,X) = true)

LINUS transforma de nuevo a Prolog:

daughter(X,Y) :- female(X), parent(Y,X).

Es simplemente un ejemplo de Pick & Mix

Resumen de métodos: expresividad

EJEMPLO. Aprender el problema no. 47 de Bongard (I.Q. tests) :

E+= { shape(case1, s1-1, triangle, small, up). shape(case1, s1-2, circle, large, n_a).
shape(case2, s2-1, triangle, large, up). shape(case2, s2-2, circle, small, n_a).
shape(case2, s2-3, square, large, n_a). in(s1-1,s1-2). left(s2-1,s2-2). }

E-= { left(s1-1,s1-2). in(s2-1,s2-2). }

Podríamos transformarla a un problema de atributos (proposicional):

caso	clase	shape1	size1	conf1	shape2	size2	conf2	1 in 2	1 left to 2	shape3	size3	conf3	1 in 3	2 in 3	1 left to 3	2 left to 3	shape4	...
1	+	triangle	small	up	circle	large	-	yes	no	-	-	-	-	-	-	-	-	...
2	+	triangle	large	up	circle	small	-	no	yes	square	large	-	-	-	-	-	-	...

Problemas:

- Muchos atributos (y muchos de ellos vacíos).
- Ambigüedad (existen múltiples representaciones para el mismo ejemplo). Una mala solución puede depender de esta representación.

  P.ej.: Clase = + if shape1 = triangle

El aprendizaje relacional se necesita estrictamente cuando los ejemplos consisten de un número *variable* de objetos y de las relaciones entre estos objetos son importantes.

Resumen de métodos: expresividad

EJEMPLOS.

- MEMBER:

```
member(X,[X|Z]).
member(X,[Y|Z]):- member(X,Z).
```

- RELATIVE:

```
ancestor(X,Y):- parent(X,Y).
ancestor(X,Y):- parent(X,Z), ancestor(Z,Y).
relative(X,Y) :- ancestor(X,W), ancestor(Y,W).
```

- REACH:

```
reach(X,Y):- link(X,Y).
reach(X,Y):- link(X,Z), reach(Z,Y).
```

La recursividad se requiere cuando la profundidad (el nivel) de las relaciones no se conoce a priori (objetos que contienen o se relacionan con un número variable de objetos).

Resumen de métodos: expresividad

Modelos Estructurales de Grammar Learning:

- Aprendizaje de autómatas aceptadores de gramáticas.
- Gramáticas regulares estocásticas.
- Lenguajes probabilísticos: cadenas de Markov, algoritmo de Viterbi

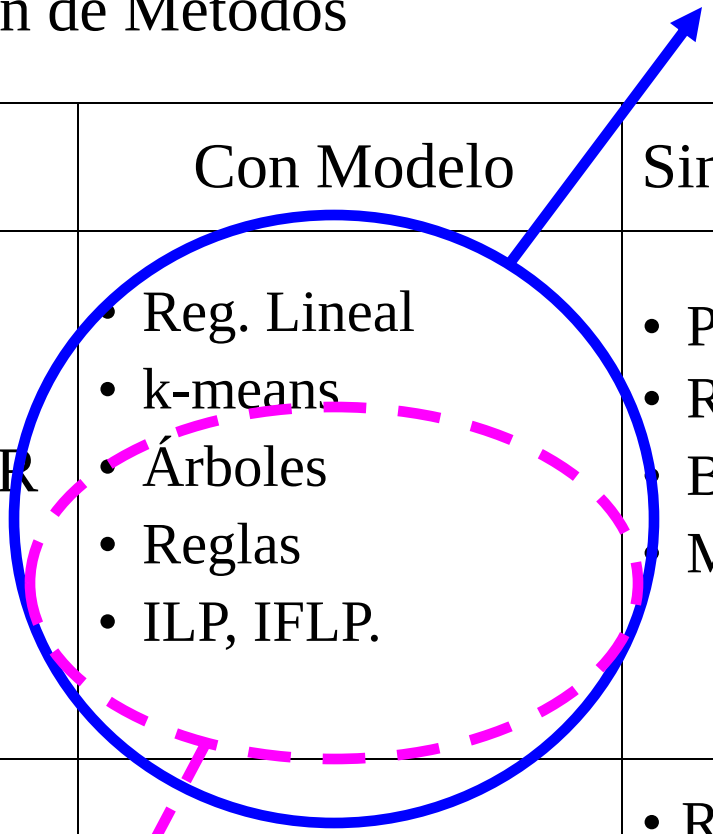
Más información "Aprendizaje y Percepción" (semestre 4B)

Aprendizaje multirelacional:

- **IFP (Inductive Functional Programming), ILP (Inductive Logic Programming), IFLP (Inductive Functional Logic Programming):**

Resumen de Métodos: conocimiento

Resumen de Métodos



	Con Modelo	Sin Modelo o no inteligible
EAGER	• Reg. Lineal • k-means • Árboles • Reglas • ILP, IFLP.	• Perceptron Learning, ANN. • Radial Basis Functions. • Bayes Classifiers. • Métodos kernel y SVM
LAZY		• Reg. Lineal Pond. Local • CBR • k-NN (Nearest Neighbour).

1.3. Técnicas que generan modelos comprensibles: árboles de decisión y sistemas de reglas

Ventajas de modelos en forma de reglas

El conocimiento extraído es un conjunto de reglas

- Comprensibilidad
- Revisión y mantenimiento más fácil (se pueden añadir, modificar o eliminar reglas).
- Integración con sistemas de reglas y sistemas expertos.
- Codificación directa en aplicaciones software.
- Importación y exportación en bases de conocimiento (PMML, RuleML, etc.).
- Detección automática de inconsistencias.
- Fusión con otros modelos más sencilla.

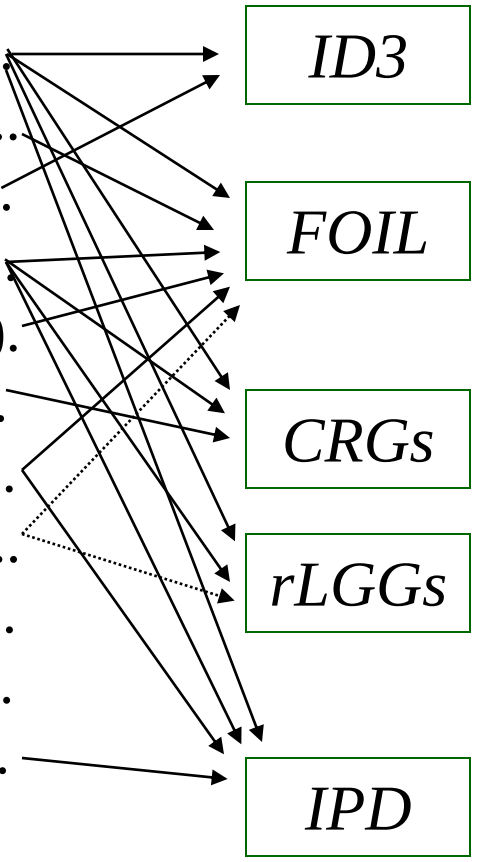
Árboles de Decisión

- Sistemas de atributos o proposicionales (ID3, C4.5, CART):
  - No permiten recursividad.
  - No permiten estructura en los atributos.
  - Condiciones muy limitadas.
- Sistemas relacionales (FOIL, FFOIL, FOIDL, IPD, TILDE):
  - Tienen limitaciones.
  - Necesitan más preparación en los datos.
  - Son más lentos.

Árboles de Decisión

TIPOS DE CONDICIONES:

- Uso de igualdades con constantes.....
- Uso de desigualdades (< >) con constantes.....
- Uso de comparaciones (<, >) con constantes.....
- Comparación de variables en positivo (X=Y)....
- Uso de negación (o operador < > con variables).
- Introducción de funciones (X = f(Y,Z,...))
- Introducción de funciones recursivas.....
- Pick & Mix libre.....
- Pick & Mix con uso de enlaces (CAj p.ej.).....
- Invención de funciones/predicados.....
- Generadas por algoritmos de aprendizaje.....



TIPOS DE RETORNO:

- Constante
- Con variable

Árboles de Decisión

Construcción:

- SPLITTING ALGORITHM (ID3, C4.5, CART, ASSISTANT, IPD)
  - Llamado también DIVIDE-AND-CONQUER
  - Asume las ramas son disjuntas (no solapan).

- COVERING ALGORITHM (AQ, CN2, FOIL)
  - Llamado también SEPARATE-AND-CONQUER

Árboles de Decisión

SPLITTING ALGORITHM (ID3, C4.5, CART, ASSISTANT)

```
Algorithm CART(Target_function, Pos, Neg)
    node = f(X1, X2, ..., Xn)
    split_and_conquer(∅, &node, Pos, Neg);
    return node;
endalgorithm

Function split_and_conquer(condition, node, Pos, Neg);
    if Neg <> ∅ then      // Añadir condiciones para especializar la regla
       Candidate_Split := generar posibles condiciones
       Best_Split := seleccionar la mejor condición según el criterio de ganancia.
       Link Node with every node of Best_Split (as children)
       for each node ∈ Best_Split do
          cond:= condition of node;
          Pos:= Pos which are true under cond.    Neg:= Neg which are true under cond.
          split_and_conquer(cond, &node, Pos, Neg);
       endfor;
    endif
    return node;
endfunction
```

con sólo dos clases

Árboles de Decisión

COVERING ALGORITHM (AQ, CN2, FOIL, FFOIL)

con sólo dos clases

FOIL(Target_predicate, Predicates, Pos, Neg)
Learned_rules := ∅
while Pos <> ∅ **do** // Aprender una nueva regla
    NewRule :- el target_predicate con todo variables y sin condiciones.
    NewRuleNeg :- Neg;
    while NewRuleNeg <> ∅ **do** // Añadir condiciones para especializar la regla
        Candidate_Literals :- generar posibles condiciones (utilizando el cjto. Predicates)
        Best_Literal :- seleccionar el mejor literal utilizando el criterio de ganancia.
        Añadir este Best_Literal a las condiciones de NewRule;
        Quitar a NewRuleNeg los ej. neg. que ya no son cubiertos (incumplen Best_Literal)
    endwhile
    Learned_rules :- Learned_rules ∪ { NewRule}
    Pos :- Pos - {miembros de Pos cubiertos por la nueva regla).
endwhile;
Retornar Learned_rules;

Árboles de Decisión

SPLITTING vs COVERING

Ventajas del SPLITTING:

- Aprovecha caminos ya abiertos (más eficiente)
- Poda más sencilla.
- Representación generalmente más corta y eficiente. Se pueden utilizar DECISION LISTS (uso del cut) en vez de DECISION TREES.

Ventajas del COVERING:

- Permite hacer coberturas no totales (interesante cuando un atributo tiene muchos valores y sólo algunos son significativos).
- Es menos voraz y las últimas ramas se ajustan más a los ejemplos. Esto también es una desventaja, porque las últimas ramas suelen hacer overfitting.

Árboles de Decisión

Funciones de Ganancia:

ID3, C4.5 (y ≅FOIL) (Quinlan 1993):

C: nº de clases,
p(*E*,*j*): proporción de los casos de *E* de la clase *j*
T: test (split) considerado.
E_i: evidencia que cumple cada uno de los *k* resultados del test.

$$\text{Info}(E) = -\sum_{j=1}^{C} p(E, j) \times \log_2(p(E, j))$$

$$Gain(E,T) = Info(E) - \sum_{i=1}^{k} \frac{|E_i|}{|E|} \times Info(E_i)$$

$$Split(E,T) = -\sum_{i=1}^{k} \frac{|E_i|}{|E|} \times \log_2\left(\frac{|E_i|}{|E|}\right)$$

$$GainRatio(E,T) = \frac{Gain(E,T)}{Split(E,T)}$$

De las particiones (splits) con al menos GAIN medio, se selecciona el que tenga el GAINRATIO mayor.

Árboles de Decisión

Funciones de Ganancia:

Ganancia Mejorada para Valores continuos:

C4.5 (Quinlan 1996)

$$AdjGain(E,T) = Gain(E,T) \times \frac{\log_2(N-1)}{|E|} \text{ for continuous splits}$$

$$AdjGain(E,T) = Gain(E,T) \text{ for non-continuous splits}$$

$$AdjGainRatio(E,T) = \frac{AdjGain(E,T)}{Split(E,T)}$$

Árboles de Decisión

Funciones de Ganancia:

CART (Breiman et al. 1984):

Dados los datos de entrenamiento *T* que caen en un node, y tenemos n clases, el índice gini(T) se define como:

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

Cuando se parte *T* en dos subconjuntos *T1* y *T2* con tamaños *N1* y *N2*, el índice después del split se define como:

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

Se elige el valor menor.

Árboles de Decisión

Funciones de Ganancia:

Para dos clases:

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

$$= 1 - p^+(1 - p^-) - p^-(1 - p^+) = 1 - p^+ + p^+ p^- - p^- + p^+ p^- =$$

$$= 2 p^+ p^-$$

Con lo que tenemos (para dos hijos):

$$\text{Gini}(E) = \left(\frac{|P|}{|E|}\right)\left(\frac{|N|}{|E|}\right) = \left(\frac{|P||N|}{|E|^2}\right)$$

$$Gain(E, cond) = Gini(E) - [(r(Gini(E_{neg}) + (1 - r)Gini(E_{pos}))]$$

$$with \;\; r = \frac{|E_{neg}|}{|E|}$$

Donde E_{neg} son los que no cumplen cond

Árboles de Decisión

Otras Medidas para Split:

- Ortogonalidad de GID3 (Fayyad 1994)
- DKM (Dietterich et al. 1996)

$$DKM(T) = \sqrt{p^+ p^-}$$

- MSE, LogLoss, AUC, etc…

La mayoría de los métodos se benefician del uso de smoothing, tanto en el cálculo de la probabilidad de las clases en cada nodo, como en el cálculo de la probabilidad de cada nodo.

Árboles de Decisión

Incorporación de Recursividad:

Chequeo extensional. FOIL o FFOIL:

- Cuando se intenta introducir una llamada recursiva en el árbol, su valor de verdad se extrae de los ejemplos positivos.
- Esto causa problemas, p.ej.
  - member(X,[Y|Z]) :- member(X,Z)
  - no cubriría el ejemplo member(a, [c,b,a]) con respecto a los otros si member(a, [b,a]) no estuviera en los ejemplos.

Chequeo intensional. FOIDL:

- Cuando se intenta introducir una llamada recursiva en el árbol, su valor de verdad se extrae de los ejemplos positivos y de llamadas a la propia función.
  - En este caso bastaría con tener member(a, [a])

También hay sistemas que permiten utilizar las ramas cerradas como verdaderas y otros no. En este caso sería suficiente unos cuantos member(x, [x]) que se hubieran generalizado.

Árboles de Decisión

Incorporación de Recursividad. Chequeo de Terminación

Aproximación a terminación. (FOIL o FFOIL):

- Excluye algunas soluciones interesantes.

Límite de pasos en la comprobación de cláusulas recursivas. (FOIDL):

- Ralentiza los algoritmos. Se suele hacer que el límite de pasos dependa del tamaño del ejemplo a probar.

Árboles de Decisión

PODA DE ÁRBOLES:

Una vez realizado el árbol, muchos sistemas pospodan (postpruning) el árbol, con el objetivo de obtener árboles más cortos (con menos condiciones) y, por tanto, más generales.

- Poda de árboles por redundancia: (pueden existir condiciones superfluas, debido a la manera como se construye el árbol)
- Poda de árboles por ruido o principio MDL: (se poda porque se supone que algunos nodos han sido introducidos porque puede haber ruido).

Árboles de Decisión

PODA DE ÁRBOLES:

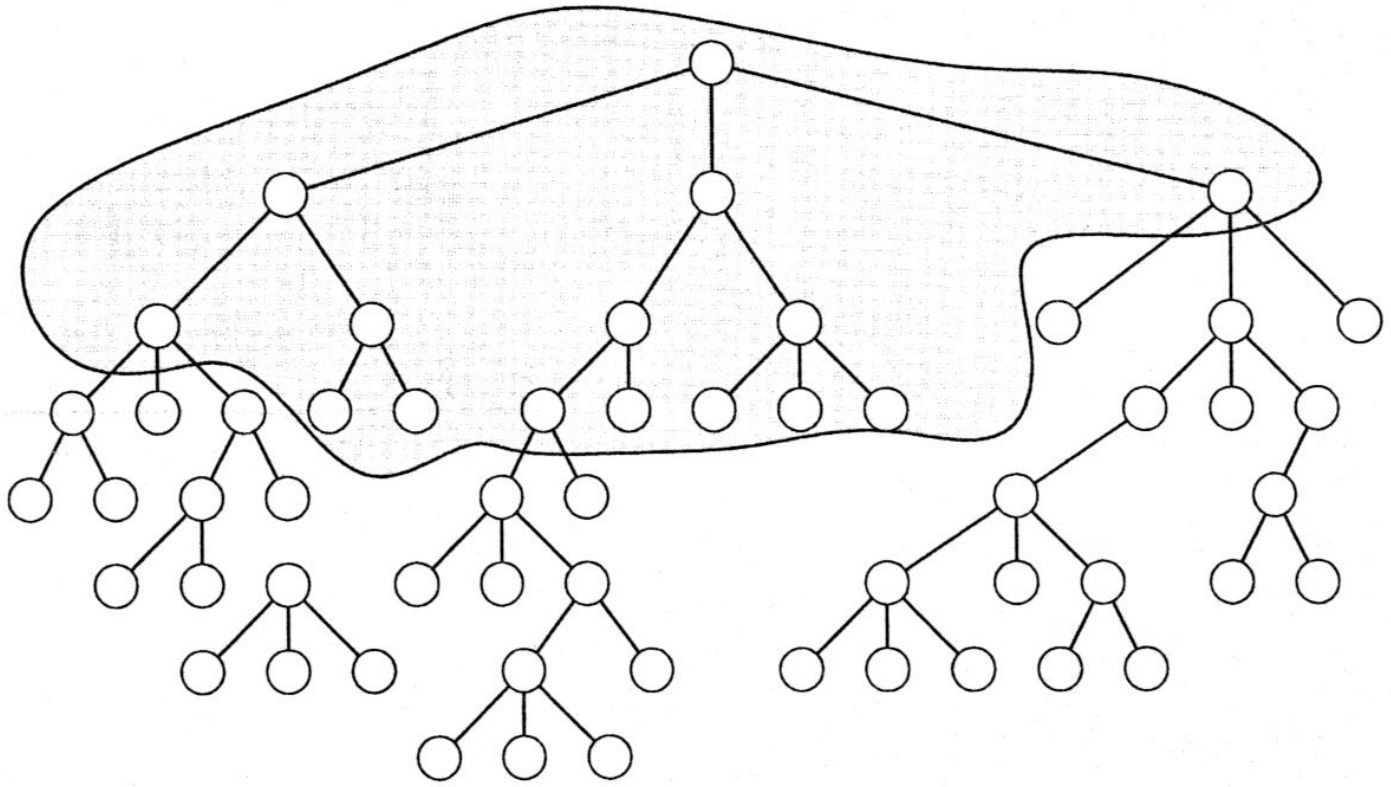


Figure 11.10 The challenge of pruning is to find the best subtree.

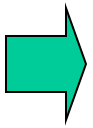
Árboles de Decisión

Poda de árboles por redundancia:

Ejemplo:

Clase	Atributos y Valores				
	Is_Smiling	Holding	Has_tie	Head_shape	Body_shape
friendly	yes	balloon	yes	square	square
friendly	yes	flag	yes	octagon	octagon
unfriendly	yes	sword	yes	round	octagon
unfriendly	yes	sword	no	square	octagon
unfriendly	no	sword	no	octagon	round
unfriendly	no	flag	no	round	octagon

friendly if Is_Smiling=yes ^ Holding<>sword
unfriendly if Is_Smiling=no
unfriendly if Is_Smiling=yes ^ Holding=sword

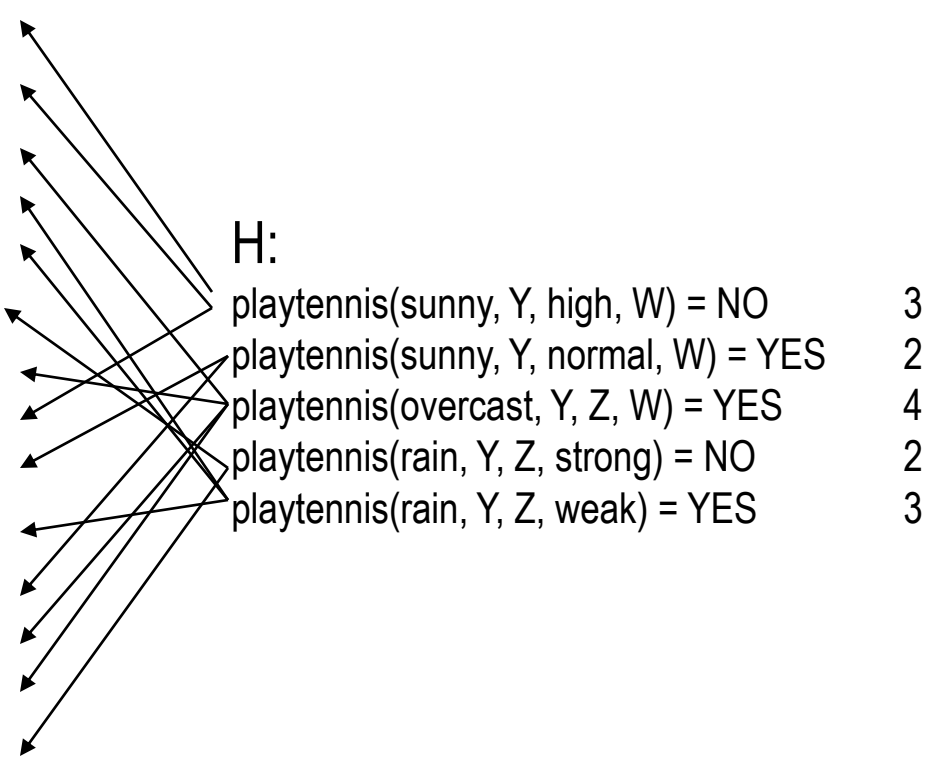


unfriendly if Is_Smiling=no
unfriendly if Holding=sword

Árboles de Decisión

Poda de árboles por ruido o principio MDL:

Ex.	Sky	Temp	Humidity	Wind	PlayTennis
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overc.	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overc.	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overc.	Mild	High	Strong	Yes
13	Overc.	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No



Árboles de Decisión

Poda de árboles por ruido o principio MDL:

Ex.	Sky	Temp	Humidity	Wind	PlayTennis
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overc.	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overc.	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	**Sunny**	**Cool**	**Normal**	**Weak**	**No**
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overc.	Mild	High	Strong	Yes
13	Overc.	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

ruido

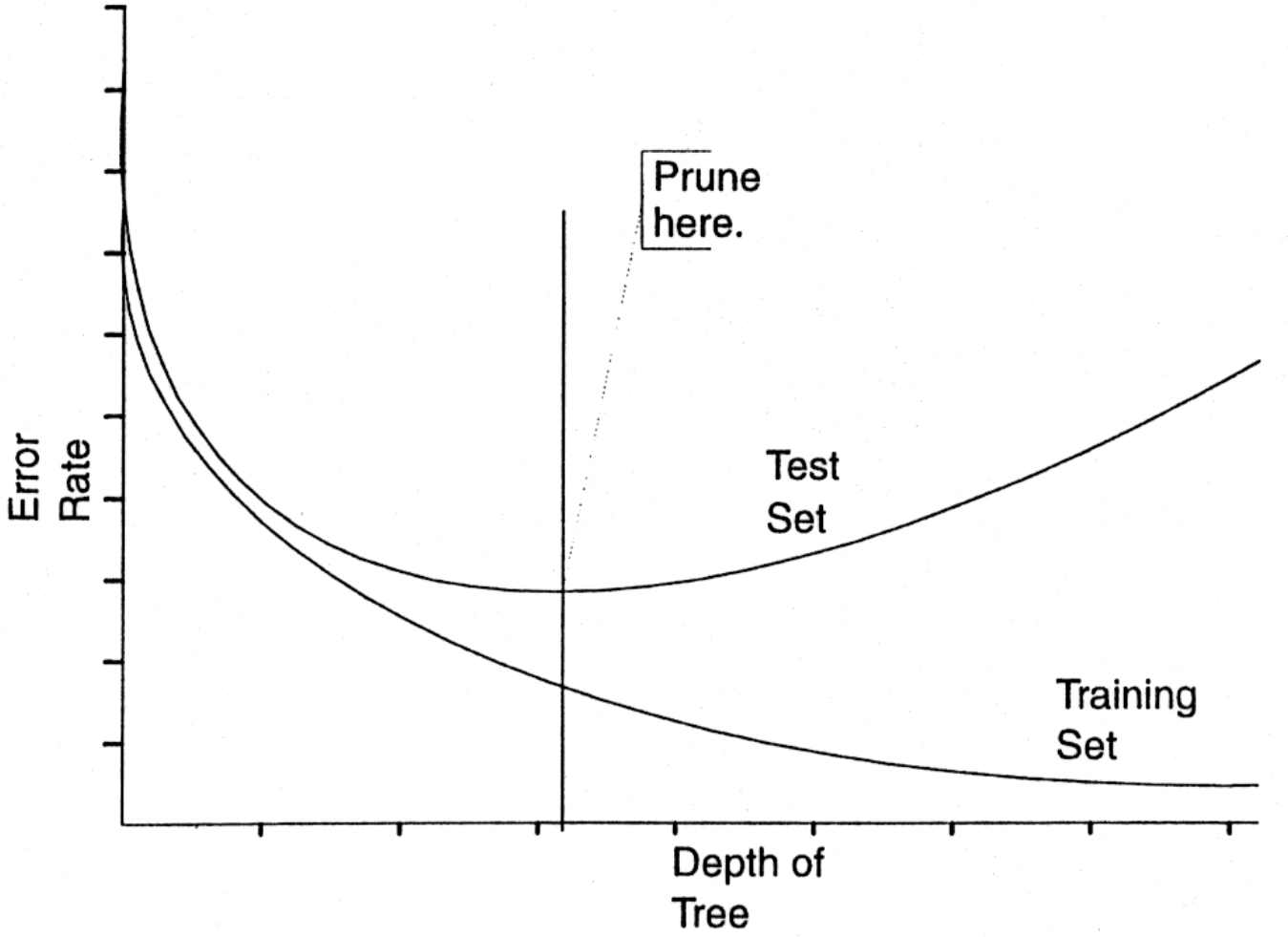
H:

	nº de ejs.
playt(sunny, Y, high, W) = NO	3
playt(sunny, mild, normal, W) = YES	1
playt(overcast, Y, Z, W) = YES	4
playt(rain, Y, Z, strong) = NO	2
playt(rain, Y, Z, weak) = YES	3
playt(sunny, cool, normal, W) = NO	1

Es preferible podar esas dos reglas con poca cobertura (1).

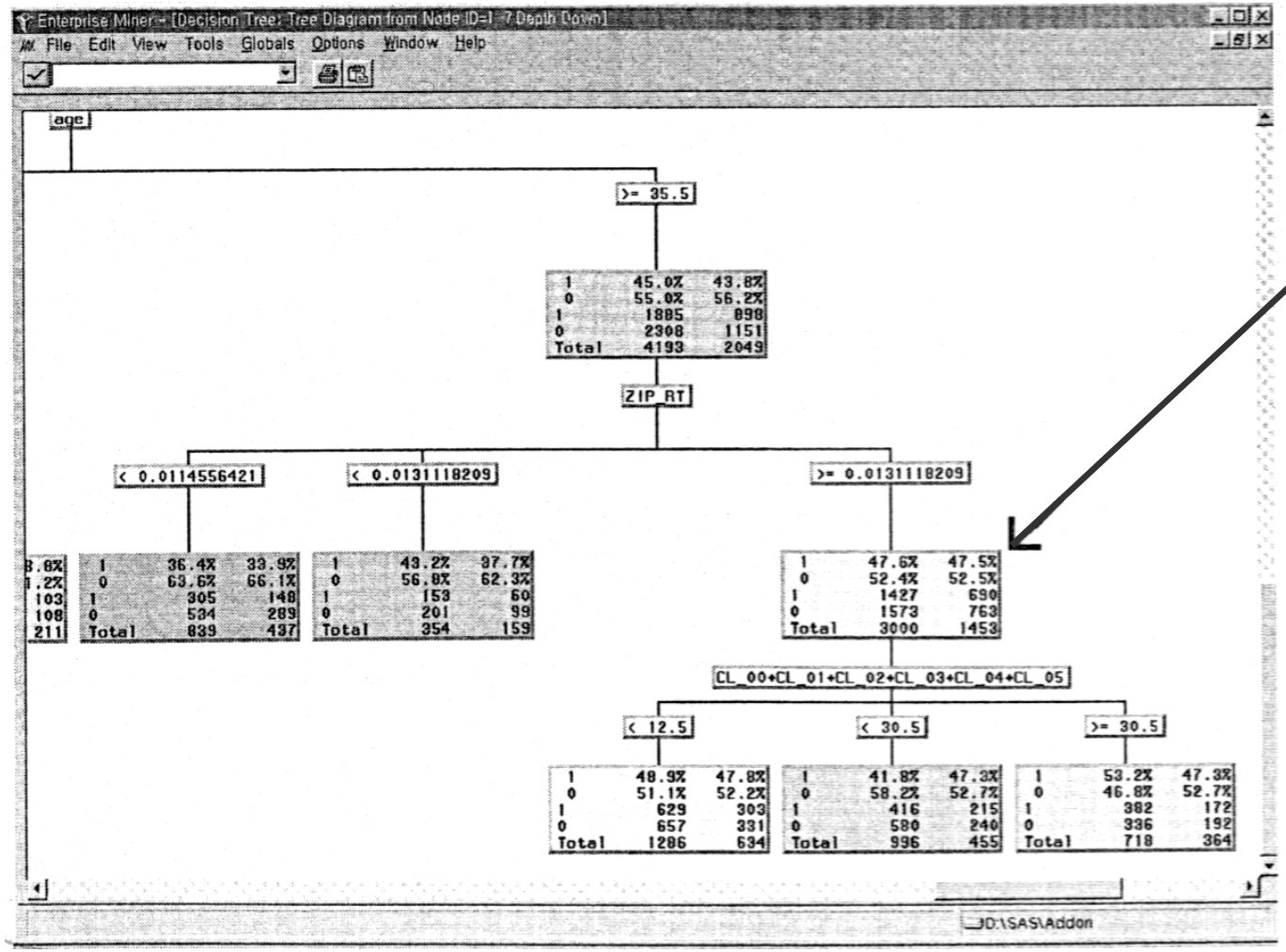
Árboles de Decisión

Poda de árboles utilizando validación cruzada:



Árboles de Decisión

Poda de árboles utilizando validación cruzada:



Para ver qué hojas podar se examina la diferencia de cada subárbol en el comportamiento con el training y test set:

Sistemas de Reglas

Basados normalmente en sistemas de cobertura:

- CN2 es el más conocido, pero existen muchas variantes:
  - Con diferentes formas de representación de las reglas (fuzzy, proposicional, primer orden, …)
  - Con solape o sin solape
  - Basadas en reglas de asociación

- Existen sistemas de reglas fuzzy y otros formalismos que permiten extraer reglas comprensibles de otros modelos.

1.4. El caso de la Minería de Datos

Minería de Datos: cuando los datos son BD

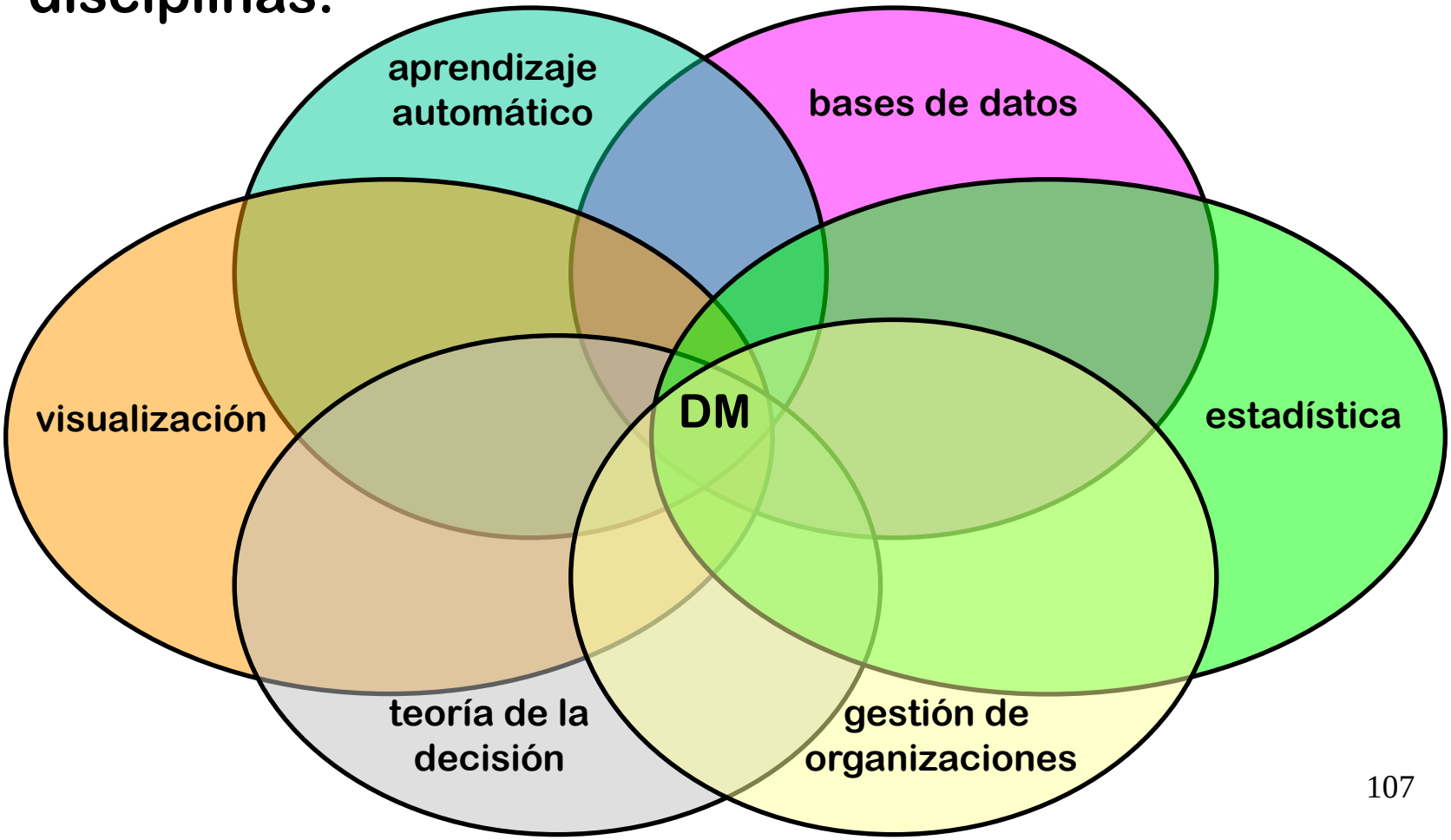
Cuando los datos provienen de bases de datos:

- **"Descubrimiento de Conocimiento a partir de Bases de Datos" (KDD, del inglés *Knowledge Discovery from Databases*).**

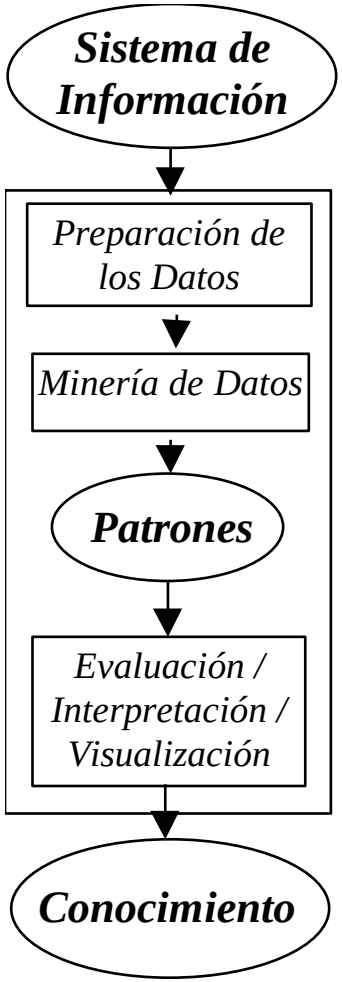
  "*proceso no trivial de identificar patrones válidos, novedosos, potencialmente útiles y en última instancia comprensibles a partir de los datos*". Fayyad et al. 1996

Relación de DM con Otras Disciplinas

- KDD nace como interfaz y se nutre de diferentes disciplinas:



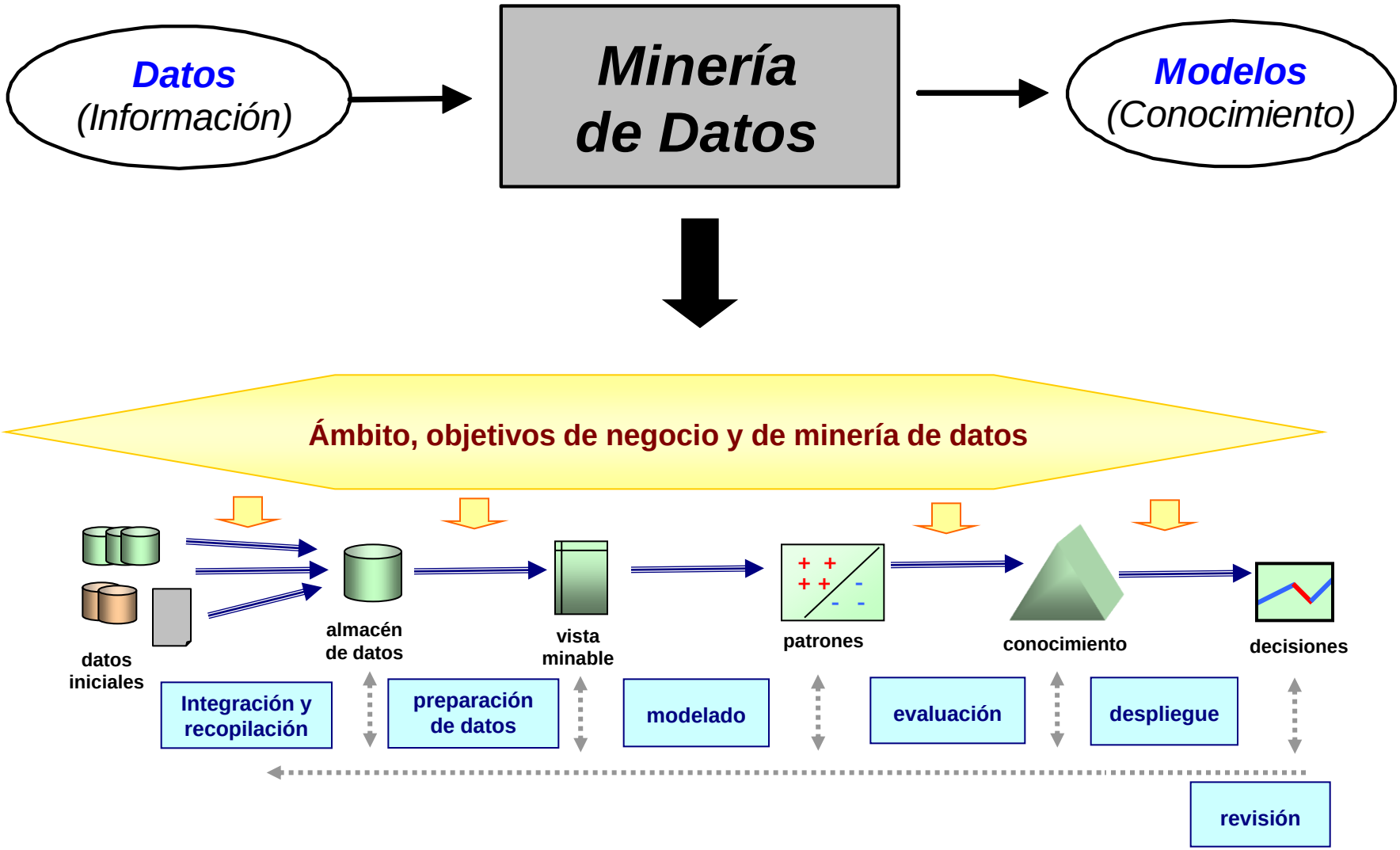
El Proceso del KDD. FASES



KDD

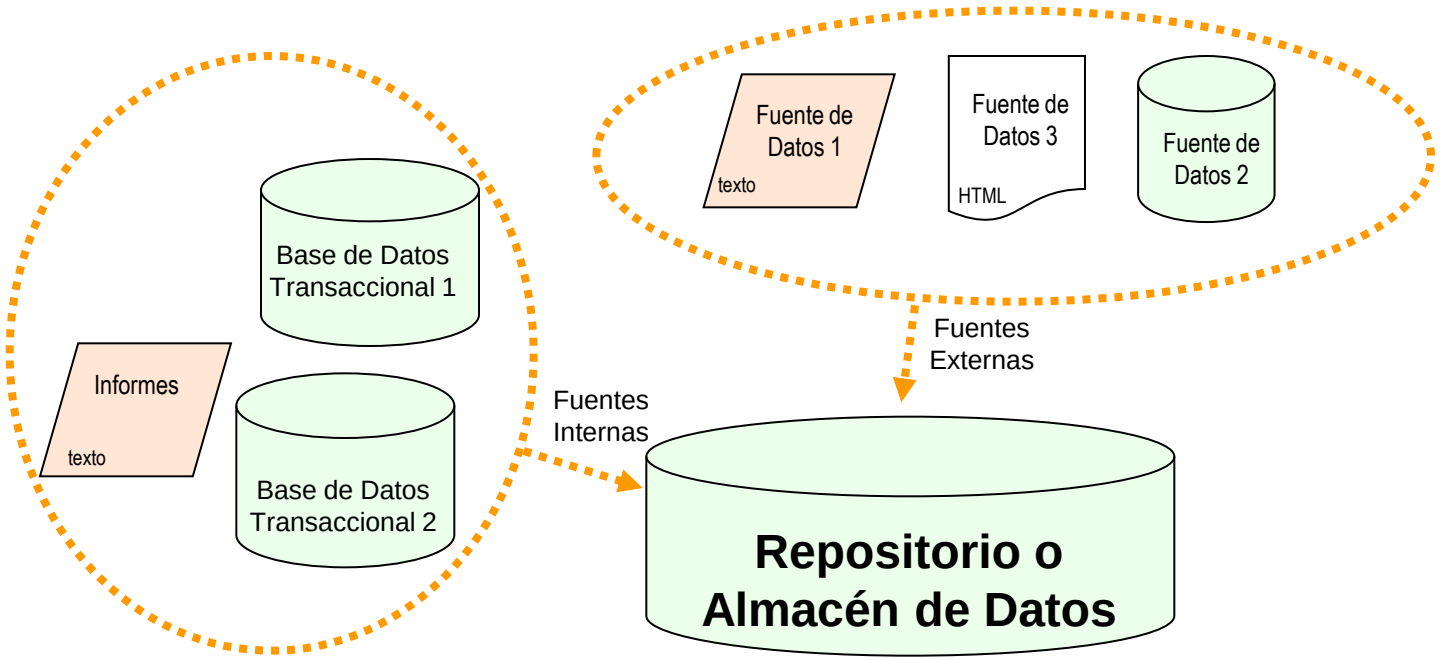
1. **Determinar las fuentes de información que pueden ser útiles y dónde conseguirlas.**
2. **Diseñar el esquema de un almacén de datos (Data Warehouse) que consiga unificar de manera operativa toda la información recogida.**
3. **Implantación del almacén de datos que permita la "navegación" y visualización previa de sus datos, para discernir qué aspectos puede interesar que sean estudiados.**
4. **Selección, limpieza y transformación de los datos que se van a analizar. La selección incluye tanto una criba o fusión horizontal (filas) como vertical (atributos).**
5. **Seleccionar y aplicar el método de minería de datos apropiado.**
6. **Evaluación, interpretación, transformación y representación de los patrones extraídos.**
7. **Difusión y uso del nuevo conocimiento.**

El Proceso del KDD. FASES



Integración de Datos

- **Recogida de Información**



Fases del KDD: Preparación de Datos

Limpieza (data cleansing) y criba (selección) de datos:

- **Se deben eliminar el mayor número posible de datos erróneos o inconsistentes (limpieza) e irrelevantes (criba).**

Métodos estadísticos casi exclusivamente.

- **resúmenes e histogramas (detección de datos anómalos).**
- **selección de datos (muestreo, ya sea verticalmente, eliminando atributos, denominado "selección de características", u horizontalmente, eliminando tuplas, denominado "muestreo").**
- **redefinición de atributos (agrupación o separación).**

Fases del KDD: Preparación de Datos

La selección y la limpieza pueden acompañarse de "transformación" de atributos (numerización, discretización, ...).

- El resultado es un conjunto de filas y columnas denominado:

VISTA MINABLE

- La vista minable integra datos de diferentes fuentes, los limpia, selecciona y transforma, y los tipa, con el fin de prepararlos para la modelización.

Fases del KDD: La Minería de Datos

Patrones a descubrir:

- Una vez recogidos los datos de interés, un explorador puede decidir qué tipo de patrón quiere descubrir.
- El tipo de conocimiento (*tarea*) que se desea extraer va a marcar claramente las *técnicas posibles* de minería de datos a utilizar.
  - **Selección del algoritmo** o algoritmos (técnicas) a aplicar para obtener el modelo.
  - Selección de los **valores de los parámetros** del algoritmo.
  - **Aplicación/Entrenamiento** del algoritmo.

Fases del KDD: Evaluación y Validación

La fase anterior produce una o más hipótesis de modelos.

- Para seleccionar y validar estos modelos es necesario el uso de **criterios de evaluación de hipótesis**. Por ejemplo:

  1ª Fase: Comprobación de la precisión del modelo en un **banco de ejemplos independiente** del que se ha utilizado para aprender el modelo. Se puede elegir el mejor modelo.

  2ª Fase: Se puede realizar una **experiencia piloto** con ese modelo. Por ejemplo, si el modelo encontrado se quería utilizar para predecir la respuesta de los clientes a un nuevo producto, se puede enviar un mailing a un subconjunto de clientes y evaluar la *fiabilidad del modelo*.

Fases del KDD: Interpretación y Difusión

El despliegue del modelo a veces es trivial pero otras veces requiere un proceso de implementación o interpretación:

- El modelo puede requerir **implementación** (p.ej. tiempo real detección de tarjetas fraudulentas).
- El modelo es descriptivo y requiere **interpretación** (p.ej. una caracterización de zonas geográficas según la distribución de los productos vendidos).
- El modelo puede tener muchos usuarios y necesita **difusión**: el modelo puede requerir ser expresado de una manera comprensible para ser distribuido en la organización (p.ej. las cervezas y los productos congelados se compran frecuentemente en conjunto $\Rightarrow$ ponerlos en estantes distantes).

Fases del KDD: Actualización y Monitorización

Los procesos derivan en un mantenimiento:

- Actualización: Un modelo válido puede dejar de serlo: cambio de contexto (económicos, competencia, fuentes de datos, etc.).
- Monitorización: Consiste en ir revalidando el modelo con cierta frecuencia sobre nuevos datos, con el objetivo de detectar si el modelo requiere una actualización.

Producen realimentaciones en el proceso KDD.

Más información

Bibliografía

Asignatura AMD