CIFF
Centro Internacional
de Formación Financiera

CIFF Trustees:

Santander

Universidad de Alcalá

Business Intelligence
&
Data Mining

Profesor:
Pedro Pasquau

Octubre 2015

MASTER EN BUSINESS ANALYTICS & BIG DATA
2015– 2016

Business Analytics & DataMining

A.- Data Mining

- Qué es Data Mining.
- Proceso Data Mining
- Técnicas-Algoritmos de Data Mining
- Herramienta de Data Mining: Introdución a WEKA

Business Analytics & DataMining

A.- Data Mining

- Qué es Data Mining.
- Proceso Data Mining
- Técnicas-Algoritmos de Data Mining
- Herramienta de Data Mining: Introdución a WEKA

3.3. Técnicas de Minería de Datos

- 3.3.1. El Problema de la Extracción Automática de Conocimiento.
- 3.3.2. Evaluación de Hipótesis
- 3.3.3. Técnicas no supervisadas y descriptivas.
- 3.3.4. Técnicas supervisadas y predictivas.

El Problema de la Extracción Automática de Conocimiento

La minería de datos no es más que un caso especial de aprendizaje computacional inductivo.

¿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***.

Aprendizaje Inductivo: razonamiento hipotético de casos particulares a casos generales.

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?
- ¿Cómo afecta la presencia de ruido?

Taxonomía de Técnicas de DM

Cualquier problema de aprendizaje **inductivo** se puede presentar (más o menos directamente) de cualquiera de estas cuatro formas.

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*: El contexto *B* es muy importante. El objetivo es explicar la evidencia respecto a *B*.

Inductivos: Predictivos, Descriptivos

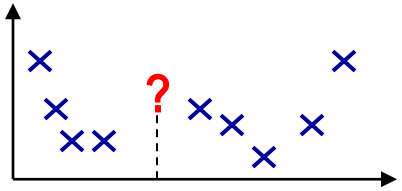
Abductivos: Explicativos

Taxonomía de Técnicas de DM

Ejemplos:

Predictivos

- *Interpolación*:



f(2.2)=?

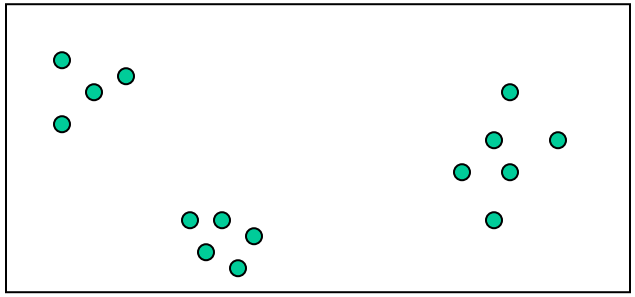
- *Predicción secuencial*: 1, 2, 3, 5, 7, 11, 13, 17, 19, ... ?
- *Aprendizaje supervisado*:

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

4 2 -> ?

Descriptivos

- *Segmentación (Aprendizaje no supervisado)*:



¿Cuántos grupos hay?
¿Qué grupos formo?

- ***Análisis Exploratorio: Correlaciones, Asociaciones y Dependencia***

Taxonomía de Técnicas de DM

PREDICTIVO: 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.

Taxonomía de Técnicas de DM

PREDICTIVO: 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.*

Taxonomía de Técnicas de DM

PREDICTIVO: Aprendizaje supervisado (Clasificación).

- 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.
  - Rules (CN2)
  - Pseudo-relational: Supercharging, Pick-and-Mix.
  - Relational: ILP, IFLP, *SCIL*.

Similarity-Based

Fence and Fill

Técnicas de Aprendizaje Automático

DESCRIPTIVO: Análisis Exploratorio

- Técnicas:
  - Estudios correlacionales
  - Asociaciones.
  - Dependencias.
  - Detección datos anómalos.
  - Análisis de dispersión.

Taxonomía de Técnicas de DM

DESCRIPTIVO: Segmentación (Aprendizaje no supervisado)

- Técnicas de *clustering*:
  - k-means (competitive learning).
  - redes neuronales de Kohonen
  - EM (Estimated Means) (Dempster et al. 1977).
  - Cobweb (Fisher 1987).
  - AUTOCLASS
  - ...

Similitud/Distancia

Un concepto importante en el aprendizaje supervisado (clasificación) y no supervisado (segmentación) es el concepto de similitud:

- La razón de este uso es que, intuitivametne, datos similares tendrán clases/grupos 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.

Similitud/Distancia

- 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

Similitud/Distancia

- Muchísimas formas de calcular la distancia:
  - ***Distancias por Diferencia:***
    ejemplo: if $x=y$ then $D=0$ els $D=1$
  - ***Distancia de Edición:***
  - ***Distancias Específicas:*** *para los ejemplos complejos de CBR.*

Valores Discretos

3.3.1. El Problema de la Extracción Automática de Conocimiento.

3.3.2. Evaluación de Hipótesis

3.3.3. Técnicas no supervisadas y descriptivas.

3.3.4. Técnicas supervisadas y predictivas.

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

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

caso continuo (p.ej.error cuadrático medio)

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

- SAMPLE ERROR :

caso discreto

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

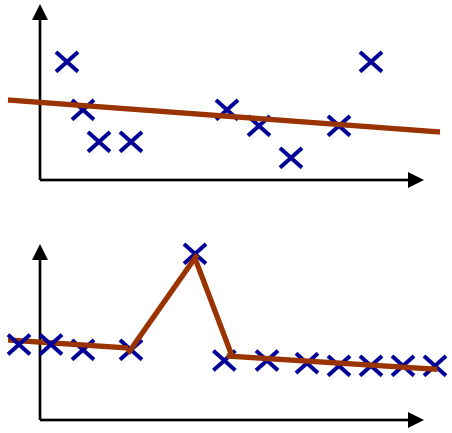
caso continuo (p.ej.error cuadrático medio)

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

- 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?

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 el cjto. de hipótesis *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

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**:

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:
- *K*-fold cross-validation:
- 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 *training 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 $n \geq 30$, 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 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 mucho conocimiento previo irrelevante 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 o del dominio: 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.

Evaluación de Hipótesis

Evaluación de Modelos Descriptivos:

- Reglas de asociación: evaluación sencilla:

  dos parámetros (support, confidence).

- No supervisados: mucho más compleja que en los predictivos.

  concepto de error difícil de definir.

  En los métodos basados en distancia se pueden mirar ciertos parámetros:

  - distancia entre bordes de los clusters
  - distancia entre centros (de haberlos)
  - radio y densidad (desv. típica de la dist.) de los clusters.

Para cada ejemplo a agrupar se comprueba su distancia con el centro o con el borde de cada cluster.

3.3.1. El Problema de la Extracción Automática de Conocimiento.
3.3.2. Evaluación de Hipótesis
3.3.3. Técnicas no supervisadas y descriptivas.
3.3.4. Técnicas supervisadas y predictivas.

Métodos Descriptivos

Correlación y Asociaciones (análisis exploratorio o *link analysis*):

- 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).

Métodos Descriptivos

Correlaciones y Estudios Factoriales:

Permiten establecer relevancia/irrelevancia de factores y si aquélla es positiva o negativa respecto a otro factor o variable a estudiar.

Ejemplo (Kiel 2000): Estudio de visitas: 11 pacientes, 7 factores:

- Health: salud del paciente (referida a la capacidad de ir a la consulta). (1-10)
- Need: convicción del paciente que la visita es importante. (1-10)
- Transportation: disponibilidad de transporte del paciente al centro. (1-10)
- Child Care: disponibilidad de dejar los niños a cuidado. (1-10)
- Sick Time: si el paciente está trabajando, puede darse de baja. (1-10)
- Satisfaction: satisfacción del cliente con su médico. (1-10)
- Ease: facilidad del centro para concertar cita y eficiencia de la misma. (1-10)
- No-Show: indica si el paciente no se ha pasado por el médico durante el último año (0-se ha pasado, 1 no se ha pasado)

Métodos Descriptivos

Correlaciones y Estudios Factoriales. Ejemplo (cont.):

Matriz de correlaciones:

	Health	Need	Transp'tion	Child Care	Sick Time	Satisfaction	Ease	No-Show
Health	1							
Need	-0.7378	1						
Transportation	0.3116	-01041	1					
Child Care	0.3116	-01041	1	1				
Sick Time	0.2771	0.0602	0.6228	0.6228	1			
Satisfaction	0.22008	-0.1337	0.6538	0.6538	0.6257	1		
Ease	0.3887	-0.0334	0.6504	0.6504	0.6588	0.8964	1	
No-Show	0.3955	-0.5416	-0.5031	-0.5031	-0.7249	-0.3988	-0.3278	1

Coeficientes de Regresión:

Indica que un incremento de 1 en el factor Health aumenta la probabilidad de que no aparezca el paciente en un 64.34%

Independent Variable	Coefficient
Health	.6434
Need	.0445
Transportation	-.2391
Child Care	-.0599
Sick Time	-.7584
Satisfaction	.3537
Ease	-.0786

Métodos Descriptivos

Reglas de Asociación y Dependencia:

La terminología no es muy coherente en este campo (Fayyad, p.ej. suele llamar asociaciones a todo y regla de asociación a las dependencias):

Asociaciones:

Se buscan asociaciones de la siguiente forma:

$$(X_1 = a) \leftrightarrow (X_4 = b)$$

De los n casos de la tabla, que las dos comparaciones sean verdaderas o falsas será cierto en r_c casos:

Un parámetro T_c (confidence):

$$T_c = \text{certeza de la regla} = r_c/n$$

si consideramos valores nulos, tenemos también un número de casos en los que se aplica satisfactoriamente (diferente de T_c) y denominado T_s.

Métodos Descriptivos

Reglas de Asociación y Dependencia de Valor:

Dependencias de Valor:

Se buscan dependencias de la siguiente forma (if *Ante* then *Cons*):

P.ej. if (X1 = a, X3=c, X5=d) then (X4=b, X2=a)

De los n casos de la tabla, el antecendente se puede hacer cierto en r_a casos y de estos en r_c casos se hace también el consecuente, tenemos:

Dos parámetros T_c (confidence/accuracy) y T_s (support):

T_c = certeza de la regla = r_c/r_a, fuerza o confianza $P(Cons/Ante)$

T_s = mínimo nº de casos o porcentaje en los que se aplica satisfactoriamente (r_c o r_c/n respectivamente).

Llamado también prevalencia: $P(Cons \wedge Ante)$

Métodos Descriptivos

Reglas de Asociación y Dependencia de Valor. Ejemplo:

DNI	Renta Familiar	Ciudad	Profesión	Edad	Hijos	Obeso	Casado
11251545	5.000.000	Barcelona	Ejecutivo	45	3	S	S
30512526	1.000.000	Melilla	Abogado	25	0	S	N
22451616	3.000.000	León	Ejecutivo	35	2	S	S
25152516	2.000.000	Valencia	Camarero	30	0	S	S
23525251	1.500.000	Benidorm	Animador Parque Temático	30	0	N	N

Asociaciones:

Casado e (Hijos > 0) están asociados (80%, 4 casos).

Obeso y casado están asociados (80%, 4 casos)

Dependencias: (Hijos > 0) → Casado (100%, 2 casos).

Casado → Obeso (100%, 3 casos)

Métodos Descriptivos

Reglas de Asociación y Dependencia de Valor:

Condiciones que se suelen imponer:

$T_C > 95\%$

$T_S > 20$ (absoluto) o 50% (relativo)

Nótese que la búsqueda de asociaciones con estas condiciones no es un problema inductivo, ya que se trata de un problema completamente determinado, sin criterios de evaluación y relativamente simple.

Complejidad de los algoritmos de asociaciones y dependencias:

- Temporal: bajo ciertas condiciones de dispersión y para atributos discretos se pueden encontrar en casi tiempo lineal (Agrawal et al. 1996).

Métodos Descriptivos

Algoritmos de búsqueda de asociaciones y dependencias.
La mayoría se basa en descomponer el problema en dos fases:

- FASE A: BÚSQUEDA DE "LARGE ITEMSETS". Se buscan conjuntos de atributos con 'support' >= al support deseado, llamados 'large itemsets' (conjuntos de atributos grandes). De momento no se busca separarlos en parte izquierda y parte derecha.

- FASE B: ESCLARECIMIENTO DE DEPENDENCIAS (REGLAS). Se hacen particiones binarias y disjuntas de los itemsets y se calcula la confianza de cada uno. Se retienen aquellas reglas que tienen confianza >= a la confianza deseada.

Propiedad: cualquier subconjunto de un conjunto grande es también grande.

Métodos Descriptivos

Algoritmos de búsqueda de asociaciones.
FASE A:
Método genérico de búsqueda de "LARGE ITEMSETS"
Dado un support mínimo s_{min}:

1. i=1 (tamaño de los conjuntos)
2. Generar un conjunto unitario para cada atributo en S_i.
3. Comprobar el support de todos los conjuntos en S_i. Eliminar aquellos cuyo support < s_{min}.
4. Combinar los conjuntos en S_i para crear conjuntos de tamaño i+1 en S_{i+1}.
5. **Si** S_i no es vacío **entonces** i:= i+1. Ir a 3.
6. **Si no**, retornar $S_2 \cup S_3 \cup \ldots \cup S_i$

Hay refinamientos que permiten una mejor paralelización (dividen en subproblemas con menos tuplas y luego comprueban para todo el problema). El más famoso es el algoritmo "APRIORI" (Agrawal & Srikant 1994).

Métodos Descriptivos

Algoritmos de búsqueda de asociaciones. Ejemplo:

FASE A:

tabla:

Fila	1	2	3	4	5
1	x		x	x	
2		x	x		x
3	x	x	x		x
4		x			x

support = 2

confidence = 0.75

S_1 = { {1}, {2}, {3}, {4}, {5} } S'_1:support = { {1}:2, {2}:3, {3}:3, {5}:3 }

S_2 = { {1,2}, {1,3}, {1,5}, {2,3}, {2,5}, {3,5} } S'_2:support = { {1,3}:2, {2,3}:2, {2,5}:3, {3,5}:2 }

S_3 = { {1,2,3}, {1,2,5}, {1,3,5}, {2,3,5} } S'_3:support = { {2,3,5}:2 }

$S_{final} = S'_2 \cup S'_3$ = { {1,3}, {2,3}, {2,5}, {3,5}, {2,3,5} }

FASE B:

Se evalúa la confianza:

{1}→{3} : 1 {3}→{1} : 0.67

{2}→{3} : 0.67 {3}→{2} : 0.67

{2}→{5} : 1 {5}→{2} : 1

{3}→{5} : 0.67 {5}→{3} : 0.67

{2,3}→{5} : 1 {2,5}→{3} : 0.67 {3,5}→{2} : 1

Métodos Descriptivos

Otros tipos de asociaciones:

- Asociaciones entre jerarquías. Si existen jerarquías entre los ítems (p.ej. las familias de productos de un comercio o de un supermercado) a veces sólo es interesante buscar asociaciones inter-jerarquía y no intra-jerarquía. Esto puede reducir mucho el espacio de búsqueda.
- Asociaciones negativas. A veces nos interesa conocer asociaciones negativas, p.ej. "80% de los clientes que compran pizzas congeladas no compran lentejas". El problema es mucho más difícil en general, porque, cuando hay muchos ítems, existen muchas más combinaciones que no se dan que las que se dan.
- Asociaciones con valores no binarios y/o continuos: se deben binarizar. P.ej. Si se tiene un atributo a con k posibles valores v_1, ..., v_k ($k > 2$) se sustituye por k atributos con la condición ($a = v_i$).
  Con los atributos continuos se discretizan en rangos (0-5, 6-10, 11-15, ...) y luego se hace el mismo procedimiento.
- Asociaciones relacionales (Dehaspe and de Raedt 1997b).

Métodos Descriptivos

Patrones Secuenciales:

Se trata de establecer asociaciones del estilo:

"*si compra X en T comprará Y en T+P*"

Ejemplo:

Transaction Database

Customer	Transaction Time	Purchased Items
John	6/21/97 5:30 pm	Beer
John	6/22/97 10:20 pm	Brandy
Frank	6/20/97 10:15 am	Juice, Coke
Frank	6/20/97 11:50 am	Beer
Frank	6/21/97 9:25 am	Wine, Water, Cider
Mitchell	6/21/97 3:20 pm	Beer, Gin, Cider
Mary	6/20/97 2:30 pm	Beer
Mary	6/21/97 6:17 pm	Wine, Cider
Mary	6/22/97 5:05 pm	Brandy
Robin	6/20/97 11:05 pm	Brandy

Métodos Descriptivos

Patrones Secuenciales:

Ejemplo (cont.):

Customer Sequence

Customer	Customer Sequences
John	(Beer) (Brandy)
Frank	(Juice, Coke) (Beer) (Wine, Water, Cider)
Mitchell	(Beer, Gin, Cider)
Mary	(Beer) (Wine, Cider) (Brandy)
Robin	(Brandy)

Métodos Descriptivos

Patrones Secuenciales:

Ejemplo (cont.):

Mining Results

Sequential Patterns with Support >= 40%	Supporting Customers
(Beer) (Brandy)	John, Mary
(Beer) (Wine, Cider)	Frank, Mary

Métodos Descriptivos

Patrones Secuenciales:

Métodos Representativos (Agrawal Srikant 1995, 1996)

- *AprioriAll*
- *AprioriSome*
- *DynamicSome*

Problema: los usuarios quieren especificar restricciones sobre el tiempo máximo y mínimo entre eventos secuenciales.

Extensiones:

- Minería de patrones secuenciales con restricciones.

  P.ej. Sólo permitir las secuencias si los elementos adyacentes (p.ej. compras) suceden en un intervalo menor a dos meses.

Métodos Descriptivos

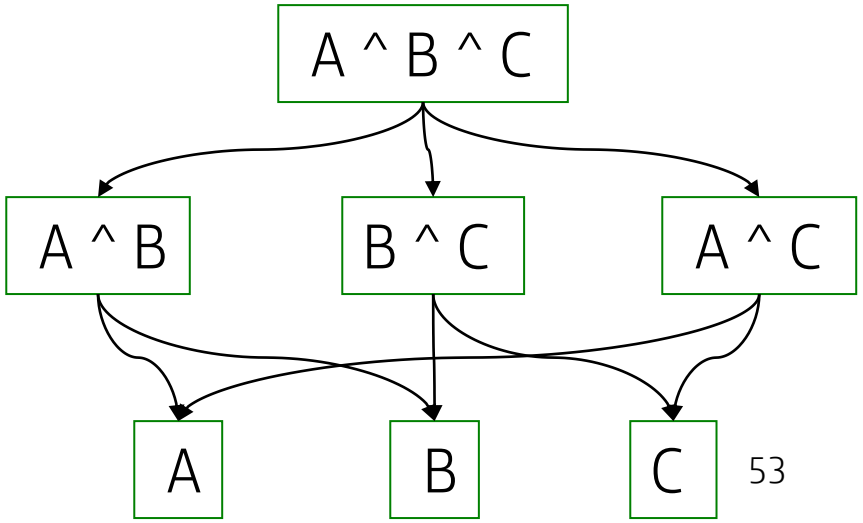
Dependencias Funcionales:

$A \wedge B \wedge C \rightarrow D$

Significa: para los mismos valores de A, B y C tenemos un solo valor de D.
Es decir D es función de A, B y C.

Si representamos la parte izquierda como un conjunto de condiciones, podemos establecer una relación de orden entre las dependencias funcionales.

Esto genera un semi-retículo.



La búsqueda se realiza en este retículo.
(Mannila & Räihä 1994)
coste exponencial
FDEP (Flach & Savnik 1999) incluye: a) simple top-down algorithm,
b) bottom-up algorithm, and
c) bi-directional algorithm.

Métodos Descriptivos

Diferencias asociaciones/dependencias, dependencias funcionales y clasificación:

1. Asociaciones y Dependencias: A=a ^ B=b ^ C=c --> D=d ^ E=e
2. Asociaciones Negativas. A=a ^ B=b ^ C=c --> D<>d ^ E<>e
3. Dependencias Funcionales: A ^ B ^ C --> D

   Si existe una tupla tal que A=X ^ B=Y ^ C=Z ^ D=W
   entonces para cualquier otra tupla que
   A=X ^ B=Y ^ C=Z entonces D=W.
   O dicho de otra manera...

```
(   SELECT MAX(COUNT(DISTINCT D))
    FROM R
    GROUP BY A, B, C;          ) = 1;
```

4. Clasificación: establecen (clarifican) una dependencia funcional. (puede ser un conjunto de dependencias de valor (1))

Métodos Descriptivos
Aprendizaje No Supervisado

Clustering (Segmentación):

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

Métodos de Agrupamiento:

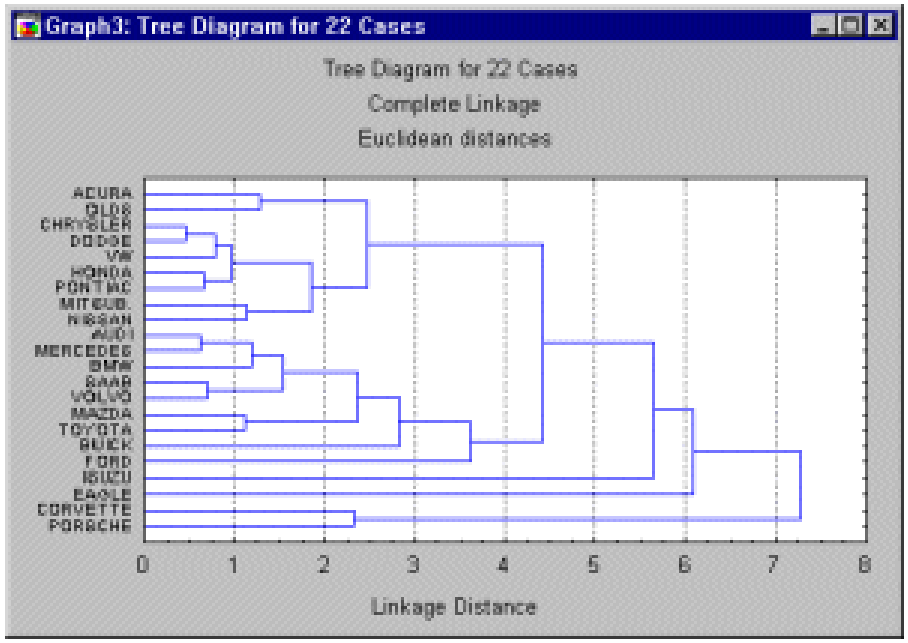
- Jerárquicos: los datos se agrupan de manera arborescente (p.ej. el reino animal).
- No jerárquicos: generar particiones a un nivel.
  - (a) Paramétricos: se asumen que las densidades condicionales de los grupos tienen cierta forma paramétrica conocida (p.e. Gaussiana), y se reduce a estimar los parámetros.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

Métodos Descriptivos
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*:



Métodos Descriptivos
Aprendizaje No Supervisado

Clustering (Segmentación). Métodos jerárquicos:

Minimal Spanning Tree Clustering Algorithm

Algoritmo (dado un número de clusters deseado C).

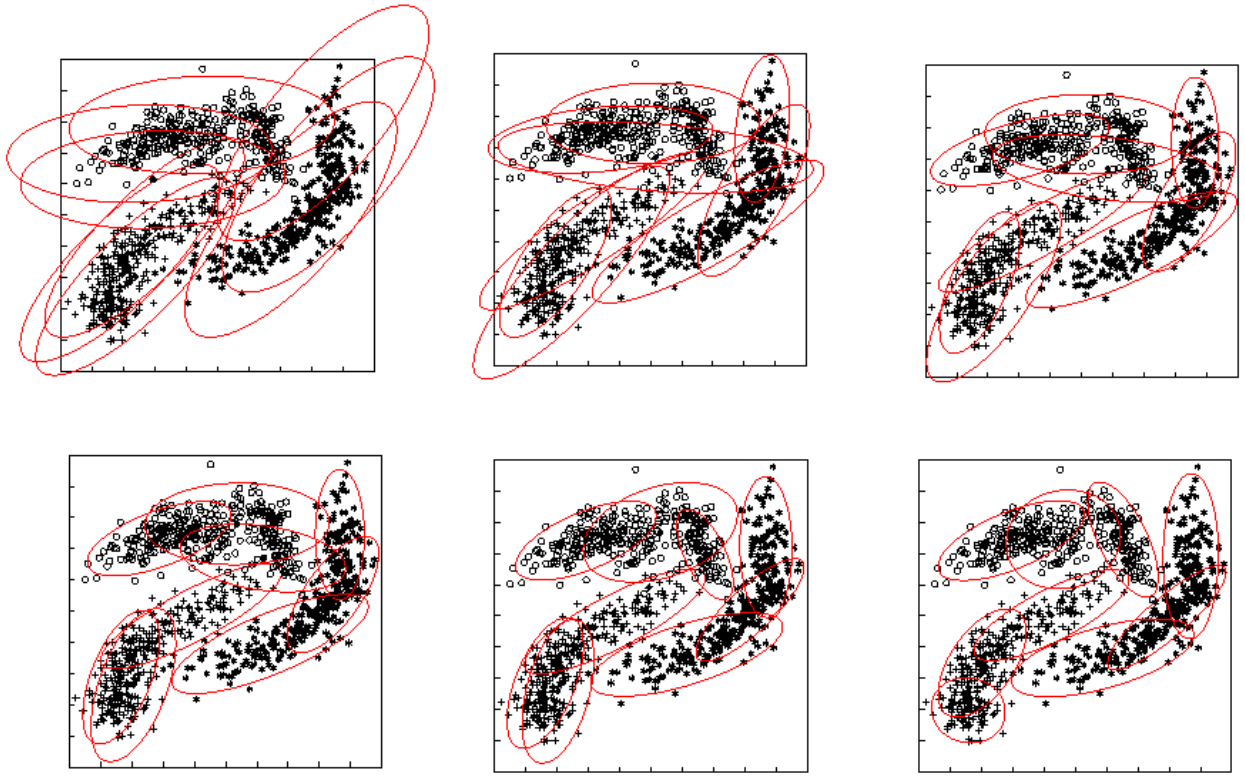
Inicialmente considera cada ejemplo como un clúster.

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

Métodos Descriptivos
Aprendizaje No Supervisado

Clustering (Segmentación). Métodos paramétricos:

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



Métodos Descriptivos
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:

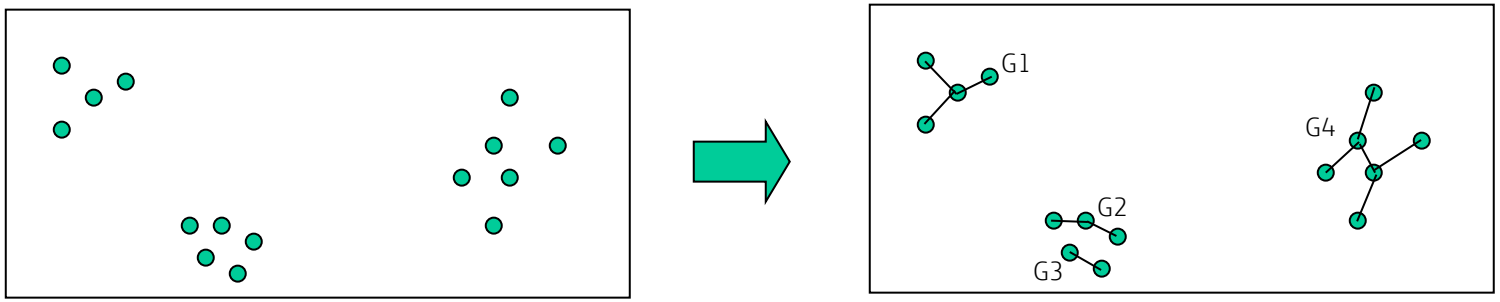
- El algoritmo Cobweb (Fisher 1987).
- El algoritmo AUTOCLASS (Cheeseman & Stutz 1996)

Métodos Descriptivos
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.

Métodos Descriptivos
Aprendizaje No Supervisado

Clustering (Segmentación). Métodos No Paramétricos

k-means clustering:

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

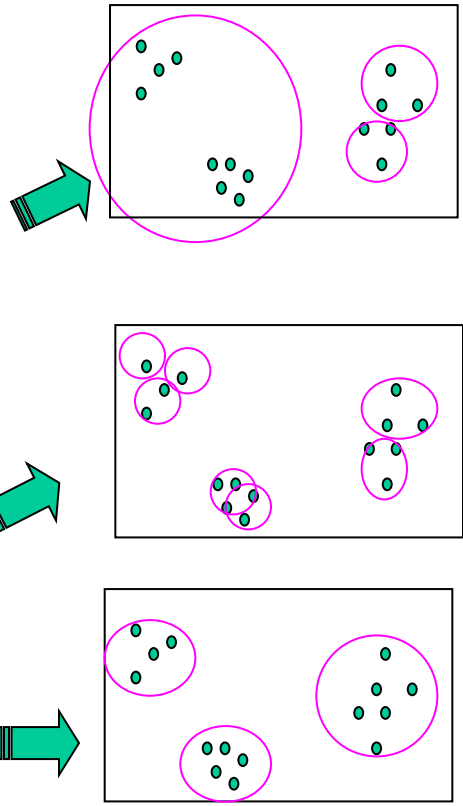
- Algoritmo:
  1. Dividir aleatoriamente los ejemplos en *k* conjuntos y calcular la media (el punto medio) de cada conjunto.
  2. Reasignar cada ejemplo al conjunto con el punto medio más cercano.
  3. Calcular los puntos medios de los *k* conjuntos.
  4. Repetir los pasos 2 y 3 hasta que los conjuntos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

Métodos Descriptivos
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.



Métodos Descriptivos
Aprendizaje No Supervisado

Clustering (Segmentación). Métodos No Paramétricos

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.

Métodos Descriptivos
Aprendizaje No Supervisado

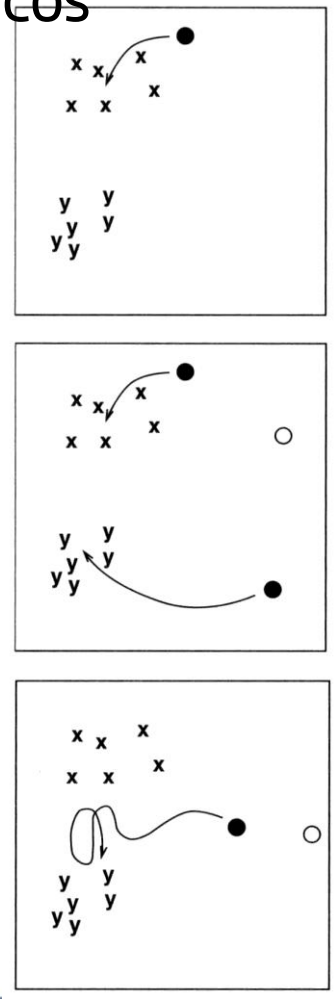
Clustering (Segmentación). Métodos No Paramétricos

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.



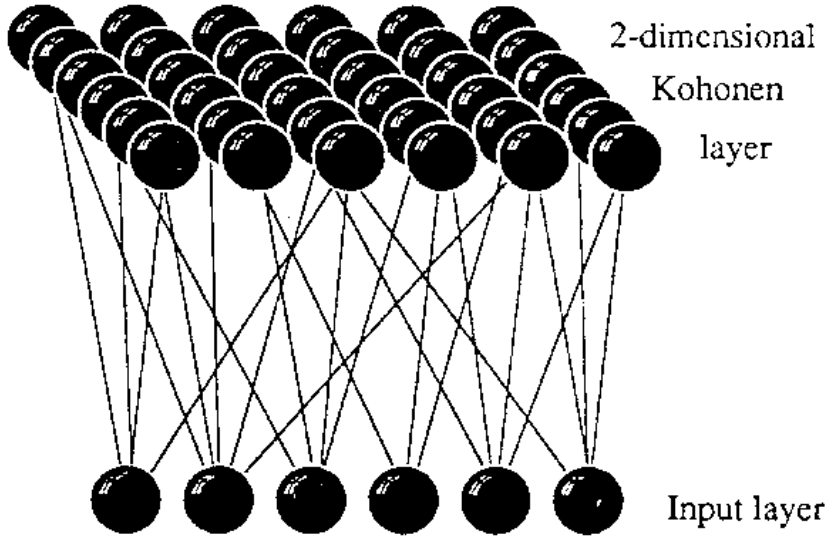
Variación muy popular: LVQ (linear-vector quantization) (Koh 1984).

Métodos Descriptivos
Aprendizaje No Supervisado

Clustering (Segmentación). Métodos No Paramétricos

SOM (Self-Organizing Maps) o Redes Kohonen

También conocidos como LVQ (linear-vector quantization) o redes de memoria asociativa (Kohonen 1984).



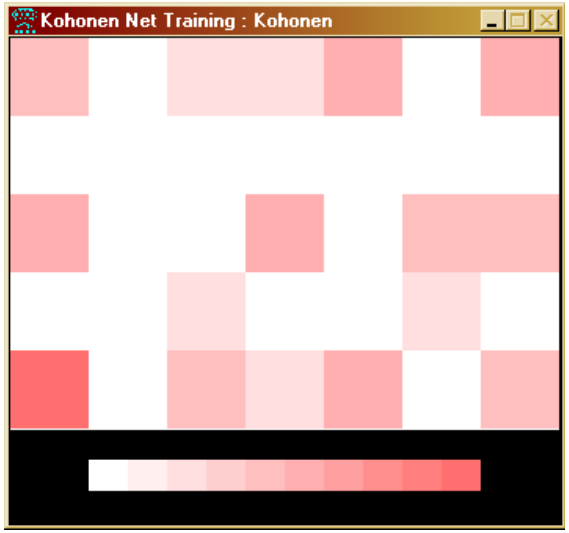
La matriz de neuronas de la última capa forma un grid bidimensional.

Métodos Descriptivos
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.

Otros Métodos Descriptivos

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.

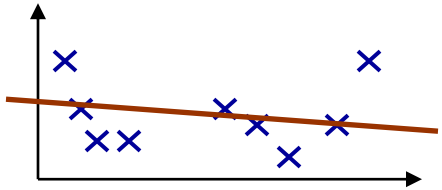
3.3.1. El Problema de la Extracción Automática de Conocimiento.
3.3.2. Evaluación de Hipótesis
3.3.3. Técnicas no supervisadas y descriptivas.
3.3.4. Técnicas supervisadas y predictivas.

Métodos Predictivos. 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 $y = w_0 + w_1 x$

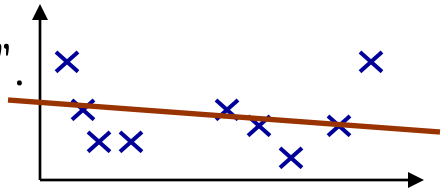
Error típico de una regresión lineal simple:

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

Métodos Predictivos. 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.

Métodos Predictivos.
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

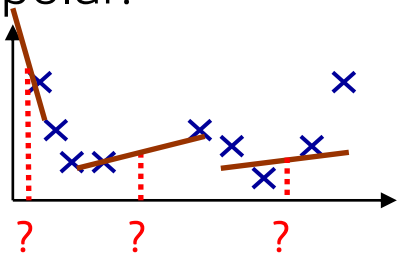
Se añaden dimensiones, combinando las dadas. P.ej. si tenemos cuatro dimensiones: x_1, x_2, x_3 (además de y) podemos definir $x_4 = x_1 \cdot x_2$, $x_5 = x_3^2$, $x_6 = x_1^{x_2}$ y obtener una función lineal de x_1, x_2, x_3, x_4, x_5, x_6

Métodos Predictivos. 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)

Métodos Predictivos. 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)

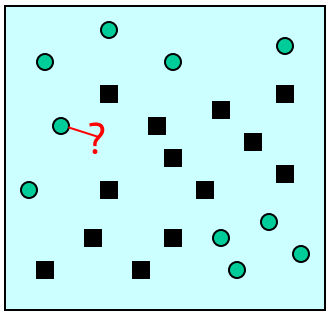
Algoritmos mucho más sofisticados (cadenas de Markov, VQ)

- Algoritmo MARS (Multiple Adaptive Regression Splines) (Friedman 1991).

Métodos Predictivos. Aprendizaje Supervisado

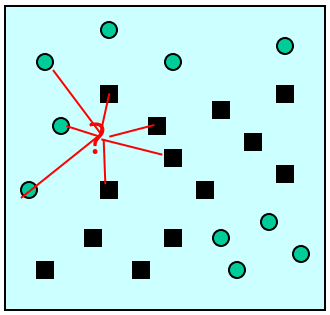
k-NN (Nearest Neighbour):

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

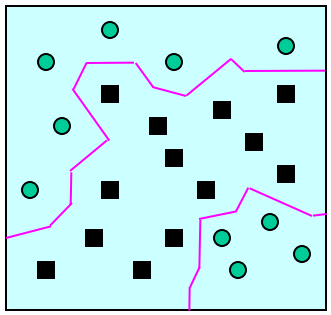
1-nearest neighbor



Clasifica cuadrado

7-nearest neighbor

(Poliédrica o de Voronoi)



PARTICIÓN DEL 1-nearest neighbor

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

Aprendizaje Supervisado

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

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

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:

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

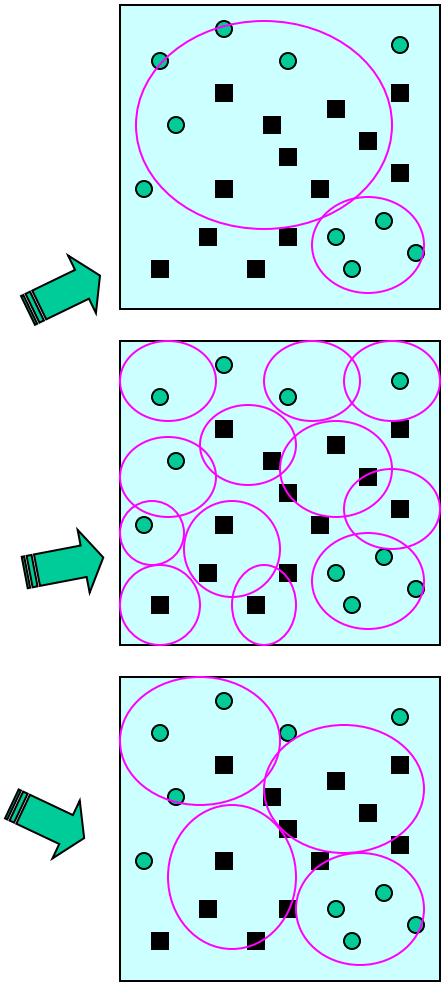
donde los x_i son los k vecinos más próximos y $f(\cdot)$ es la función que da el valor real de cada uno.

Aprendizaje Supervisado

(On-line) k-means clustering:

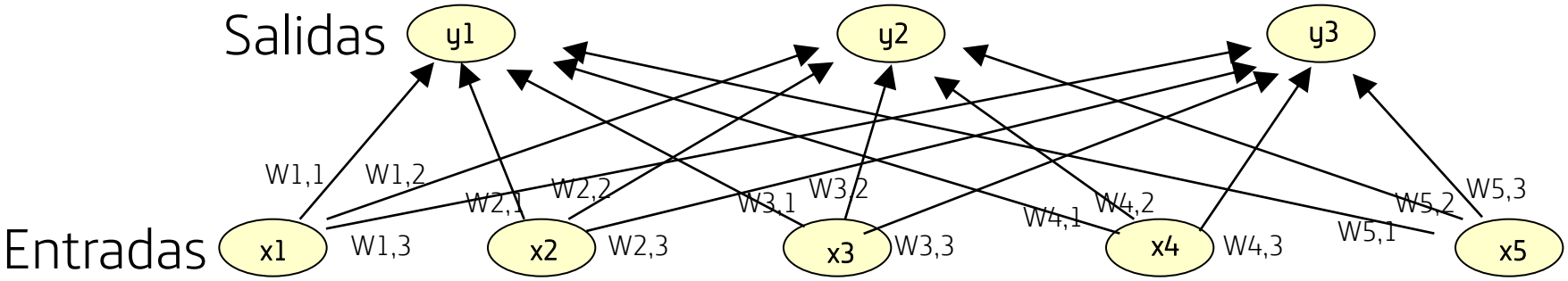
- Aunque lo vimos como una técnica no supervisada, también se puede utilizar para aprendizaje supervisado, si se utiliza convenientemente.

- Elegir un *k* mayor que el número de clases pero no mucho mayor.



Aprendizaje Supervisado

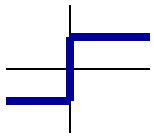
Perceptron Learning.



- Computan una función lineal para cada y_j es:

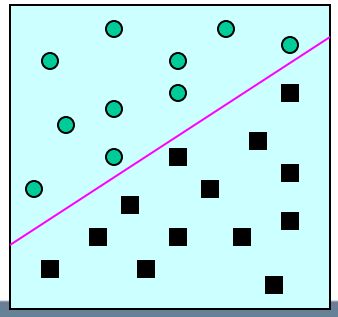
$$y'_j = \sum_{i=1}^{n} w_{i,j} \cdot x_i$$

Se añade un threshold escalón:

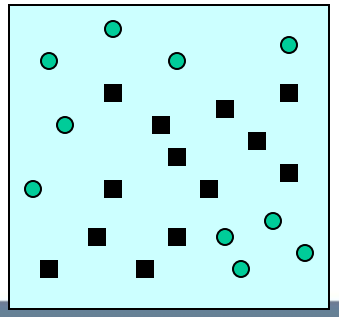


$$output_j = \mathrm{sgn}(y'_j)$$

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



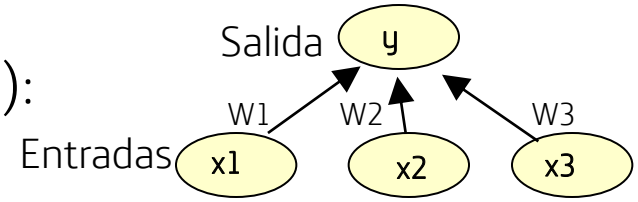
PARTICIÓN LINEAL POSIBLE



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 $(x_k, y_k)^t$ se calcula la matriz de error $(e^t_k = y^t_k - y'^t_k)$
  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*:

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

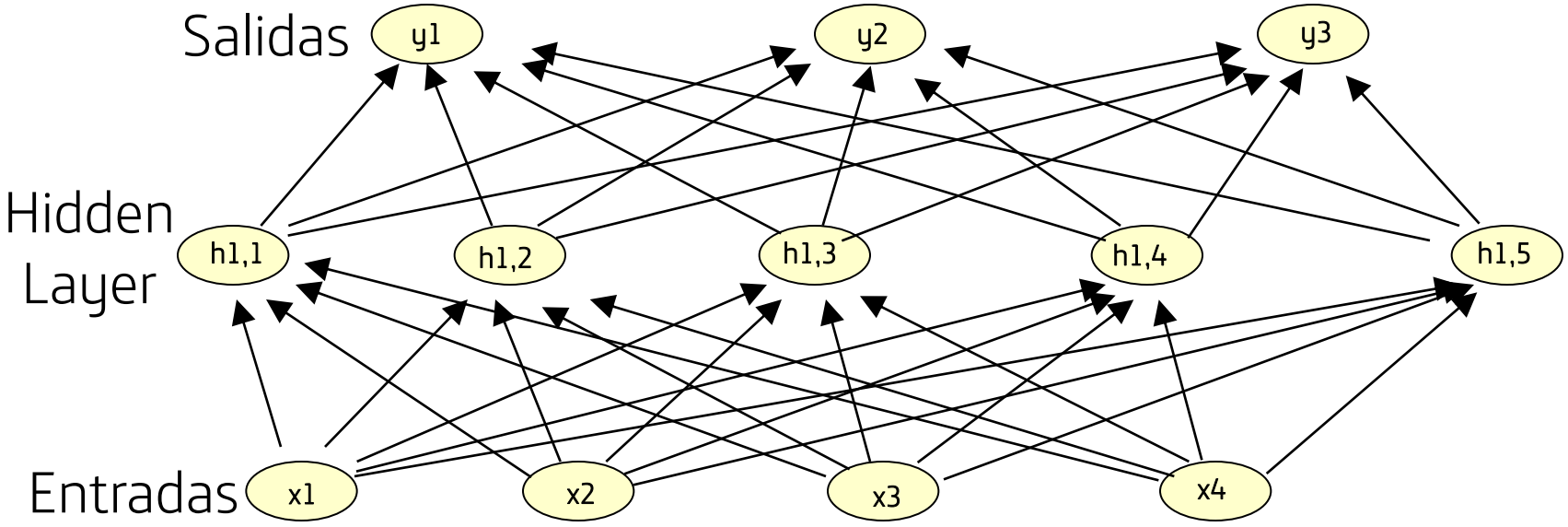
  5. t:= t+1, ir a 2 hasta que no queden ejemplos o el error medio se ha reducido a un valor deseado.

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

Aprendizaje Supervisado

Multilayer Perceptron (redes neuronales artificiales, ANN).

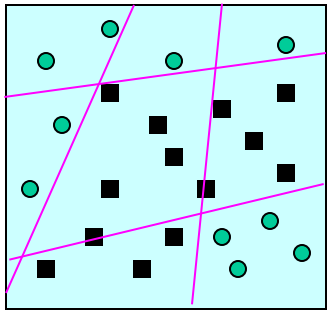
- El perceptron de una capa no es capaz de aprender las funciones más sencillas.
- Se añaden capas internas, se introducen diferentes funciones de activación e incluso recientemente se introducen bucles y retardos.



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.

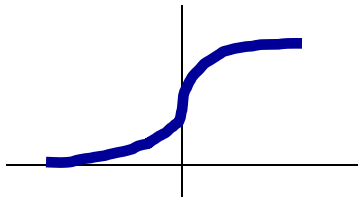
- Pero, ¿cómo entrenar este tipo de red?

Aprendizaje Supervisado

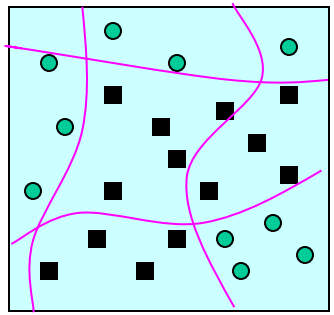
Multilayer Perceptron (redes neuronales artificiales, ANN).

- 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:

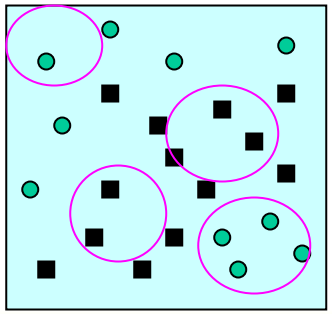
$$\delta_h = o_h(1-o_h) \sum_{k \in downstream(h)} (w_{k,h} \times \delta_k)$$

- 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.



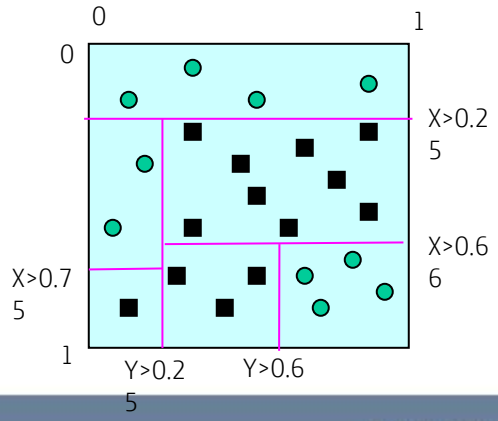
PARTICIÓN HIPERESFÉRICA CON 4 centros.

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

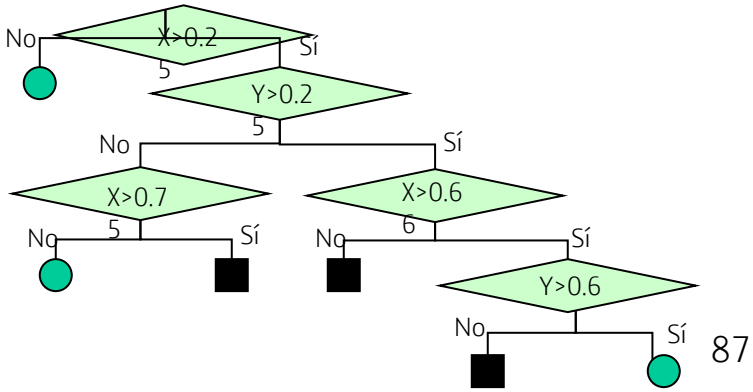
Aprendizaje Supervisado

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

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



PARTICIÓN CUADRICULAR.



Aprendizaje Supervisado

Árboles de Decisión.

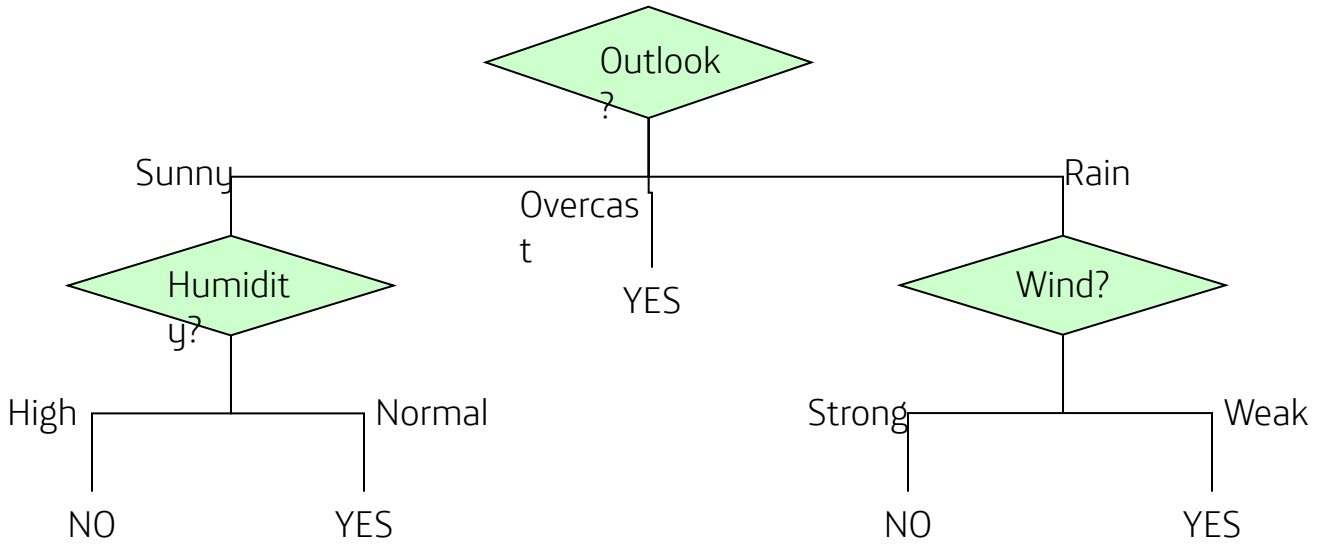
- Ejemplo C4.5 con datos discretos:

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

Aprendizaje Supervisado

Árboles de Decisión.

- Ejemplo C4.5 con datos discretos:



- Representación Lógica:

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

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

Aprendizaje Supervisado

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

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

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

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

Aprendizaje Supervisado

Naive Bayes Classifiers.

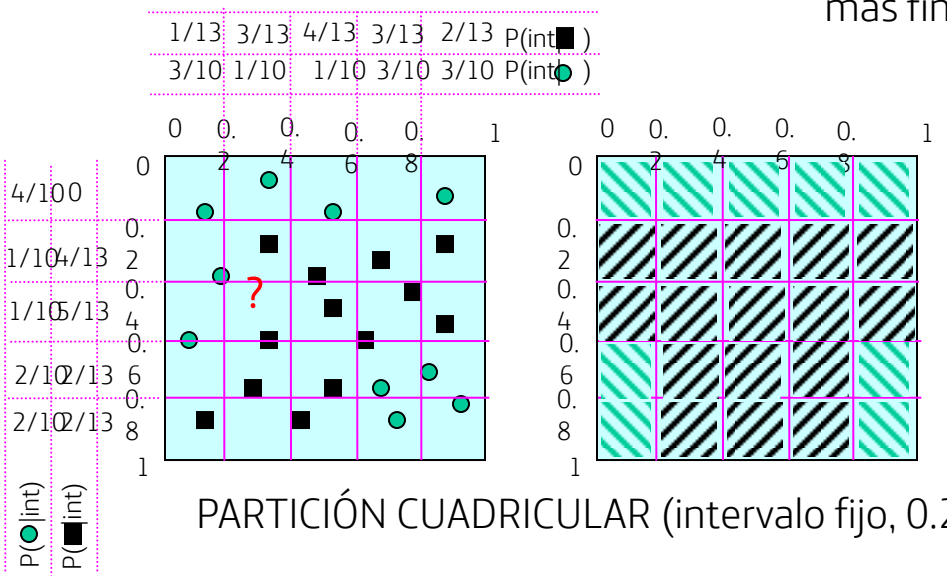
$$\arg\max_{c_i \in C} P(c_i \mid (x_1, x_2, ..., x_m)) \underset{Bayes}{=} \arg\max_{c_i \in C} \frac{P((x_1, x_2, ..., x_m) \mid c_i) \cdot P(c_i)}{P(x_1, x_2, ..., x_m)} =$$

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

- Asumiendo independencia entre los atributos, tenemos:

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

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



PARTICIÓN CUADRICULAR (intervalo fijo, 0.2).

P(●)=10/23= 0.435
P(■)=13/23= 0.565

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

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

Aprendizaje Supervisado

Naive Bayes Classifiers.

- Otra manera es hacer los intervalos continuos y calcular la frecuencia acumulada f($c_i / x_j \leq t$). Tenemos f($c_i / s < x_j \leq t$) = f($c_i / x_j \leq t$) – f($c_i / x_j \leq s$).
- 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 valor del atributo a priori.
- Generalmente p se escoge uniformemente, es decir, si hay k valores 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).

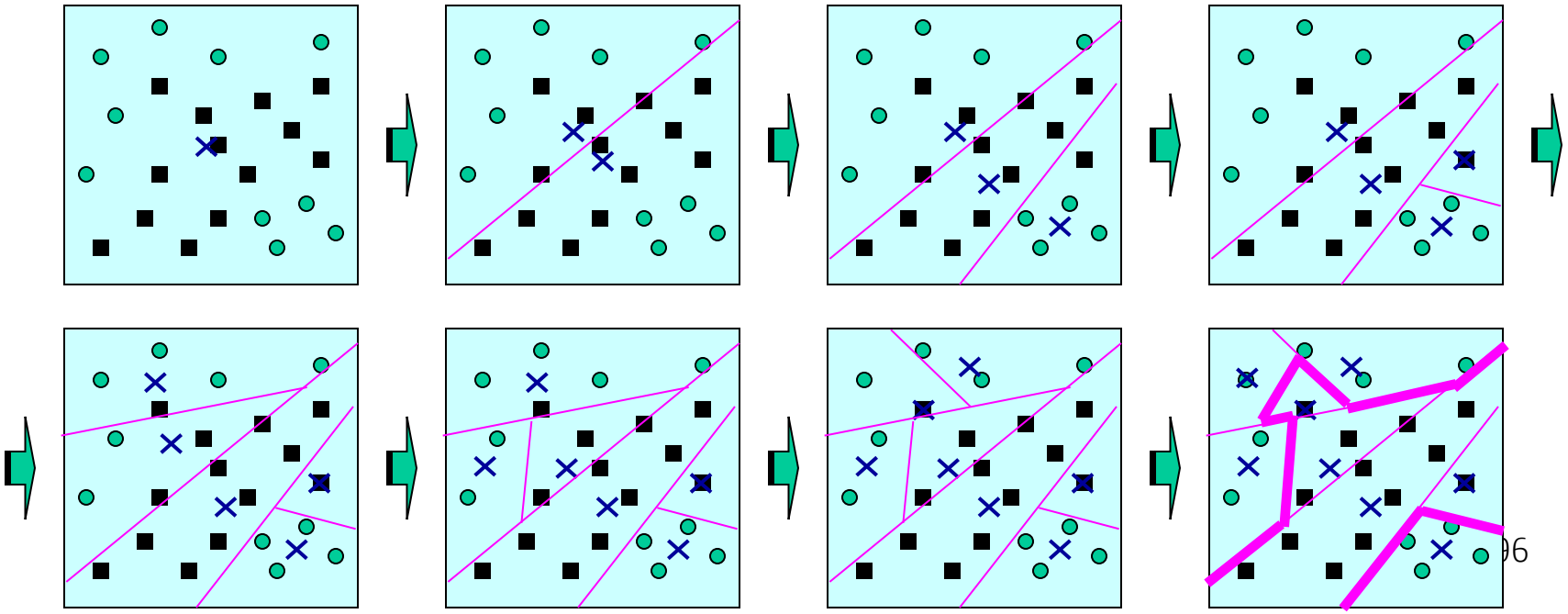
Aprendizaje Supervisado

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.

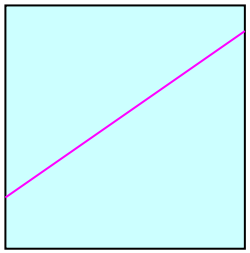
Aprendizaje Supervisado

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

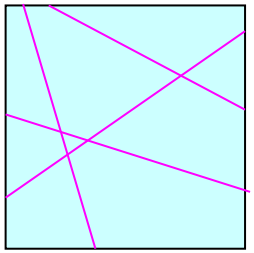


Aprendizaje Supervisado

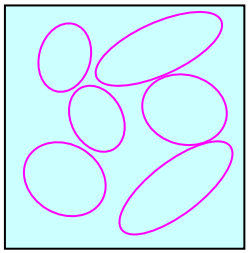
Comparación de representación:



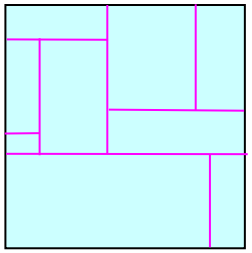
Perceptron / LMS



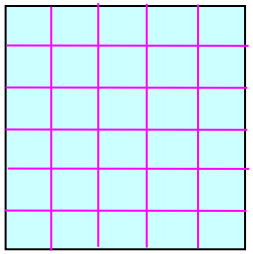
Redes Neuronales Multicapa



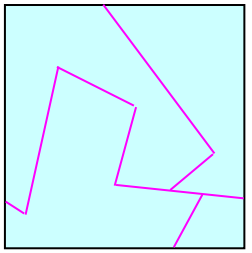
RBF



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 / Similarity-based:

- 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).

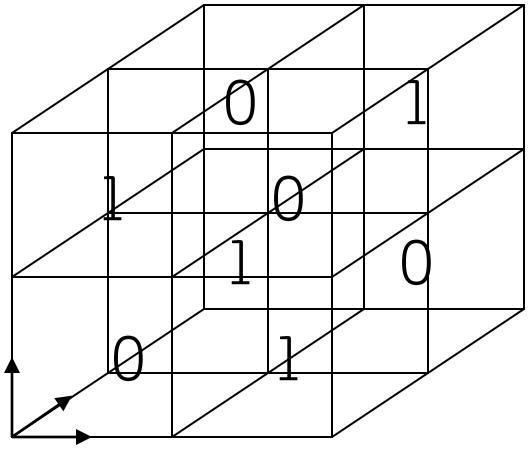
Aprendizaje Supervisado

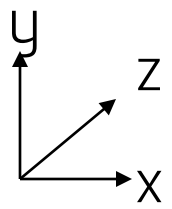
La *continuidad* hacia problemas relacionales.

- Ejemplo. Paridad

$x\ y\ z \rightarrow$ clase

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





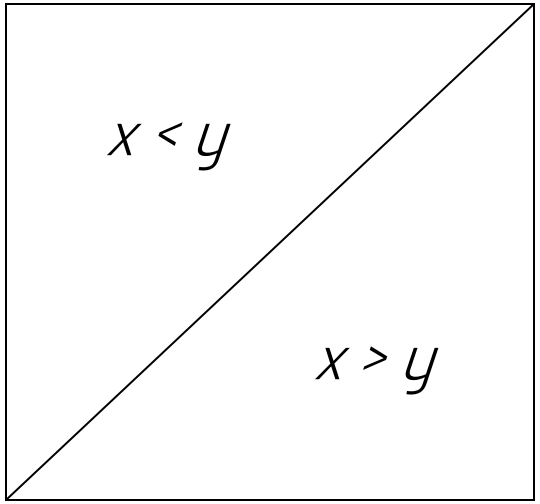
¡¡¡¡No hay manera de hacer grupos geométricos!!!!
La paridad es un problema relacional PURO.

Aprendizaje Supervisado

La *continuidad* hacia problemas relacionales.

- Ejemplo. ($x > y$)

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

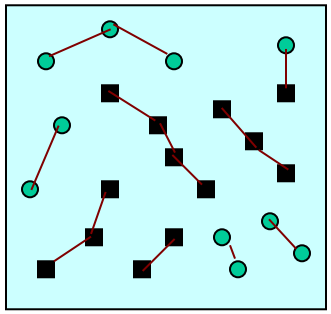
$$RGS(f) = \frac{\sum_{i=1}^{n} \frac{\sum_{e_j : dist(e_i, e_j) < r} eq(f(e_i), f(e_j))}{\left| e_j : dist(e_i, e_j) < r \right|}}{n}$$

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

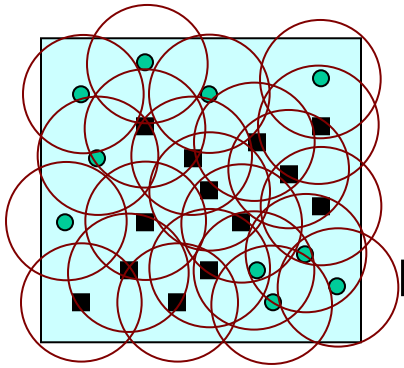
También se puede calcular una RGS' que no incluye el propio punto.

Aprendizaje Supervisado

- Ejemplos:

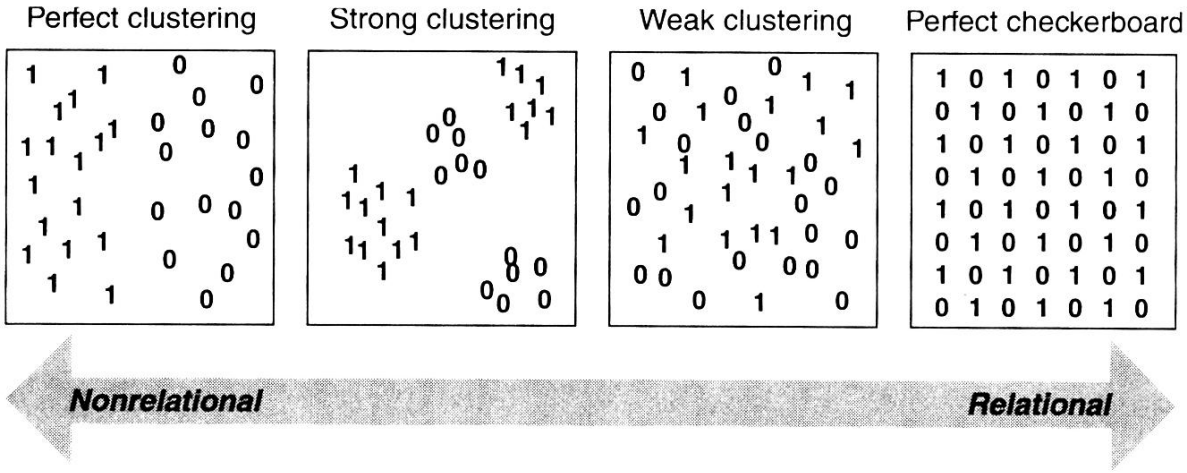


GS = 21/23 = 0.91



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 (Thornton 2000):.

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, lo que hace que parte de la solución se disponga antes de empezar (lo mismo ocurre con muchos sistemas ILP similares, como FOIL, o con la regresión logarítmica).
- 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

Condiciones de Pick & Mix:

- En minería de datos, generalmente los datos sugieren las creación de nuevos campos (columnas) por pick & mix.
- Ejemplos:
  - height^2/weight (índice de obesidad)
  - debt/earnings
  - passengers * miles
  - credit limit - balance
  - population / area
  - minutes of use / number of telephone calls
  - activation_date - application_date
  - number of web pages visited / total amount purchased
- Es conveniente añadirlas, pero siempre una sola combinación. No poner *x/y* si ya se ha puesto *x-y*.

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):

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 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

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	-	-	-	-	-	-	...

Aprendizaje Supervisado

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 Supervisado. Sobremuestreo

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

Aprendizaje Supervisado. Sobremuestreo

Sobremuestreo (oversampling / balancing):

- El sobremuestreo consiste en filtrar los ejemplos (tuplas) de las clases con mayor proporción, manteniendo las tuplas de las clases con 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?

- 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.

Aprendizaje Supervisado. Macro-average

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:

$$acc(h) = \frac{aciertos_{clase1} / total_{clase1} + aciertos_{clase2} / total_{clase2} + ... + aciertos_{clase-n} / total_{clase-n}}{nº\ clases}$$

(conocido como *macro-averaged accuracy*)

De esta manera se obtiene un resultado mucho más compensado

Aprendizaje Supervisado. 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.

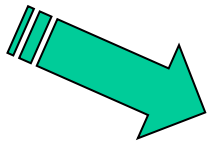
  - Ejemplo: clasificar una partida de neumáticos en perfectas condiciones como defectuoso o viceversa.

Aprendizaje Supervisado.
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":

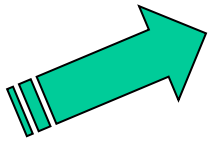
COST		*actual*		
		low	medium	high
predicted	low	0€	5€	2€
	medium	200€	-2000€	10€
	high	10€	1€	-15€



Coste total:

-29787€

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



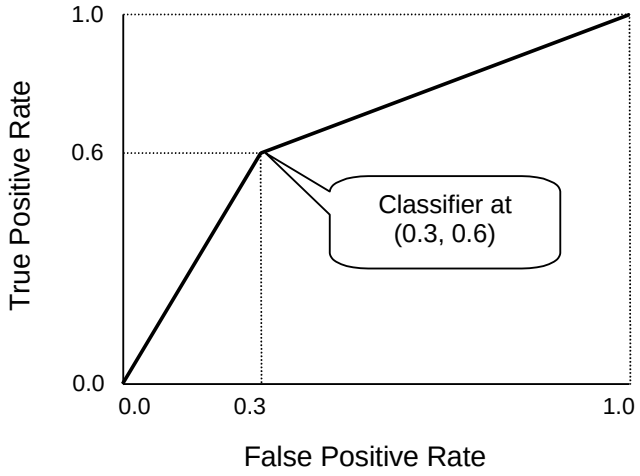
Aprendizaje Supervisado. Análisis ROC.

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	F
Predicted	T	30	30
	F	20	70

- Tendríamos TPR= 0.6 y FPR= 0.3.



Aprendizaje Supervisado. Matrices de Coste y Confusión.

Errores de Clasificación y Mailings:

- Más aún… Existen técnicas específicas para evaluar la conveniencia de campañas de 'mailings' (propaganda por correo selectiva):
- EJEMPLO: Una compañía quiere hacer un mailing para fomentar la compra de productos. En caso de respuesta positiva, los clientes suelen comprar productos por valor medio de 100€. Si un 55% suelen ser costes de producción (fijos y variables), tenemos que por cada respuesta positiva hay una ganancia media de 45€.
- Cada mailing cuesta 1€ (portes, folletos) y el conjunto de la campaña (indep. del número) tendría un coste base 20.000€.
- Con un 1.000.000 de clientes, en el que el 1% responde, ¿cómo podemos evaluar y aplicar un modelo que nos dice (ordena) los mejores clientes para la campaña?

Aprendizaje Supervisado.
Matrices de Coste y Confusión.

Errores de Clasificación y Mailings. Ejemplo:

Tabla mostrando el beneficio para cada decil:

DECILE	GAINS	CUM	LIFT	SIZE	SIZE(YES)	SIZE(NO)	PROFIT
0%	0.0%	0%	0.000	0	0	0	−($20,000)
10%	30.0%	30%	3.000	100,000	3,000	97,000	+ $15,000
20%	20.0%	50%	2.500	200,000	5,000	195,000	+ $5,000
30%	15.0%	65%	2.167	300,000	6,500	293,500	−($27,500)
40%	13.0%	78%	1.950	400,000	7,800	392,200	−($69,000)
50%	7.0%	85%	1.700	500,000	8,500	491,500	−($137,500)
60%	5.0%	90%	1.500	600,000	9,000	591,000	−($215,000)
70%	4.0%	94%	1.343	700,000	9,400	690,600	−($297,000)
80%	4.0%	98%	1.225	800,000	9,800	790,200	−($379,000)
90%	2.0%	100%	1.111	900,000	10,000	890,000	−($470,000)
100%	0.0%	100%	1.000	1,000,000	10,000	990,000	−($570,000)

Coste Campaña

20.000 --> 20.000

100.000 x 1 --> 100.000

Total: 120.000

Benef. Campaña

3.000 x 45 --> 135.000

Benef. Netos:

15.000

Coste Campaña

20.000 --> 20.000

200.000 x 1 --> 100.000

Total: 220.000

Benef. Campaña

5.000 x 45 --> 225.000

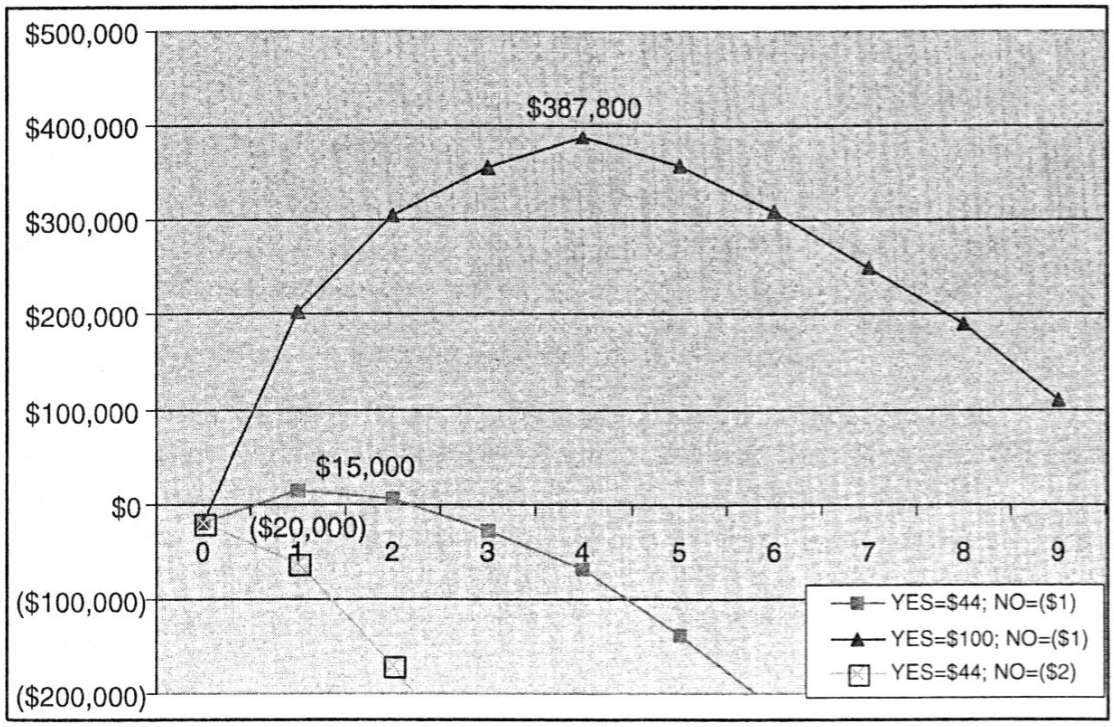
Benef. Netos:

5.000

Aprendizaje Supervisado. Matrices de Coste y Confusión.

Errores de Clasificación y Mailings. Ejemplo (cont.):

Gráfica mostrando el beneficio para tres campañas diferentes:



Aprendizaje Supervisado. Matrices de Coste y Confusión.

Errores de Clasificación y Mailings:

- En este tipo de problemas (si son binarios o ordenados) es preferible hacer hipótesis con clases continuas (estimaciones), porque permiten combinar con los costes de una manera más detallada.

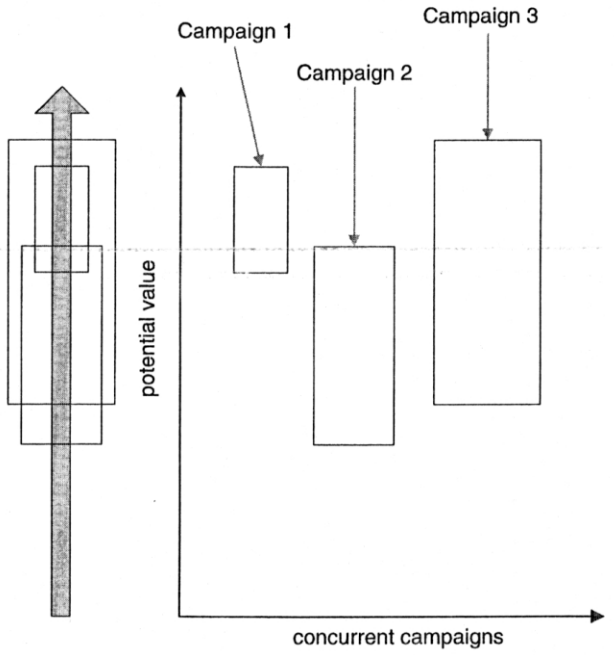
- P.ej. es preferible un modelo que determine en una escala de 0 a 10 lo bueno que es un cliente, que un modelo que determine si un cliente es malo o bueno.

Aprendizaje Supervisado. Mailings.

Secuenciación de Mailings:

No sobrecargar los clientes con demasiados mensajes de márketing… O bien acabarán ignorándolos o bien se cambiarán de compañía.

El mismo pequeño grupo de gente se elige una y otra vez y otros no se eligen nunca.

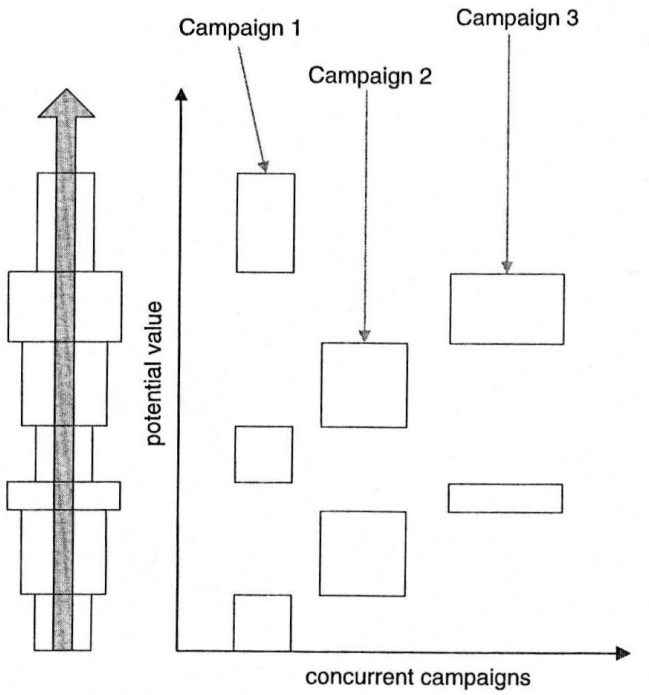


Aprendizaje Supervisado. Mailings.

Secuenciación de Mailings:

- Hay que intentar abarcar mejor los clientes:

Ahora todos los clientes participan en una campaña.



Métodos Predictivos
Combinación de Hipótesis

Combinación de Hipótesis:

- *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:
  - WEIGHTING MAJORITY: el valor se obtiene haciendo la media (caso continuo) o la mediana (caso discreto).
  - 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).

Métodos Predictivos
Combinación 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*).

Métodos Predictivos
Combinación 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:

$$P(false \mid D) = \sum_{h_i \in H} P(false \mid h_i) P(h_i \mid D) = 0.6$$
$$P(true \mid D) = \sum_{h_i \in H} P(true \mid h_i) P(h_i \mid D) = 0.4$$

Métodos Predictivos
Combinación 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.

Métodos Predictivos

Combinación 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.

- *RANDOMISING*: 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, etc.). Luego se *combinan* las distintas hipótesis.

Métodos Predictivos
Combinación 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.