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

T.1 INTRODUCCIÓN

José Hernández Orallo

Programas:

- Programación Declarativa e Ingeniería de la Programación (Dep. de Sistemes Informàtics i Computació)

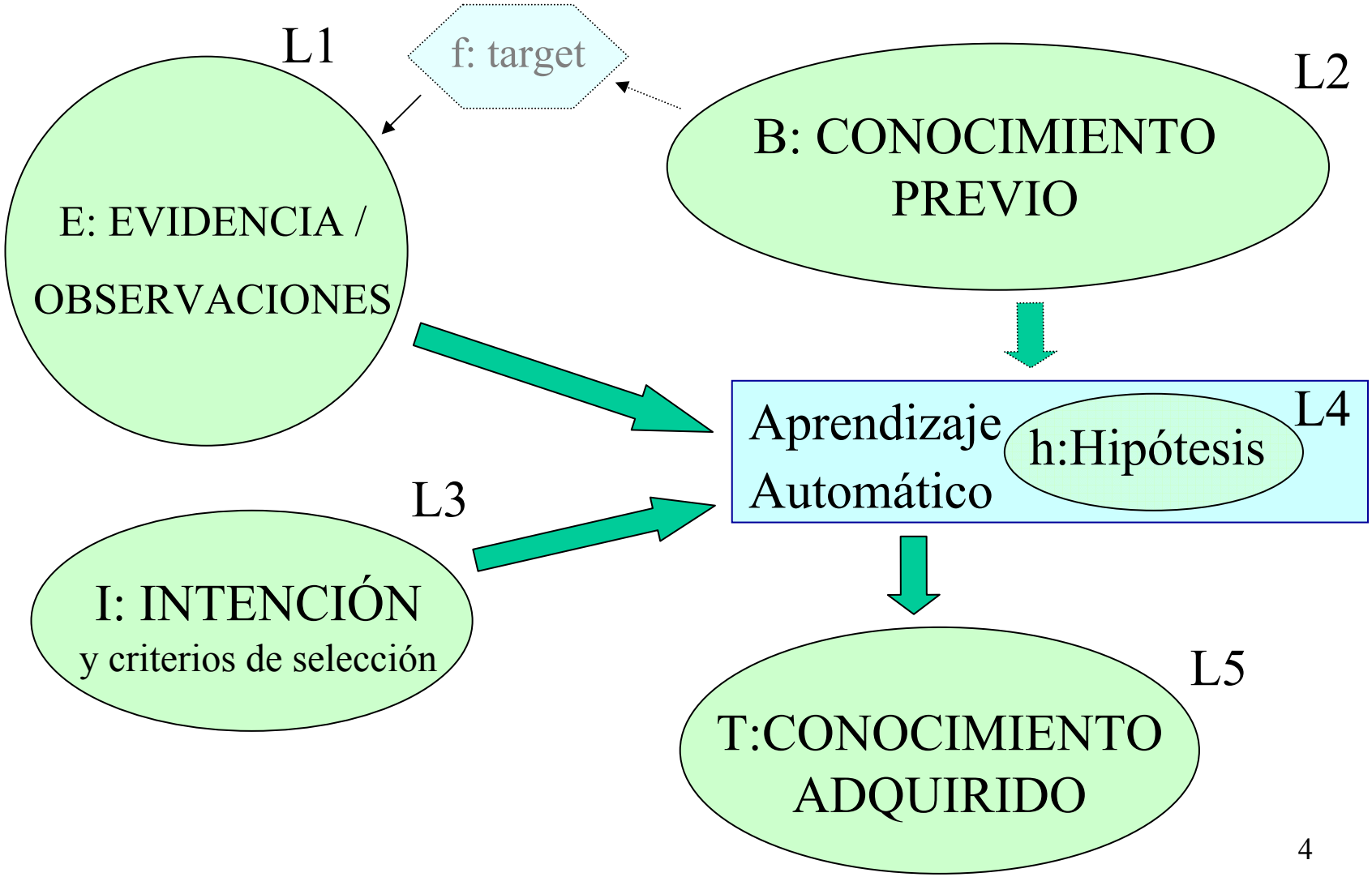
Objetivos

- 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.
- Estudiar el problema de evaluación de hipótesis.
- 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

1.1. El Problema de la Extracción Automática de Conocimiento.
1.2. Técnicas de Aprendizaje Automático.
1.3. Evaluación de Hipótesis
1.4. Inducción Declarativa y Árboles de Decisión.

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: Target* Theory or Function: 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?
- ¿El aprendizaje es computación determinista?
- ¿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?

- Informalmente, cuestión de matices.

- Generalmente "información" y "datos" se pueden referir a cualquier cosa. "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

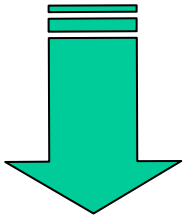
¿Cómo se *representan* evidencia, conocimiento previo, intenciones, hipótesis y conocimiento adquirido (L1, L2, L3, L4 y L5)?

- 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

¿El aprendizaje está predeterminado?

El espacio de hipótesis *H*, junto con el conocimiento previo y el criterio de selección, es lo que **determina** el sesgo (*inductive bias)* del aprendizaje y, por tanto, la solución.



Visión nomológica de la inducción: el aprendizaje computacional es un proceso deductivo determinista.

(que puede no tener solución o ser no computable)

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.
- Incremental: los datos se presentan poco a poco.
  - No interactivo: No se puede actuar sobre la evidencia a recibir.
  - Interactivo: Se puede actuar sobre la evidencia a recibir.
    - Preguntas a un profesor (query-learning).
    - Interacción con un oráculo (la realidad).

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.

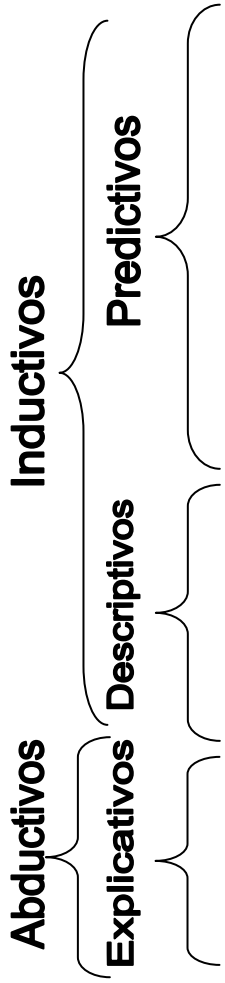
El Problema de la Extracción Automática de Conocimiento

¿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

Técnicas de Aprendizaje Automático

Clasificación de las técnicas de aprendizaje:



- ***Interpolación***: una función continua sobre varias dimensiones
- ***Predicción secuencial***: las observaciones están ordenadas secuencialmente. Se predice el siguiente valor de la secuencia. Caso particular de interpol. con 2 dim., una discreta y regular.
- ***Aprendizaje supervisado***: cada observación incluye un valor de la clase a la que corresponde. Se aprende un clasificador. Caso particular de interpolación: la clase (imag. función) es discreta.
- ***Aprendizaje no supervisado***: el conjunto de observaciones no tienen clases asociadas. El objetivo es detectar regularidades en los datos de cualquier tipo: agrupaciones, contornos, asociaciones, valores anómalos.
- ***Abducción o Aprendizaje Analítico***: *B* es muy importante. El objetivo es explicar la evidencia respecto a *B*.

Diapositiva 14

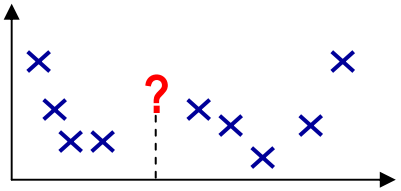
JHO2 jorallo; 08/03/2003

Técnicas de Aprendizaje Automático

Ejemplos:

Predictivos

- ***Interpolación***:



f(2.2)=?

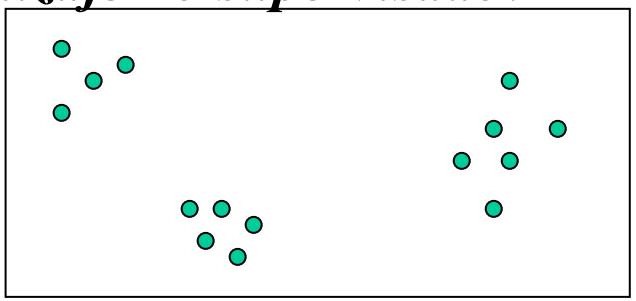
- ***Predicción secuencial***: 1, 2, 3, 5, 7, 11, 13, 17, 19, ... ?
- ***Aprendizaje supervisado***:

1 3 -> 4.
3 5 -> 8.
7 2 -> 9.

4 2 -> ?

Descriptivos

- ***Aprendizaje no supervisado***:



¿Cuántos grupos hay?

Técnicas de Aprendizaje Automático

Interpolación y Predicción Secuencial.

- Generalmente las mismas técnicas:
  - ***Datos continuos (reales):***
    - ***Regresión Lineal***:
      - Regresión lineal global (clásica).
      - Regresión lineal ponderada localmente.
    - ***Regresión No Lineal***: logarítmica, pick & mix, ...
  - ***Datos discretos:***
    - No hay técnicas específicas: se suelen utilizar técnicas de algoritmos genéticos o algoritmos de enumeración refinados.

Técnicas de Aprendizaje Automático

Aprendizaje supervisado.

Dependiendo de si se estima una función o una correspondencia:

- clasificación: se estima una función (las clases son disjuntas).
- categorización: se estima una correspondencia (las clases pueden solapar).

Dependiendo del número y tipo de clases:

- clase *discreta*: se conoce como "clasificación".

  Ejemplo: determinar el grupo sanguíneo a partir de los grupos sanguíneos de los padres.

  - si sólo tiene dos valores (V y F) se conoce como "concept learning".
    Ejemplo: Determinar si un compuesto químico es cancerígeno.

- clase *continua* o discreta ordenada: se conoce como "estimación". " (o también "regresión").

  Ejemplo: estimar el número de hijos de una familia a partir de otros ejemplos de familias.

Técnicas de Aprendizaje Automático

Aprendizaje supervisado.

- Técnicas:
  - k-NN (Nearest Neighbor).
  - k-means (competitive learning).
  - Perceptron Learning.
  - Multilayer ANN methods (e.g. backpropagation).
  - Radial Basis Functions.
  - Decision Tree Learning (e.g. ID3, C4.5, CART).
  - Bayes Classifiers.
  - Center Splitting Methods.
  - Pseudo-relational: Supercharging, Pick-and-Mix.
  - Relational: ILP, IFLP, *SCIL*.

Similarity-Based

Fence and Fill

Técnicas de Aprendizaje Automático

Aprendizaje no supervisado.

- Técnicas:
  - *clustering (segmentación)*.
    - (algunas propias, otras adaptadas de clasificación)
  - Estudios correlacionales / Asociaciones.
  - Dependencias.
  - Detección datos anómalos.
  - Análisis de dispersión.

Técnicas de Aprendizaje Automático

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

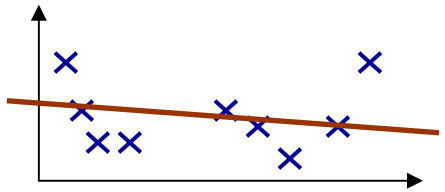
TÉCNICA	PREDICTIVO / SUPERVISADO		DESCRIPTIVO / NO SUPERVISADO		
	Clasificación	Regresión	Clustering (agrup.)	Reglas asociación	Otros (factoriales, correl, dispersión)
Redes Neuronales	✓	✓	✓ *		
Árboles de Decisión	✓ (c4.5)	✓ (CART)	✓		
Kohonen			✓		
Regresión lineal (local, global), exp..		✓			
Reg.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

Interpolación y Predicción Secuencial

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):

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

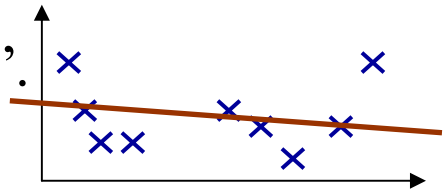
obteniendo $y = w_0 + w_1 x$

Error típico de una regresión lineal simple:

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

Interpolación y Predicción Secuencial

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.

Interpolación y Predicción Secuencial

Regresión No Lineal.

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

$$y = w_0 + w_1 x_1 + \ldots + w_m x_m$$

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

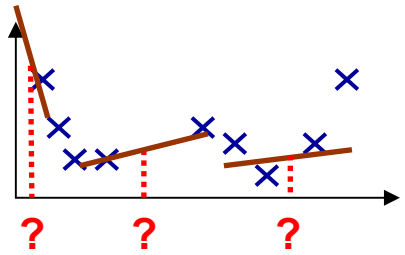
Se añaden dimensiones, combinando las dadas. P.ej. si tenemos cuatro dimensiones: x_1, x_2, x_3 (además de y) podemos definir $x_4 = x_1 \cdot x_2$, $x_5 = x_3^2$, $x_6 = x_1^{x_2}$ y obtener una función lineal de x_1, x_2, x_3, x_4, x_5, x_6

Interpolación y Predicción Secuencial

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)

Interpolación y Predicción Secuencial

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).

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

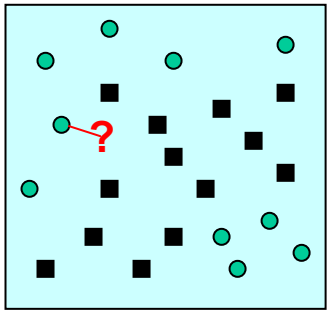
  - ***Distancias por Diferencia:***
    ejemplo: if $x=y$ then $D=0$ else $D=1$
  - ***Distancia de Edición:***
  - ***Distancias Específicas:*** *para los ejemplos complejos de CBR.*

  } Valores Discretos

Aprendizaje Supervisado

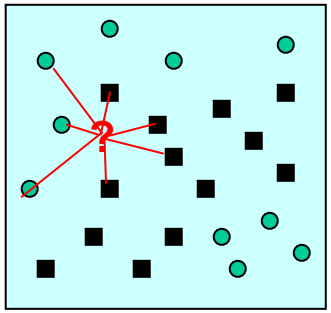
k-NN (Nearest Neighbour):

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.



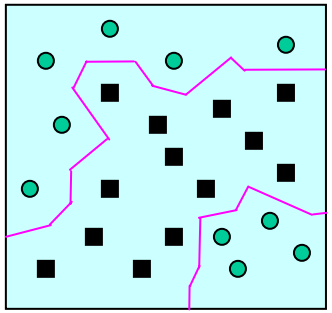
Clasifica círculo

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

***k*-means clustering**:

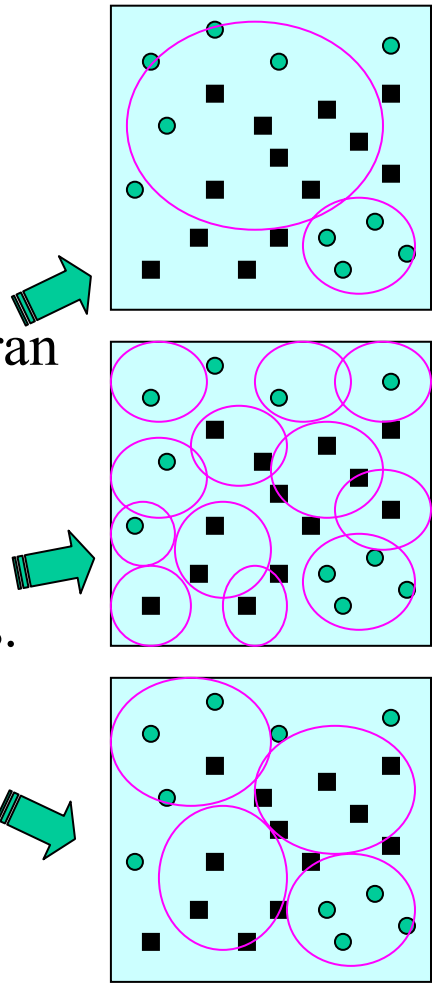
- Se utiliza para encontrar los *k* puntos más densos en un conjunto arbitrario de puntos.

- 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 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 Supervisado

***On-line k*-means clustering** (competitive learning):

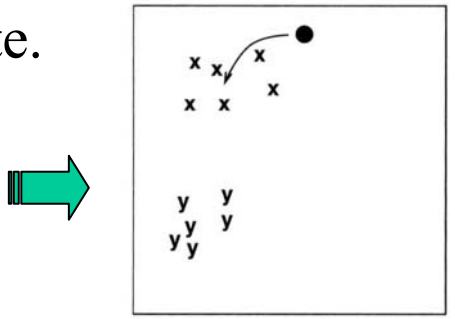
- Refinamiento incremental del anterior.

- Algoritmo:
  1. Inicializar aleatoriamente *k* puntos, llamados *centros*.
  2. Elegir el siguiente ejemplo y ver cuál es el centro más cercano. Mover el centro hacia el ejemplo. (p.ej. Distancia/2)
  3. Repetir el paso 2 para cada ejemplo.
  4. Repetir los pasos 2 y 3 hasta que los ejemplos capturados por cada centro no varíen.

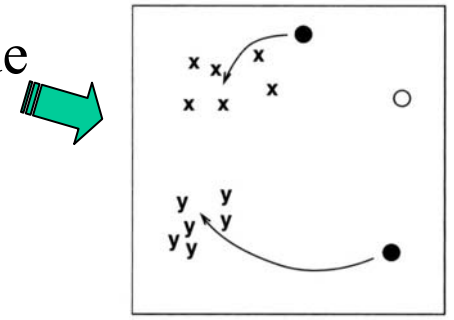
Aprendizaje Supervisado

El valor de *k* se suele determinar heurísticamente.

- Problemas:
  - Si *k* se elige muy pequeño, hay grupos que se quedan sin centro.

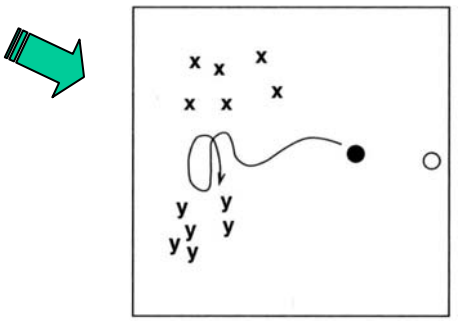


  - Si *k* se elige muy grande, hay centros que se quedan huérfanos.



Aunque esto es preferible a...

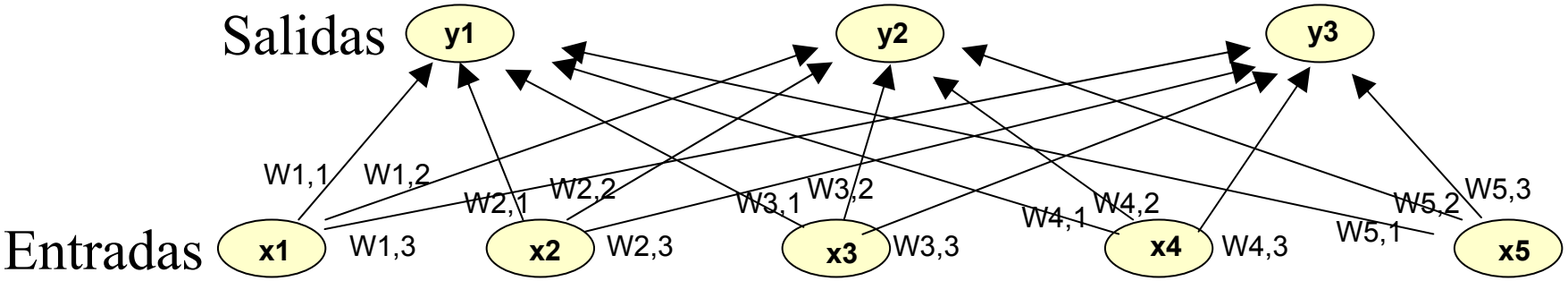
  - Incluso con *k* exacto, puede haber algún centro que quede huérfano.



- Variación muy popular: LVQ (learning vector quantization) (Kohonen 1984).

Aprendizaje Supervisado

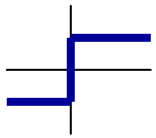
Perceptron Learning (McCulloch & Pitts 1943, Widrow & Hoff 1960, Rosenblatt 1962).



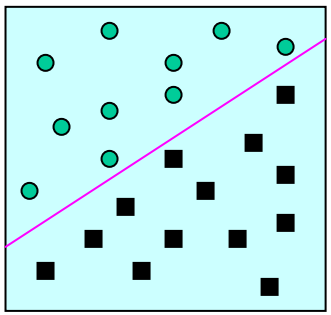
- Computan una función lineal para cada y_j es:

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

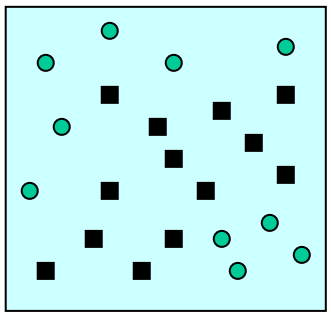
Se añade un threshold escalón: $output_j = \text{sgn}(y'_j)$



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



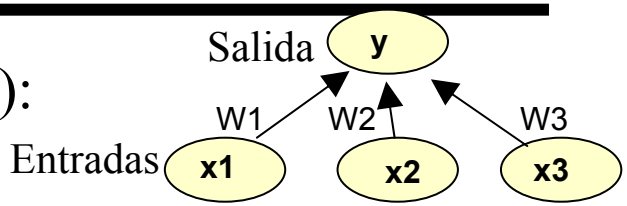
PARTICIÓN LINEAL POSIBLE



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):

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

  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*:

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

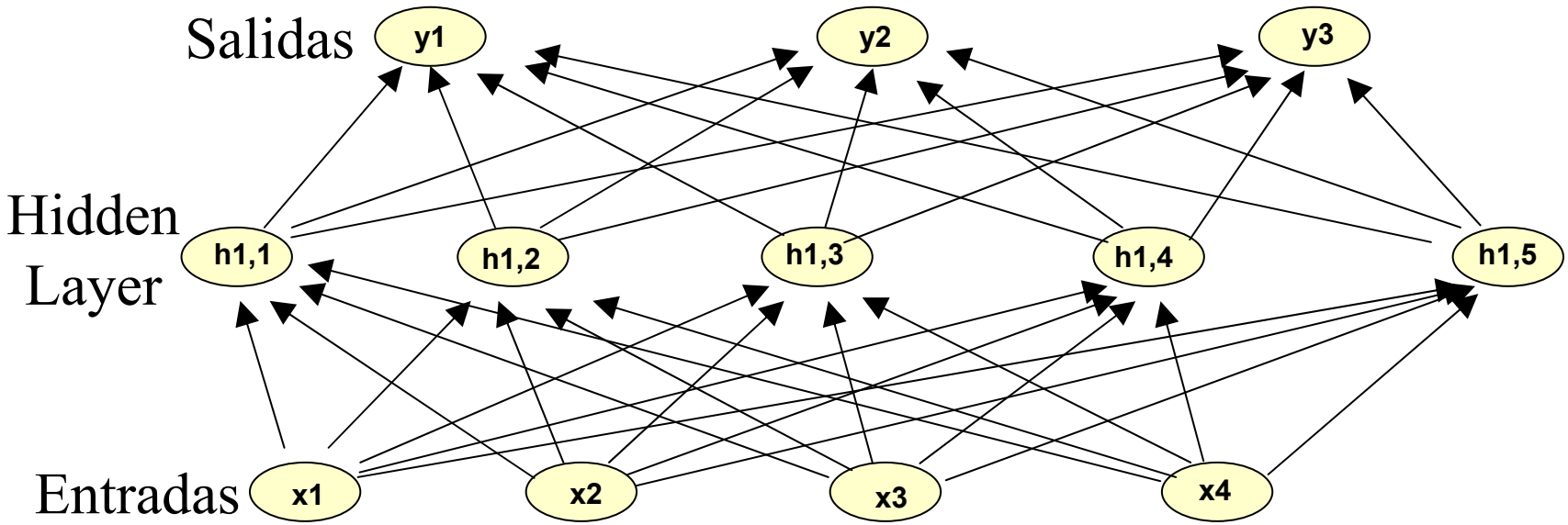
  5. t:= t+1, ir a 2 hasta que no queden ejemplos o el error medio se ha reducido a un valor deseado.

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

Aprendizaje Supervisado

Multilayer Perceptron (redes neuronales artificiales, ANN).

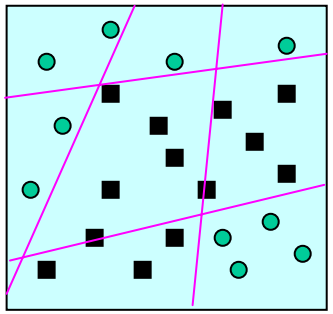
- El perceptron de una capa no es capaz de aprender las funciones más sencillas.
- Se añaden capas internas, se introducen diferentes funciones de activación e incluso recientemente se introducen bucles y retardos.



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.

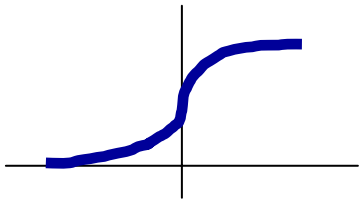
- Pero, ¿cómo entrenar este tipo de red?

Aprendizaje Supervisado

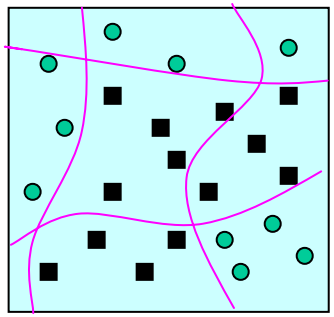
Multilayer Perceptron (redes neuronales artificiales, ANN).

- 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:

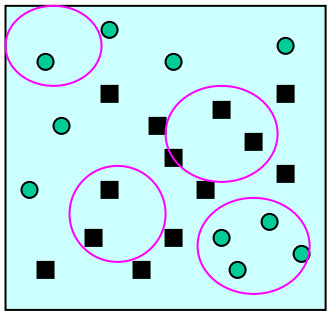
$$\delta_h = o_h(1-o_h) \sum_{k \in downstream(h)} (w_{k,h} \times \delta_k)$$

- 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.



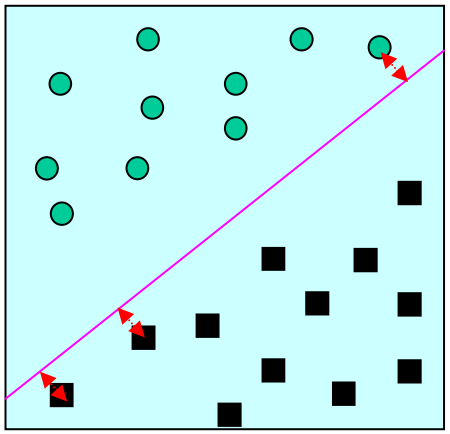
PARTICIÓN HIPERESFÉRICA CON 4 centros.

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

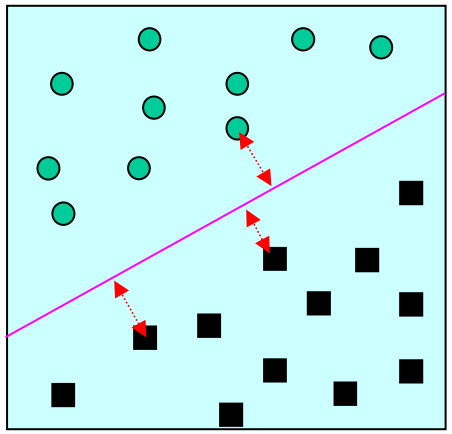
Aprendizaje Supervisado

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.

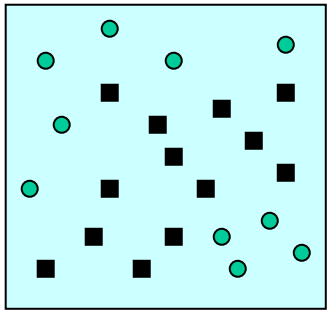


Separa perfectamente, pero además los ejemplos más cercanos (vectores soporte) están lo más lejos posible de la frontera.

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?

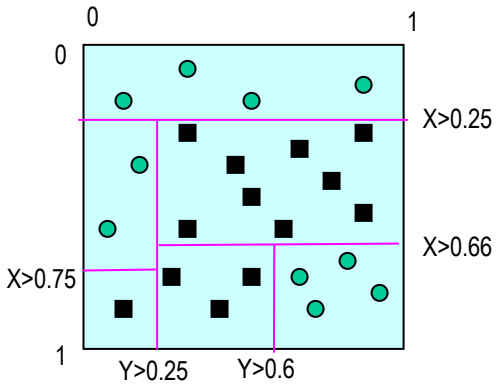


- Se aplica una función núcleo (“kernel”) que suele aumentar el número de dimensiones de tal manera que los datos sean separables.

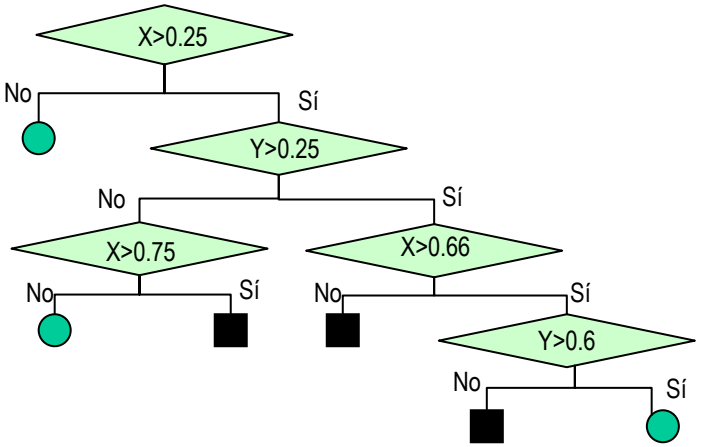
Aprendizaje Supervisado

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.

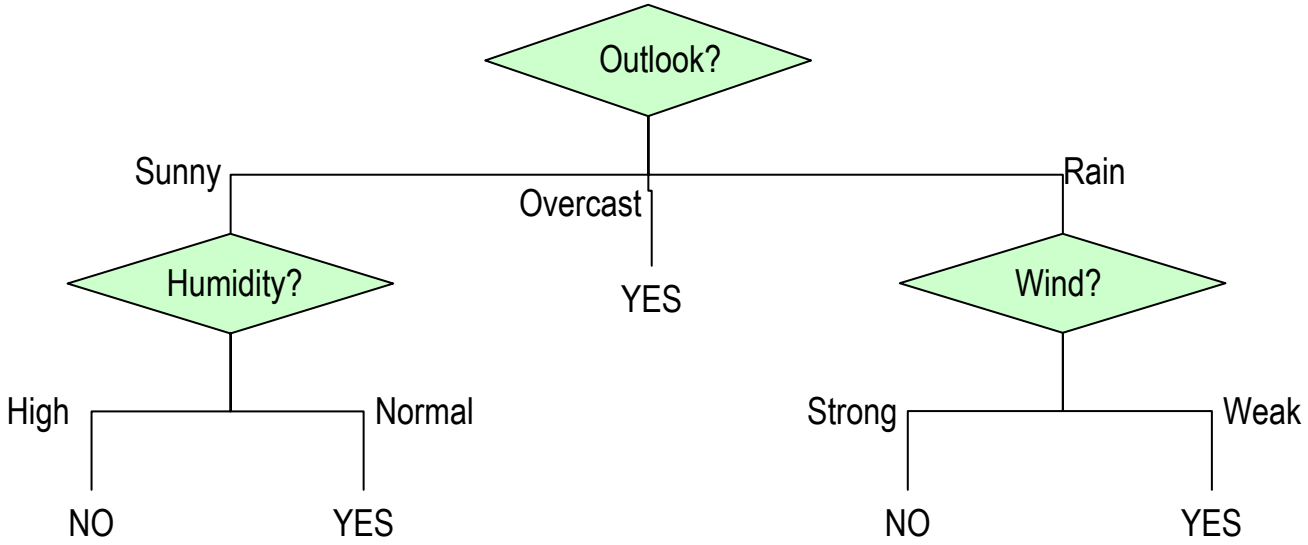
- Ejemplo C4.5 con datos discretos:

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

General-to-specific: Candidate-Elimination Algorithm (Mitchell).

- Algoritmo (para sólo dos clases):

Inicializar G al conjunto de hipótesis maximalmente generales.
Inicializar S al conjunto de hipótesis maximalmente específicas.
Para cada ejemplo *d*, hacer

- Si *d* es un ejemplo positivo
  - Eliminar de *G* cualquier hipótesis inconsistente con *d*.
  - Para cada hipótesis *s* en *S* que no sea consistente con *d*:
    - Eliminar *s* de *S*.
    - Añadir a *S* todas las generalizaciones minimales *h* de *s* tales que:
      - *h* es consistente con *d*, y algún miembro de *G* es más general que *h*.
    - Eliminar de *S* cualquier hipótesis que sea más general que otra hipótesis de *S*.
- Si *d* es un ejemplo negativo
  - Eliminar de *S* cualquier hipótesis inconsistente con *d*.
  - Para cada hipótesis *g* en *G* que no sea consistente con *d*:
    - Eliminar *g* de *G*.
    - Añadir a *G* todas las especializaciones minimales *h* de *g* tales que:
      - *h* es consistente con *d*, y algún miembro de *S* es más específico que *h*.
    - Eliminar de *G* cualquier hipótesis que sea más específica que otra hipótesis de *G*.

Aprendizaje Supervisado

General-to-specific: Candidate-Elimination Algorithm (Mitchell).

- Ejemplo:

Example	Sky	AirTemp	Humidity	Wind	Water	Forecast	EnjoySport
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes

Cjtos Iniciales: S_0 = {<∅, ∅, ∅, ∅, ∅, ∅>} G_0 = {<?, ?, ?, ?, ?, ?>}

Ejemplo 1: S_1 = {<Sunny, Warm, Normal, Strong, Warm, Same>} G_1 = {<?, ?, ?, ?, ?, ?>}

Ejemplo 2: S_2 = {<Sunny, Warm, ?, Strong, Warm, Same>} G_2 = {<?, ?, ?, ?, ?, ?>}

Ejemplo 3: S_3 = {<Sunny, Warm, ?, Strong, Warm, Same>} G_3 = {<Sunny, ?, ?, ?, ?, ?>, <?, Warm, ?, ?, ?, ?>, <?, ?, ?, ?, ?, Same> }

Ejemplo 4: S_4 = {<Sunny, Warm, ?, Strong, ?, ?>} G_4 = {<Sunny, ?, ?, ?, ?, ?>, <?, Warm, ?, ?, ?, ?> }

El espacio de hipótesis válidas es, por tanto:

Más generales que “Sky=Sunny AND AirTemp=Warm AND Win=Strong”.

Más específicas que “Sky=Sunny OR AirTemp=Warm”.

Aprendizaje Supervisado

Naive Bayes Classifiers.

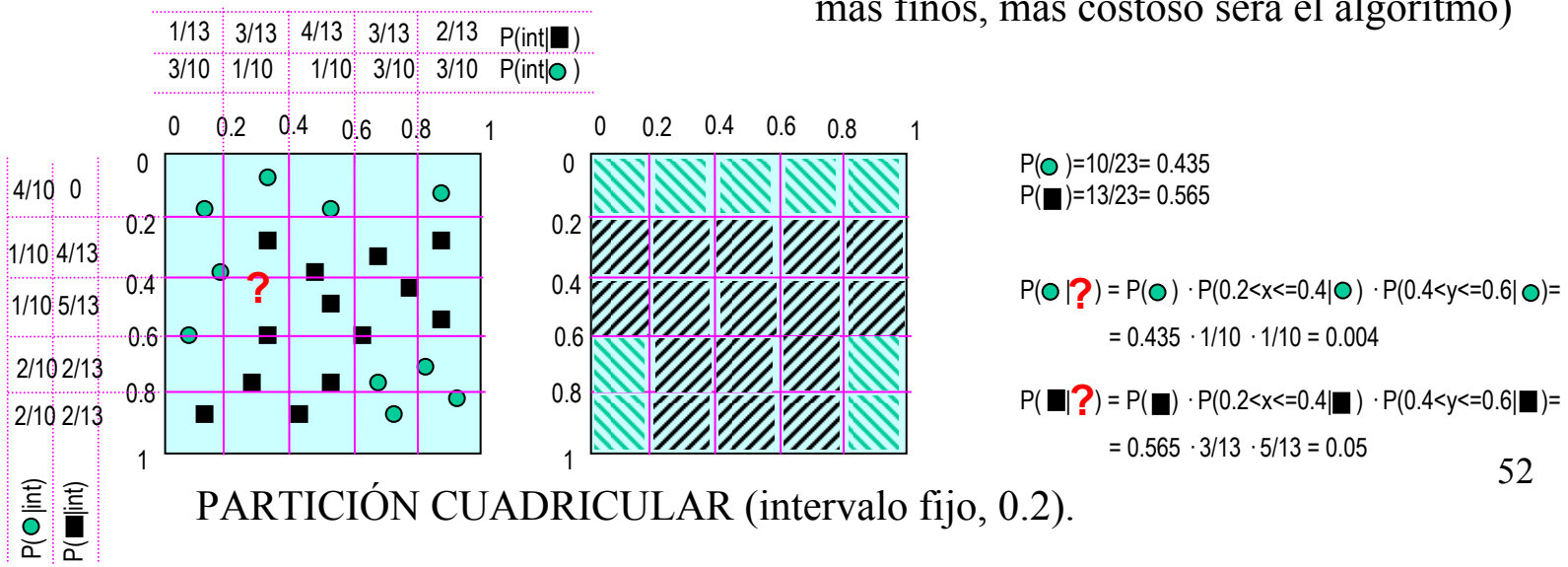
$$\arg\max_{c_i \in C} P(c_i \mid (x_1, x_2, ..., x_m)) \underset{Bayes}{=} \arg\max_{c_i \in C} \frac{P((x_1, x_2, ..., x_m) \mid c_i) \cdot P(c_i)}{P(x_1, x_2, ..., x_m)} =$$

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

- 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)



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:

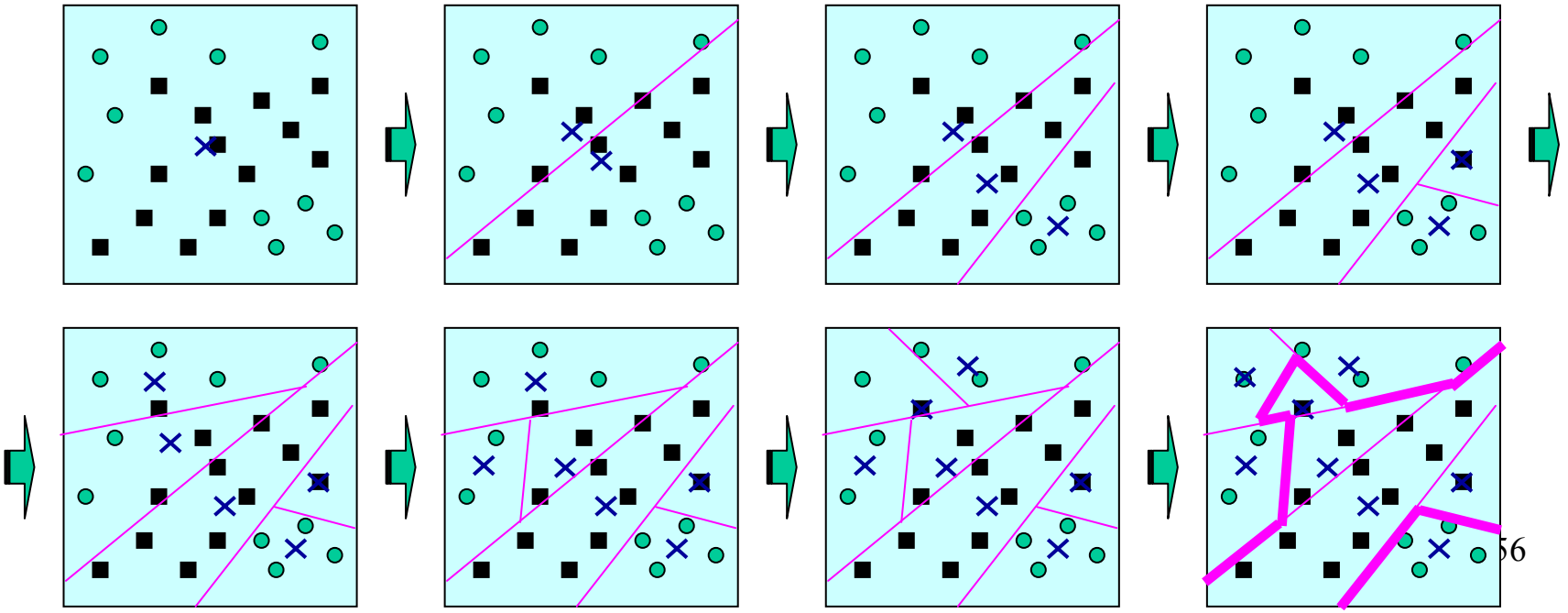
$$\frac{n_c + mp}{n + m}$$

- 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 Supervisado

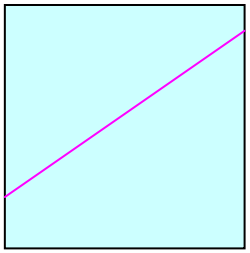
Center Splitting (es un híbrido LVQ/C4.5).

1. Inicializar el primer centro en la media de los ejemplos.
2. Asignar todos los ejemplos a su centro más cercano.
3. Si hay algún centro que tiene ejemplos de diferente clase, borrar el centro y crear tantos nuevos centros distintos como clases haya, cada uno siendo la media de los ejemplos de la clase. Ir a 2.

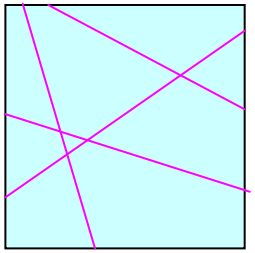


Aprendizaje Supervisado

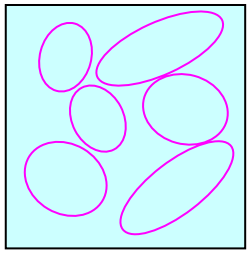
Comparación de representación:



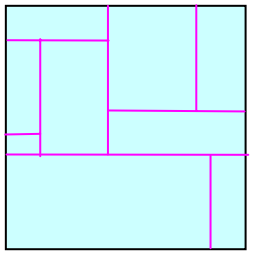
Perceptron / LMS



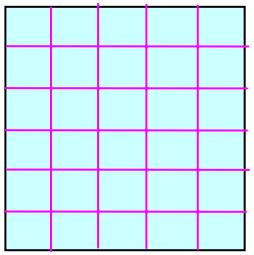
Redes Neuronales Multicapa



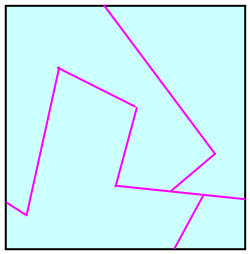
RBF



C4.5/ID3/CART



Naive Bayes Classifier



k-NN, LVQ, CS

Aprendizaje Supervisado

Comparación de métodos no relacionales:

- k-NN:
  - Muy fácil de usar
  - Eficiente si el nº de ejemplos no es excesivamente grande.
  - El valor de *k* no es muy importante.
  - Gran expresividad de la partición.
  - Inteligible sólo visualmente.
  - Robusto al ruido pero no a atributos no significativos (las distancias aumentan, conocido como "the curse of dimensionality")
- RBF (combinaciones de k-means clustering + perceptron):
  - Preferibles a cualquiera de las dos técnicas por separado.
  - Difícil de ajustar el *k*.
  - Poca inteligibilidad.
- Redes neuronales (multicapa):
  - El número de capas y elementos por capa difíciles de ajustar.
  - Apropiado para clases discretas o *continuas*.
  - Poca inteligibilidad.
  - Muy sensibles a outliers (datos anómalos).
  - Se necesitan muchos ejemplos.

Aprendizaje Supervisado

Comparación de métodos no relacionales (cont.):

- Naive Bayes:
  - Muy fácil de usar.
  - Muy eficiente.
  - NO HAY MODELO.
  - Robusto al ruido.
- Árboles de decisión: (C4.5):
  - Muy fácil de usar.
  - Admite atributos discretos y continuos.
  - La clase debe ser discreta y finita. (aunque tb. existen los árboles de regresión que permiten clase continua)
  - Es tolerante al ruido, a atributos no significativos y a *missing attribute values*.
  - Alta inteligibilidad.
- CS (*Center Splitting*):
  - Muy fácil de usar.
  - Preferible sobre k-NN si hay muchos ejemplos.
  - Inteligible sólo visualmente.
  - Sufre también “the curse of dimensionality”.

Aprendizaje Supervisado

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**

Aprendizaje Supervisado

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).

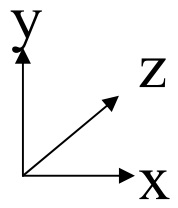
Aprendizaje Supervisado

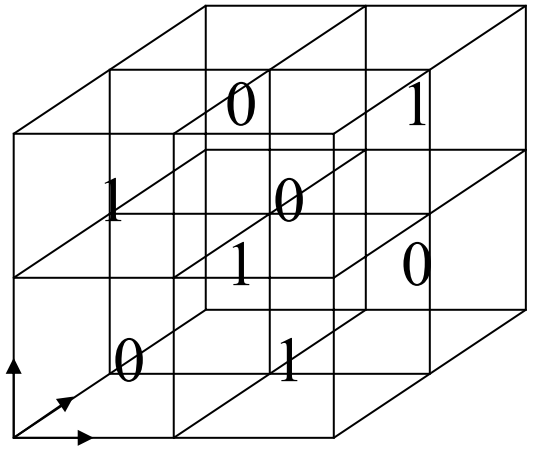
La *continuidad* hacia problemas relacionales.

- Ejemplo. Paridad

$x\ y\ z \rightarrow$ clase

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





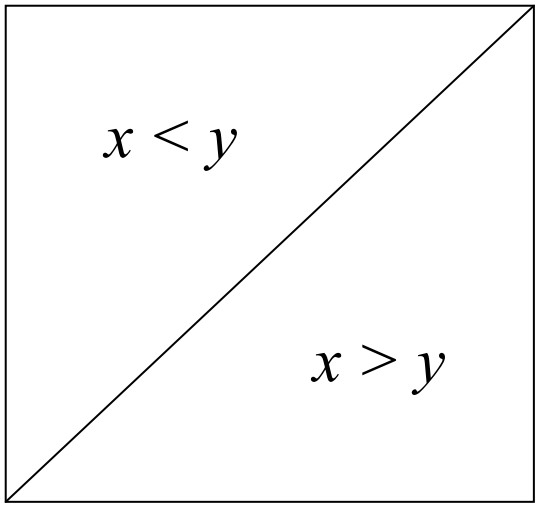
¡¡¡¡No hay manera de hacer grupos geométricos!!!!
La paridad es un problema relacional PURO.

Aprendizaje Supervisado

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.

Aprendizaje Supervisado

¿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.

Aprendizaje Supervisado

Funciones Heurísticas.

- Separabilidad Geométrica Radial:

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

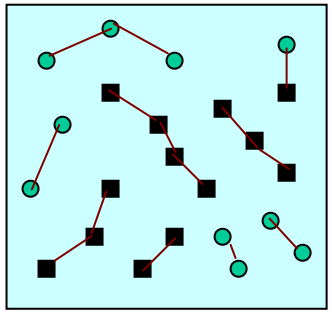
$$RGS(f) = \frac{\sum_{i=1}^{n} \frac{\sum_{e_j : dist(e_i, e_j) < r} eq(f(e_i), f(e_j))}{\left| e_j : dist(e_i, e_j) < r \right|}}{n}$$

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

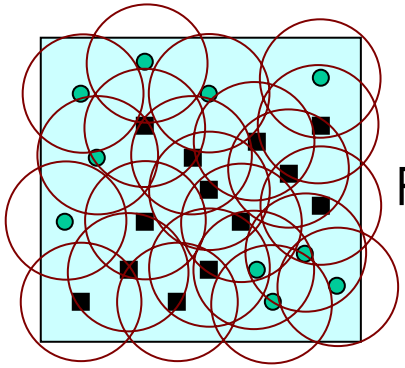
También se puede calcular una RGS' que no incluye el propio punto.

Aprendizaje Supervisado

- Ejemplos:

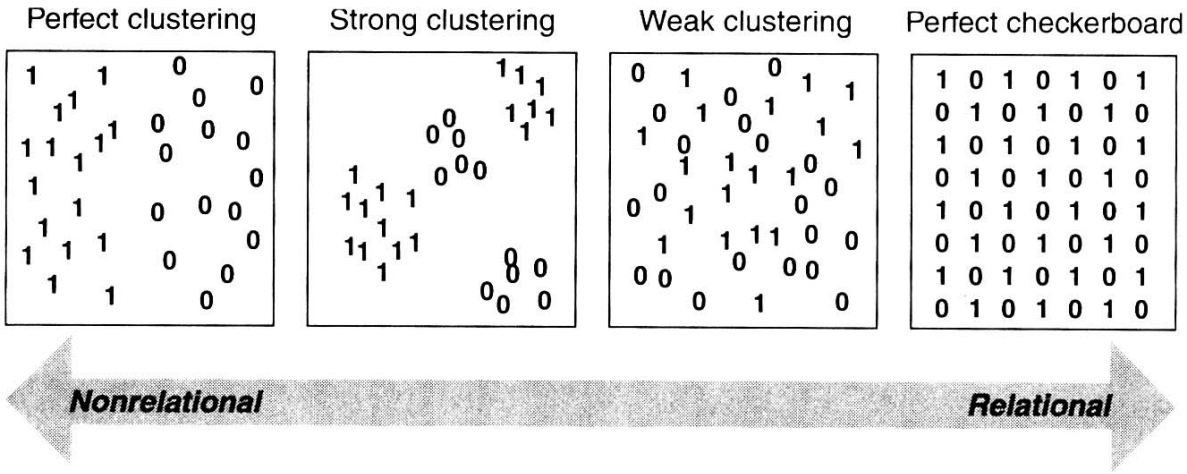


GS = 21/23 = 0.91



RGS = 18.44/23 = 0.8

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



Aprendizaje Supervisado

Métodos pseudo-relacionales

- Una manera de poder abordar problemas con poca separabilidad es transformar los datos, mediante recodificaciones o aumento de la dimensionalidad.
    - **Super-charging**: aumentar la dimensionalidad separando todos los atributos en dígitos binarios.
    - **Pick and Mix**: crear nuevos atributos como combinación de los ya existentes.
    - **SCIL**: iterar múltiples veces un método clásico fence & fill seguido de recodificación.

Aprendizaje Supervisado

Super-charging, ejemplo:

- Problema MONKS2 del repositorio UCI se define como:

  la salida es cierta si "exactamente dos de las seis variables de entrada tienen su primer valor".

- Si se aplica ID3 directamente, el resultado es un 67,9% de acierto sobre los datos de prueba.

  CODIFICACIÓN: se obtienen 17 dimensiones separando las posibles igualdades (x_1=a, x_2=b, x_3=c, y_1= 1, y_2=4, ...)

- Si se aplica ahora ID3 el resultado es un 77%.
- Si se aplica *backpropagation* con un número suficiente de unidades ocultas, consigue 100%.
- PROBLEMAS:
  - A veces 'sobreajusta' (overfits), dando patrones irreales.
  - Además los modelos aprendidos son difícilmente inteligibles.
  - Incrementar el número de atributos aumenta el coste temporal.

Aprendizaje Supervisado

Pick-and-Mix: Algunos problemas son relacionales debido simplemente a relaciones *triviales* entre los atributos.

- Si tenemos en el conocimiento previo o de base *B* algunas de estas relaciones triviales, podríamos añadir nuevos atributos a los datos combinando los atributos originales con estas funciones triviales.
- Esta solución es especialmente útil para modelos matemáticos y físicos, ya que se pueden utilizar las operaciones básicas (+, –, /, *) sobre los atributos y conseguir modelos no lineales.
- En general el número de funciones de *B* a probar se dispara, o éstas deben estar muy bien elegidas.
- Sin embargo, aun probando muchísimas funciones en el conocimiento previo, no resuelven otros problemas relacionales, como el de la paridad o el MONKS2.

Aprendizaje Supervisado

Pick-and-Mix, ejemplo:

El sistema BACON (Langley 1978, 1979, et al. 1983).

Descubrimiento de la ley de Kepler (y^2/d^3 es una constante):

Planeta	***y***	***d***	***y/d***	***(y/d)/d***	***((y/d)/d)y***	***(((y/d)/d)y)/d***
Mercurio	0.24	0.39	0.62	1.61	0.39	1.00
Venus	0.61	0.72	0.85	1.18	0.72	1.00
Tierra	1.00	1.00	1.00	1.00	1.00	1.00
Marte	1.88	1.52	1.23	0.81	1.52	1.00
Ceres	4.60	2.77	1.66	0.60	2.76	1.00
Júpiter	11.86	5.20	2.28	0.44	5.20	1.00
Saturno	29.46	9.54	3.09	0.32	9.54	1.00
Urano	84.01	19.19	4.38	0.23	19.17	1.00
Neptuno	164.80	30.07	5.48	0.18	30.04	1.00
Plutón	248.40	39.52	6.29	0.16	39.51	1.00
T.Beta	680.00	77.22	8.81	0.11	77.55	1.00

(*y* representa el periodo del planeta, *d* representa la distancia al sol, ambos valores normalizados respecto a la tierra)

Aprendizaje Supervisado

SCIL Learning (Thornton 2000):

- Algoritmo Genérico: Truth From Trash, TFT (Thornton 2000):

  1. Utilizar cualquier método fence-and-fill para los datos.
  2. Si la partición es satisfactoria, salir.
  3. Si no, recodificar los datos de entrada, de tal manera que los ejemplos de la misma clase estén más cerca que antes.
  4. Volver al punto 1 con los nuevos datos.

Aprendizaje Supervisado

SCIL Learning (Thornton 2000):

- Instancia de TFT: Algoritmo SCIL (split-centers-in-layers):

1. Utilizar Center-Splitting.
2. Si la partición es satisfactoria, salir.
3. Si no, recodificar los datos de entrada, de la siguiente manera:

(*suponemos los centros se numeran, ordenados por clases*)

Dado un ejemplo $(x_1, x_2, ..., x_n)$, se calcula un nuevo ejemplo (x'_1, x'_2)

x'_1:= nº de orden del centro original más próximo.

x'_2:= proximidad (o distancia) al ejemplo original más próximo.

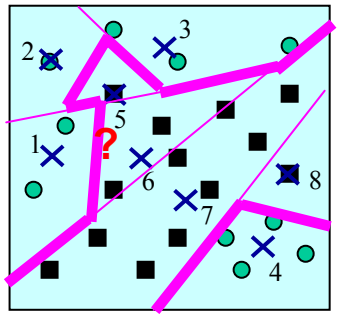
Nótese que a partir de esta iteración, la dimensión es siempre 2.

4. Volver al punto 1 con los nuevos datos.

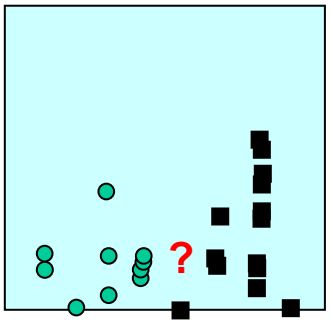
Aprendizaje Supervisado

SCIL Learning (Thornton 2000):

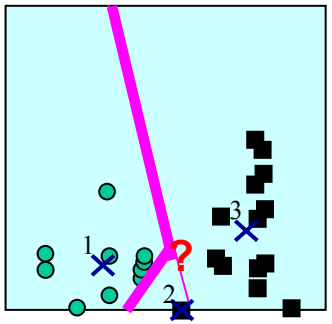
Primer resultado de CS:



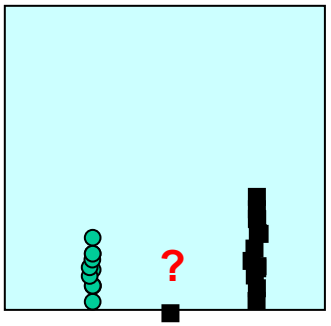
Recodificación



Segundo resultado de CS:



Recodificación



Resultados poco inteligibles.

Aprendizaje Supervisado

Pero estos métodos pseudo-relacionales 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...

Aprendizaje Supervisado

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

Aprendizaje Supervisado

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.

Aprendizaje Supervisado

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).

Aprendizaje Supervisado

Aprendizaje Recursivo:

Modelos Estructurales de Grammar Learning:
Es uno de los campos más antiguos de ML:
(las frases gramaticales son de la clase true)

- 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 Supervisado

Aprendizaje Relacional y Recursivo:

- **IFP (Inductive Functional Programming).** Se aprenden reglas de la forma:

  $$g(f(a), X) \rightarrow b$$

  - Existen aproximaciones con LISP, el lenguaje ML y otros (70s).
- **ILP (Inductive Logic Programming).** El lenguaje representacional es cláusulas de Horn:

  $$p(X,Y,b) :- q(f(X,Y), c)$$

  - Inicio en los 80 (Shapiro) y gran desarrollo en la década de los 90.
- **IFLP (Inductive Functional Logic Programming):**

  $$g(f(a),X) \rightarrow b :- p(X,b) = true, q(X,X) = a$$

  - Mayor naturalidad y expresividad. Ventaja con problemas de clasif.
- Aprendizaje en **Orden Superior**. Algún intento con el lenguaje Escher. Todavía en pañales.

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:

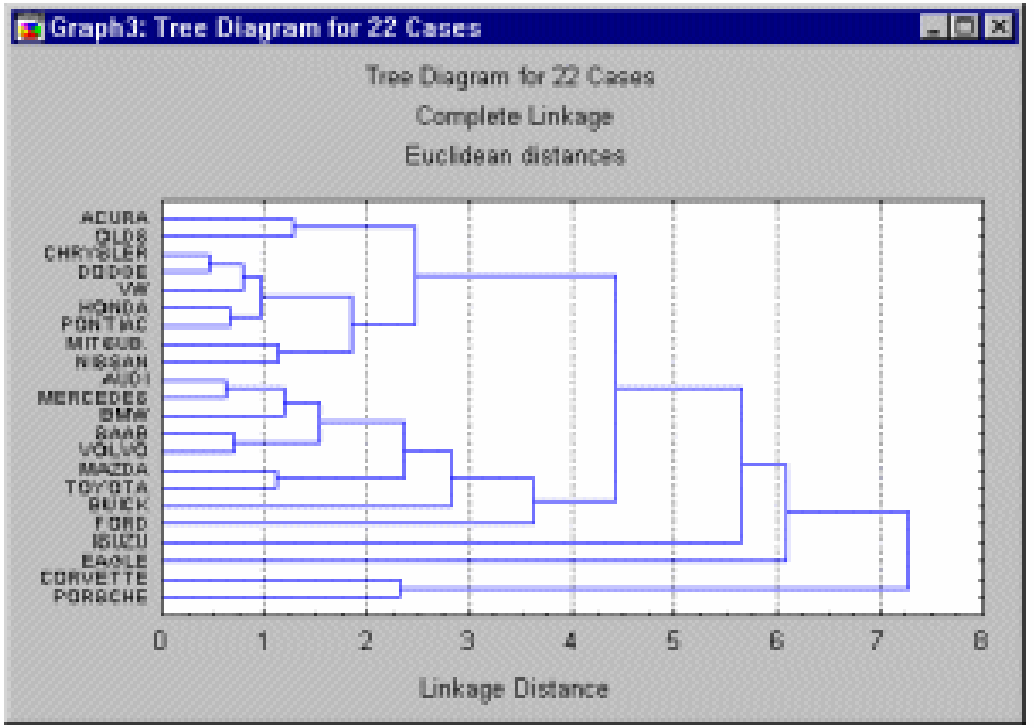
- Jerárquicos: los datos se agrupan de manera arborescente (p.ej. el reino animal).
- No jerárquicos: generar particiones a un nivel.
  - (a) Paramétricos: se asume que las densidades condicionales de los grupos tienen cierta forma paramétrica conocida (p.ej. Gaussiana), y se reduce a estimar los parámetros.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

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).

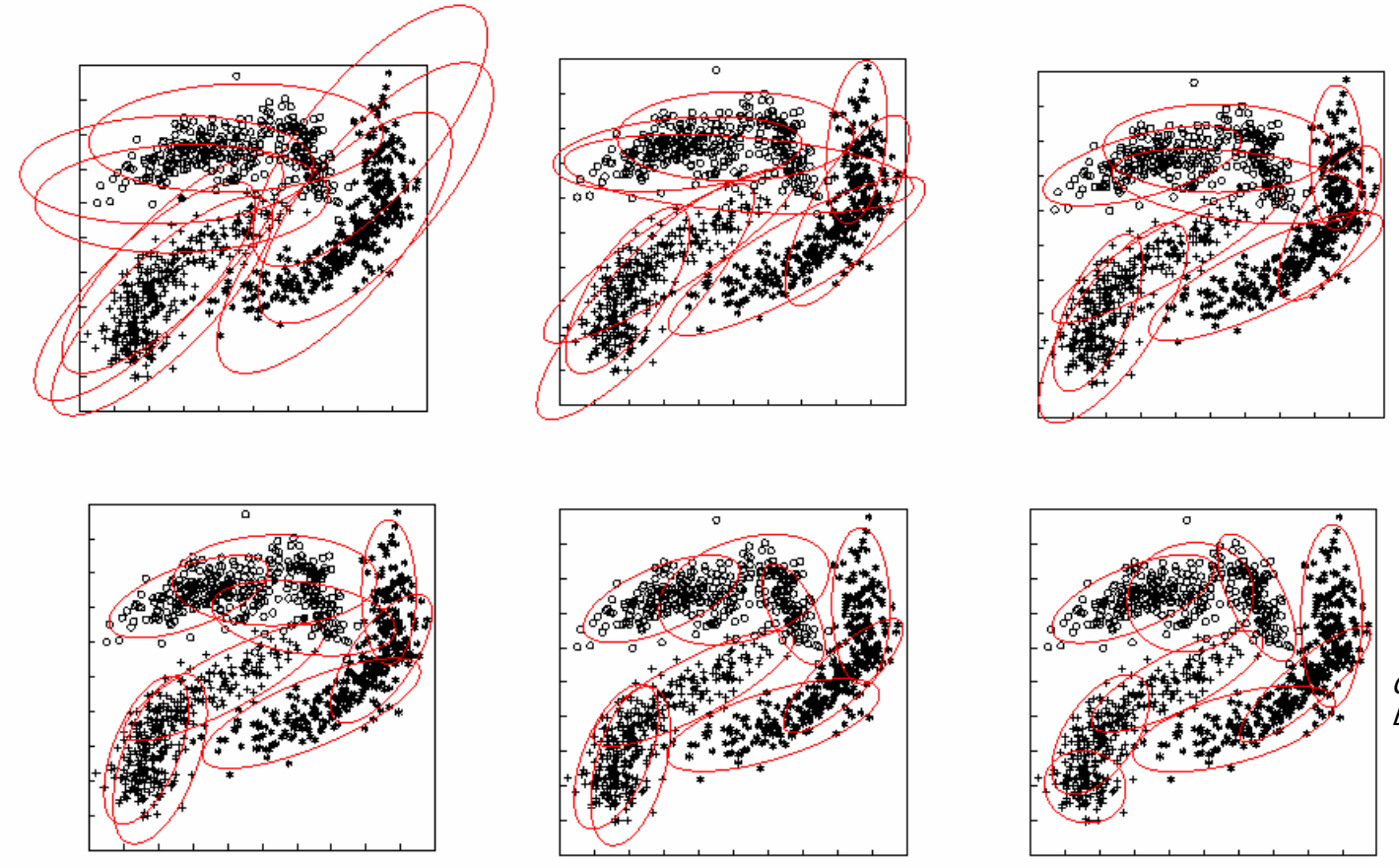
Inicialmente considera cada ejemplo como un cluster.

- 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:

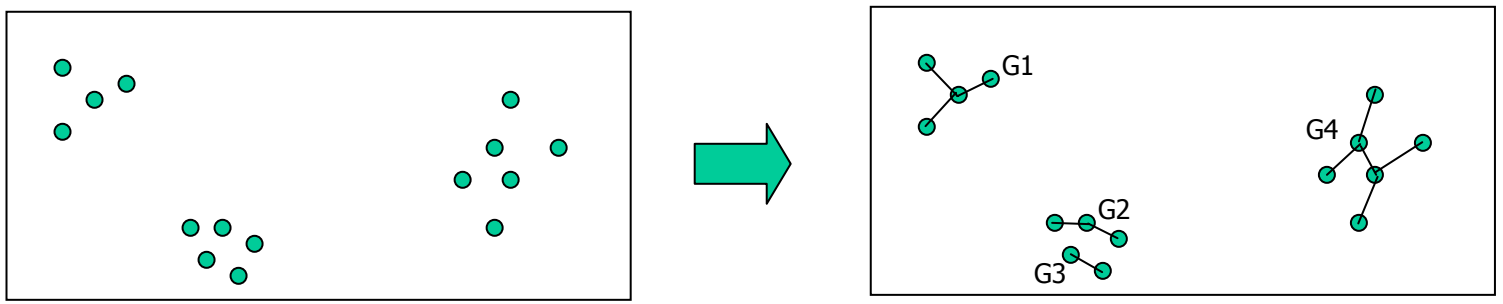
- El algoritmo Cobweb (Fisher 1987).
- El algoritmo AUTOCLASS (Cheeseman & Stutz 1996)

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.

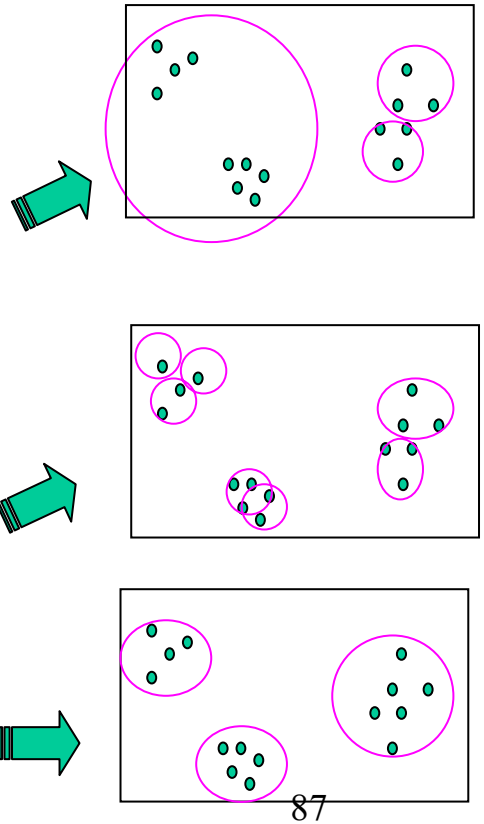
Existen variantes: k-NN o como el spanning tree que para de agrupar cuando llega a un número de grupos.

Aprendizaje No Supervisado

Clustering (Segmentación). Métodos No Paramétricos

***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, algunas veces, algún grupo use dos centros y dos grupos separados tengan que compartir centro.
  - Si *k* se elige muy grande, la generalización es pobre y las agrupaciones futuras serán malas.
  - Determinar el *k* ideal es difícil.



Aprendizaje No Supervisado

Clustering (Segmentación). Métodos No Paramétricos

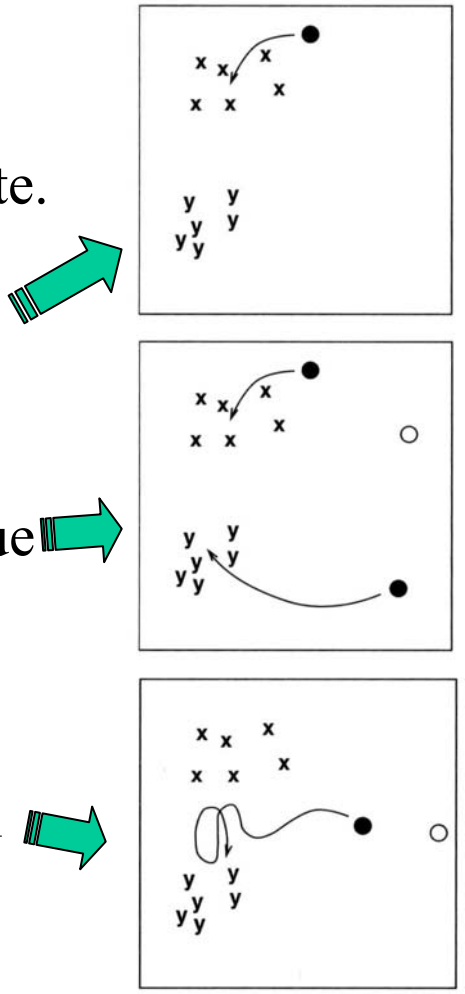
On-line k-means clustering:

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.

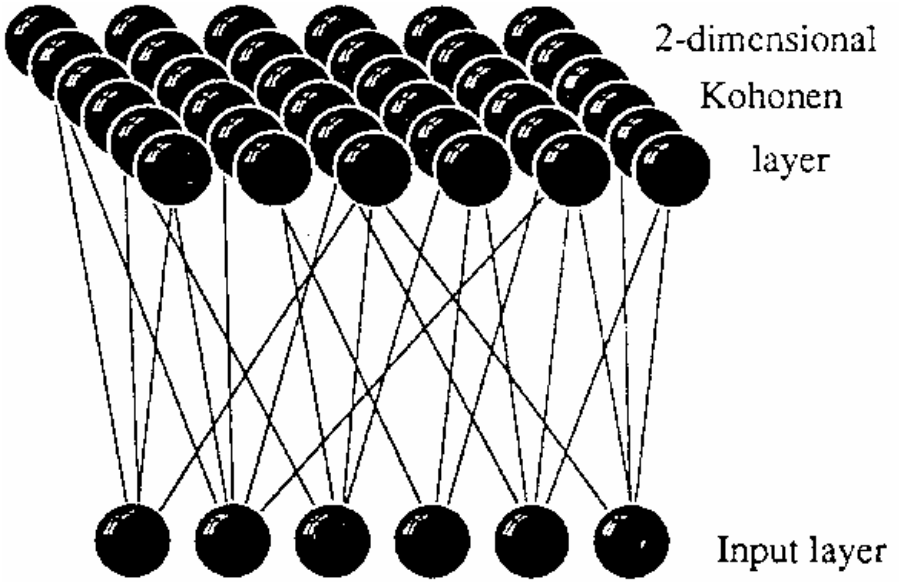


Aprendizaje No Supervisado

Clustering (Segmentación). Métodos No Paramétricos

SOM (Self-Organizing Maps) o Redes Kohonen

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



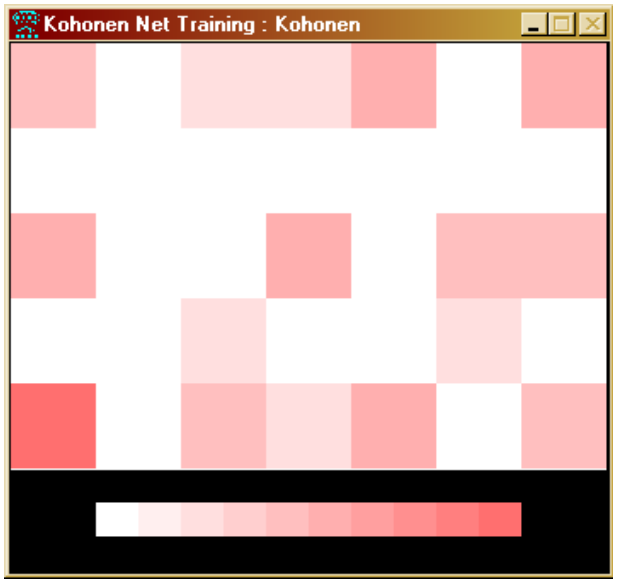
La matriz de neuronas de la última capa forma un grid bidimensional.

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.

Aprendizaje Analítico (Abductivo)

Aprendizaje Analítico (Abductivo o Explicativo)

El objetivo es explicar los datos; buscar una justificación respecto al conocimiento previo. Las hipótesis generalmente se conocen o se derivan de *B*.

P.ej. *B*: Si llueve el césped está mojado.

Si no hay nubes no puede llover

Si se riega el césped, el césped está mojado

D: El césped está mojado

¿por qué?

H_1: Ha llovido, H_2: Se ha regado el césped

Aprendizaje Analítico (Abductivo)

Aprendizaje Analítico (Abductivo o Explicativo)

EBL (Explanation-Based Learning) and KBR (Knowledge-Based Reasoning)

- EBL realiza una generalización de los ejemplos basada en el conocimiento previo.
- En realidad, EBL simplemente *reformula* el conocimiento que ya se sabía. Obtiene una versión más operativa.
- Es en realidad un método deductivo. Conexión con transformación/especialización de programas.

(se puede ver como aprendizaje pero no es inducción)

Aprendizaje Analítico (Abductivo)

Aprendizaje Analítico. Ejemplo **PROLOG-EBG** (Kedar-Cabelli and McCarty, M.T 1987). Refinamiento del EBG de Mitchell 1986. :

B= { safetostack(X,Y) :- ¬fragile(Y). safetostack(X,Y) :- lighter(X,Y).
lighter(X,Y) :- weight(X, WX), weight(Y,WY), lessthan(WX,WY).
weight(X,W) :- volume(X, V), density(X,D), equal(W, times(V,D)).
weight(X,5) :- type(X, endtable). fragile(X) :- material(X, glass). }

E_t^+ (target) = { safetostack(obj1,obj2). }

E_c^+ (context) = { on(obj1, obj2). type(obj1, box). type(obj2, endtable). color(obj1, red).
color(obj2, blue). volume(obj1,2). owner(obj1, fred). owner(obj2, louise).
density(obj1, 0.3). material(obj1, cardboard). material(obj2, wood). }

- Se lanza a Prolog el ejemplo target de E_t^+ (i.e. safetostack(obj1,obj2).).
- Cada *demostración* de safetostack(obj1,obj2) respecto de B y E_c^+ constituye una explicación a E_t^+ (formada sólo de los hechos de E_c^+)

volume(obj1,2), density(obj1, 0.3), type(obj2, endtable)

Aprendizaje Analítico (Abductivo)

Aprendizaje Analítico. Ejemplo PROLOG-EBG (continuación):

- Se generaliza y se obtiene una regla:

```
safetostack(X,Y) :- volume(X,2), density(X, 0.3), type(Y, endtable)
```

- Se puede calcular la "preimagen más débil"

  Preimagen más débil: la preimagen más débil de una conclusión *C* con respecto a una demostración *P* es el conjunto más general de aserciones iniciales *A*, tal que *A* implica *C* según *P*.

  Existe un algoritmo (llamado de regresión, Waldinger 1977) para calcularla.

- Se obtiene la regla:

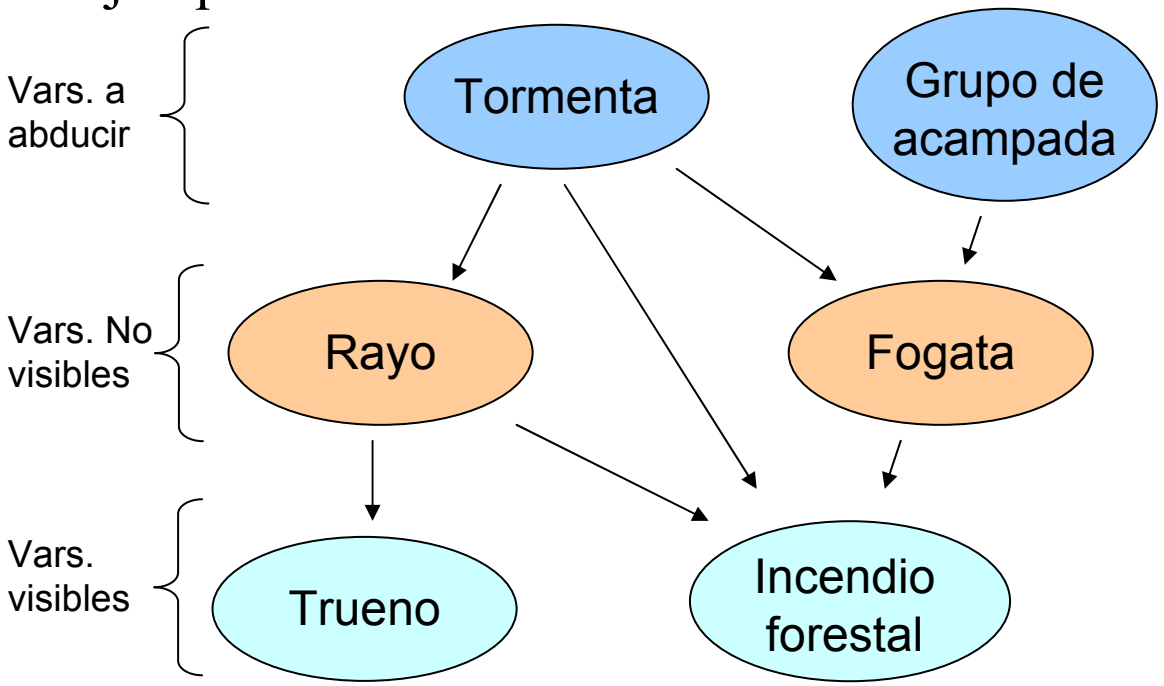
```
safetostack(X,Y)  :-  volume(X,VX), density(X, DX), equal(WX, times(VX,DX)),
                      lessthan(WX, 5), type(Y, endtable)
```

Aprendizaje Analítico (Abductivo)

Redes Bayesianas (o Bayesian Belief Networks)

En su versión más simple, son grafos acíclicos dirigidos de proposiciones en las que las flechas representan dependencias probabilísticas y la ausencia de flechas (directa o indirectamente) indica independencia.

Ejemplo:



	T, G	T,¬G	¬T,G	¬T,¬G
F	0.4	0.1	0.8	0.2
¬F	0.6	0.9	0.2	0.8

Aprendizaje Analítico (Abductivo)

Uso Redes Bayesianas

Si la red está ya aprendida (todas las dependencias establecidas), se puede utilizar para:

- abducir las probabilidades de las causas iniciales (nivel superior) de unas observaciones (nivel inferior).
- deducir las probabilidades de las consecuencias finales (nivel inferior) a partir del conocimiento de unas causas (nivel superior).

Dependiendo del número de variables conocidas y desconocidas y de la estructura de la red el problema puede ser más o menos complicado (en el peor de los casos es NP-hard).

Aprendizaje Analítico (Abductivo)

Aprendizaje de Redes Bayesianas

Las redes bayesianas generalmente no se aprenden a mano. Existen métodos para aprenderlas:

- Si se conoce la estructura y se observan TODAS las variables: el aprendizaje es sencillo, como un clasificador bayesiano Naive.
- Si se conoce la estructura y las variables internas no se conocen, el problema es muy parecido al aprendizaje de una red neuronal con hidden layers. Se puede utilizar un gradient ascent, donde tras cada paso los pesos se normalizan para que definan una probabilidad:

$$\sum_j w_{ijk} = 1 \qquad \forall i, j, k : 0 \le w_{ijk} \le 1$$

- Si no se conoce la estructura: tema de investigación muy abierto.
- También tema abierto si la red es de 1er orden en vez de proposicional.

Aprendizaje Analítico (Abductivo)

Aprendizaje Abductivo en ILP

No hay una línea de separación con el aprendizaje inductivo.

EJEMPLO:

B:
has_wings(X) :- bird(X).
has_beak(X) :- bird(X).
bird(X) :- vulture(X).
carnivore(X) :- vulture(X).

E+: **has_wings(tweety), has_beak(tweety)**

Basta hacer: $B \cup \overline{E^+} \models \overline{\bot} = \overline{bird(tweety)} \wedge \overline{vulture(tweety)}$

$$\bot = bird(tweety) \vee vulture(tweety)$$

De aquí podemos considerar las siguientes hipótesis:

H1:	**bird(tweety)**	**H3:**	**bird(X)**
H2:	**vulture(tweety)**	**H4:**	**vulture(X)**

¿Cuál es más plausible? (ALP: Kakas, abduction: Poole)...

Esto es especialmente útil en abducción. En inducción ignora muchas posibles soluciones, aunque puede valer para empezar la búsqueda.

Otras Técnicas

Computación Evolutiva

Algoritmos Genéticos:

- Las hipótesis (individuos) se representan como secuencias de bits.
- Existen poblaciones y operadores de crossover, mutación y selección.
- Existe una función de Fitness.

Programación Genética:

- Los individuos son programas en algún lenguaje de programación o de reglas.

VENTAJAS: Muy flexibles a la hora de cambiar criterios de selección, representación, etc. Altamente paralelizables...

INCONVENIENTES: Necesitan mucha memoria. Sólo aseguran convergencia en situaciones muy concretas.

Otras Técnicas

Fuzzy Logic (Zadeh 1965)

- Basado en reglas fuzzy (conjuntos fuzzy).
- Muy Expresivo: if x_1 is very small and x_2 is big then class = 3
- Muy utilizado en control.

Aprendizaje: más reciente.

- Híbridos interesantes con ANN (neuro-fuzzy) y programación evolutiva (fuzzy-genetic)...
- Esta área recibe nombres fantasmas: “soft computing” (Zadeh 1994), “computational intelligence”.
- Existen métodos de de-fuzzificación.

Transparencias sobre el tema: “http://www.abo.fi/~rfuller/nfs.html”.

Resumen de Métodos

Resumen de Métodos

	Con Modelo	Sin Modelo o no inteligible
EAGER	• Reg. Lineal • k-means. • ID3, C4.5, CART. • Center Splitting. • Pick-and-Mix. • ILP, IFLP.	• Perceptron Learning, ANN. • Radial Basis Functions. • Bayes Classifiers. • Supercharging • SCIL.
LAZY		• Reg. Lineal Pond. Local • CBR • k-NN (Nearest Neighbour).

Resumen de Métodos

Diferencia métodos Lazy / Eager

LAZY LEARNING: se generaliza para cada pregunta o predicción requerida.

- No se construye modelo. Optimización local.
- El tiempo de respuesta empieza a degradarse cuando el número de ejemplos es muy grande (no construye modelo). P.ej. k-NN.

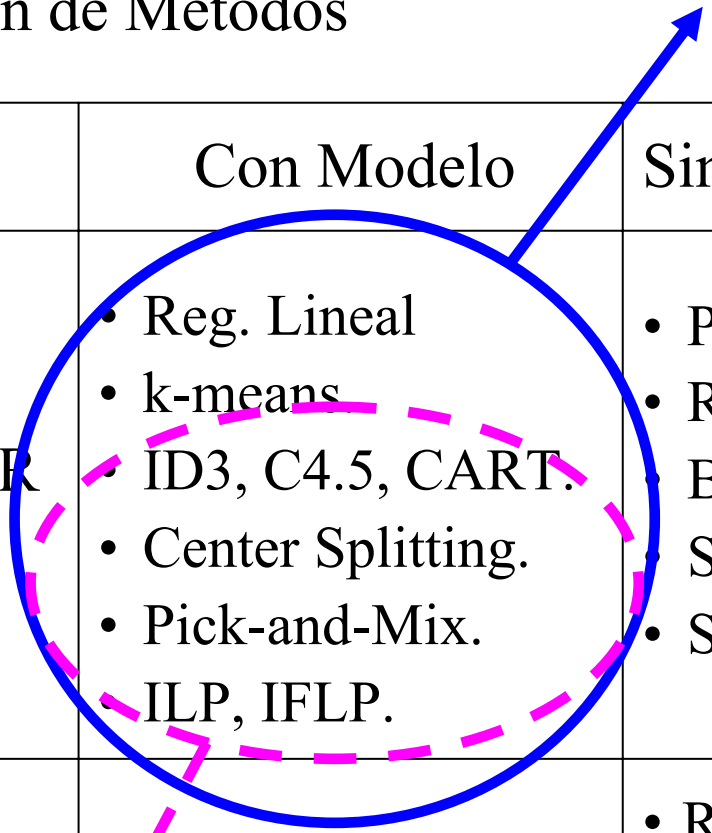
EAGER LEARNING: se generaliza a medida que se ven los ejemplos.

- Se construye modelo. Optimización global.

Para espacios de hipótesis iguales, lazy puede representar funciones más complejas.

Resumen de Métodos

Resumen de Métodos



	Con Modelo	Sin Modelo o no inteligible
EAGER	• Reg. Lineal • k-means. • ID3, C4.5, CART. • Center Splitting. • Pick-and-Mix. • ILP, IFLP.	• Perceptron Learning, ANN. • Radial Basis Functions. • Bayes Classifiers. • Supercharging • SCIL.
LAZY		• Reg. Lineal Pond. Local • CBR • k-NN (Nearest Neighbour).

Resumen de Métodos

Software Disponible:

- Librerías Genéricas:
  - MLC++ en C++. (Kohavi et al. 1994) (http://www.sgi.com/tech/mlc/)
  - WEKA en Java. (http://www.cs.waikato.ac.nz/ml/weka)

    Acompañado por el libro: "Data Mining: practical machine learning tools and techniques with Java implementations" Morgan Kaufmann
  - ML-Lisp en LISP (ftp://ftp.cs.utexas.edu/pub/mooney/ml-progs/)
  - Otros (data-mining): http://www.cs.bham.ac.uk/~anp/software.html
- Software Particular:
  - C4.5 (http://www.cse.unsw.edu.au/~quinlan/)
  - Progol, Golem y FOIL (http://www.cs.york.ac.uk/aig/ai/)
  - ILP systems (http://www-ai.ijs.si/~ilpnet2/systems/)
  - FLIP & SMILES (http://www.dsic.upv.es/~flip/)

Evaluación de Hipótesis

El problema del aprendizaje NO está especificado completamente.

- Si sólo nos basamos en la evidencia, una solución al problema sería *cualquier hipótesis que cubre la evidencia*.
- Si el lenguaje es expresivo, pueden existir infinitas hipótesis.
- Objetivo: Elegir la hipótesis *h* que MINIMIZA EL ERROR de la hipótesis *h* respecto la función objetivo *f*,

¿Qué error?

Evaluación de Hipótesis

Medidas de Error para evaluar Hipótesis

D : dominio

S : *sample* (*muestra*)

- TRUE ERROR:

caso discreto

$$error_{\mathsf{D}}(h) = \Pr_{x \in \mathsf{D}}[f(x) \neq h(x)]$$

caso continuo (p.ej.error cuadrático medio)

$$error_{\mathsf{D}}(h) = \lim_{S \to \mathsf{D}} \frac{1}{n} \sum_{x \in S} (f(x) - h(x))^2$$

- SAMPLE ERROR :

caso discreto

$$error_{train}(h) = \frac{1}{n} \sum_{x \in trainSet} \partial(f(x) \neq h(x))$$

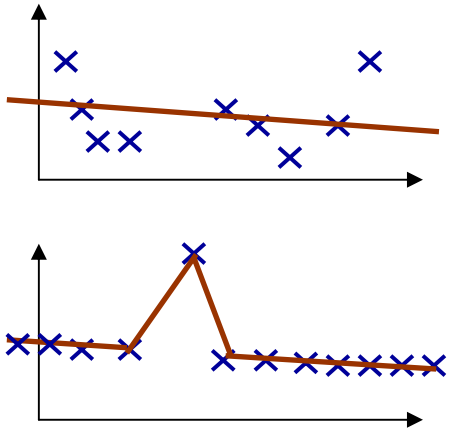
caso continuo (p.ej.error cuadrático medio)

$$error_{train}(h) = \frac{1}{n} \sum_{x \in trainSet} (f(x) - h(x))^2$$

donde (δ(true)=1, δ(false)=0) y n= |trainSet|

Evaluación de Hipótesis

- Problemas típicos:
  - under-fitting
    (sobregeneralización o subajuste)
  - over-fitting
    (sobreespecialización o superajuste).



- Definición de over-fitting: Una hipótesis $h \in H$ sobre-especializa o superajusta si existe una hipótesis alternativa $h' \in H$ tal que:

$$error_{train}(h) < error_{train}(h')$$

Sample or train error

y

$$error_{D}(h) > error_{D}(h')$$

True error

Evaluación de Hipótesis

- Problema: *f* (la función objetivo) no se conoce!!!
- Podemos calcular el SAMPLE ERROR pero no el TRUE ERROR.
- Si nos fijamos sólo en toda la muestra y minimizamos el SAMPLE ERROR, aparecerán dos problemas:
  - si la evidencia es sólo positiva: under-fitting o sobregeneralización.
  - Si la evidencia tiene más de una clase: over-fitting o sobreespecialización.

Evaluación de Hipótesis

¿Qué hipótesis elegimos?

- APROXIMACIONES:
  - Asumir distribuciones a priori. — *En frío*
  - Criterio de simplicidad, de descripción o transmisión mínimas. — *En caliente*
  - Separar: Training Set y Test Set. — *En frío*
    - Cross-validation.
  - Basadas en refuerzo. — *En caliente*

Otras preguntas importantes:

¿Cómo sabemos lo bien que se comportará en el futuro?

¿Cómo podemos comparar algoritmos de aprendizaje?

Evaluación de Hipótesis

Evaluación por técnicas bayesianas.

- La mejor hipótesis es la más probable.
- Basadas en el teorema de Bayes. Despejan $P(h|D)$.
- La distribución de hipótesis a priori $P(h)$ y la probabilidad de unas observaciones respecto a cada hipótesis $P(D|h)$ deben ser conocidas.
- Son sólo técnicas evaluadoras aunque si el conjunto de hipótesis H es reducido se pueden utilizar en algoritmos de aprendizaje.
- Permiten acomodar hipótesis probabilísticas tales como “este paciente de neumonía tiene un 93% de posibilidades de recuperarse”.
- Muchas veces no se conoce $P(h)$ o incluso $P(D|h)$. Se hacen suposiciones: distribución uniforme, normal o universal.

Evaluación de Hipótesis

Teorema de Bayes, MAP y Maximum Likelihood:

- $P(h|D)$: probabilidad de una hipótesis dado un cjto. de datos.
- $P(h)$: probabilidad a priori de las hipótesis.
- $P(D|h)$: probabilidad de D dada la hipótesis.
- $P(D)$: probabilidad a priori de los datos (sin otra información).

- Teorema de Bayes: (prob. a posteriori a partir de a priori)

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

- Criterio MAP (Maximum a Posteriori) (h es indep. de $P(D)$):

El Naive Bayes Classifier es un caso particular de esto.

$$h_{MAP} = \arg\max_{h \in H} P(h \mid D) = \arg\max_{h \in H} \frac{P(D \mid h)P(h)}{P(D)} = \arg\max_{h \in H} P(D \mid h)P(h)$$

- Maximum Likelihood (asumiendo $P(h)$ uniforme):

$$h_{ML} = \arg\max_{h \in H} P(D \mid h)$$

Evaluación de Hipótesis

Evaluación bayesiana:

Si H es pequeño y conocido:

- Se puede asumir la distribución uniforme:

$$P(h) = \frac{1}{|H|}$$

Si H es infinito:

- La distribución uniforme no está bien definida (P=0).
- Aunque el maximum likelihood se puede seguir utilizando.

Evaluación de Hipótesis

El principio MDL (Minimum Description Length):

- Asumimos $P(h)$ como la distribución universal (Occam's Razor):

$$P(h) = 2^{-K(h)}$$

donde $K(\cdot)$ es la complejidad descripcional (Kolmogorov) de H.

FORMALIZACIÓN DE LA NAVAJA DE OCCAM:

"Las hipótesis con mínima descripción más pequeña son más probables".

- Asumimos $P(D|h)$ de la misma manera:

$$P(D \mid h) = 2^{-K(D|h)}$$

Evaluación de Hipótesis

El principio MDL:

- A partir de MAP tenemos:

$$h_{MAP} = \arg\max_{k \in H} P(D \mid h)P(h) = \arg\max_{k \in H} \log\left[P(D \mid h)P(h)\right] =$$

$$= \arg\max_{k \in H} \log P(D \mid h) + \log P(h) = \arg\max_{k \in H} \log 2^{-K(D|h)} + \log 2^{-K(h)} =$$

$$= \arg\max_{k \in H} (-K(D \mid h) - K(h))$$

- Resulta en:

$$h_{MDL} = \arg\min_{k \in H} (K(h) + K(D \mid h))$$

PRINCIPIO MDL: La hipótesis más probable es la que minimiza la suma de su descripción y la descripción de los datos respecto a ella.

Evaluación de Hipótesis

Diferencia entre el principio MDL y la navaja de Occam:

Es significativa si la hipótesis no es determinista y/o si sólo hay ejemplos positivos.

Ejemplo 1:

D= E+ = (1, 3, 5, 7, 11, 13, 15, 17, 19, 21, 23, ...) la evidencia es una serie.

H_{Occam}= "los números impares en orden ascendente".

H_{MDL} = "los números impares en orden ascendente" y K(D|H)=0.

Ejemplo 2:

D= E+ = {1, 3, 5, 7, 11, 13, 15, 17, 19, 21, 23} la evidencia es un cjto.

H_{Occam}= "cualquier número es válido" $\Rightarrow$ UNDERFITTING

H_{MDL} = "los primeros 11 números impares" y K(D|H)=0.

Ejemplo 3:

D= E+ = { 24, 8, 4, 16, 2040, 2, 124, 48 } la evidencia es un cjto.

H_{Occam}= "cualquier número es válido" $\Rightarrow$ UNDERFITTING

H_{MDL} = E+, y K(D|H)=0 ó H_{MDL} = 0, y K(D|H)=E+ $\Rightarrow$ OVERFITTING

Aunque si *n* es grande $H_{MDL} = 2*(E^+/2)$, y K(D|H)=0

Evaluación de Hipótesis

LEARNING AS COMPRESSION:

- El éxito en la práctica principio MDL y la formalización de la navaja de Occam bajo la complejidad descripcional han llevado a ver el aprendizaje como compresión (eliminación de redundancia).
- Ventajas del MDL:
  - Fácil de aplicar
  - Permite guiar los algoritmos (de menos a más complejidad de hipótesis). Permite evaluar *en caliente*.
- Inconvenientes del MDL:
  - Si los datos son aleatorios o poco numerosos, no hay redundancia que explotar y la mejor hipótesis son los datos mismos (SUPER OVERFITTING).
  - No es computable en general.
  - No determina qué partes de la teoría son mejores o peores (sólo da un valor para toda la hipótesis).
  - Sólo resuelve a veces el problema de aprender con sólo E^+.

Evaluación de Hipótesis

Voting y el Clasificador Bayesiano Óptimo:

- Una pregunta es: "Qué hipótesis es más probable?"
- Otra pregunta es: "Qué predicción es más probable?"
- Consideremos una evidencia D y tres hipótesis h_1, h_2 y h_3, cuyas probabilidades a posteriori se han calculado:

$$P(h_1 \mid D) = 0.4, \quad P(h_2 \mid D) = 0.3, \quad P(h_3 \mid D) = 0.3$$

- Para la próxima observación h_1 la clasifica como positiva (true), mientras que h_2 y h_3 la clasifican como negativa (false).
- Según MAP y suponiendo $P(h)$ uniforme, la mejor hipótesis es h_1 y la nueva observación debería clasificarse como positiva...
  - Sin embargo...

La mejor clasificación de una nueva instancia se obtiene combinando las predicciones de las distintas hipótesis consideradas (*voting*).

Evaluación de Hipótesis

Voting y el Clasificador Bayesiano Óptimo:

- Justificación:

$$P(v_j \mid D) = \sum_{h_i \in H} P(v_j \mid h_i) P(h_i \mid D)$$

- Para el ejemplo anterior:

$$P(h_1 \mid D) = 0.4, \quad P(h_2 \mid D) = 0.3, \quad P(h_3 \mid D) = 0.3$$

$$P(false \mid h_1) = 0, \quad P(false \mid h_2) = 1, \quad P(false \mid h_3) = 1$$

$$P(true \mid h_1) = 1, \quad P(true \mid h_2) = 0, \quad P(true \mid h_3) = 0$$

- Por tanto:

$$P(false \mid D) = \sum_{h_i \in H} P(false \mid h_i) P(h_i \mid D) = 0.6$$

$$P(true \mid D) = \sum_{h_i \in H} P(true \mid h_i) P(h_i \mid D) = 0.4$$

Evaluación de Hipótesis

Voting y el Clasificador Bayesiano Óptimo:

- No existe otro método de clasificación mejor si se conocen el espacio de hipótesis y su distribución a priori.

- Es importante resaltar que las predicciones que se van generando por el clasificador bayesiano óptimo forman una nueva hipótesis $h_{OBC.}$

- Esta hipótesis puede ser que incluso no pertenezca a H!!! Esto permite sobrepasar el límite de expresividad inicial marcado por H.

- Sin embargo, el mayor problema es que la hipótesis h_{OBC} muchas veces no es representable y mucho menos inteligible y no proporciona conocimiento, sólo buenas predicciones.

Evaluación de Hipótesis

PARTICIÓN DE LA MUESTRA

- Evaluar una hipótesis sobre los mismos datos que han servido para generarla da siempre resultados muy optimistas.

Solución: PARTIR EN: Training Set y Test Set.

- Si los datos disponibles son grandes (o ilimitados) :
  - *Training Set*: cjto. con el que el algoritmo aprende una o más hipótesis.
  - *Test Set*: cjto. con el que se selecciona la mejor de las anteriores y se estima su validez.
- Para problemas con *clase discreta*, se calcula la "accuracy", que se mide como el porcentaje de aciertos sobre el test set.
- Para problemas con *clase continua*, se utiliza la media del error cuadrático u otras medidas sobre el test set.

Evaluación de Hipótesis

PARTICIÓN DE LA MUESTRA (Cross-validation).

Si los datos disponibles son pequeños, partir los datos en dos cjtos restringe el número de ejemplos disponibles para el algoritmo → peores resultados.

SOLUCIONES:

- **2-fold cross-validation**: se parte en 2 cjtos. *S1* y *S2* de igual tamaño. Se entrena el algoritmo con *S1* y se evalúan las hipótesis *H1* con *S2*. Se entrena luego el algoritmo con *S2* y se evalúan las hipótesis *H2* con *S1*. Se selecciona la hipótesis con la mejor precisión o el menor error.
- ***K*-fold cross-validation**: los *n* datos se parten *k* veces ($k<n$) en 2 subcjtos. (normalmente *n/k* para test y *n-n/k* para entrenamiento, excluyendo cada vez un subconjunto distinto) y el algoritmo se ejecuta *k* veces. Se elige la mejor solución de las *k* veces.
- **Leave one out cross-validation**: Cuando $k = n$. Normalmente no hace falta apurar tanto y con K-fold para un *k* suficiente está bien

Además, la accuracy media obtenida de los k casos nos da una estimación de cómo se va a comportar la hipótesis para datos nuevos.

Evaluación de Hipótesis

Una vez obtenida una hipótesis...

¿cómo obtener su precisión (accuracy) para datos futuros?

- Utilizar la precisión para el *test data* puede ser una aproximación, ¿pero cuán buena?

- La estadística nos da soluciones para esto:
  - Suponiendo la muestra *S* de *n* ejemplos, la hipótesis *h* es discreta y son independientes.
  - Si n ≥ 30, nos permite aproximar la distribución binomial con la normal.
  - Calculado el $error_s(h)$ sobre la muestra como *nºerrores/n*

Evaluación de Hipótesis

Podemos obtener un intervalo de confianza a un nivel c:

$$error_S(h) \pm Z_c \cdot \sqrt{\frac{error_S(h)(1-error_S(h))}{n}}$$

donde Z_c es la constante obtenida de la tabla de confianzas de la normal.

- Algunos valores de la tabla normal:

Nivel de confianza c:	50%	68%	80%	90%	95%	98%	99%
Constante Z_c:	0.67	1.00	1.28	1.64	1.96	2.33	2.58

Evaluación de Hipótesis

EJEMPLO:

- Considerando que una hipótesis da 12 errores sobre 40 ejemplos, tenemos un $error_S(h) = 0.30$.
- Tenemos, por tanto, que con confianza 95% (Zc = 1.96), el intervalo del error será:

$$0.30 \pm 0.14$$

- lo que quiere decir que, para el 95% de otras muestras de 40 ejemplos que probáramos, el error estaría dentro de ese intervalo.

En general, una mejor regla para saber si se puede aplicar la evaluación anterior es que:

$$n \cdot error_S(h)(1 - error_S(h)) \geq 5$$

(si no, habría que utilizar la dist. binomial)

Evaluación en Aprendizaje Supervisado

Sobremuestreo (oversampling):

En problemas de clasificación sobre bases de datos es posible que haya muchísima más proporción de algunas clases sobre otras. Esto puede ocasionar que haya muy pocos casos de una clase:

Problema: la clase escasa se puede tomar como ruido y ser ignorada por la teoría. Ejemplo: si un problema binario (*yes* / *no*) sólo hay un 1% de ejemplos de la clase *no*, la teoría "todo es de la clase yes" tendría un 99% de precisión (accuracy).

Soluciones:

- Utilizar sobremuestro...
- Análisis ROC

Evaluación en Aprendizaje Supervisado

Sobremuestreo (oversampling / balancing):

- El sobremuestreo/submuestreo consiste en replicar/filtrar los ejemplos (tuplas) de las clases con menor/mayor proporción, manteniendo las tuplas de las clases con mayor/menor proporción.
- Esto, evidentemente, cambia la proporción de las clases, pero permite aprovechar a fondo los ejemplos de las clases más raras.

¿Cuándo se debe usar sobremuestreo/submuestreo?

- Cuando una clase es muy extraña: p.ej. predecir fallos de máquinas, anomalías, excepciones, etc.
- Cuando todas las clases (especialmente las escasas) deben ser validadas. P.ej. si la clase escasa es la de los clientes fraudulentos.

Pegas: hay que ser muy cuidadoso a la hora de evaluar los modelos.

Evaluación en Aprendizaje Supervisado

Macro-average:

- Una alternativa al sobremuestreo consiste en calcular la precisión de una manera diferente.
- Habitualmente, la precisión (accuracy) se calcula:

$$acc(h) = aciertos \,/\, total$$

(conocido como *micro-averaged accuracy*)

- La alternativa es calcular la precisión como:

$$acc(h) = \frac{\sum aciertos_{clase\_i} \,/\, total_{clase\_i}}{n^{o}\, clases}$$

(conocido como *macro-averaged accuracy*)

De esta manera se obtiene un resultado mucho más compensado

Evaluación. Matrices de Coste y Confusión

Errores de Clasificación (confusión de clases) :

- En muchos casos de minería de datos, el error de clasificación sobre una clase no tiene las mismas consecuencias económicas, éticas o humanas que con otras.

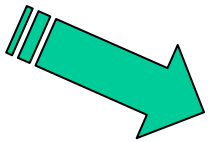
    - Ejemplo: clasificar una partida de neumáticos en perfectas condiciones como defectuoso o viceversa.

Evaluación. Matrices de Coste y Confusión

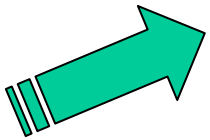
Matrices de Confusión y Coste:

- Existen técnicas para ponderar las clases → se combinan las "matrices de confusión" con las "matrices de costes":

COST		*actual*		
		low	medium	high
predicted	low	0€	5€	2€
	medium	200€	-2000€	10€
	high	10€	1€	-15€



ERROR		*actual*		
		low	medium	high
predicted	low	20	0	13
	medium	5	15	4
	high	4	7	60



Coste total:

-29787€

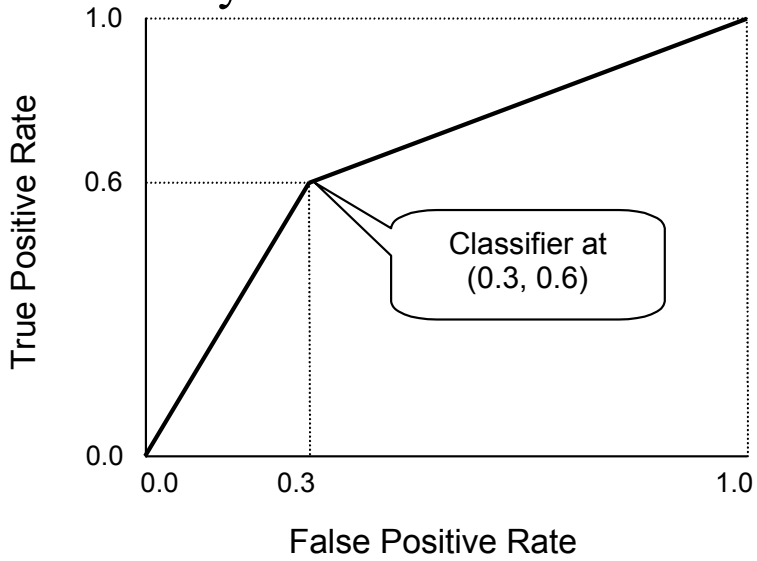
Aprendizaje Supervisado.
Análisis ROC.

Análisis ROC (Receiver Operating Characteristic):

- Se basa en dibujar el "true-positive rate" en el eje *y* y el "false-positive rate" en el eje *x*. Por ejemplo, dada la siguiente matriz de confusión:

		Actual T	Actual F
Predicted	T	30	30
	F	20	70

- Tendríamos TPR= 0.6 y FPR= 0.3.



El AUC (Area Under the ROC Curve) se usa para seleccionar la mejor solución

Evaluación de Hipótesis: score functions

Score Functions (funciones de evaluación):

continuo

- MSE (Minimum Squared Error): $error_{train}(h) = \frac{1}{n} \sum_{x \in trainSet} (f(x) - h(x))^2$

discreto

- Accuracy/Error $error_{train}(h) = \frac{1}{n} \sum_{x \in trainSet} \partial(f(x) \neq h(x))$
- Macro-averaged accuracy
- AUC (ROC)
- MSE y LogLoss adaptados para clases discretas.

Evaluación de Hipótesis

¿Cómo comparar dos (o más) algoritmos de aprendizaje?

Paired tests:

- Se trata de pasar a los dos algoritmos L_A y L_B el mismo *training set* y comparar el resultado con el mismo *test set*.
- Una manera obvia es cross-validation: disponer dos subconjuntos disjuntos de la muestra : training set (S_0) y test set T_0.
- La diferencia entre los dos se mide:

$$error_{T_0}(L_A(S_0)) - error_{T_0}(L_B(S_0))$$

- Una manera mejor es hacer un *k*-fold cross-validation y calcular la media de la diferencia de los *k* casos. Esto obliga, evidentemente, a ejecutar cada algoritmo *k* veces.

Evaluación de Hipótesis

¿Sobre qué ejemplos se comparan los algoritmos?

- Bancos de prueba:
  - Librerías de ejemplos clásicos (recopilados de artículos anteriores, de bases de datos u otras fuentes).
    - UCI ML datasets repository: http://www.ics.uci.edu/~mlearn/MLRepository.html
  - Generadores. Generan hipótesis y/o ejemplos para esa hipótesis.
    - Se puede variar la distribución a priori.
    - Para lenguajes universales, se ha de utilizar la distribución universal (gramática generativa).
    - En algunos casos, permiten estimar $P(D|h)$ y $P(h)$.

    Ejemplo: www.datasetgenerator.com

Evaluación de Hipótesis

Combinación de Hipótesis (Prodromidis et al. 1997):

- *VOTING/ARBITER/COMBINER:*
  - Se utiliza DISTINTOS algoritmos para aprender distintas hipótesis sobre todo el conjunto de los datos.
  - Luego se *combinan* las distintas hipótesis.

- Maneras de COMBINAR hipótesis:
  - WEIGHTED MAJORITY: el valor se obtiene haciendo la media (caso continuo) o la mediana (caso discreto), generalmente ponderando por la fiabilidad de predicción.
  - STACKING/CASCADE: se utiliza cada hipótesis como una variable y se utiliza otro algoritmo (p.ej. una red neuronal para asignar diferentes pesos a las diferentes hipótesis).

Evaluación de Hipótesis

Potenciación mediante Combinación de Hipótesis:

- *BOOSTING:* Se utiliza el MISMO algoritmo para aprender distintas hipótesis sobre los mismos datos, aumentando el peso de aquellos ejemplos que han sido clasificados incorrectamente. Luego se *combinan* las distintas hipótesis.

- *BAGGING*: Se utiliza el MISMO algoritmo para aprender distintas hipótesis sobre *n* muestras de *m* de los *m* datos con reemplazamiento (bootstrap). Luego se *combinan* las distintas hipótesis.

- *RANDOMIZATION*: Se utiliza el MISMO algoritmo para aprender distintas hipótesis sobre los mismos datos, pero variando aleatoriamente alguna característica del algoritmo (restringiendo los atributos que se usan cada vez, variando el criterio de selección, haciendo pick-and-mix, etc.). Luego se *combinan* las distintas hipótesis.

Evaluación de Hipótesis

Ejemplo: Boosting (reiteración):

- *A veces unos malos resultados se pueden mejorar mediante la técnica de BOOSTING:*

  Se da peso a los ejemplos y para cada iteración se aprende una nueva hipótesis, siendo los ejemplos reponderados para que el sistema se centre en los ejemplos que han sido mal clasificados.

- Algoritmo más simple:
  - Dar a todos los ejemplos el mismo peso.
  - De i:1 hasta T (nº de reiteraciones)
    - Aprender una hipótesis h_i con los pesos actuales.
    - Disminuir los pesos de los ejemplos que h_i clasifica correctamente.
  - Después se promedia una solución ponderada a partir de las T hipótesis, teniendo en cuenta la precisión de cada una.

Evaluación de Hipótesis

DATOS IMPERFECTOS:

- Tipos de Datos Imperfectos:
  - Ruido:
    - en la evidencia o ejemplos de entrenamiento.
      - Valores erróneos de argumentos de los ejemplos.
      - Clasificación errónea de algún ejemplo.
    - en el conocimiento previo.
  - Ejemplos de entrenamiento muy dispersos.
  - Conocimiento previo correcto pero inapropiado.
    - Existencia de muchos predicados irrelevantes para el problema a aprender.
    - Conocimiento previo insuficiente para el problema a aprender (algunos predicados auxiliares serían necesarios).
  - Argumentos faltantes en los ejemplos.

Evaluación de Hipótesis

DATOS IMPERFECTOS:

- Consecuencias:
  - Ruido o dispersión de datos ⇒ OVERFITTING.
    - Es necesario podar las hipótesis, eliminado partes de la hipótesis muy ad-hoc (cubren uno o pocos ejemplos). El criterio MDL es un buen método para esto.
  - Conocimiento previo inapropiado ⇒ INTRATABILIDAD
    - Demasiado conocimiento previo: se necesita metaconocimiento o priorización de los predicados.
    - Poco conocimiento previo: se necesita invención de nuevos funciones/conceptos/predicados.
  - Argumentos faltantes en los ejemplos ⇒ Se pierde tamaño de muestra si no se es capaz de aprovecharlos.
    - Los sistemas basados en árboles de decisión los tratan.

Revisión de Teorías

Y ahora, si, supuestamente, hemos encontrado una buena hipótesis...

¿qué vamos a hacer con ella?

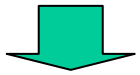
- Utilizarla para hacer predicciones.
- Combinarla con otras hipótesis obtenidas separadamente.

Pero, recordemos el escándalo de la inducción: las hipótesis nunca pueden ser confirmadas...

- ¿qué pasa si la hipótesis se ve refutada o su precisión empieza a ser muy baja, o bien es inconsistente con otra hipótesis?

APARECEN INCONSISTENCIAS

¿Hay que tirarlo todo y volver a empezar?



REVISIÓN

Incrementalidad y Revisión de Teorías

RAZONES DE INCREMENTALIDAD

- Tener todos los ejemplos a la vez es imposible. Ejemplo: un programa de aprendizaje de un robot.
- Almacenar todos los ejemplos puede ser imposible, con lo que muchos ejemplos vistos en el pasado tienen que comprimirse en las hipótesis inductivas actuales.
- Incluso aún pudiendo ser técnicamente posible almacenar todos los ejemplos, es más eficiente no hacerlo.
- La incrementalidad es necesaria cuando el número de ejemplos necesarios es desconocido.
- Imprescindible la incrementalidad para los sistemas interactivos.

Incrementalidad y Revisión de Teorías

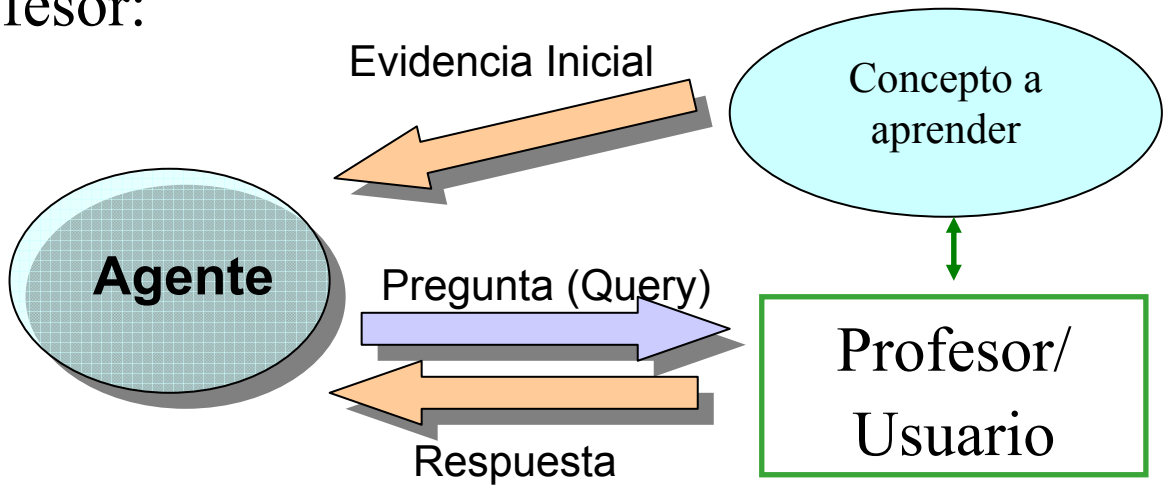
Casos en aprendizaje incremental:

- Para cada nuevo ejemplo positivo:
  - HIT: La hipótesis es consistente y cubre el nuevo ejemplo. Normalmente no se revisa (aunque pudiera ser que otra hipótesis pase a tener mejor optimalidad según un criterio de selección).
  - NOVELTY: La hipótesis es consistente pero no cubre el nuevo ejemplo. La hipótesis deberá ampliarse (extenderse) para cubrir el ejemplo y es posible que otra hipótesis pase a ser la mejor.
  - ANOMALY: La hipótesis es inconsistente con el ejemplo. La hipótesis debe revisarse (modificarse) para ser consistente y cubri el nuevo ejemplo (y los anteriores).
- Para cada nuevo ejemplo negativo:
  - Sólo existen HIT y ANOMALY.

Query Learning

Query Learning (Angluin 1987)(Angluin 1988)

- El sistema de aprendizaje puede hacer preguntas a un profesor:



- La complejidad computacional del aprendizaje es menor.

Tradicionalmente las preguntas consideradas eran simplemente preguntas sobre la clase de algún ejemplo (preguntas no cuantificadas).

Query Learning

Query Learning con ILP

- La prog. lógica permite expresar preguntas más ricas:
  - Preguntas básicas: interrogar al profesor acerca del valor de verdad de un átomo básico. No hay cuantificadores.
  - Preguntas existenciales: del tipo "*qué instancias de las variables hacen del átomo un ejemplo positivo*".

    (∃X) append(X,[1,2],[0,1,2]) **respuesta: X=0**
    (∃X,Y) append(X,Y,[0,1,2]) **4 respuestas**

    TIPO I: respuesta *si* o *no*
    (∃X) append(X,[1,2],[0,1,2]) **respuesta: sí**
    TIPO II: respuesta *instancia de las variables*
    (∃X) append(X,[1,2],[0,1,2]) **respuesta: X=0**

  - Preguntas de corrección: el sistema da al usuario una cláusula de (o toda) la hipótesis y le pregunta acerca de su corrección.

Query Learning

Query Learning con ILP. Otras ventajas:

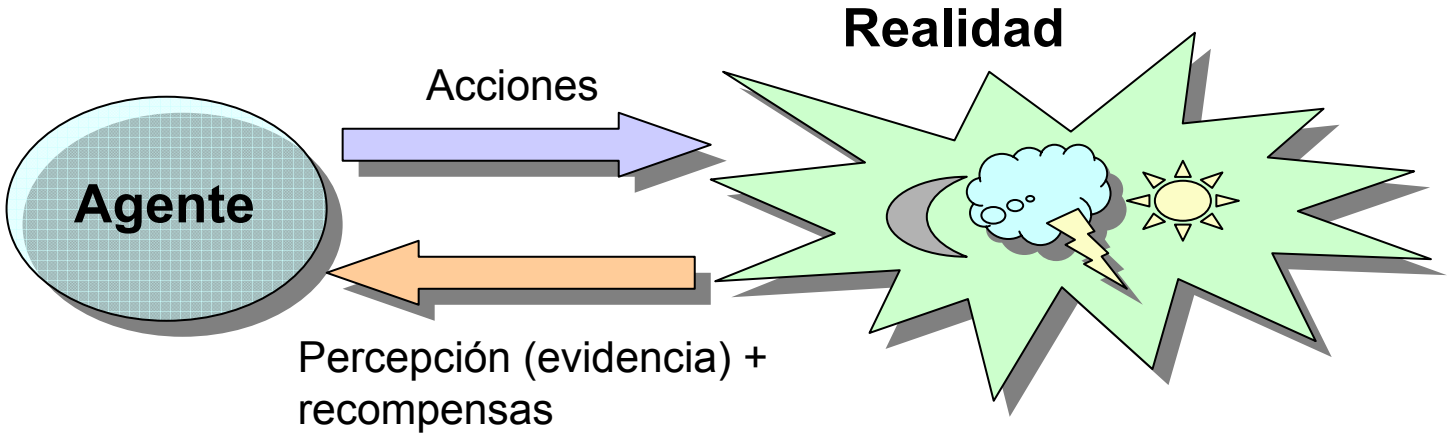
El profesor puede ser:

- un sistema automático (¡una base de datos!)
- un ser humano (¡las expresiones lógicas son fácilmente inteligibles!)

Muy útil cuando tratemos lenguajes de consulta inductivos...

Aprendizaje por Refuerzo

- Tipo de aprendizaje incremental interactivo que se basa en recompensas (y penalizaciones):

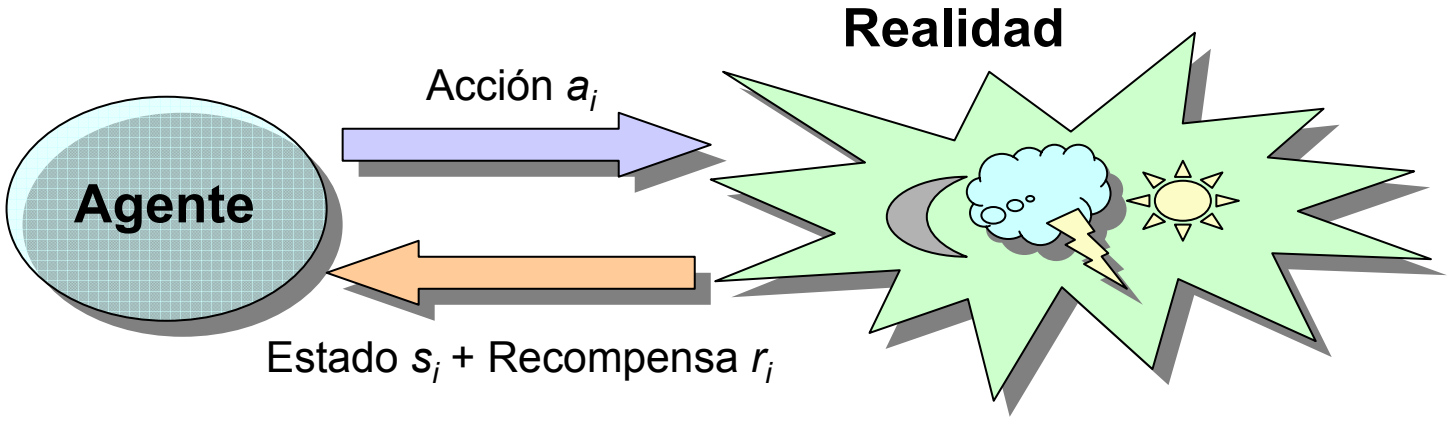


- Ya no se trata de cubrir o no la evidencia con más o menos error, sino de maximizar la recompensa del entorno.

(*a veces se conoce como aprendizaje débilmente supervisado*)

Aprendizaje por Refuerzo

- Simplificación. Hay unos estados y acciones predeterminados, junto con unas recompensas:



$$s_0 \xrightarrow[r_0]{a_0} s_1 \xrightarrow[r_1]{a_1} s_2 \xrightarrow[r_2]{a_2} \dots$$

- OBJETIVO: aprender las acciones que maximizan

$$r_0 + \gamma r_1 + \gamma^2 r_2 + \dots \text{, donde } 0 \le \gamma \le 1$$

Aprendizaje por Refuerzo

- Aplicaciones:
  - robótica.
  - control.
  - aprendizaje de políticas de juegos.
  - investigación operativa.
  - personalización de interfaces y aplicaciones.
- Técnicas:
  - Markov Decision Processes: si el estado y la recompensa sólo dependen del estado y la acción anterior.
  - La complejidad depende de si se conoce la función entre estados o la función que va de estado a recompensa. Generalmente no se conoce ninguna de las dos.
  - El agente aprende una función de evaluación (denotada generalmente por Q) de todas las acciones y sobre ella actúa.

Aprendizaje por Refuerzo

- Q learning (caso más simple):
  - Para cada s,a inicializar en la tabla $Q(s,a):=0$
  - Observar estado actual s
  - Repetir indefinidamente:
    - Seleccionar una acción a según criterio (más explorativo o más provechoso) y realizarla.
    - Recibir la recompensa inmediata r.
    - Observar el nuevo estado s'.
    - Modificar la entrada para $Q(s,a)$ como sigue:

$$Q(s,a) = r + \gamma \max_{a'} Q(s',a')$$

    - s:= s'

Aprendizaje por Refuerzo

Refuerzo Constructivo:

Acciones y Conocimiento se mezclan.

Ejemplo:

```
danger(Pipe) :- type(Pipe, T), pression(Pipe, P),
                max_pression_allowed(T, MP), MP > P.
action(open(Valve2)) :- is_in(Valve2, Pipe1), danger(Pipe1).
action(open(Valve2)) :- is_in(Valve2, Pipe1), flow_into(Pipe2, Pipe1),
                        danger(Pipe2).
max_pression_allowed(category1, 2000).
max_pression_allowed(category2, 300).
```

Cada vez que el agente abre correctamente la válvula, recibe refuerzo $r_{sí,sí}$. Si no abre y no tiene que abrir recibe un pequeño refuerzo $r_{no,no}$ (todo va bien).

Si abre la válvula cuando no toca, recibe una penalización $p_{sí,no}$. Si no abre cuando hay que abrirla recibe una super-penalización $p_{no,sí}$.

Aprendizaje por Refuerzo

Refuerzo Constructivo:

- ¿Qué revisamos? (problema de propagación de refuerzo y de asignación de culpa (blame assignment)):

Supongamos: $r_{sí,sí} = 20$, $r_{no,no} = 1$, $r_{sí,no} = -5$, $r_{no,sí} = -100$

Supongamos la distribución de la evidencia:

$|e_{sí,sí,categ1}|=2$ $|e_{no,no,categ1}|=100$ $|e_{sí,no,categ1}|=10$ $|e_{no,sí,categ1}|=1$

$|e_{sí,sí,categ2}|=5$ $|e_{no,no,categ2}|=90$ $|e_{sí,no,categ2}|=30$ $|e_{no,sí,categ2}|=0$

Para la categ1 hay un reward total de: $2\cdot20 + 100\cdot1 + 10\cdot(-5) + 1\cdot(-100)= -10$

Para la categ2 hay un reward total de: $5\cdot20 + 90\cdot1 + 30\cdot(-5) + 0\cdot(-100)= 40$

Da una idea de qué hay que revisar...

```
max_pression_allowed(category1, 2000).
max_pression_allowed(category2, 300).
```

Esta aproximación (a ojo) presenta varios problemas:

- ad-hoc para este problema.
- algunos casos solapan ($e_{no,no,categ1}$, $e_{no,no,categ2}$)

Aprendizaje por Refuerzo

Refuerzo Constructivo:

- Soluciones más genéricas:
- Medidas de evaluación que determinen que partes están más reforzadas que otras...
- Esto permite saber por donde hay que empezar a revisar.
  - "Constructive Reinforcement Learning" (Hernández 2000).
    - Determina el grado de refuerzo de cada regla de una teoría (puede ser una teoría lógica, funcional o lógico-funcional).
      $\Rightarrow$ permite saber qué partes revisar
    - Determina el grado de explicación que la teoría da sobre unos determinados ejemplos.
      $\Rightarrow$ permite saber qué predicciones son más seguras.
  - Se ha demostrado que el resultado es similar al MDL pero evitando algunos casos de overfitting.

Aprendizaje por Refuerzo. Más info.

- Sutton & Barto Book: Reinforcement Learning: An Introduction
  http://www-anw.cs.umass.edu/~rich/book/the-book.html

- Reinforcement Learning
  http://www.aaai.org/Pathfinder/html/reinf.html

- Reinforcement Learning Repository at MSU - Background
  http://www.cse.msu.edu/rlr/backgr.html

- Reinforcement Learning:
  http://www.leemon.com/papers/crosstalk/ct.html

- Mance E. Harmon and Stephanie S. Harmon “Reinforcement Learning: A Tutorial2
  www.nbu.bg/cogs/events/2000/Readings/Petrov/rltutorial.pdf

Referencias del Tema 1a

(Angluin 1987) Angluin, D. 1987. Learning Regular Sets from Queries and Counterexamples. *Information and Computation*, 75: 87-106.

(Angluin 1988) Angluin, D. 1988. Queries and concept learning. *Machine Learning* 2 (4): 319-342.

(Dempster et al. 1977) Dempster, A.P.; Laird, N.M.; Rubin D.B. "Maximum likelihood from incomplete data via the EM algorithm" *Journal of the Royal Statistical Society*, Series B, 39(1), 1-38, 1977

(Kedar-Cabelli & McCarty) Kedar-Cabelli S.T., McCarty L.T. *Explanation-Based Generalization as Resolution Theorem Proving*. in: Procs. 4th Int. Workshop on Machine Learning, Ivine (CA), June 22-25, Morgan Kaufmann 1987, 383-389.

(Fahlman & Lebiere 1990) Fahlman, S.; Lebiere, C. "The Cascade-correlation learning architecture" in D.S. Tovrezky (ed.). Advances in Neural Information Processing Systems, Morgan Kaufmann 1990.

(Fisher 1987) Fisher, D. (1987) ``Knowledge Acquisition Via Incremental Conceptual Clustering," Machine Learning, 2, 139-172. Reprinted in Readings in Machine Learning, J. Shavlik & T. Dietterich (Eds.), 267-283, Morgan Kaufmann, 1990.

(Hernández 2000a) Hernández-Orallo, J. 2000. Constructive Reinforcement Learning, *International Journal of Intelligent Systems*, 15(3): 241-264.

(Kakas et al. 1993) Kakas, A.C.; Kowalski, R.A.; Toni, F. "Abductive Logic Programming" *Journal of Logic and Computation*, 2(6), 719-770, 1993.

(Kohonen 1984) Kohonen, T. "Self-Organization and Associative Memory" Springer Verlag 1984.

Referencias del Tema 1a

(Langley 1978a) Langley, P. "BACON.1 A general discovery system" in Proc. of the 2nd National Conf. of the Canadian Society for Computational Studies in Intelligence, 1978.

(Langley 1978b) Langley, P. "Rediscovering physics with bacon-3" in Proc. Of the Sixth International Joint Conference on Artificial Intelligence, vol. I, Morgan Kaufmann, 1978.

(Langley et al. 1983) Langley, P.; Bradshaw, G.; Simon, H. "Rediscovering chemistry with the BACON system" in R. Michalski, J. Carbonell, and T. Mitchell (eds.) *Machine Learning: An Artificial Intelligence Approach*. Tioga, 1983.

(Langley et al. 1987) Langley, P.; Simon, H.A.; Bradshaw, G.L.; Zytkow, J.M. *Scientific Discovery. Computational Explorations of the Creative Processes,* MIT Press, 1987. 7-3/544

(McCulloch and Pitts 1943) McCulloch, W.S.; Pitts, W. "A logical calculus of the ideas immanent in nervous activity" *Bulletin of Mathematical Biophysics*, 5, 115-133, 1943.

(Minsky and Papert 1969) Minsky, M.; Papert, S. "Perceptrons: An Introduction to Computationl Geometry" (expanded edition), MIT Press, 1969.

(Poole 1993) Poole, D. "Probabilistic Horn abduction & Bayesian Networks" Art. Int., 64(1), 81-129, 1993.

(Prodromidis et al. 1997) Prodromidis, A.; Chan, P.; Stolfo, S. "Metalearning in distributed data mining systems: issues and approaches" *in Proc. of the 3rd Int'l Conf. on Knowledge Discovery and Data Mining*, 1997.

Referencias del Tema 1a

(Rosenblatt 1962) Rosenblatt, F. *Principles of neurodynamics*, New York, Spartan Books, 1962 .

(Rumelhart et al. 1986) Rumelhart, D.; Hinton, G.; William, R. "Learning representations by back-propagating errors" Nature, 323:533-536, 1986.

(Thornton 1997) Thornton, C. "Separability is a learner's best friend" in J.A. Bullinaria, D.W. Glasspool, and G. Houghton (eds.), Proc. of the Fourth Neural Computation and Psychology Workshop, Connectionist Representations, Springer-Verlag 1997.

(Widrow and Hoff 1960) Widrow, B.; Hoff, M.E. "Adaptive switching circuits" *IRE WESCON Convention Record*, 4, 96-104, 1960,

(Zadeh 1965) Zadeh, L.A. "Fuzzy sets", *Inf. Control* 8, 338-353, 1965.

(Zadeh 1994) Zadeh, L.A. "Fuzzy Logic, Neural Networks and Soft Computing", Communications of the ACM, 37(3):77-84, 1994.

(Zadeh 2000) Zadeh, L.A. *Fuzzy Sets and Fuzzy Information Granulation Theory* (Key selected papers), collected by Da Ruan and Chongfu Huang, Beijing Normal University Press, 2000.