INTELIGENCIA DE NEGOCIO

2018 - 2019

- Tema 1. Introducción a la Inteligencia de Negocio
- Tema 2. Minería de Datos. Ciencia de Datos
- **Tema 3. Modelos de Predicción: Clasificación, regresión y series temporales**
- Tema 4. Preparación de Datos
- Tema 5. Modelos de Agrupamiento o Segmentación
- Tema 6. Modelos de Asociación
- Tema 7. Modelos Avanzados de Minería de Datos.
- Tema 8. Big Data

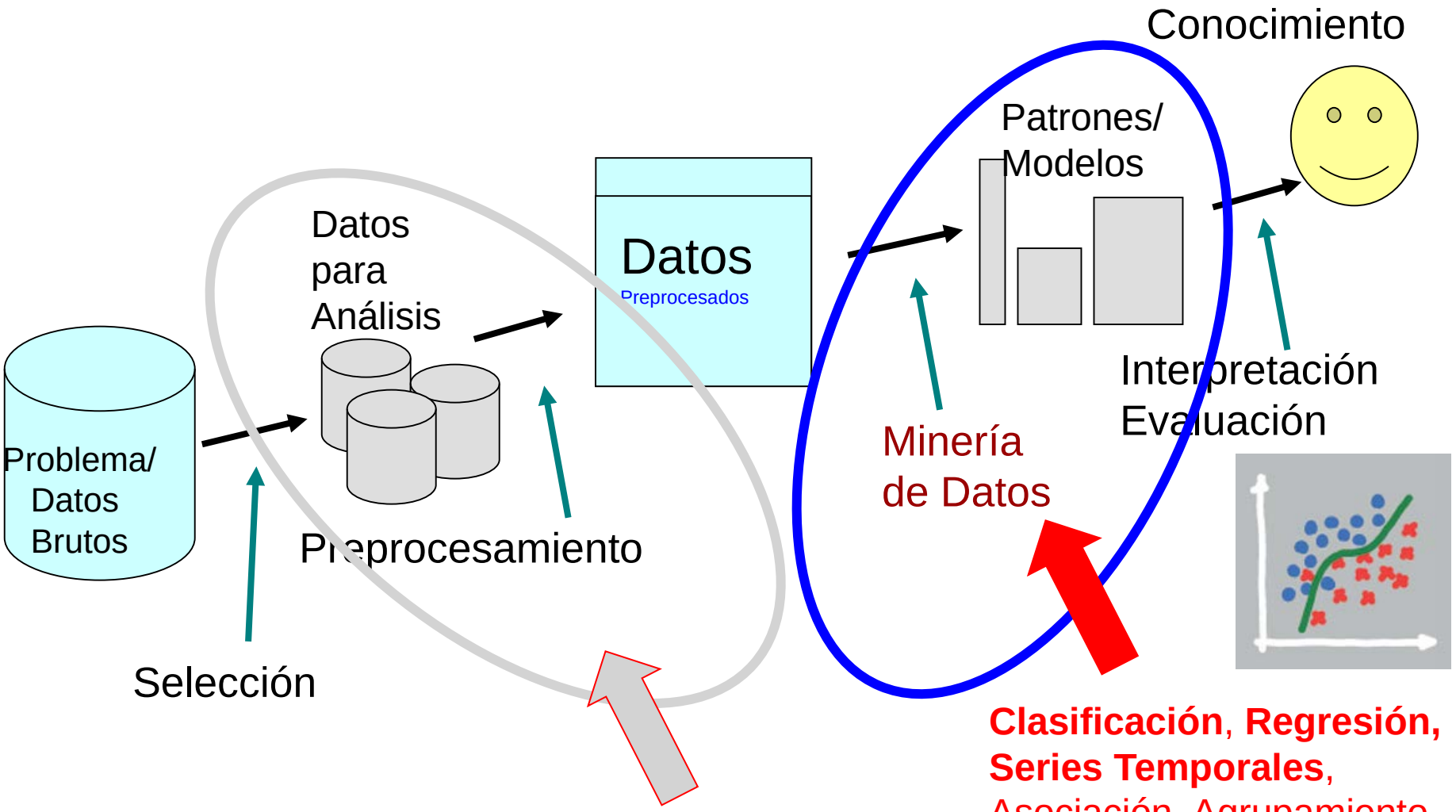
Inteligencia de Negocio

TEMA 3. Modelos de Predicción: Clasificación, regresión y series temporales

1. Clasificación
2. Regresión
3. Series Temporales

Motivación

Predicción es uno de los problemas más estudiados en minería de datos y tiene una gran presencia en problemas reales: problemas de clasificación y regresión



Objetivos. Parte I. Clasificación

- Entender qué es un problema de clasificación y un sistema de clasificación
- Conocer las fases de cualquier proceso de clasificación
- Conocer los criterios que se utilizan para evaluar un clasificador
- Entender el modo de funcionamiento de algunos clasificadores sencillos
- Conocer distintas metodologías para evaluar un sistema de clasificación
- Conocer algoritmos concretos de clasificación

Inteligencia de Negocio

TEMA 3. Modelos de Predicción: Clasificación, regresión y series temporales

Clasificación

1. **El problema de clasificación**
2. Clasificación con árboles y reglas
3. Clasificación con otras técnicas
4. Multiclasificadores

Bibliografía:

G. Shmueli, N.R. Patel, P.C. Bruce
Data mining for business intelligence (Part IV)
Wiley 2010 (2nd. edition)

V. Cherkassky, F.M. Mulier
Learning from Data: Concepts, Theory, and Methods (Sections 8 and 9)
2nd Edition, Wiley-IEE Prees, 2007

M. Zaki and W. Meira Jr.
Data Mining and Analysis: Fundamental Concepts and Algorithms (Part 4, Cap. 18 - 22)
Cambridge University Press, 2014.
http://www.dataminingbook.info/pmwiki.php

El problema de la clasificación

1. **Definición del problema de clasificación**
2. Etapas en el proceso de clasificación
3. Criterios para evaluar un clasificador
4. Algunos ejemplos de clasificadores sencillos
5. Evaluación de clasificadores

1. Definición del problema de clasificación. Ejemplo

Ejemplo: El problema de clasificación de la flor de *Iris*

- Problema simple muy conocido: *clasificación de lirios.*
- Tres clases de lirios: *setosa*, *versicolor* y *virginica*.
- Cuatro atributos: *longitud* y *anchura* de *pétalo* y *sépalo*, respectivamente.
- 150 ejemplos, 50 de cada clase.
- Disponible en
  http://www.ics.uci.edu/~mlearn/MLRepository.html

	Sepal length X_1	Sepal width X_2	Petal length X_3	Petal width X_4	Class X_5
$\mathbf{x}_1$	5.9	3.0	4.2	1.5	Iris-versicolor
$\mathbf{x}_2$	6.9	3.1	4.9	1.5	Iris-versicolor
$\mathbf{x}_3$	6.6	2.9	4.6	1.3	Iris-versicolor
$\mathbf{x}_4$	4.6	3.2	1.4	0.2	Iris-setosa
$\mathbf{x}_5$	6.0	2.2	4.0	1.0	Iris-versicolor
$\mathbf{x}_6$	4.7	3.2	1.3	0.2	Iris-setosa
$\mathbf{x}_7$	6.5	3.0	5.8	2.2	Iris-virginica
$\mathbf{x}_8$	5.8	2.7	5.1	1.9	Iris-virginica
⋮	⋮	⋮	⋮	⋮	⋮
$\mathbf{x}_{149}$	7.7	3.8	6.7	2.2	Iris-virginica
$\mathbf{x}_{150}$	5.1	3.4	1.5	0.2	Iris-setosa

setosa

versicolor

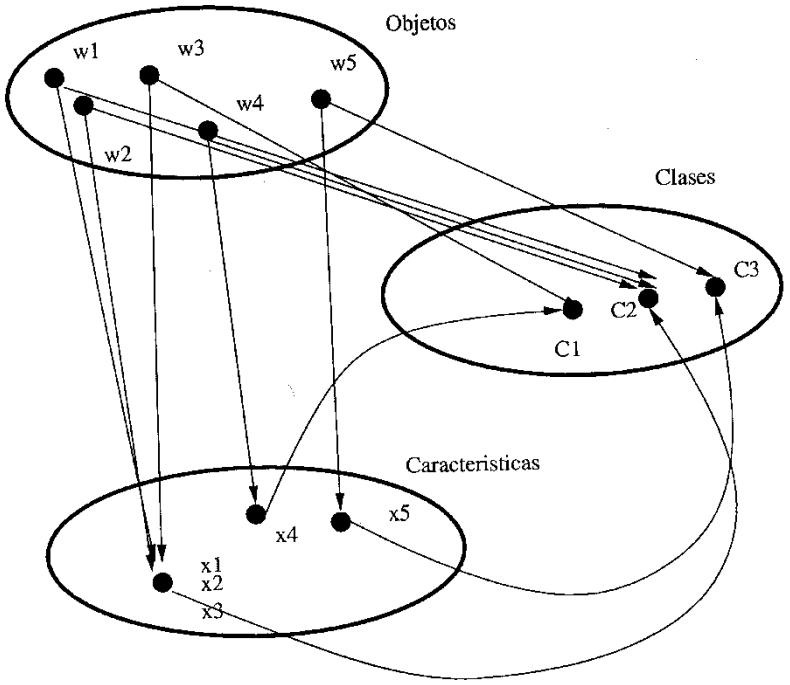
virginica

1. Definición del problema de clasificación

- Es un tipo de *aprendizaje supervisado*: Se conoce la clase verdadera de cada uno de los ejemplos que se utilizan para construir el clasificador
- El *problema de clasificación* consiste en predecir una determinada clase (categórica) para un objeto

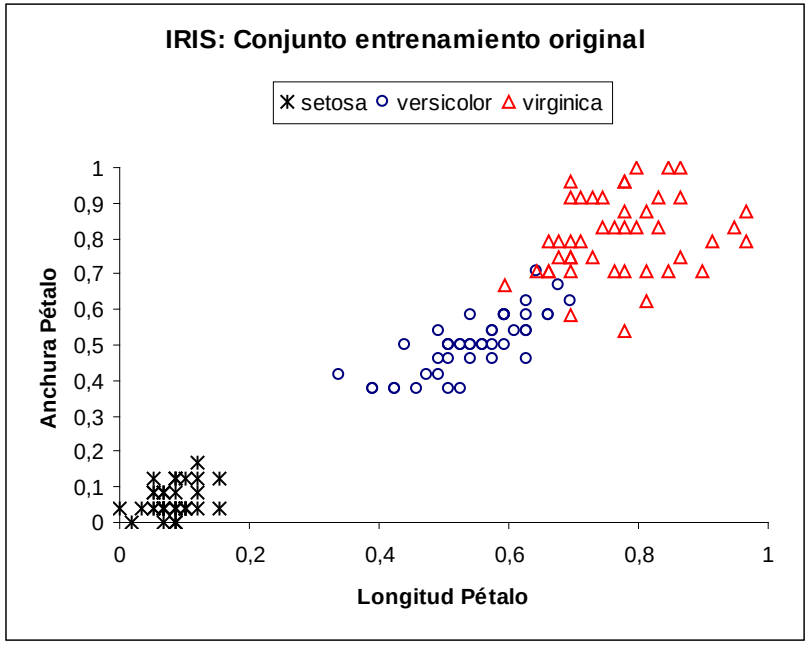
- La *tarea de clasificación:* Dados un conjunto de ejemplos ya clasificados, construir un modelo o clasificador que permita clasificar nuevos casos

El problema fundamental de la clasificación está directamente relacionado con la separabilidad de las clases.



1. Definición del problema de clasificación. Ejemplo

Ejemplos de conjuntos seleccionados sobre *Iris:*

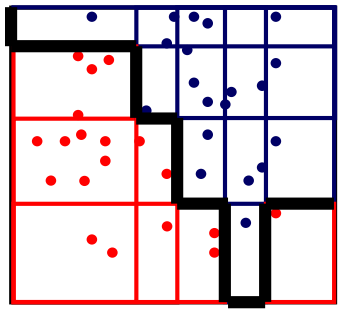


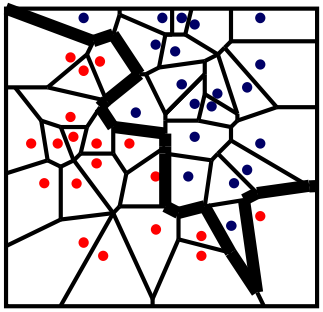
El problema fundamental de la clasificación está directamente relacionado con la separabilidad de las clases.

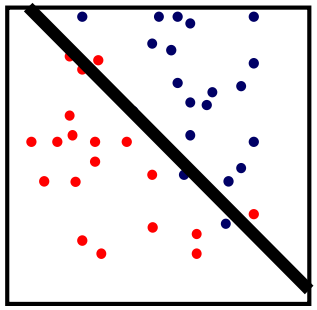
En este problema la dificultad está en la separación entre las clases versicolor y virgínica

1. Definición del problema de clasificación

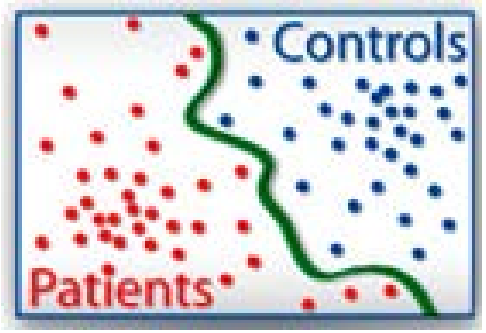
- Un *clasificador* puede ser un conjunto de reglas, un árbol de decisión, una red neuronal, etc.







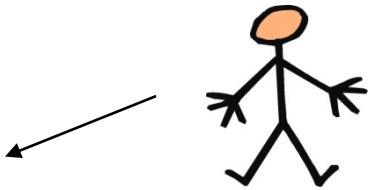
- Aplicaciones típicas:
  - Aprobación de créditos, marketing directo, detección de fraudes, diagnóstico médico…



1. Definición del problema de clasificación

- Un objeto se describe a través de un conjunto de características (variables o atributos)

$$X \rightarrow \{X_1, X_2, \ldots, X_n\} \qquad e_i \rightarrow \{x_1, x_2, \ldots, x_n, c_k\}$$



- Edad
- Astigmatismo
- Ratio de lagrimeo
- Miopía

CLASIFICACIÓN: Tipo de lentillas

-Ingresos
-Deudas
-Propiedades
…
-CLASIFICACIÓN: Conceder el crédito

- Objetivo de la tarea de clasificación: Clasificar el objetivo dentro de una de las categorías de la clase $C = \{c_1, \ldots, c_k\}$

$$f: X_1 \times X_2 \times \ldots \times X_n \rightarrow C$$

- Las características o variables elegidas dependen del problema de clasificación

Kaggle: The Home of Data Science

Digit Recognizer

Classify handwritten digits using the famous MNIST data

3 months to go · Getting Started

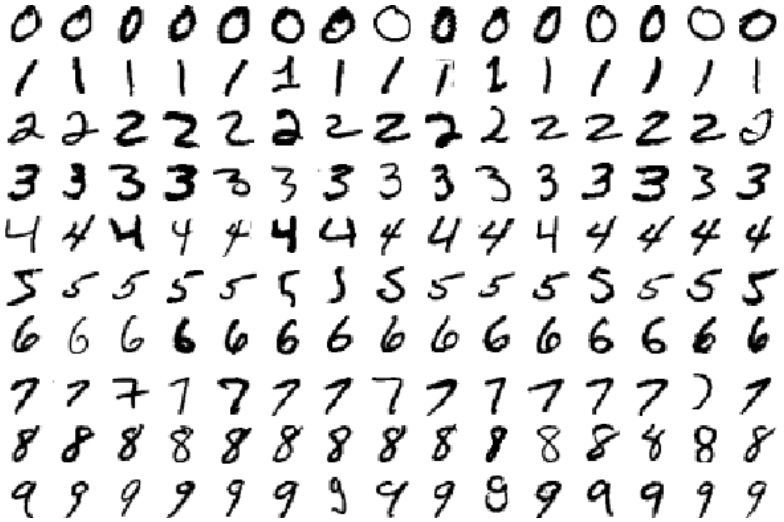
1,116 teams
6,153 kernels
Knowledge

Titanic: Machine Learning from Disaster

Predict survival on the Titanic using Excel, Python, R & Random Forests

3 months to go · Getting Started

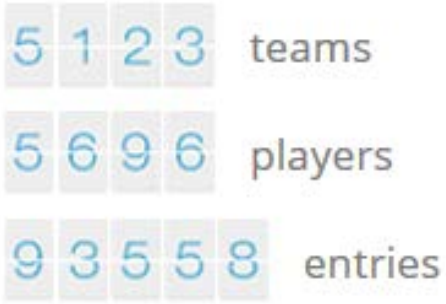
5,095 teams
9,571 kernels
Knowledge



MNIST data

Kaggle: The Home of Data Science

Santander Customer Satisfaction



Banco Santander

- Objetivo: crear un modelo que **prediga** qué clientes no están satisfechos
- Premio **de 60.000€** entre las 3 mejores soluciones
- Competición activa desde el **2 de marzo hasta el 2 de mayo, 2016**
- Conjunto de datos: 76020 instancias y 371 variables
- Datos públicos 50%

Kaggle: The Home of Data Science

otto group

$10,000 • 1,987 teams

Otto Group Product Classification Challenge

Enter/Merge by

Tue 17 Mar 2015 — Mon 18 May 2015 (39 days to go)

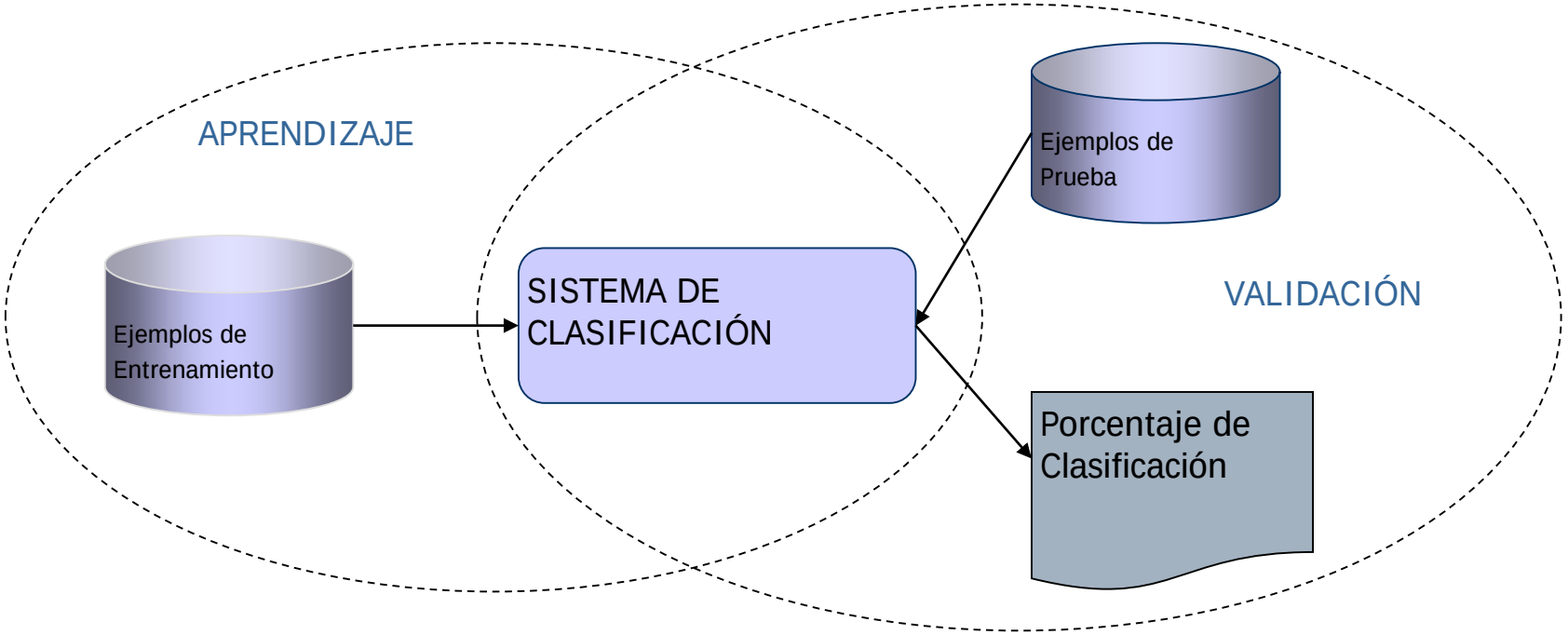
A KAGGLE competition with a Multiclass and imbalanced problem

For this competition, we have provided a dataset with 93 features for more than 200,000 products. The objective is to build a predictive model which is able to distinguish between our main product categories. The winning models will be open sourced.

El problema de la clasificación

1. Definición del problema de clasificación
2. **Etapas en el proceso de clasificación**
3. Criterios para evaluar un clasificador
4. Algunos ejemplos de clasificadores sencillos
5. Evaluación de clasificadores

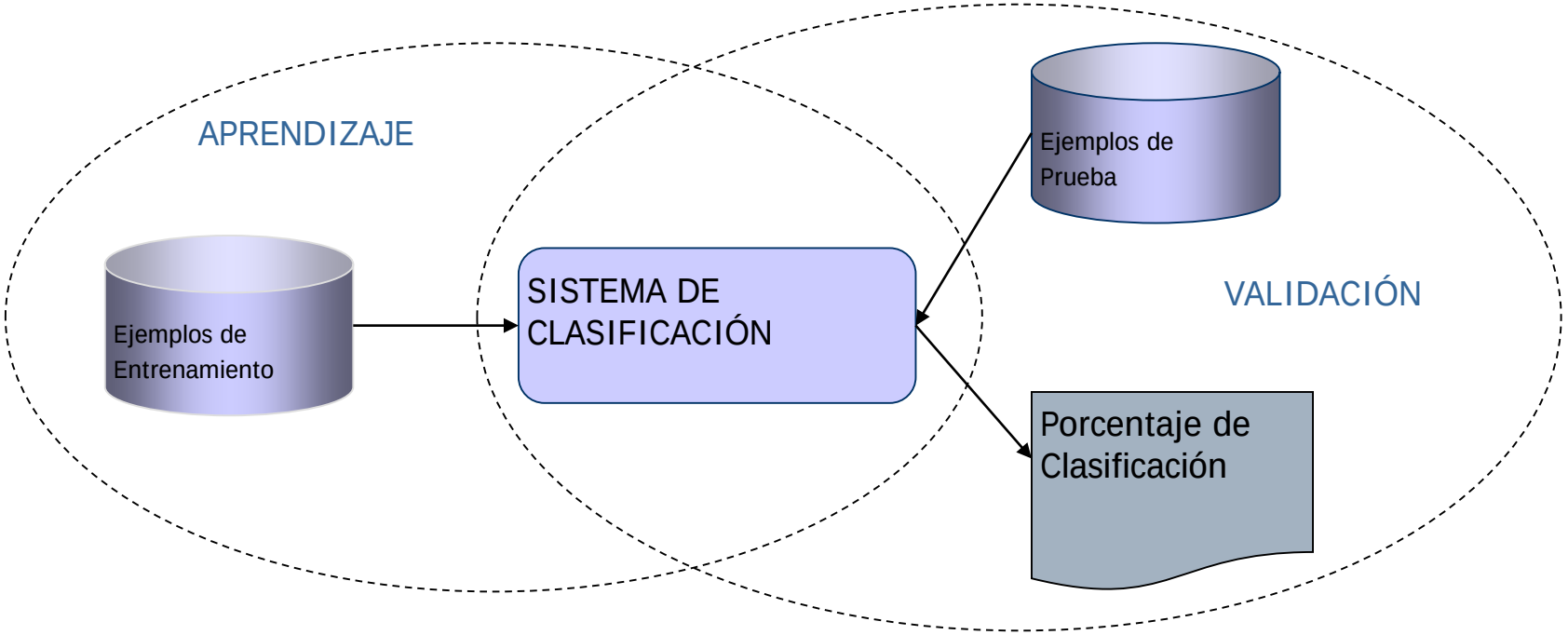
2. Etapas en el proceso de clasificación



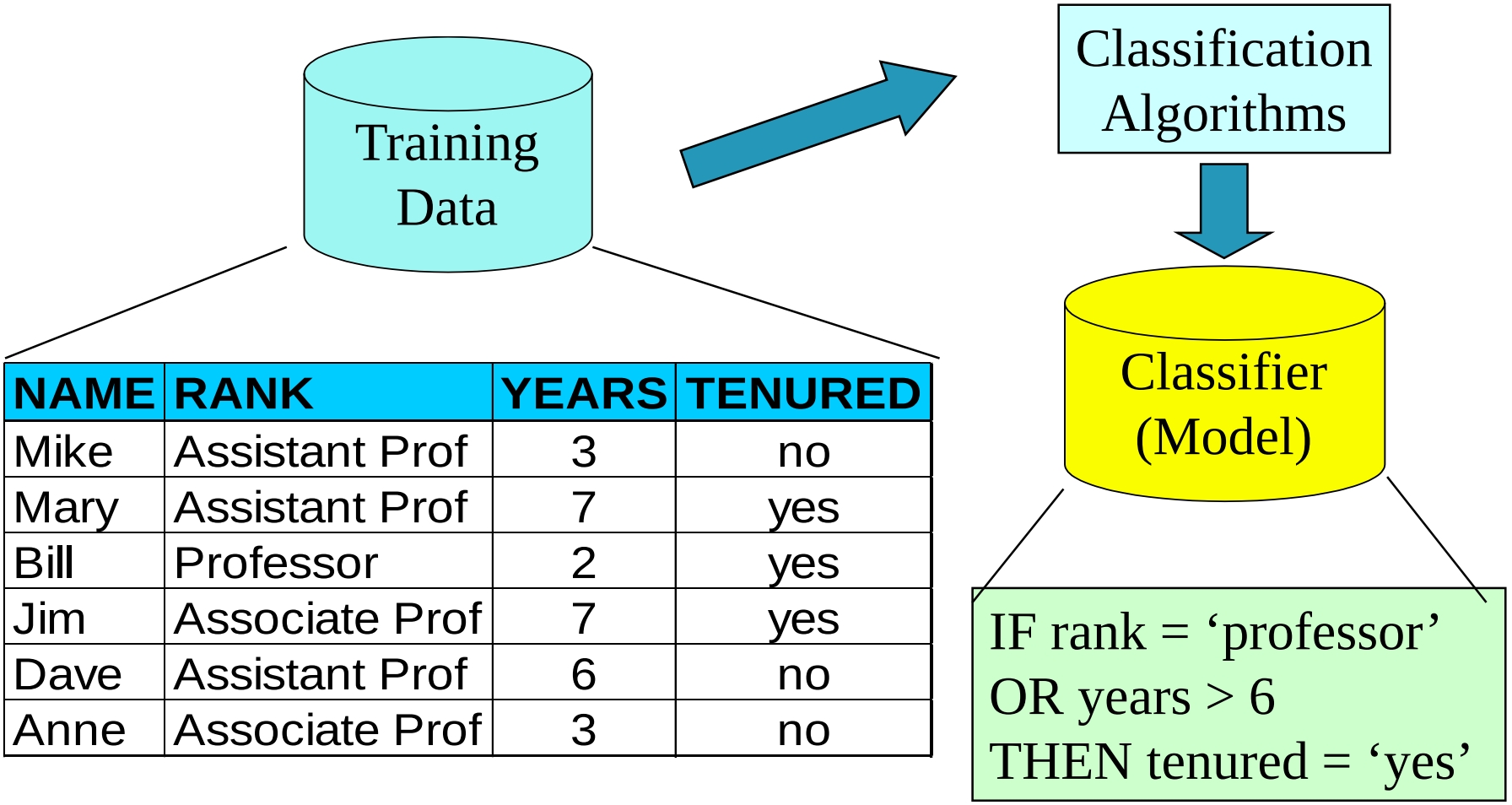
2. Etapas en el proceso de clasificación

1. Construcción del modelo que describe el conjunto de clases (predeterminadas): Fase de Aprendizaje
   - Cada ejemplo (tupla) se sabe que pertenece a una clase (etiqueta del atributo clase)
   - Se utiliza un conjunto de ejemplos para la construcción del modelo: conjunto de entrenamiento (*training set*)
   - El modelo obtenido se representa como un conjunto de reglas de clasificación, árboles de decisión, fórmula matemática, …
2. Utilización del modelo: Validación
   - Estimación de la precisión del modelo
     - Se utiliza un conjunto de ejemplos distintos de los utilizados para la construcción del modelo: conjunto de prueba (*test set*)
       - Si el conjunto de test no fuese independiente del de entrenamiento ocurría un proceso de sobreajuste (*overfitting*)
     - Para cada ejemplo de test se compara la clase determinada por el modelo con la clase real (conocida)
     - El ratio de precisión es el porcentaje de ejemplos de test que el modelo clasifica correctamente

2. Etapas en el proceso de clasificación

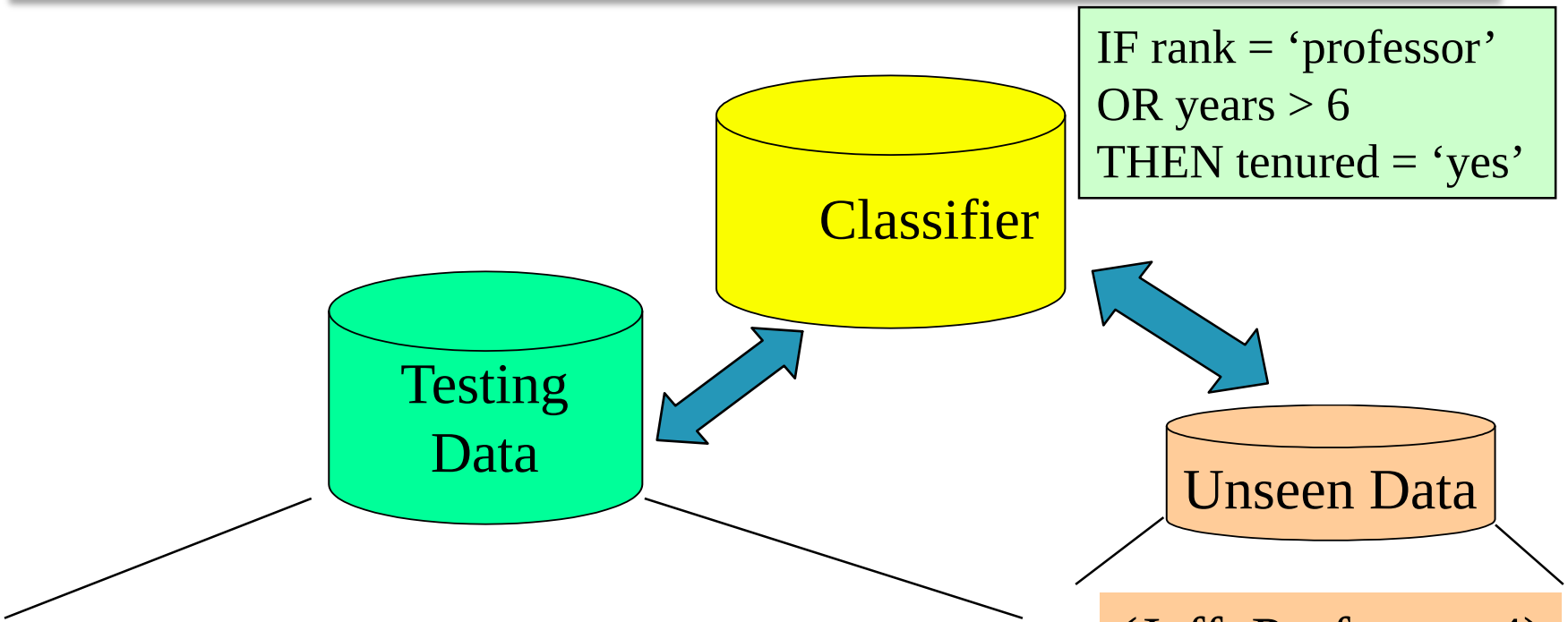


2. Etapas en el proceso de clasificación



NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

2. Etapas en el proceso de clasificación



NAME	RANK	YEARS	TENURED
Tom	Assistant Prof	2	no
Merlisa	Associate Prof	7	no
George	Professor	5	yes
Joseph	Assistant Prof	7	yes

(Jeff, Professor, 4)

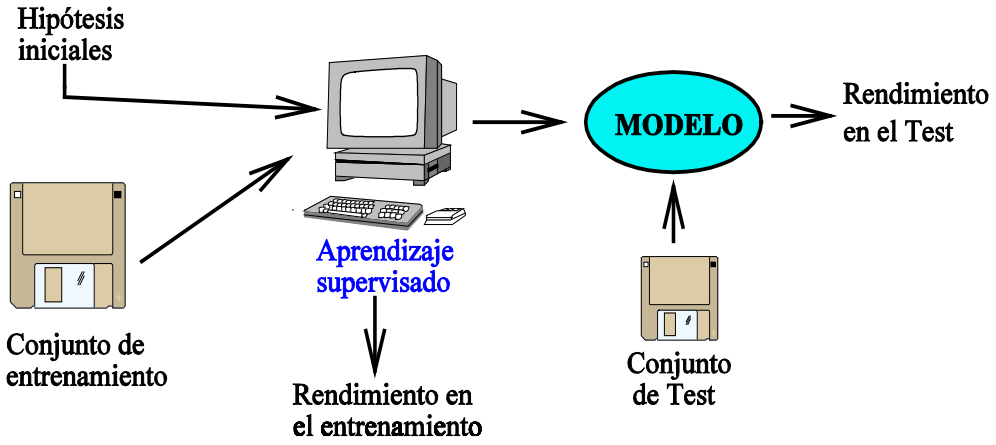
Tenured?

Yes

El problema de la clasificación

1. Definición del problema de clasificación
2. Etapas en el proceso de clasificación
3. **Criterios para evaluar un clasificador**
4. Algunos ejemplos de clasificadores sencillos
5. Evaluación de clasificadores

3. Criterios para evaluar un clasificador



- Rendimiento (matriz de confusión):

	Clasificación como	
	Si	No
Clase SI	Verdadero positivo (VP)	Falso negativo (FN)
real NO	Falso Positivo (FP)	Verdadero Negativo (VN)

N=VP+VN+FP+FN

- Tasa de acierto $s = \frac{VP + VN}{N}$
- Tasa de error $\varepsilon = 1 - s$

3. Criterios para evaluar un clasificador

- Precisión o exactitud (s, ε)
  - En ocasiones se debe considerar el costo de la clasificación incorrecta
- Velocidad
  - Tiempo necesario para la construcción del modelo
  - Tiempo necesario para usar el modelo
- Robustez: capacidad para tratar con valores desconocidos
- Escalabilidad: Aumento del tiempo necesario (en construcción y evaluación) con el tamaño de la BD
- Interpretabilidad: comprensibilidad del modelo obtenido
- Complejidad del modelo: Tamaño del árbol de clasificación, número de reglas, antecedentes en las reglas,...

3. Criterios para evaluar un clasificador

- Matriz de confusión

Dado un problema de clasificación con *m* clases, una matriz de confusión es una matriz *m* x *m* en la que una entrada $c_{i,j}$ indica el número de ejemplos que se han asignado a la clase $c_{j,}$ cuando la clase correcta es c_i

Ejemplo: Para la BD Height, si suponemos que output1 es la clasificación correcta y output2 es la que hace el clasificador, la matriz de confusión es

3. Criterios para evaluar un clasificador

Name	Gender	Height	Output1	Output2
Kristina	F	1.6m	Short	Medium
Jim	M	2m	Tall	Medium
Maggie	F	1.9m	Medium	Tall
Martha	F	1.88m	Medium	Tall
Stephanie	F	1.7m	Short	Medium
Bob	M	1.85m	Medium	Medium
Kathy	F	1.6m	Short	Medium
Dave	M	1.7m	Short	Medium
Worth	M	2.2m	Tall	Tall
Steven	M	2.1m	Tall	Tall
Debbie	F	1.8m	Medium	Medium
Todd	M	1.95m	Medium	Medium
Kim	F	1.9m	Medium	Tall
Amy	F	1.8m	Medium	Medium
Wynette	F	1.75m	Medium	Medium

	Asignación		
	Short	Medium	Tall
Short	0	4	0
Medium	0	5	3
Tall	0	1	2

Matriz de confusión

El problema de la clasificación

1. Definición del problema de clasificación
2. Etapas en el proceso de clasificación
3. Criterios para evaluar un clasificador
4. **Algunos ejemplos de clasificadores sencillos**
5. Evaluación de clasificadores

4. Algunos ejemplos de clasificadores sencillos

Edad	Miopía	Astigmatismo	Lagrimeo	Lentes
Joven	Miope	No	Reducido	Ninguna
Joven	Miope	No	Normal	Blandas
Joven	Miope	Si	Reducido	Ninguna
Joven	Miope	Si	Normal	Duras
Joven	Hipermétrope	No	Reducido	Ninguna
Joven	Hipermétrope	No	Normal	Blandas
Joven	Hipermétrope	Si	Reducido	Ninguna
Joven	Hipermétrope	Si	Normal	Duras
Pre-presbiopico	Miope	No	Reducido	Ninguna
Pre-presbiopico	Miope	No	Normal	Blandas
Pre-presbiopico	Miope	Si	Reducido	Ninguna
Pre-presbiopico	Miope	Si	Normal	Duras
Pre-presbiopico	Hipermétrope	No	Reducido	Ninguna
Pre-presbiopico	Hipermétrope	No	Normal	Blandas
Pre-presbiopico	Hipermétrope	Si	Reducido	Ninguna
Pre-presbiopico	Hipermétrope	Si	Normal	Ninguna
Presbiopico	Miope	No	Reducido	Ninguna
Presbiopico	Miope	No	Normal	Ninguna
Presbiopico	Miope	Si	Reducido	Ninguna
Presbiopico	Miope	Si	Normal	Duras
Presbiopico	Hipermétrope	No	Reducido	Ninguna
Presbiopico	Hipermétrope	No	Normal	Blandas
Presbiopico	Hipermétrope	Si	Reducido	Ninguna
Presbiopico	Hipermétrope	Si	Normal	Ninguna

4. Algunos ejemplos de clasificadores sencillos

Clasificador ZeroR

- Es el clasificador más sencillo
- Todas las instancias se clasifican como pertenecientes a la clase mayoritaria
- Se usa como caso base para realizar comparaciones (cualquier algoritmo debería al menos igualar su rendimiento)
- Ejemplo: Con la BD Lentes de contacto
  - Lentes=ninguna 15/24
  - Lentes=blandas 5/24
  - Lentes=duras 4/24

  Por tanto la regla sería: lentes=ninguna

  s = 0.625

  ε = 0.375

4. Algunos ejemplos de clasificadores sencillos

Clasificador OneR

- Objetivo: crear un clasificador formado por reglas con una única variable en el antecedente
- Se generan todas las reglas del tipo

  Si variable=valor entonces Clase = categoría
- También se utiliza como algoritmo para realizar comparaciones

```
For each attribute,
    For each value of the attribute, make a rule as follows:
        count how often each class appears
        find the most frequent class
        make the rule assign that class to this attribute-value
    Calculate the error rate of the rules
Choose the rules with the smallest error rate
```

Nota: Los valores perdidos se utilizan como un valor especial de las variables

4. Algunos ejemplos de clasificadores sencillos

Ejemplo: Con la BD lentes de contacto

	Regla		ninguna	blandas	Duras
Si edad=joven	entonces	Lentes=ninguna	4	2	2
Si edad=pre-presbiopico	entonces	Lentes=ninguna	5	2	1
Si edad=presbiopico	entonces	Lentes=ninguna	6	1	1
Si miopia=miope	entonces	Lentes=ninguna	7	2	3
Si miopía=hipermétrope	entonces	Lentes=ninguna	8	3	1
Si astigmatismo=no	entonces	Lentes=ninguna	7	5	0
Si astigmatismo=si	entonces	Lentes=ninguna	8	0	4
Si lagrimeo=reducido	entonces	Lentes=ninguna	12	0	0
Si lagrimeo=normal	entonces	Lentes=blandas	3	5	4

4. Algunos ejemplos de clasificadores sencillos

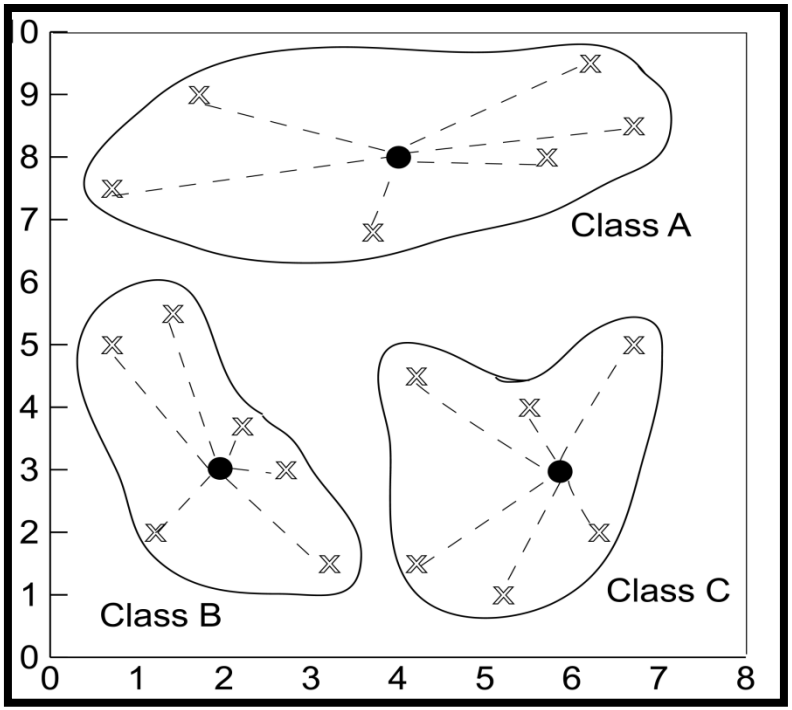
Clasificador Del Vecino Más Cercano (k-NN)

- k-NN (*k vecinos más cercanos*) es uno de los clasificadores más utilizados por su simplicidad.
- El proceso de aprendizaje de este clasificador consiste en almacenar una tabla con los ejemplos disponibles, junto a la clase asociada a cada uno de ellos.
- Ante un nuevo ejemplo a clasificar, se calcula su distancia (usaremos la Euclídea) con respecto a los *n* ejemplos existentes en la tabla, y se consideran los *k* más cercanos.
- El nuevo ejemplo se clasifica según la clase mayoritaria de los *k* ejemplos más cercanos.
- El caso más sencillo es cuando *k = 1 (1-NN)*.

4. Algunos ejemplos de clasificadores sencillos

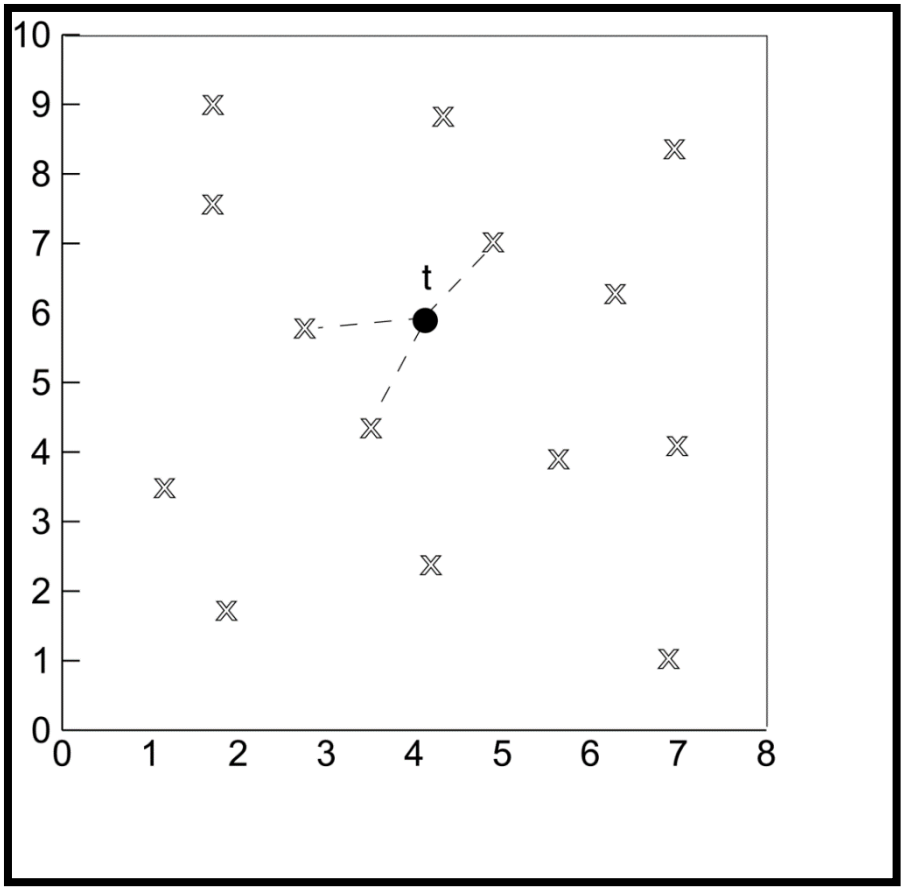
Clasificador Del Vecino Más Cercano (k-NN): Basado en distancias

Basado en Distancias



4. Algunos ejemplos de clasificadores sencillos

Clasificador Del Vecino Más Cercano (k-NN): Ejemplo para k=3



4. Algunos ejemplos de clasificadores sencillos

Clasificador Del Vecino Más Cercano (k-NN)

Dado el siguiente conjunto con 4 instancias, 3 atributos y 2 clases:

x_1: 0.4 0.8 0.2 positiva

x_2: 0.2 0.7 0.9 positiva

x_3: 0.9 0.8 0.9 negativa

x_4: 0.8 0.1 0.0 negativa

Calculamos la distancia del ejemplo con todos los del conjunto:

$$d(x_{1,}y_1) = \sqrt{(0.4-0.7)^2 + (0.8-0.2)^2 + (0.2-0.1)^2} = 0.678$$

$$d(x_{2,}y_1) = \sqrt{(0.2-0.7)^2 + (0.7-0.2)^2 + (0.9-0.1)^2} = 1.068$$

$$d(x_{3,}y_1) = \sqrt{(0.9-0.7)^2 + (0.8-0.2)^2 + (0.9-0.1)^2} = 1.020$$

$$d(x_{4,}y_1) = \sqrt{(0.8-0.7)^2 + (0.1-0.2)^2 + (0.0-0.1)^2} = 0.173$$

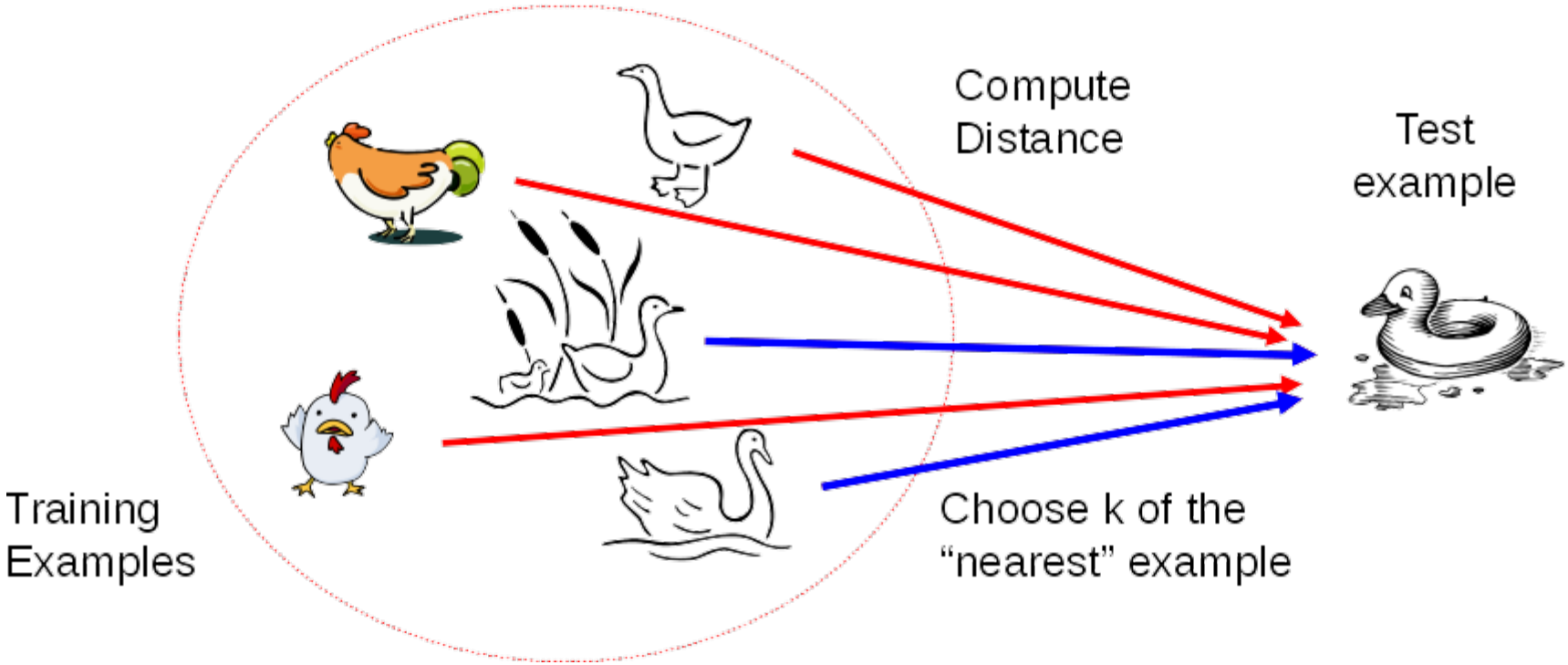
Por tanto, el ejemplo se clasificará con respecto a la clase *negativa.*

IMPORTANTE: Los atributos deben estar normalizados [0,1] para no priorizarlos sobre otros.

4. Algunos ejemplos de clasificadores sencillos

Clasificador del Vecino Más Cercano (k-NN)

- If it walks like a duck, quacks like a duck,
  then it's probably a duck



4. Algunos ejemplos de clasificadores sencillos

Clasificador Del Vecino Más Cercano (k-NN) Problema en las fronteras

Ejemplo: Diseño de un Clasificador para *Iris*

- Problema simple muy conocido: *clasificación de lirios.*
- Tres clases de lirios: *setosa*, *versicolor* y *virginica*.
- Cuatro atributos: *longitud* y *anchura* de *pétalo* y *sépalo*, respectivamente.
- 150 ejemplos, 50 de cada clase.
- Disponible en http://www.ics.uci.edu/~mlearn/MLRepository.html



4. Algunos ejemplos de clasificadores sencillos

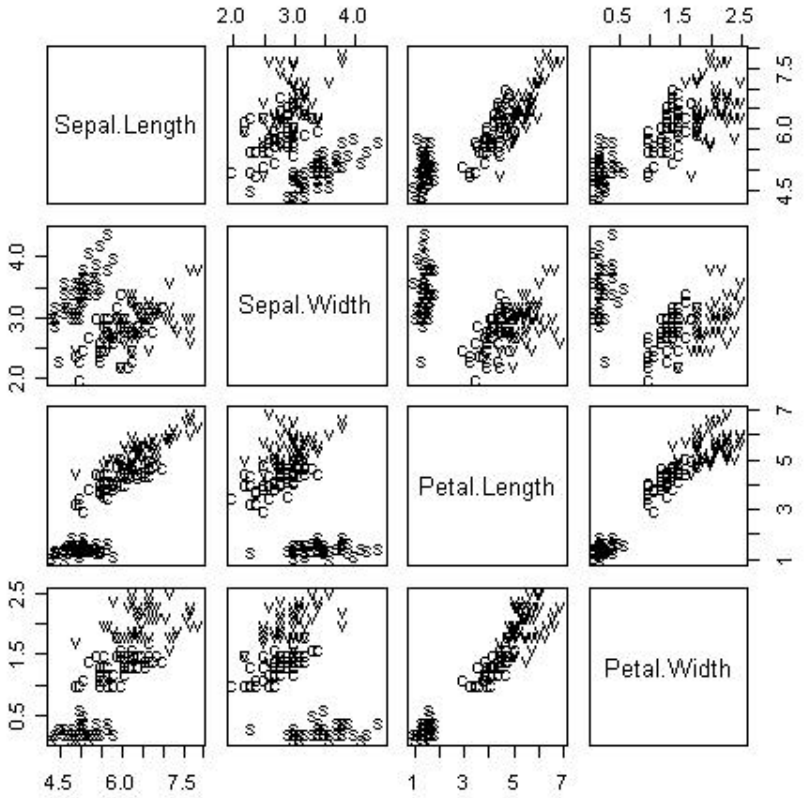
Clasificador Del Vecino Más Cercano (k-NN): Ejemplo IRIS. Problema en las fronteras

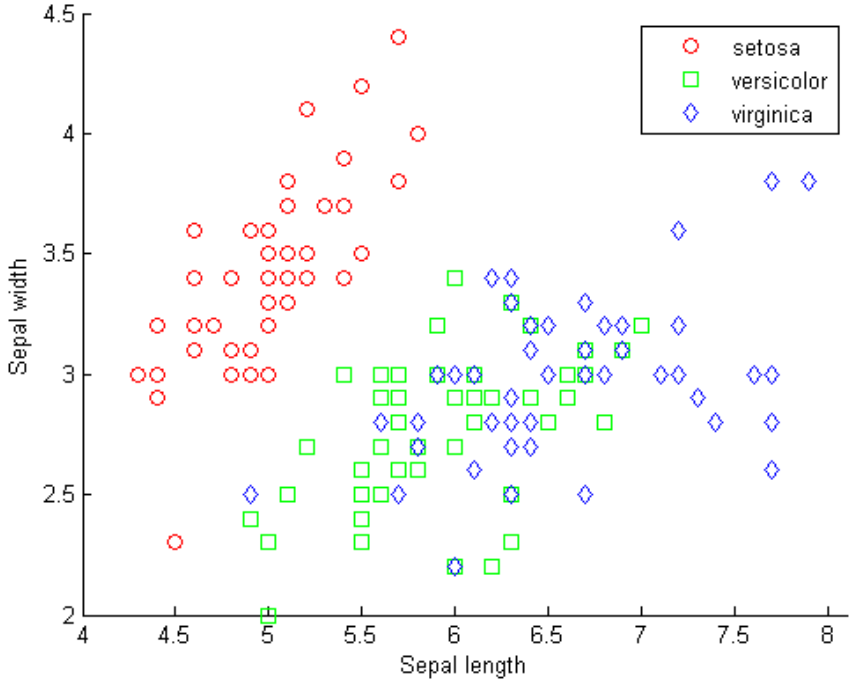
	Sepal length X_1	Sepal width X_2	Petal length X_3	Petal width X_4	Class X_5
$\mathbf{x}_1$	5.9	3.0	4.2	1.5	Iris-versicolor
$\mathbf{x}_2$	6.9	3.1	4.9	1.5	Iris-versicolor
$\mathbf{x}_3$	6.6	2.9	4.6	1.3	Iris-versicolor
$\mathbf{x}_4$	4.6	3.2	1.4	0.2	Iris-setosa
$\mathbf{x}_5$	6.0	2.2	4.0	1.0	Iris-versicolor
$\mathbf{x}_6$	4.7	3.2	1.3	0.2	Iris-setosa
$\mathbf{x}_7$	6.5	3.0	5.8	2.2	Iris-virginica
$\mathbf{x}_8$	5.8	2.7	5.1	1.9	Iris-virginica
⋮	⋮	⋮	⋮	⋮	⋮
$\mathbf{x}_{149}$	7.7	3.8	6.7	2.2	Iris-virginica
$\mathbf{x}_{150}$	5.1	3.4	1.5	0.2	Iris-setosa

4. Algunos ejemplos de clasificadores sencillos

Clasificador Del Vecino Más Cercano (k-NN). Problema en las fronteras

setosa, versicolor (C) y *virginica*

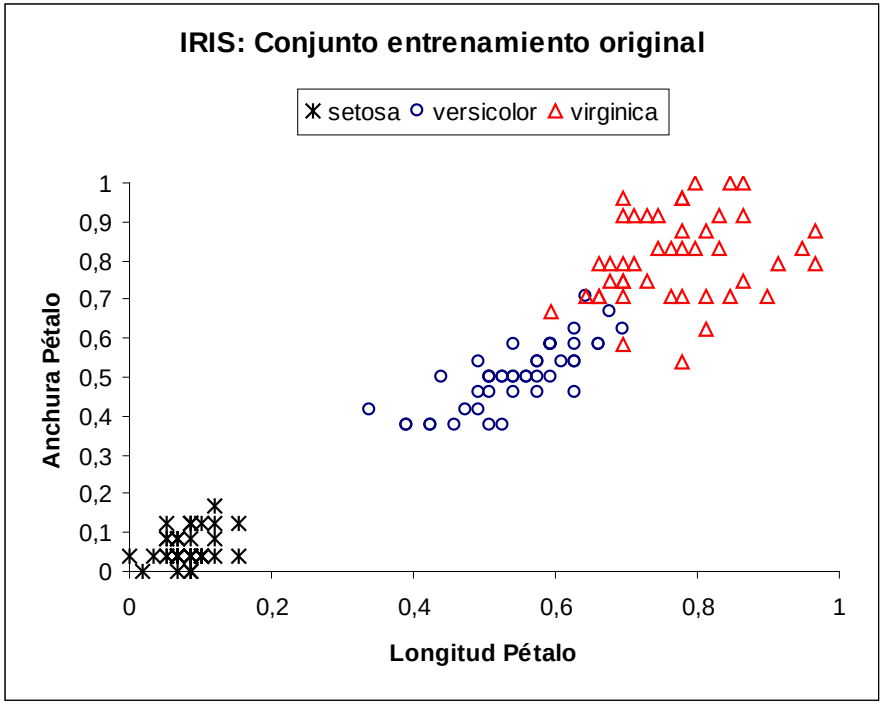


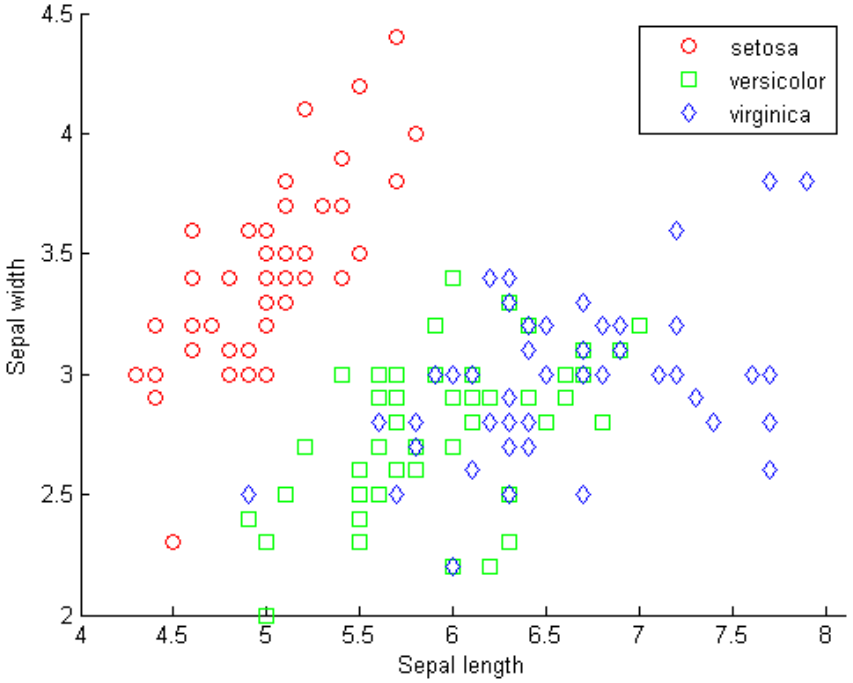


4. Algunos ejemplos de clasificadores sencillos

Clasificador Del Vecino Más Cercano (k-NN). Problema en las fronteras

setosa, versicolor (C) y *virginica*

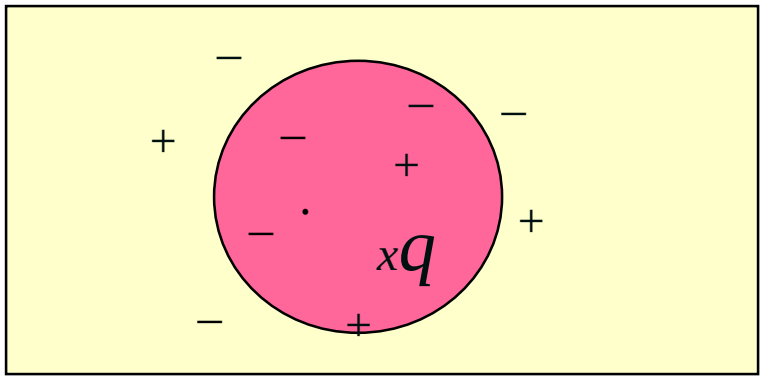


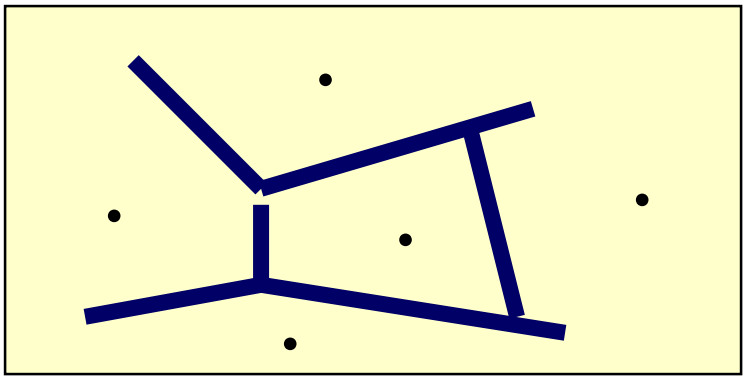


4. Algunos ejemplos de clasificadores sencillos

Clasificador Del Vecino Más Cercano (k-NN)

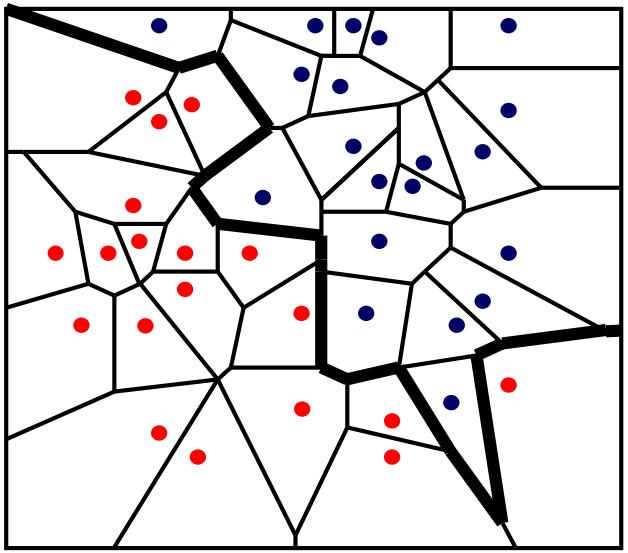
- *k*-NN devuelve la clase más repetida de entre todos los k ejemplos de entrenamiento cercanos a *xq*.
- Diagrama de Voronoi: superficie de decisión inducida por 1-NN para un conjunto dado de ejemplos de entrenamiento.

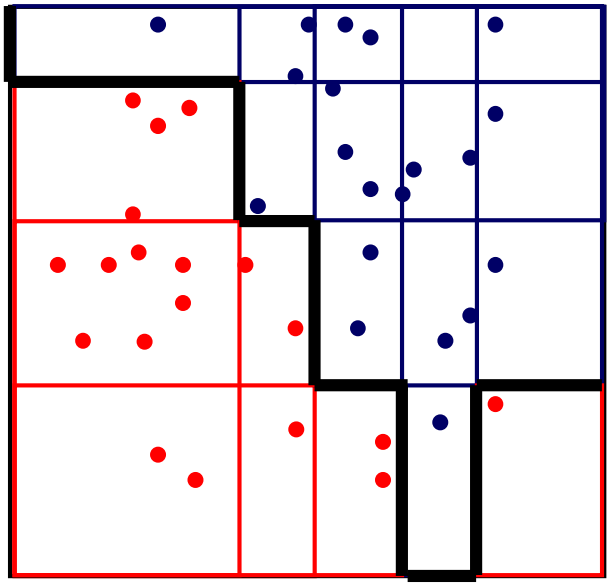




4. Algunos ejemplos de clasificadores sencillos

Clasificador Del Vecino Más Cercano (k-NN): Basado en distancias (muy diferente a los basados en particiones)





El problema de la clasificación

1. Definición del problema de clasificación
2. Etapas en el proceso de clasificación
3. Criterios para evaluar un clasificador
4. Algunos ejemplos de clasificadores sencillos
5. **Evaluación de clasificadores**

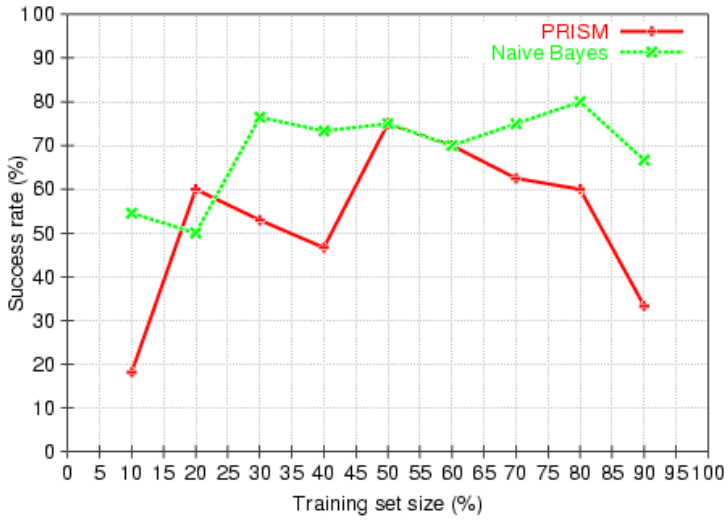
5. Evaluación de clasificadores

- El objetivo de los métodos de validación es realizar una estimación honesta de la bondad del clasificador construido
- Utilizar como bondad la tasa de acierto sobre el conjunto de entrenamiento no es realista
  - El porcentaje obtenido suele ser demasiado optimista debido a que el modelo estará sobreajustado a los datos utilizados durante el proceso de aprendizaje
- Existen distintas técnicas de validación de clasificadores, entre ellas
  - *Hold-out*
  - Validación cruzada
  - *Leave-one-out*
  - *Boostraping*

5. Evaluación de clasificadores

Hold-out

- Consiste en dividir la BD en dos conjuntos independientes: entrenamiento (CE) y test (CT)
- El tamaño del CE normalmente es mayor que el del CT (2/3, 1/3, 4/5, 1/5,...)
- Los elementos del CE suelen obtenerse mediante muestreo sin reemplazamiento de la BD inicial. El CT está formado por los elementos no incluidos en el CE



Test set (%) + Training set (%) = 100%

- Suele utilizarse en BBDD de tamaño grande

5. Evaluación de clasificadores

Validación cruzada

- Consiste en:
  1. Dividir la BD en *k* subconjuntos (*folds*), $\{S_1,\ldots,S_k\}$ de igual tamaño
  2. Aprender *k* clasificadores utilizando en cada uno de ellos un CE distinto. Validar con el CT correspondiente

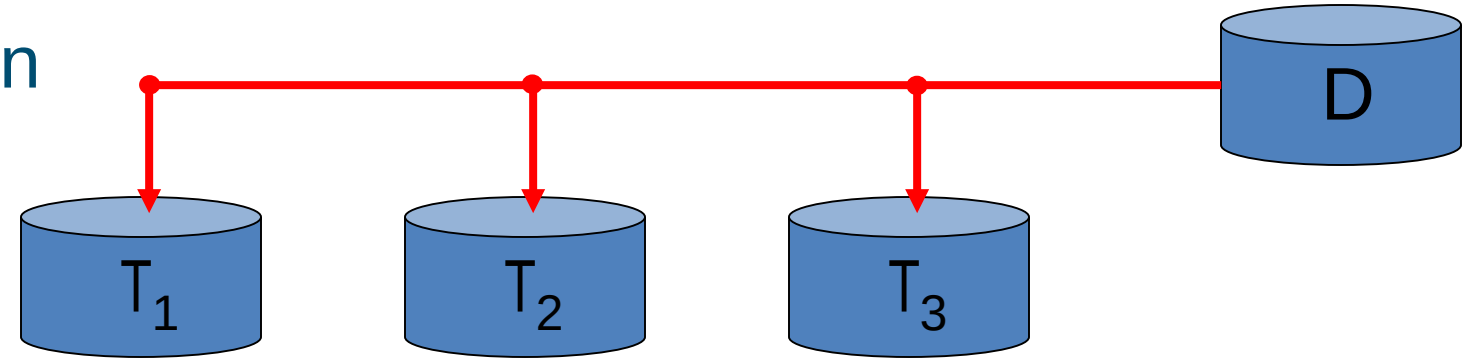
$$CE = S_1 \cup \ldots \cup S_{i-1} \cup S_{i+1} \cup \ldots S_k$$

$$CT = S_i$$

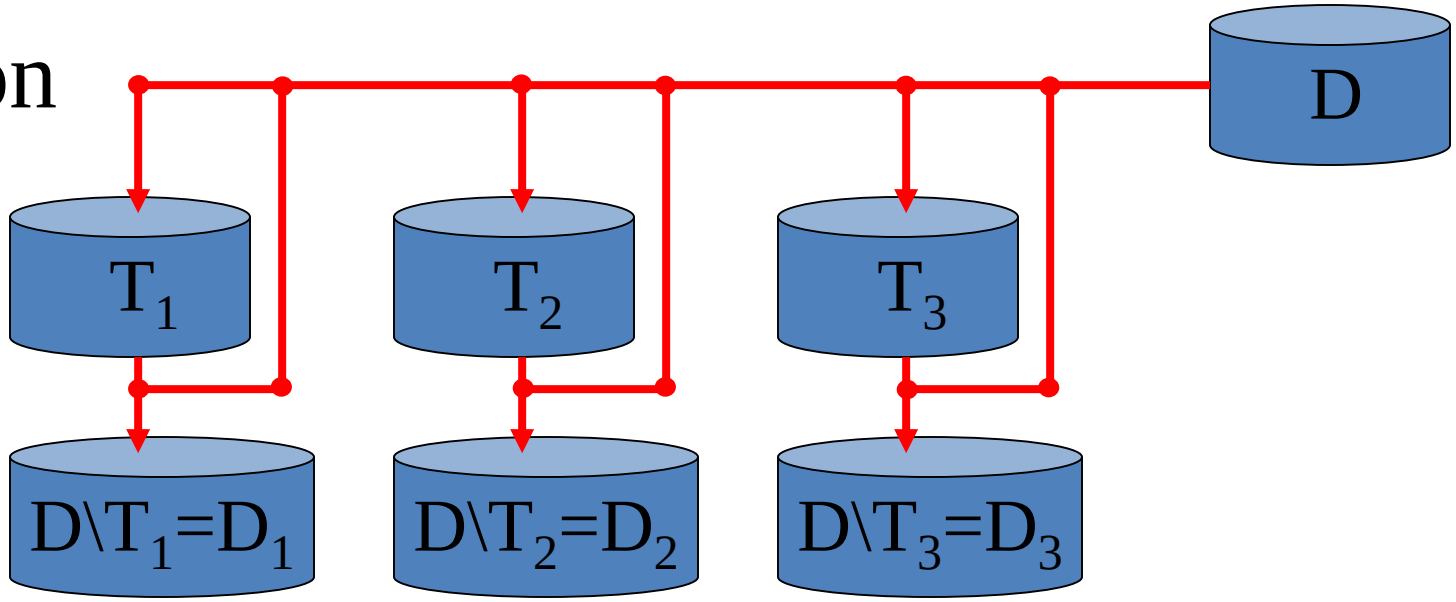
  3. Devolver como tasa de acierto (error) el promedio obtenido en las *k* iteraciones

- Validación cruzada estratificada: Los subconjuntos se estratifican en función de la variable clase
- Valores típicos de k=5,10
- Suele utilizarse en BBDD de tamaño moderado

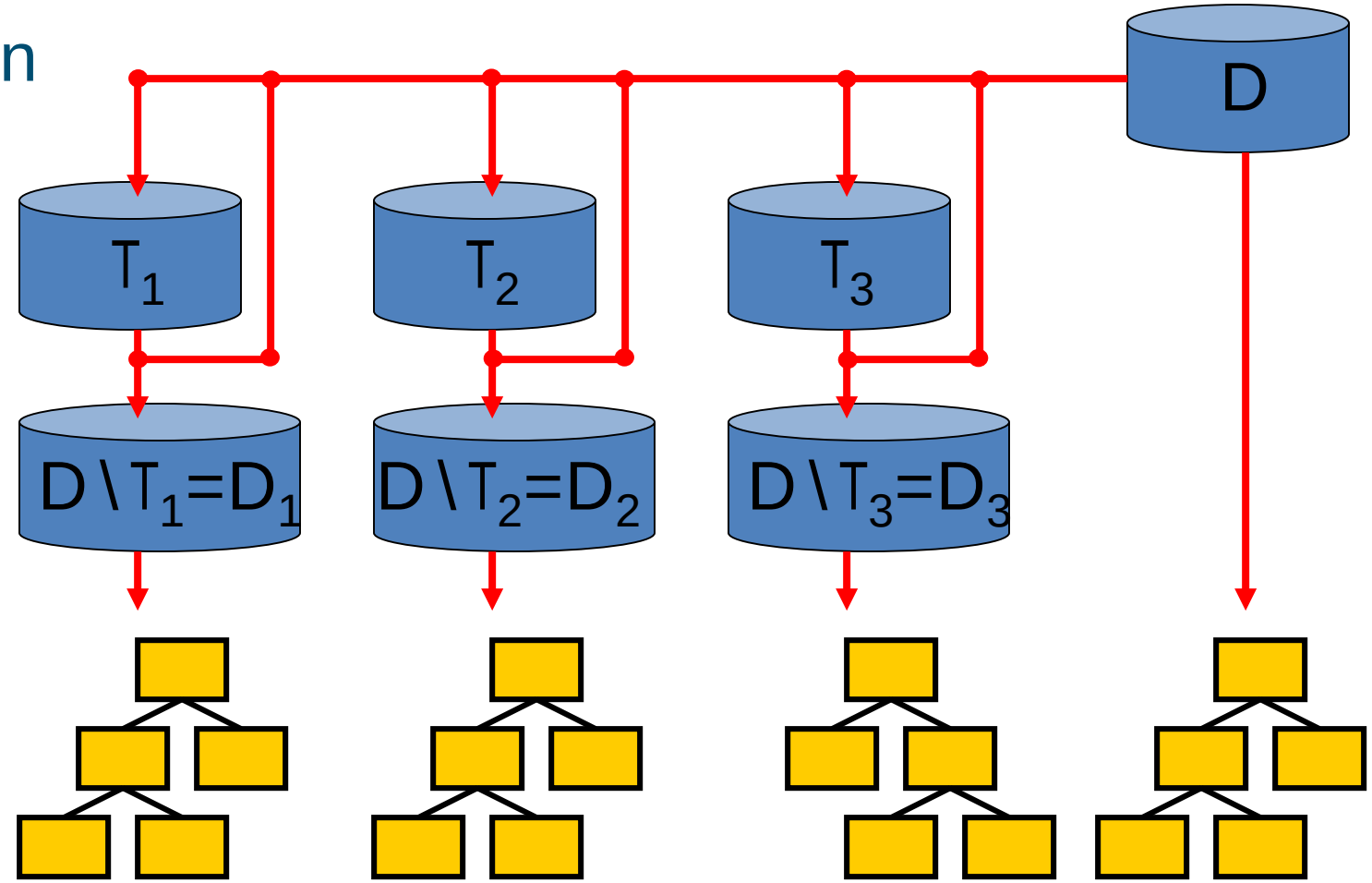
- Partition



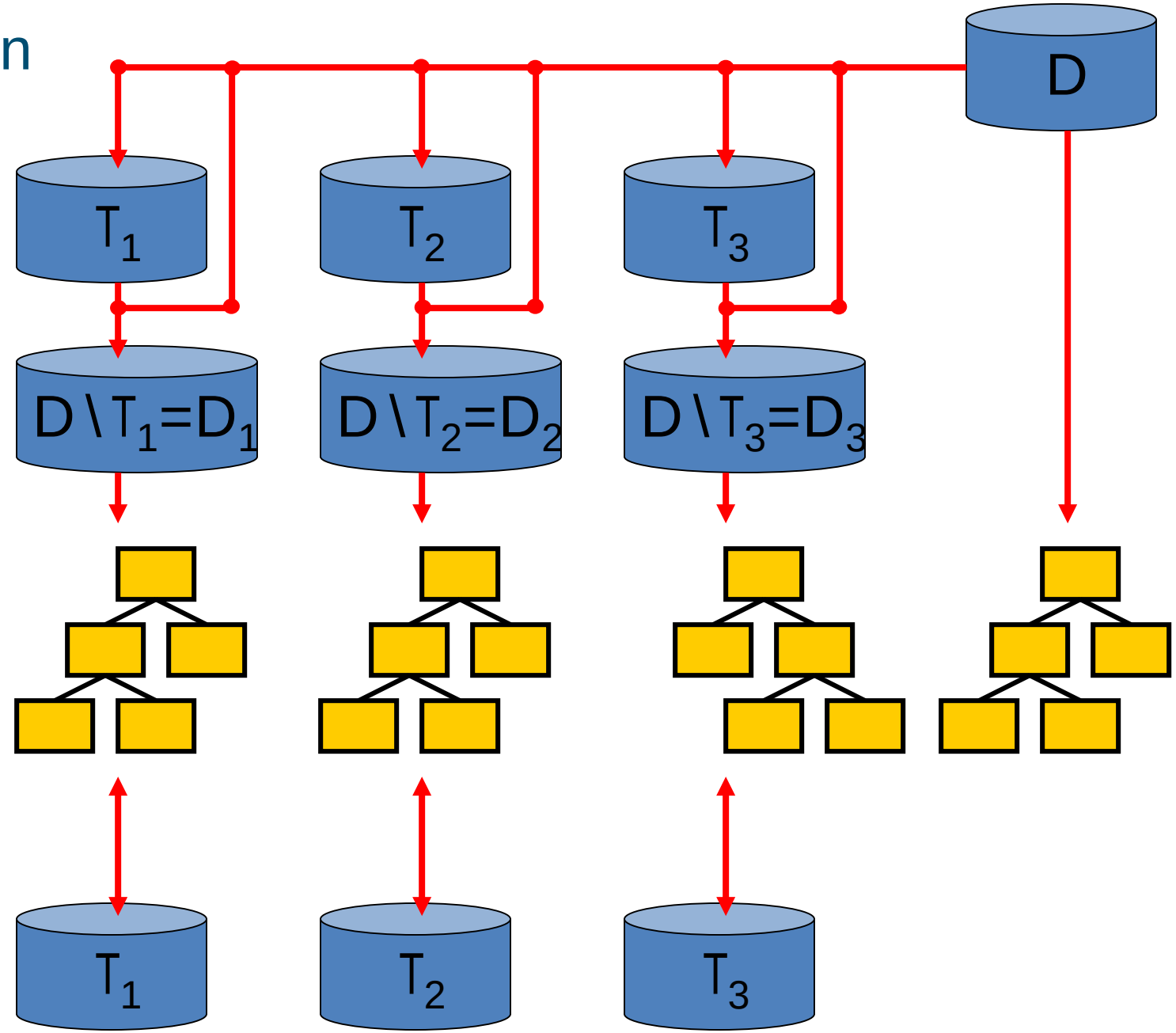
- Partition
- Train



- Partition
- Train



- Partition
- Train
- Test



5. Evaluación de clasificadores

Leaving-one-out

- Es un caso especial de validación cruzada en el que *k* es igual al número de registros
  - Tiene la ventaja de que el proceso es determinista y de que en todo momento se utiliza el máximo posible de datos para la inducción del clasificador
  - Se utiliza en BBDD muy pequeñas, debido a su alto costo computacional

5. Evaluación de clasificadores

Bootstrap

- Está basado en el proceso de muestreo con reposición
- A partir de una BD con *n* registros se obtiene un CE con *n* casos
- Como CT se utilizan los registros de la BD no seleccionados para el CE

  ¿Cuántos casos habrá en CT? ¿qué porcentaje respecto a *n*?

  - La probabilidad de que se elija un registro es *1/n*. La probabilidad de que no se elija es *1-1/n*
  - Se hacen *n* extracciones, por tanto la probabilidad de que un ejemplo no sea elegido es

$$\left(1-\frac{1}{n}\right)^{n} \approx e^{-1} = 0.368$$

  - El CE tendrá aproximadamente el 63.2% de los registros de la BD y el CT el 36.8 %
- Esta técnica se conoce como 0.632 bootstrap
- El error sobre el CT suele ser bastante pesimista por lo que se corrige

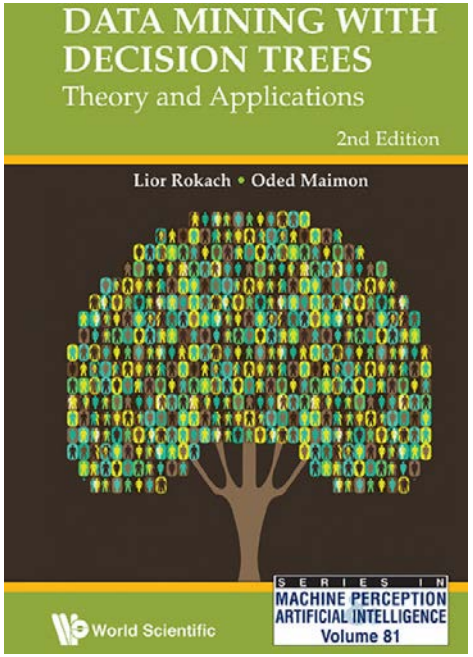
$$error = 0.632 \cdot error_{CT} + 0.368 \cdot error_{CE}$$

Inteligencia de Negocio

TEMA 3. Modelos de Predicción: Clasificación, regresión y series temporales

Clasificación

1. El problema de clasificación
2. **Clasificación con árboles y reglas**
3. Clasificación con otras técnicas
4. Multiclasificadores



Clasificación con árboles y reglas

1. **Árboles de decisión**
   - **1.1. Definición de árboles de decisión**
   - 1.2. Construcción de árboles de decisión
   - 1.3. Criterios de selección de variables
   - 1.4. Particionamiento del espacio con un árbol de decisión
   - 1.5. Ventajas e inconvenientes del uso de árboles de decisión en clasificación
   - 1.6. Algunos algoritmos de minería de datos basados en árboles de decisión

2. Clasificadores basados en reglas

1. Definición de árboles de decisión

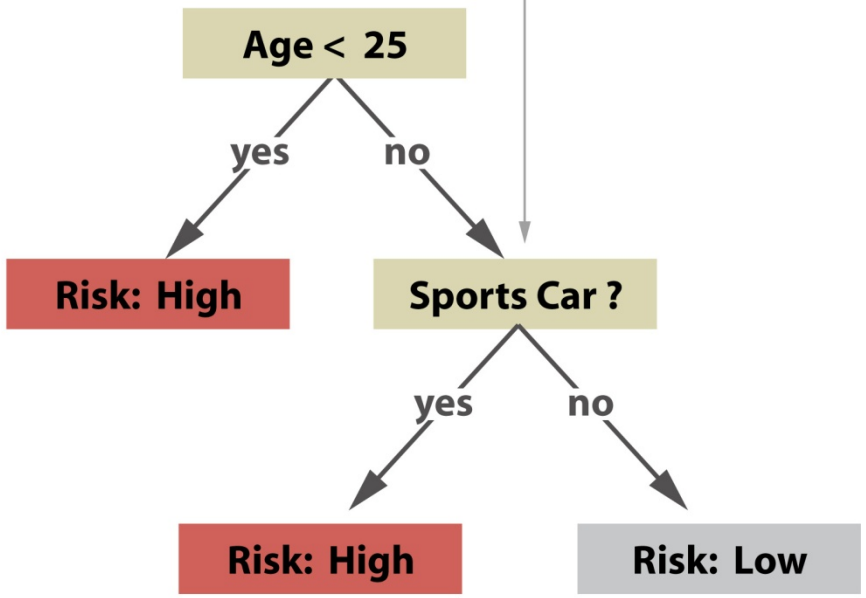
- Un árbol de decisión es un clasificador que en función de un conjunto de atributos permite determinar a qué clase pertenece el caso objeto de estudio

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

1. Definición de árboles de decisión

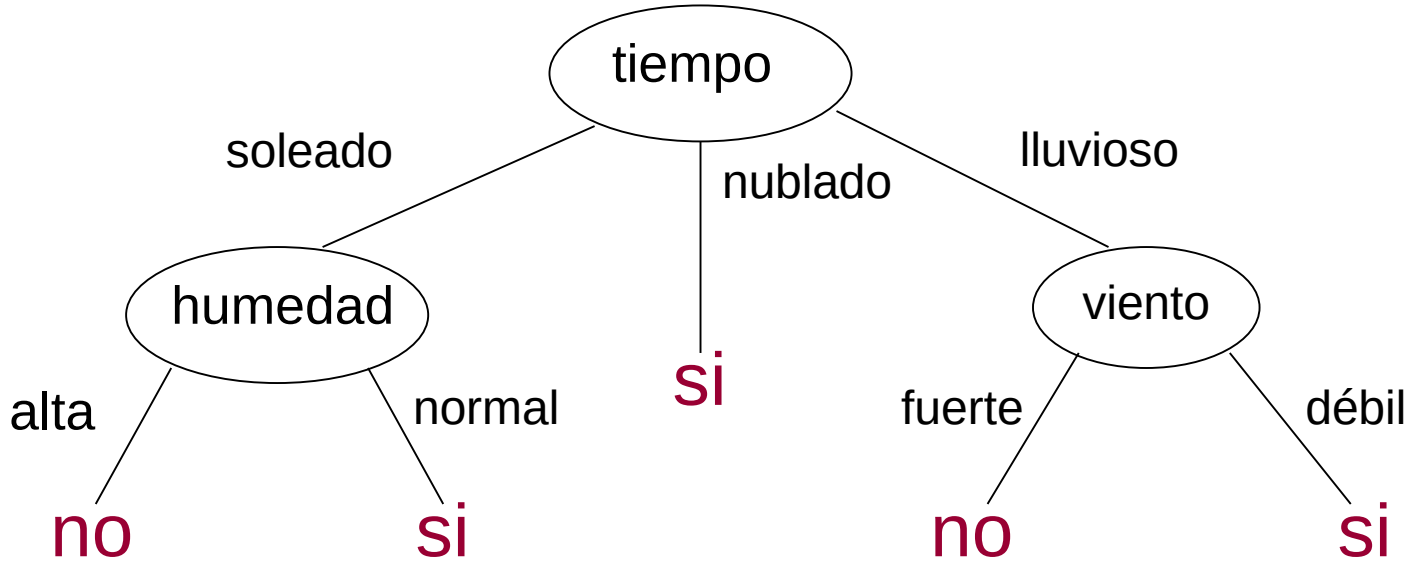
Insurance Risk Assessment

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



1. Definición de árboles de decisión

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



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

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

1. Definición de árboles de decisión

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

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

Clasificación con árboles y reglas

1. Árboles de decisión
   - 1.1. Definición de árboles de decisión
   - **1.2. Construcción de árboles de decisión**
   - 1.3. Criterios de selección de variables
   - 1.4. Particionamiento del espacio con un árbol de decisión
   - 1.5. Ventajas e inconvenientes del uso de árboles de decisión en clasificación
   - 1.6. Algunos algoritmos de minería de datos basados en árboles de decisión

2. Clasificadores basados en reglas

2. Construcción de árboles de decisión

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

2. Construcción de árboles de decisión

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

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

2. Construcción de árboles de decisión

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

ConstruirArbol (T,C,A)

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

2. Construcción de árboles de decisión

Problema de asignación de crédito

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

2. Construcción de árboles de decisión

1. Se llama al algoritmo sobre el nodo raiz

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

2. Construcción de árboles de decisión

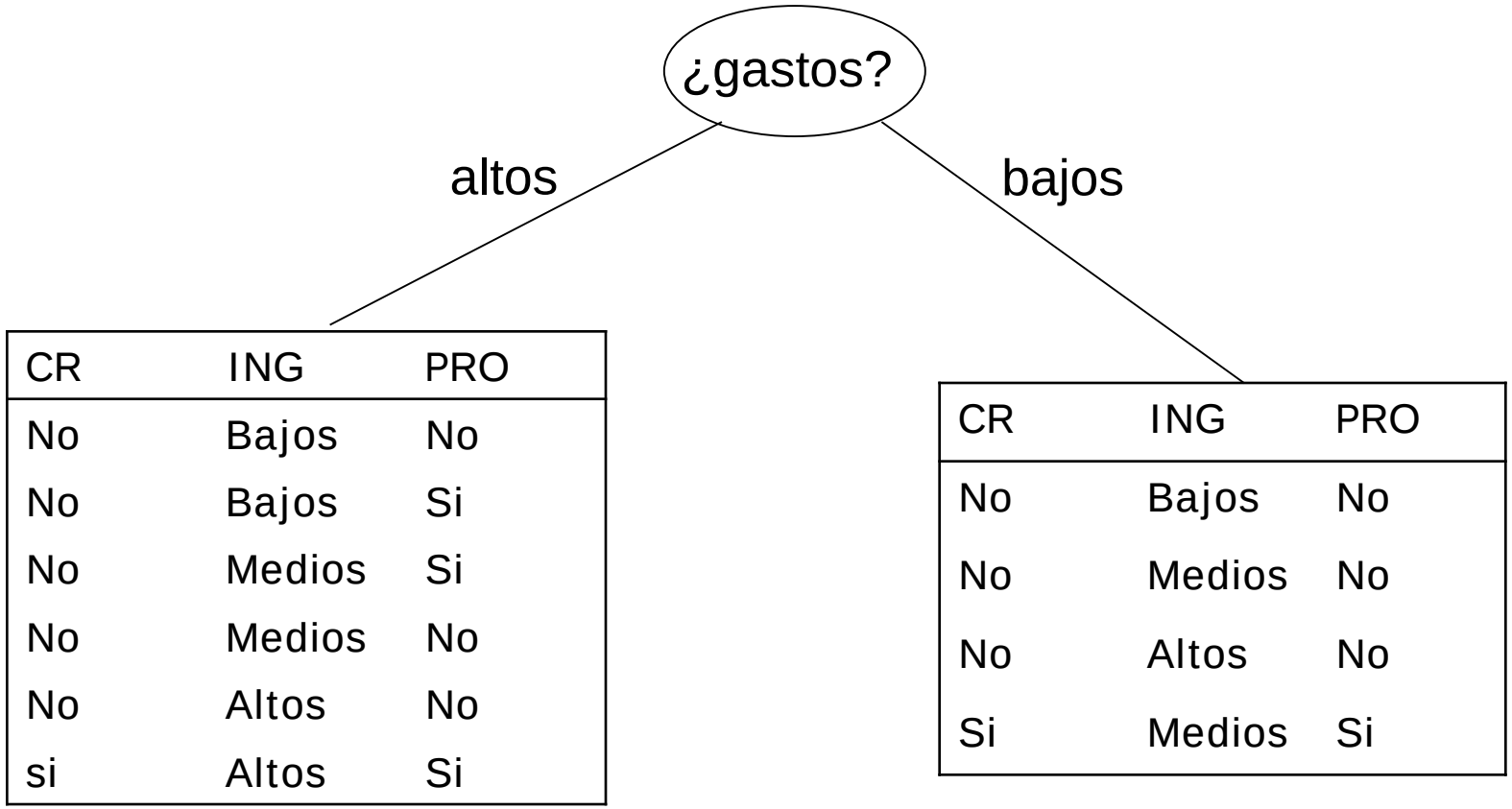
2. Seleccionamos gastos como atributo que mejor clasifica

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

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

2. Construcción de árboles de decisión

3. Preparamos los nodos para las llamadas recursivas

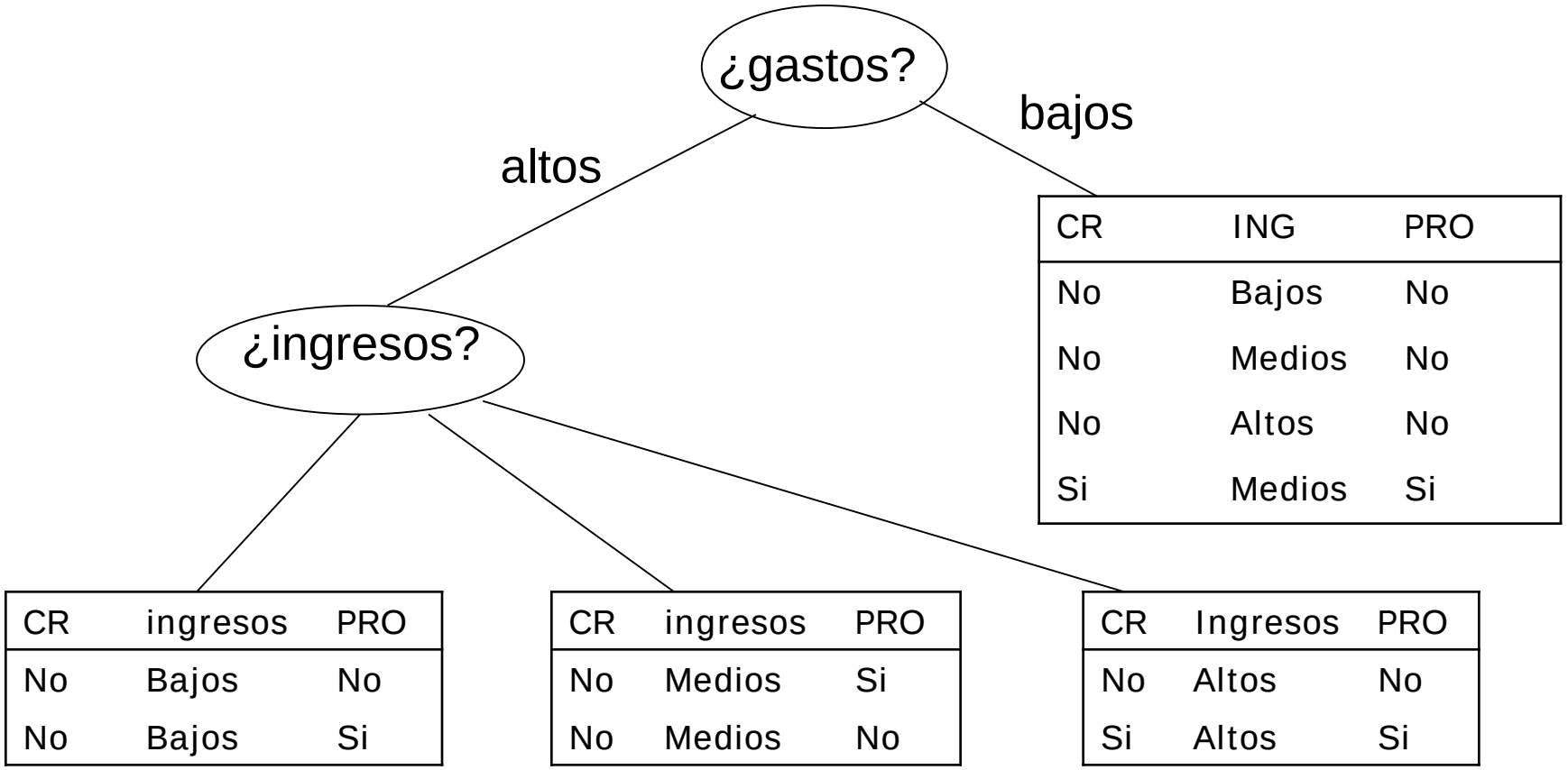


CR	ING	PRO
No	Bajos	No
No	Bajos	Si
No	Medios	Si
No	Medios	No
No	Altos	No
si	Altos	Si

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

2. Construcción de árboles de decisión

4. Seleccionamos ingresos como atributo en gastos = altos



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

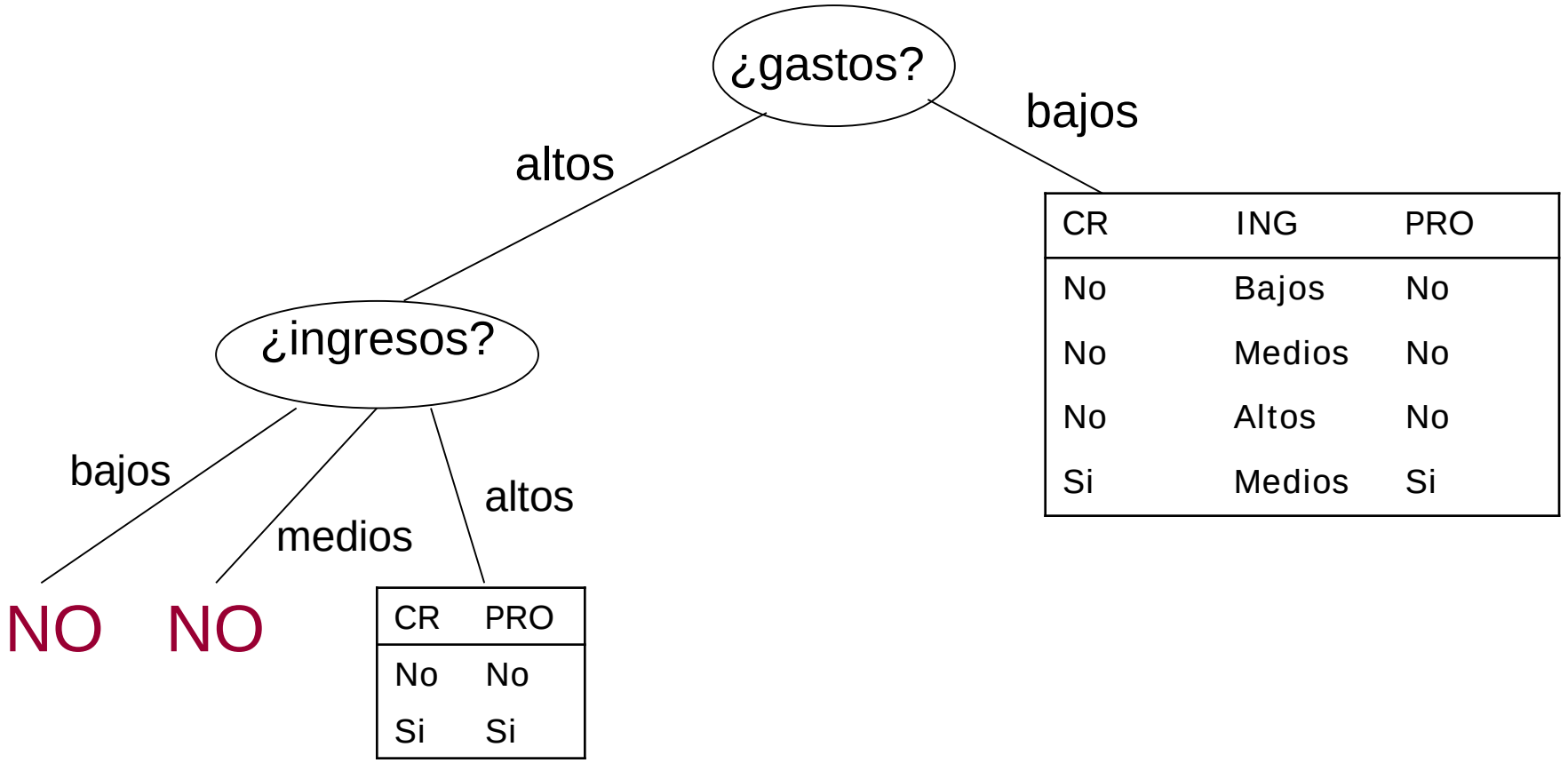
CR	ingresos	PRO
No	Bajos	No
No	Bajos	Si

CR	ingresos	PRO
No	Medios	Si
No	Medios	No

CR	Ingresos	PRO
No	Altos	No
Si	Altos	Si

2. Construcción de árboles de decisión

5. Creamos nodos hoja

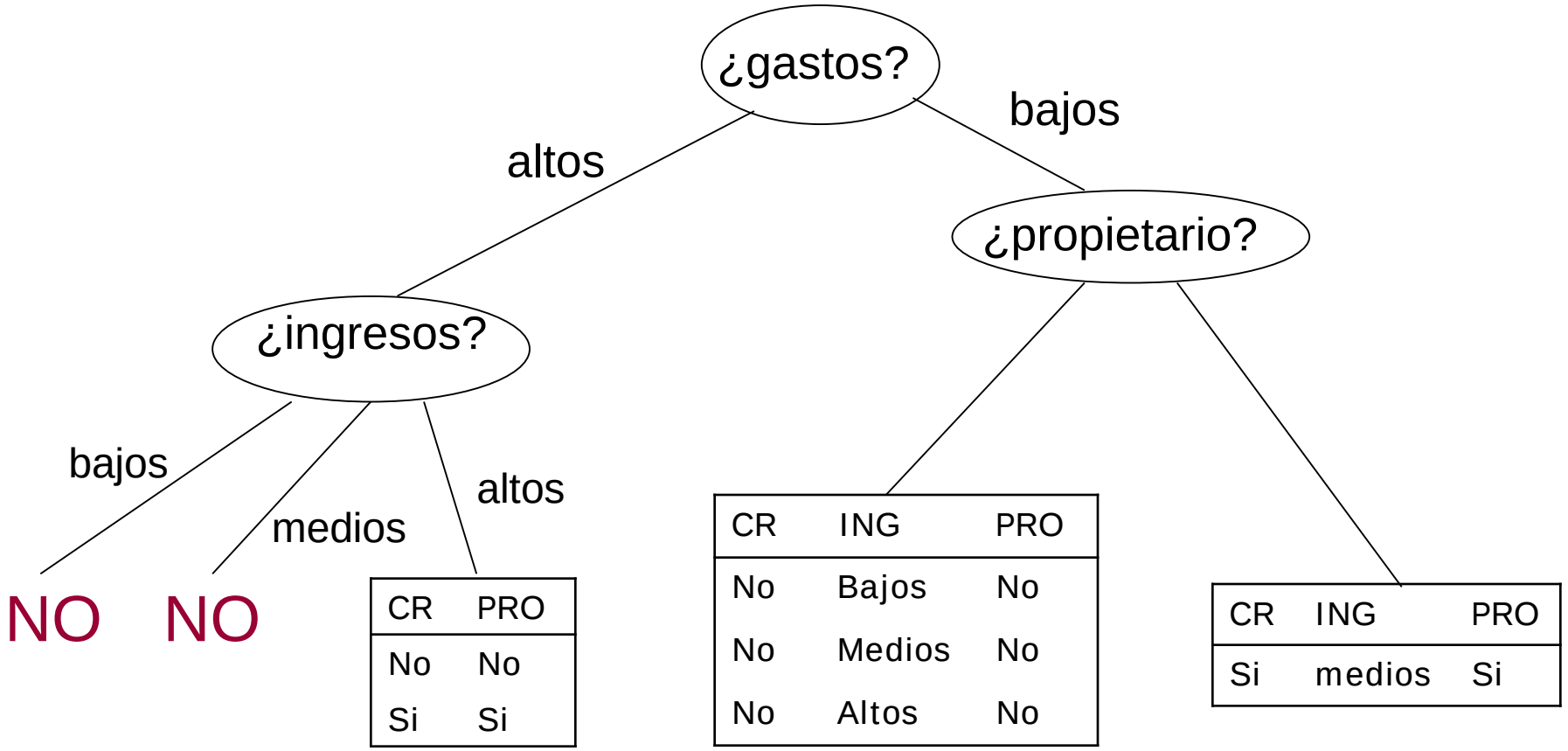


CR	ING	PRO
No	Bajos	No
No	Medios	No
No	Altos	No
Si	Medios	Si

CR	PRO
No	No
Si	Si

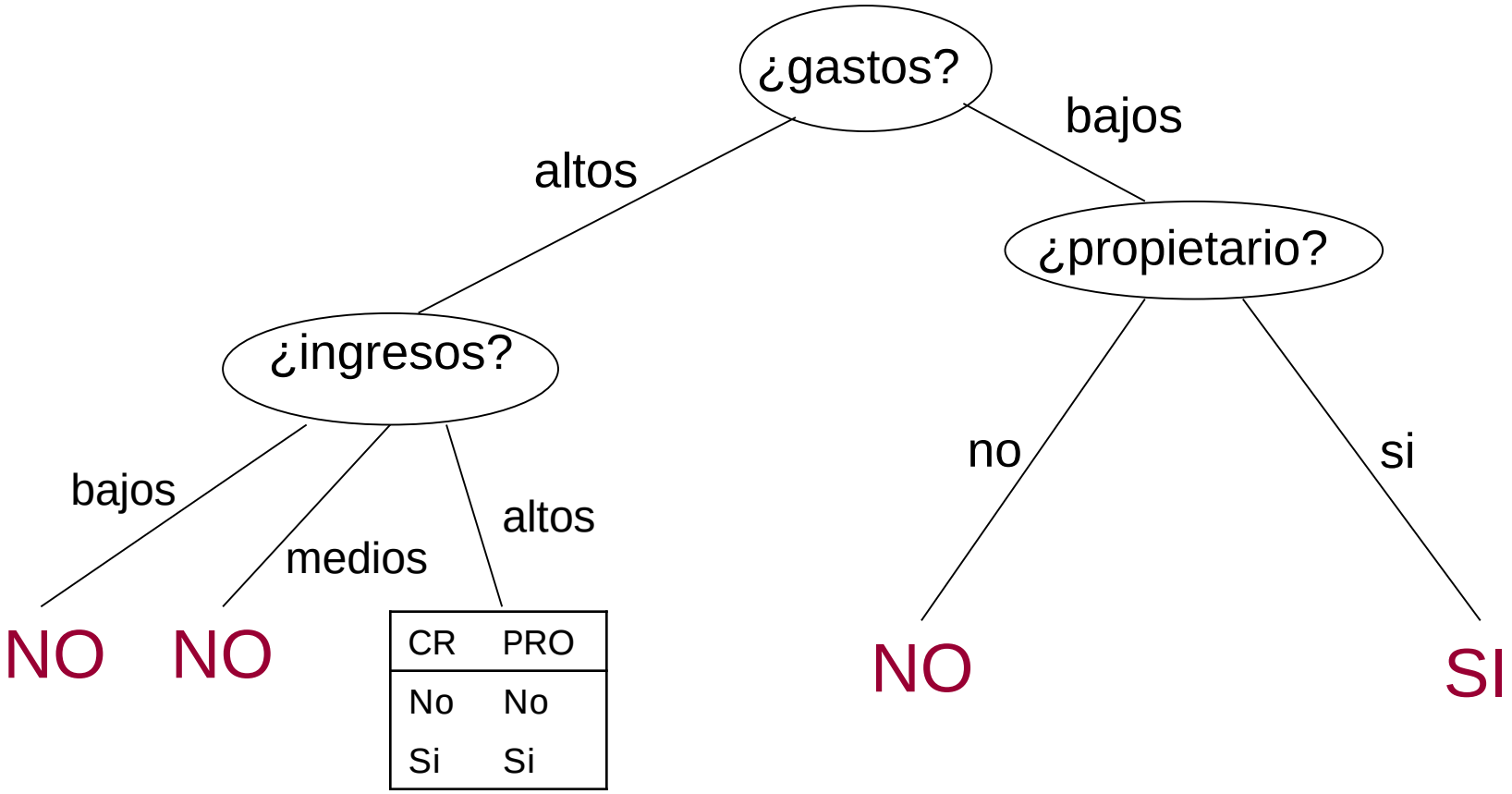
2. Construcción de árboles de decisión

6. Seleccionamos propietario como atributo en gastos = bajos



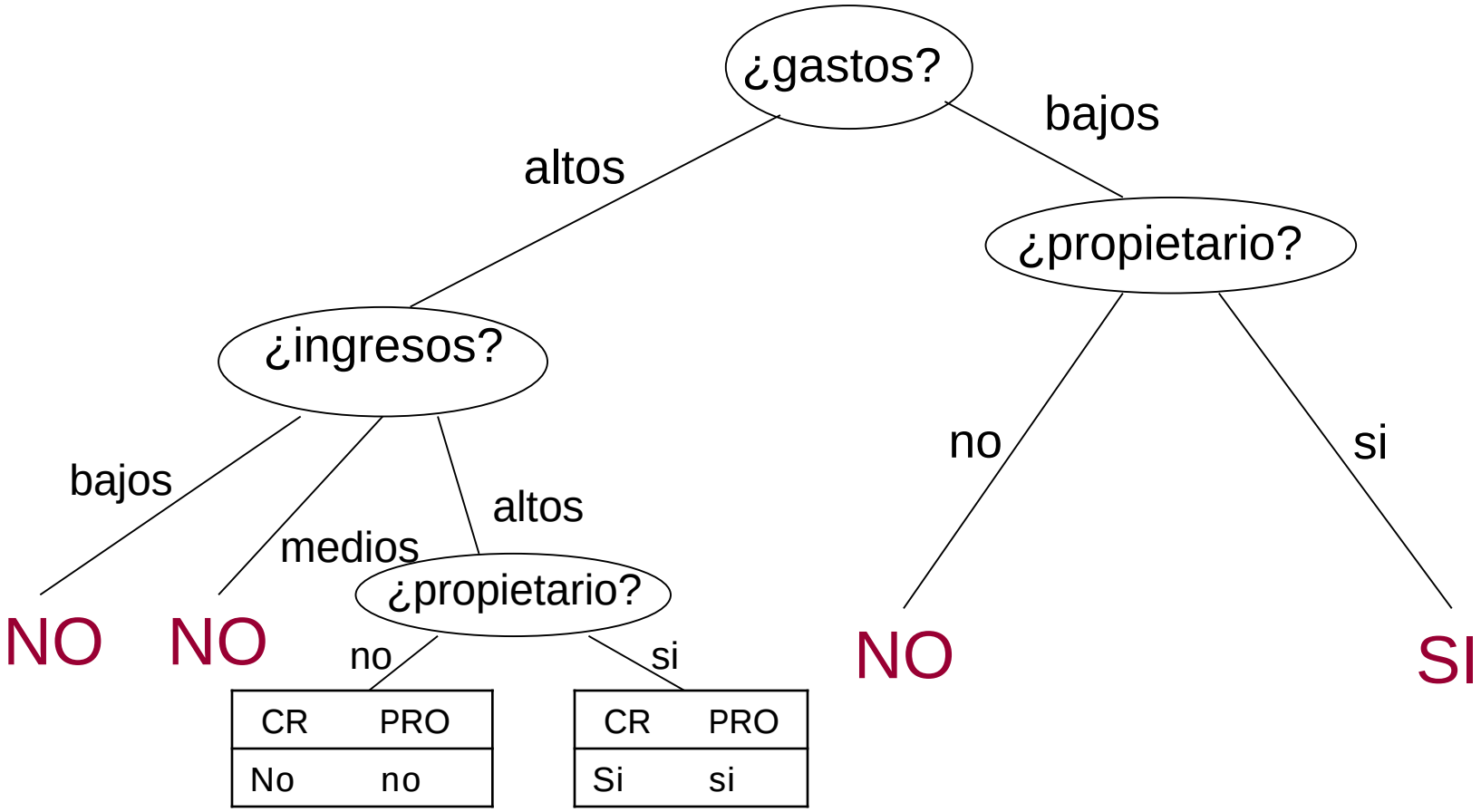
2. Construcción de árboles de decisión

7. Creamos nodos hoja



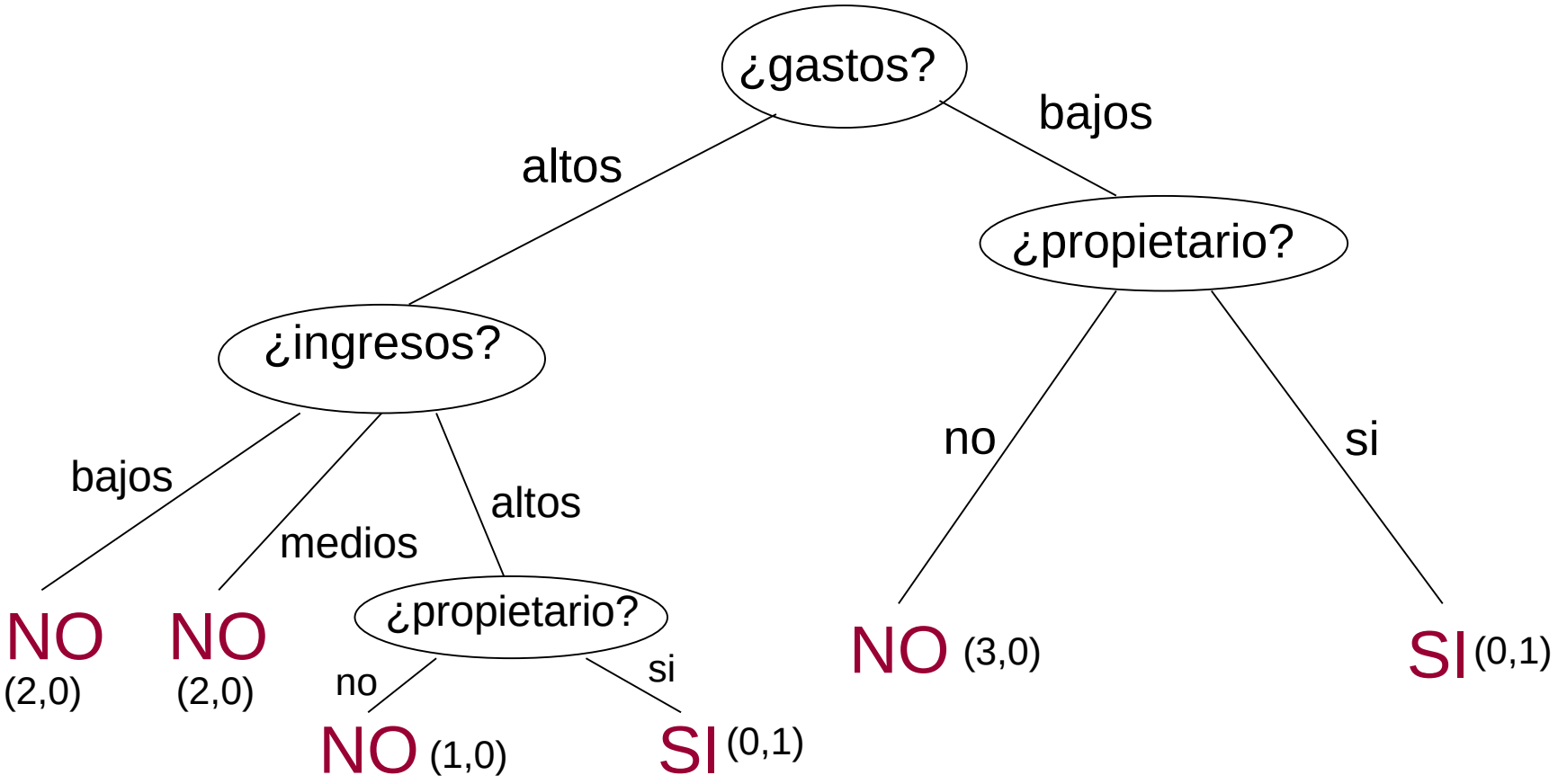
2. Construcción de árboles de decisión

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



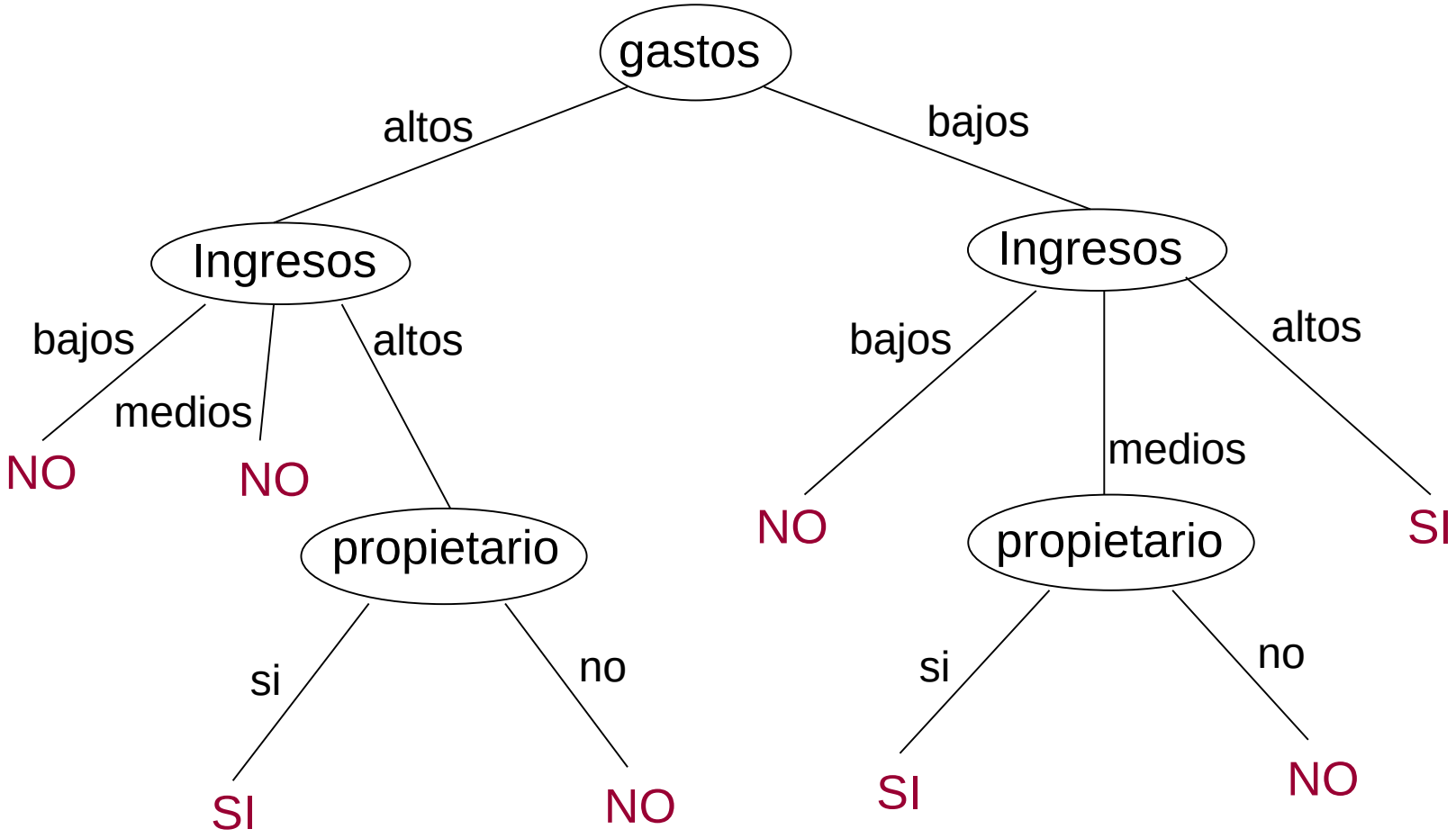
2. Construcción de árboles de decisión

9. Árbol de decisión (#no, #si)



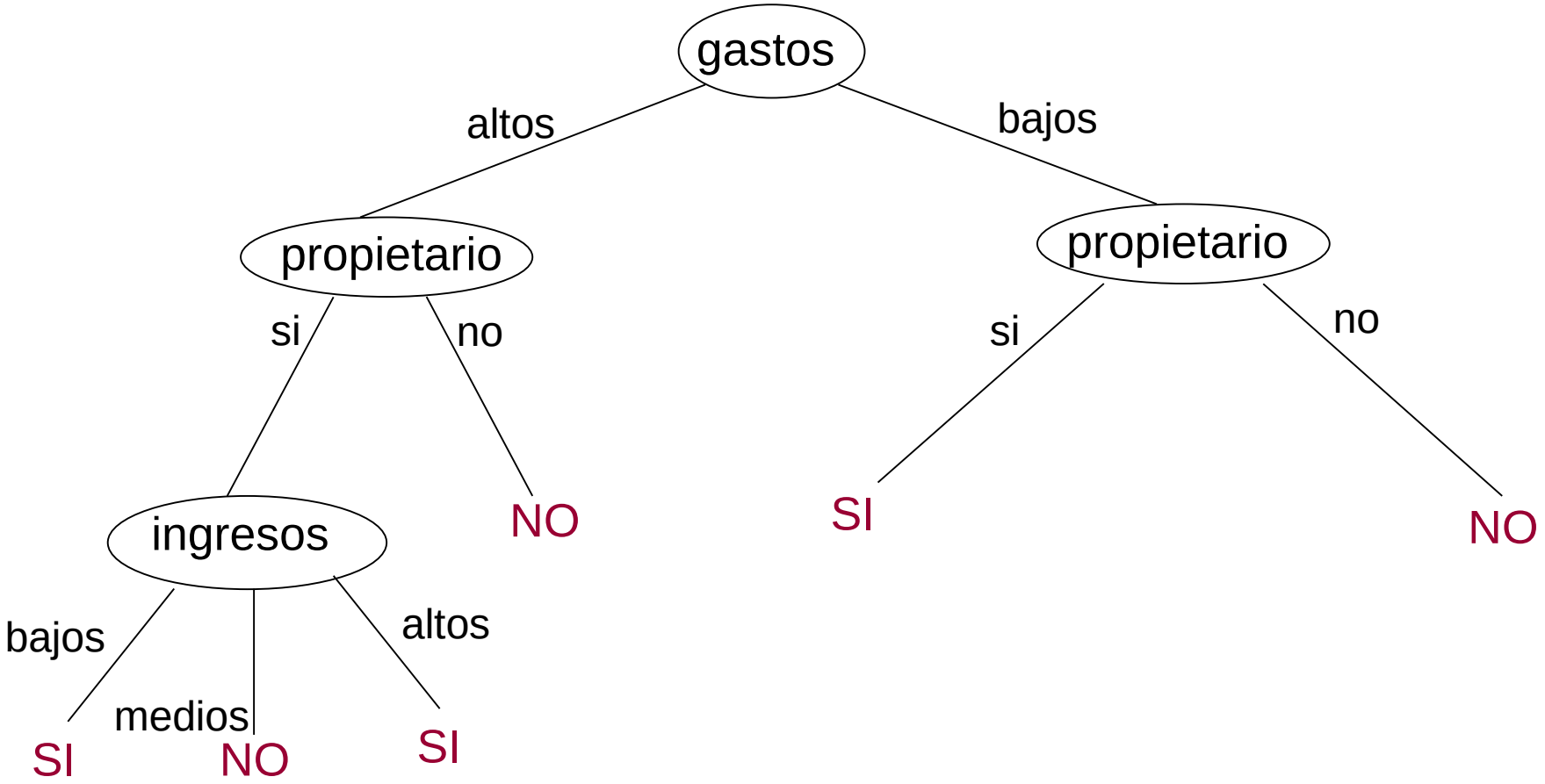
2. Construcción de árboles de decisión

Otro árbol de decisión para crédito (árbol 2)



2. Construcción de árboles de decisión

Otro árbol de decisión para crédito (árbol 3)



2. Construcción de árboles de decisión

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

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

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

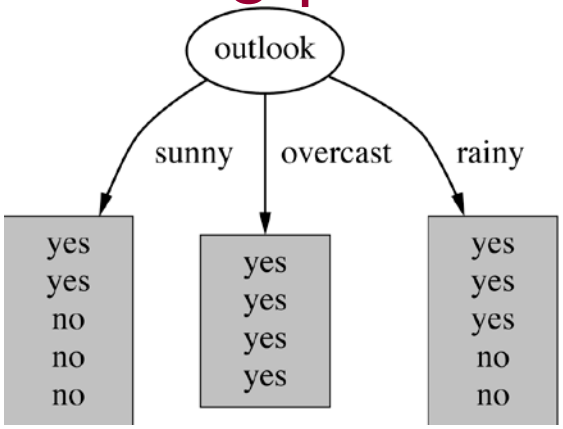
¿Cómo decidir qué atributo es el mejor?

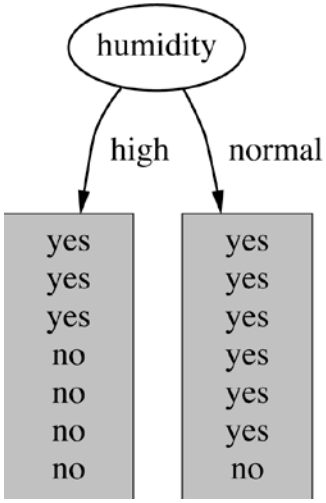
2. Construcción de árboles de decisión

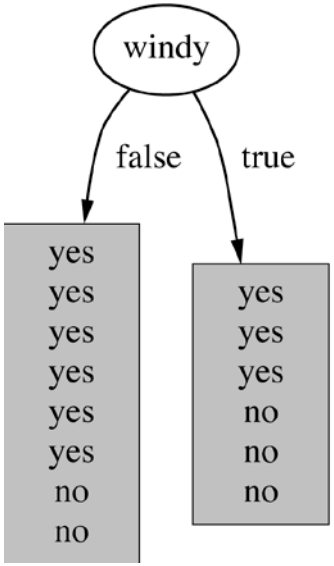
Outlook	Temperature	Humidity	Windy	Play?
sunny	hot	high	false	No
sunny	hot	high	true	No
overcast	hot	high	false	Yes
rain	mild	high	false	Yes
rain	cool	normal	false	Yes
rain	cool	normal	true	No
overcast	cool	normal	true	Yes
sunny	mild	high	false	No
sunny	cool	normal	false	Yes
rain	mild	normal	false	Yes
sunny	mild	normal	true	Yes
overcast	mild	high	true	Yes
overcast	hot	normal	false	Yes
rain	mild	high	true	No

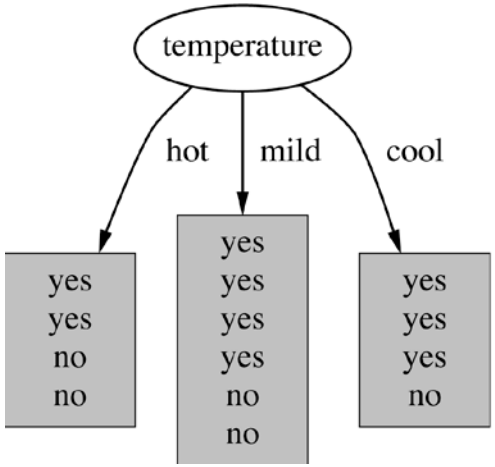
2. Construcción de árboles de decisión

¿qué atributo seleccionamos?









Clasificación con árboles y reglas

1. Árboles de decisión
   - 1.1. Definición de árboles de decisión
   - 1.2. Construcción de árboles de decisión
   - **1.3. Criterios de selección de variables**
   - 1.4. Particionamiento del espacio con un árbol de decisión
   - 1.5. Ventajas e inconvenientes del uso de árboles de decisión en clasificación
   - 1.6. Algunos algoritmos de minería de datos basados en árboles de decisión

2. Clasificadores basados en reglas

3. Criterios de selección de variables

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

- *InfoGain*: Ganancia de información (ID3) **(H es la entropía de la variable)**

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

$$H(X) = -\sum_i p(x_i) \log_2 p(x_i)$$

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

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

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

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

3. Criterios de selección de variables

Ganancia de información

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

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

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

3. Criterios de selección de variables

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

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

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

$$Gain(A) = H(p,n) - H(C/A)$$

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

3. Criterios de selección de variables

Conjunto de entrenamiento

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

3. Criterios de selección de variables

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

$$H(Clase/age) = \frac{5}{14}H(2,3) + \frac{4}{14}H(4,0) + \frac{5}{14}H(3,2) = 0.69$$

$$Gain(age) = H(p,n) - H(Clase/age)$$

De forma similar

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

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

3. Criterios de selección de variables

GINI

- Si un conjunto de datos *T* tiene ejemplos pertenecientes a *n* clases, el índice Gini, se define como

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

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

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

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

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

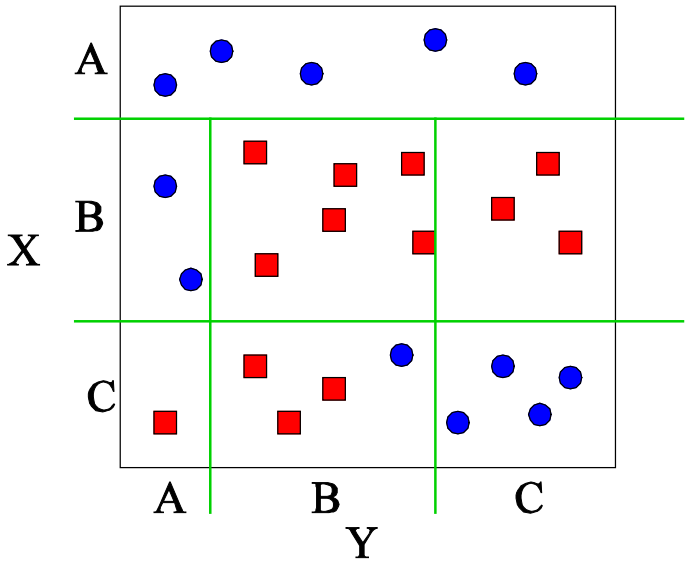
Clasificación con árboles y reglas

1. Árboles de decisión
   - 1.1. Definición de árboles de decisión
   - 1.2. Construcción de árboles de decisión
   - 1.3. Criterios de selección de variables
   - **1.4. Particionamiento del espacio con un árbol de decisión**
   - 1.5. Ventajas e inconvenientes del uso de árboles de decisión en clasificación
   - 1.6. Algunos algoritmos de minería de datos basados en árboles de decisión

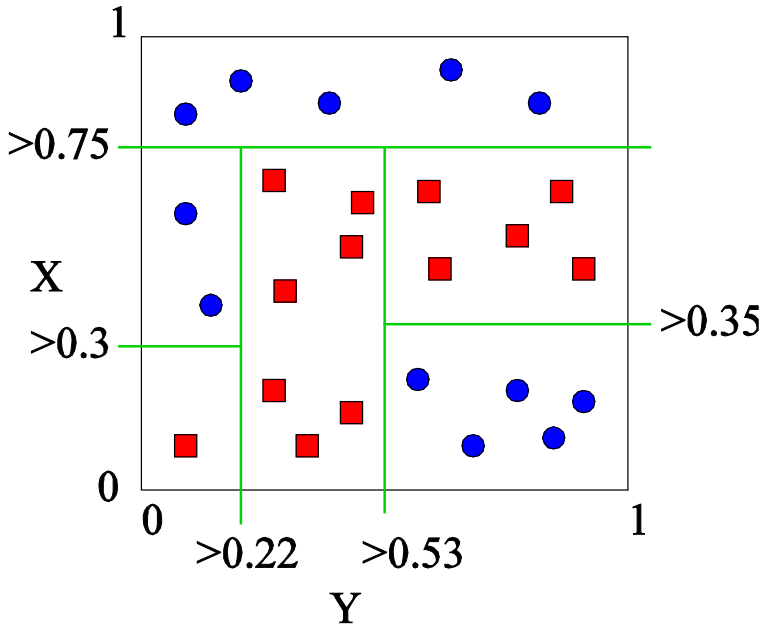
2. Clasificadores basados en reglas

4. Particionamiento del espacio de un árbol de decisión

Los árboles particionan el espacio de soluciones de forma exhaustiva

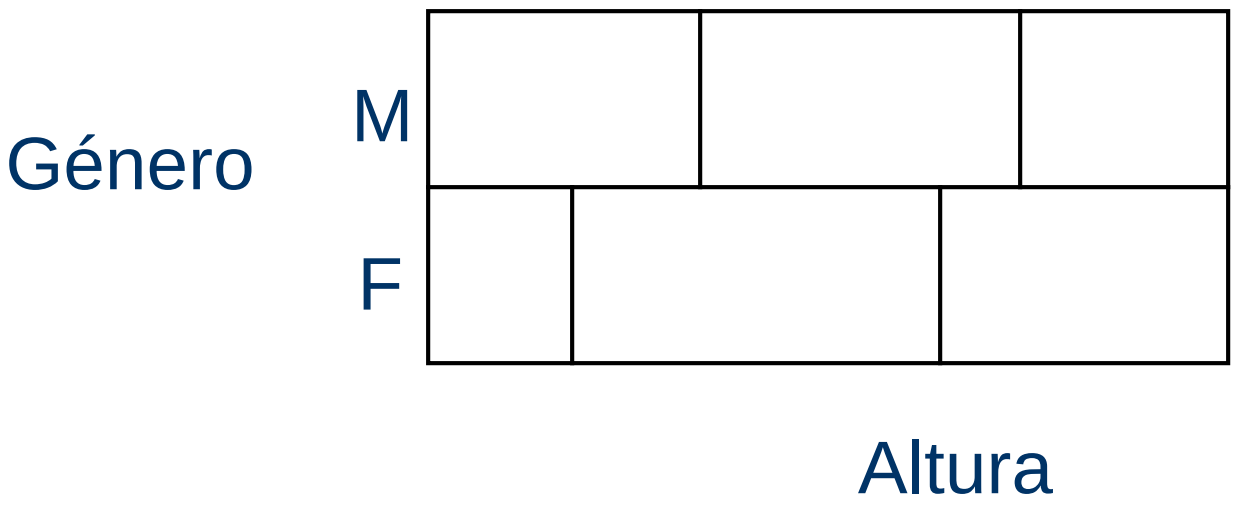


Variables X e Y discretas



Variables X e Y contínuas

4. Particionamiento del espacio de un árbol de decisión



Clasificación con árboles y reglas

1. Árboles de decisión
   - 1.1. Definición de árboles de decisión
   - 1.2. Construcción de árboles de decisión
   - 1.3. Criterios de selección de variables
   - 1.4. Particionamiento del espacio con un árbol de decisión
   - **1.5. Ventajas e inconvenientes del uso de árboles de decisión en clasificación**
   - 1.6. Algunos algoritmos de minería de datos basados en árboles de decisión

2. Clasificadores basados en reglas

5. Ventajas e inconvenientes del uso de arboles de decisión en clasificación

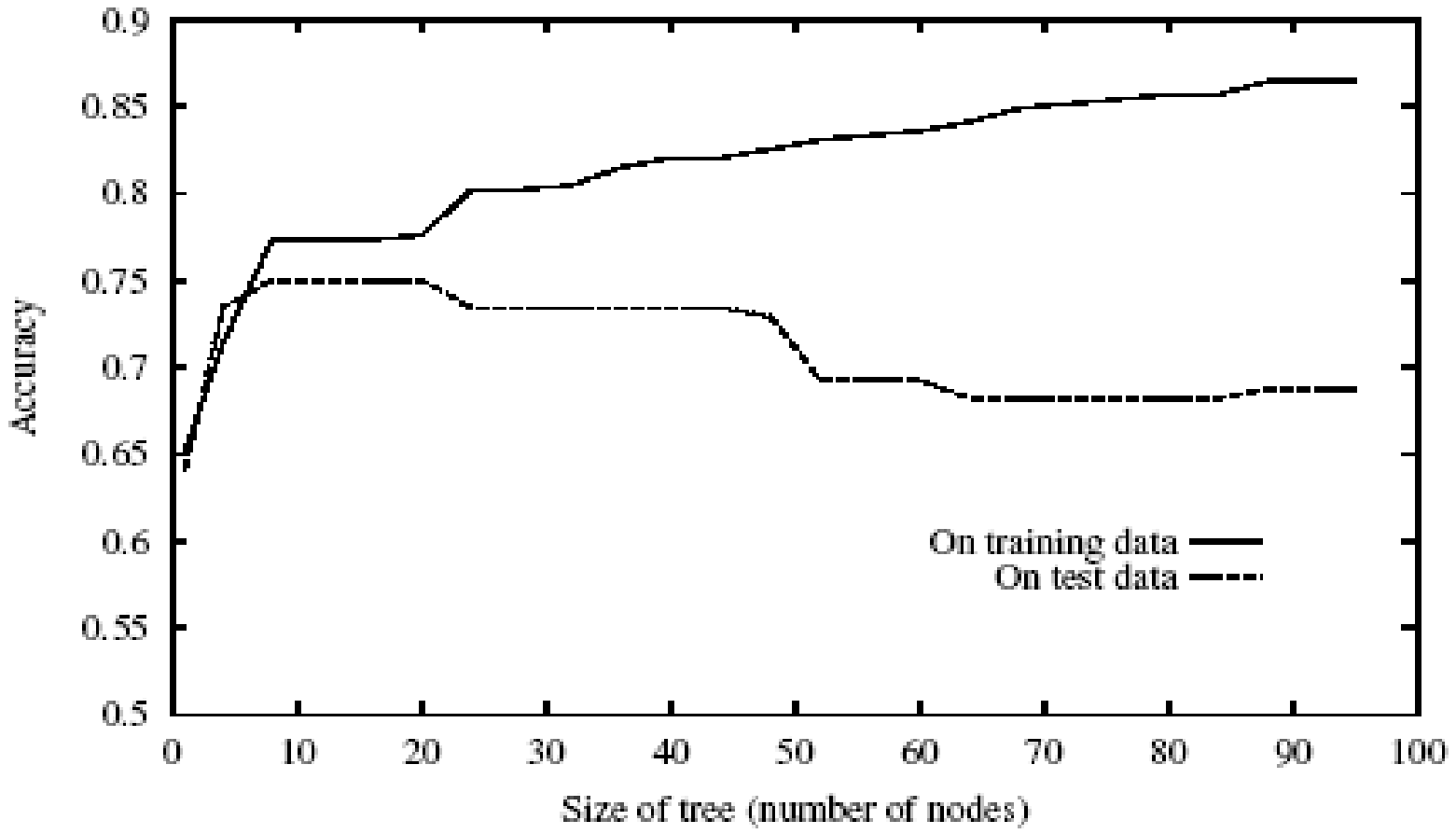
Ventajas:

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

Inconvenientes:

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

5. Ventajas e inconvenientes del uso de arboles de decisión en clasificación



Clasificación con árboles y reglas

1. Árboles de decisión
   - 1.1. Definición de árboles de decisión
   - 1.2. Construcción de árboles de decisión
   - 1.3. Criterios de selección de variables
   - 1.4. Particionamiento del espacio con un árbol de decisión
   - 1.5. Ventajas e inconvenientes del uso de árboles de decisión en clasificación
   - **1.6. Algunos algoritmos de minería de datos basados en árboles de decisión**

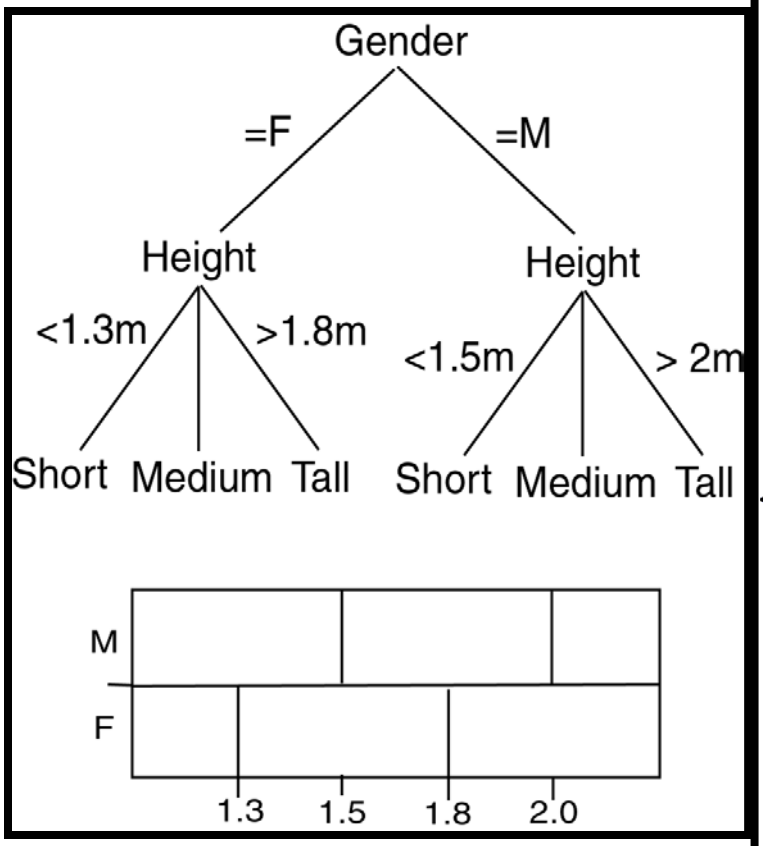
2. Clasificadores basados en reglas

6. Algunos algoritmos de minería de datos basados en árboles de decisión

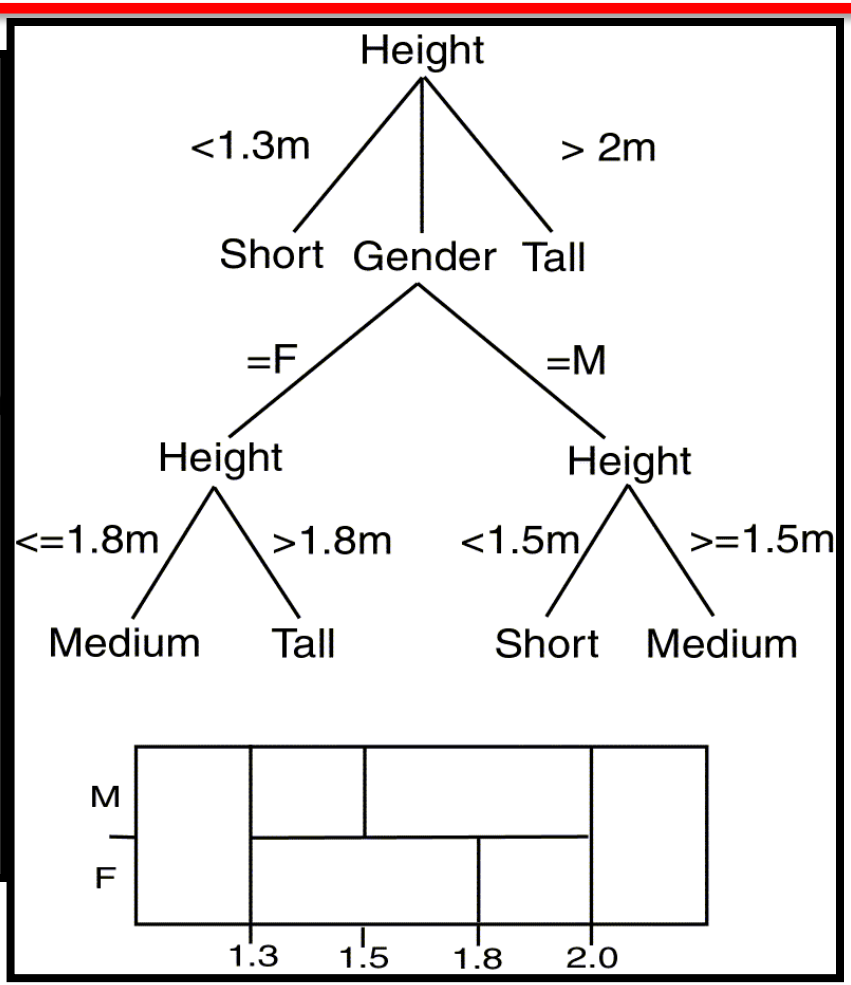
Aspectos importantes en cualquier algoritmo de MD basado en árboles de decisión:

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

6. Algunos algoritmos de minería de datos basados en árboles de decisión



Balanceado



Profundidad

6. Algunos algoritmos de minería de datos basados en árboles de decisión

ID3

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

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

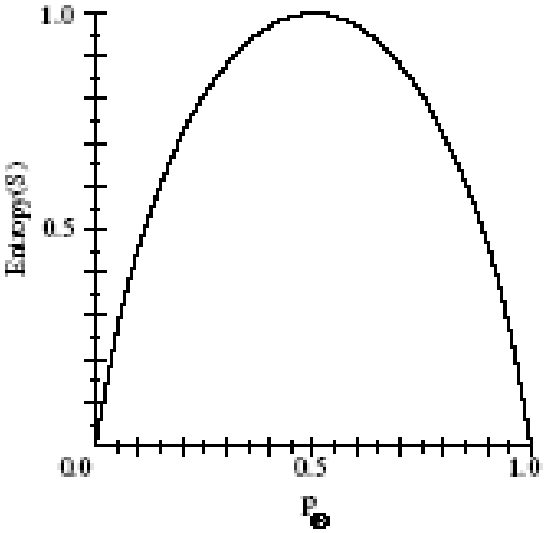
6. Algunos algoritmos de minería de datos basados en árboles de decisión: ID3

Definición de entropía. Dados:

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

La entropía mide la incertidumbre en S

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



- Con k clases:

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

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

- ID3 elige el atributo con mayor ganancia de información

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

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

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

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

Esquema algoritmo ID3

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

   Si sólo hay ejemplos de una clase, C_k

   Entonces etiquetarlo con C_k

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

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

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

Ejemplo:

Name	Gender	Height	Output1
Kristina	F	1.6m	Short
Jim	M	2m	Tall
Maggie	F	1.9m	Medium
Martha	F	1.88m	Medium
Stephanie	F	1.7m	Short
Bob	M	1.85m	Medium
Kathy	F	1.6m	Short
Dave	M	1.7m	Short
Worth	M	2.2m	Tall
Steven	M	2.1m	Tall
Debbie	F	1.8m	Medium
Todd	M	1.95m	Medium
Kim	F	1.9m	Medium
Amy	F	1.8m	Medium
Wynette	F	1.75m	Medium

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

1. Inicialmente, la entropía del conjunto es

H(S) = Entropia(S)=4/15 log (15/4) + 8/15 log(15/8) + 3/15 log (15/3) = 0.4384

Name	Gender	Height	Output1
Kristina	F	1.6m	Short
Maggie	F	1.9m	Medium
Martha	F	1.88m	Medium
Stephanie	F	1.7m	Short
Kathy	F	1.6m	Short
Debbie	F	1.8m	Medium
Kim	F	1.9m	Medium
Amy	F	1.8m	Medium
Wynette	F	1.75m	Medium

2. Elegir el atributo test con máxima ganancia de información

Para genero,

H(S,gender=F)=3/9 log(9/3) + 6/9 log (9/6) =0.2764

H(S,gender=M)=1/6 log(6/1) + 2/6 log (6/2) + 3/6 log(6/3)= 0.4392

Name	Gender	Height	Output1
Jim	M	2m	Tall
Bob	M	1.85m	Medium
Dave	M	1.7m	Short
Worth	M	2.2m	Tall
Steven	M	2.1m	Tall
Todd	M	1.95m	Medium

Gain(S,gender) = 0.4384 – 9/15·0.2764 – 6/15·0.4392 = 0.09688

Para altura, es necesario dividir en rangos: 1:(0,1.6], 2:(1.6,1.7], 3:(1.7,1.8], 4:(1.8,1.9], 5:(1.9,2.0], 6:(2.0,∞)

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

Name	Gender	Height	Output1
Kristina	F	1	Short
Jim	M	5	Tall
Maggie	F	4	Medium
Martha	F	4	Medium
Stephanie	F	2	Short
Bob	M	4	Medium
Kathy	F	1	Short
Dave	M	2	Short
Worth	M	6	Tall
Steven	M	6	Tall
Debbie	F	3	Medium
Todd	M	5	Medium
Kim	F	4	Medium
Amy	F	3	Medium
Wynette	F	3	Medium

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

H(S,height=1)=2/2 log(2/2)=0

H(S,heigth=2)=2/2 log(2/2)=0

H(S,height=3)=3/3 log(3/3)=0

H(S,height=4)=4/4 log(4/4)=0

H(S,height=5)=1/2 log 2 + 1/2 log 2 = 0.301

H(S,height=6)=2/2 log (2/2)=0

Gain(S,height) =0.4384 – 2/15 · 0.301 = 0.3983

Es mayor, por lo que se elige el atributo height

Name	Gender	Height	Output1
Kristina	F	1	Short
Kathy	F	1	Short

Name	Gender	Height	Output1
Stephanie	F	2	Short
Dave	M	2	Short

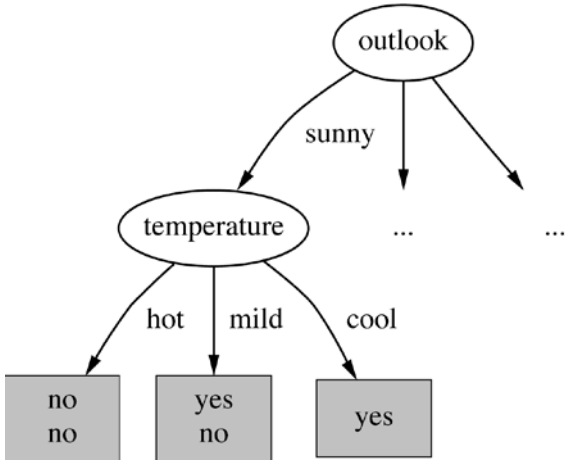
Name	Gender	Height	Output1
Debbie	F	3	Medium
Amy	F	3	Medium
Wynette	F	3	Medium

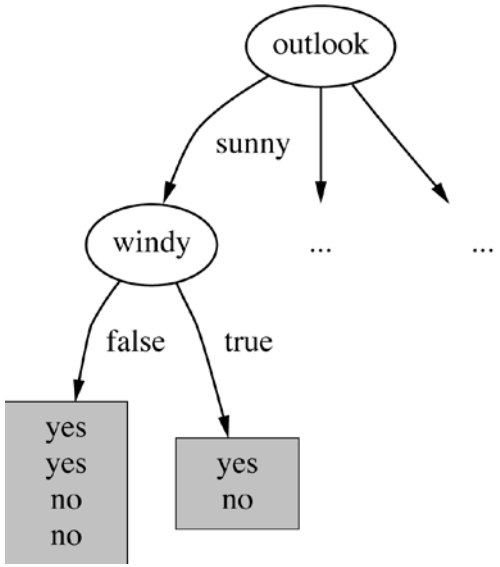
Name	Gender	Height	Output1
Maggie	F	4	Medium
Martha	F	4	Medium
Bob	M	4	Medium
Kim	F	4	Medium

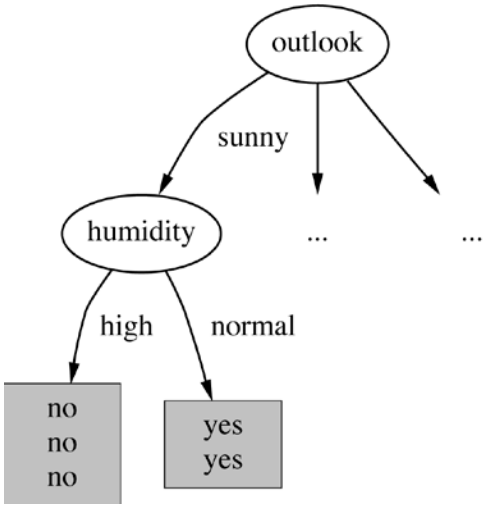
Name	Gender	Height	Output1
Jim	M	5	Tall
Todd	M	5	Medium

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

Ejemplo 2: Jugar al tenis





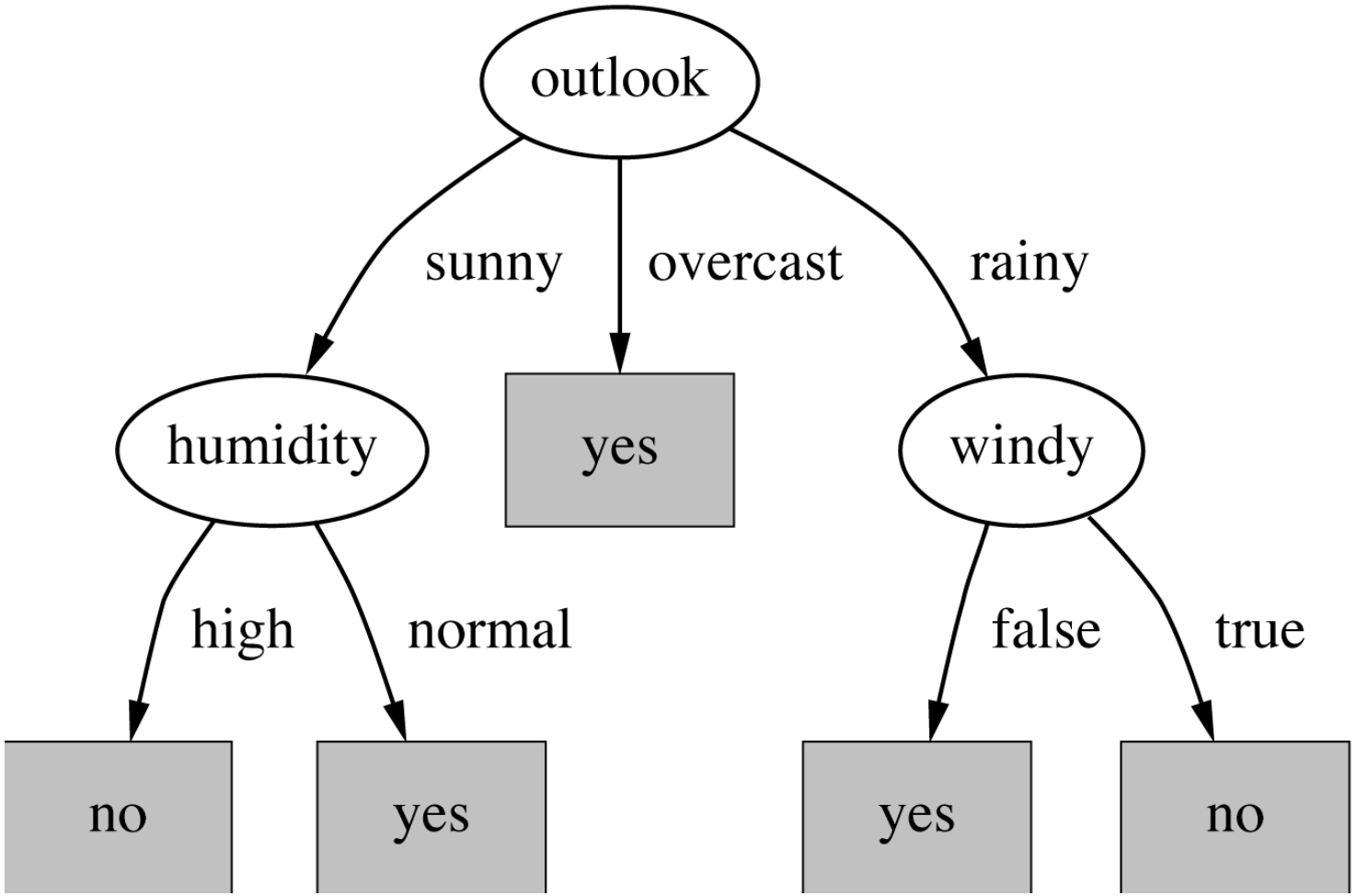


$$gain("Humidity") = 0.971$$

$$gain("Temperature") = 0.571$$

$$gain("Windy") = 0.020$$

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



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

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

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

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

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

Refinamiento de ID3

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

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

C4.5

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

Mejora a ID3 en los siguientes aspectos:

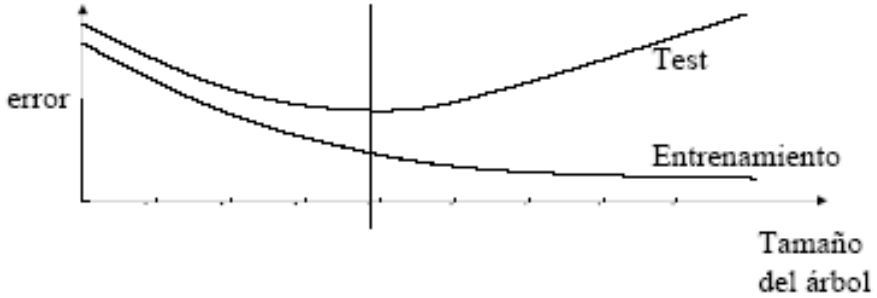
- Datos perdidos:
  - Cuando se construye el árbol, los datos perdidos se ignoran (el árbol de decisión se construye mirando sólo los registros que tienen valor para ese atributo)
  - Para clasificar un ejemplo con valor perdido, éste se predice en base a lo que se sabe sobre los valores del atributo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

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

Pre-poda:

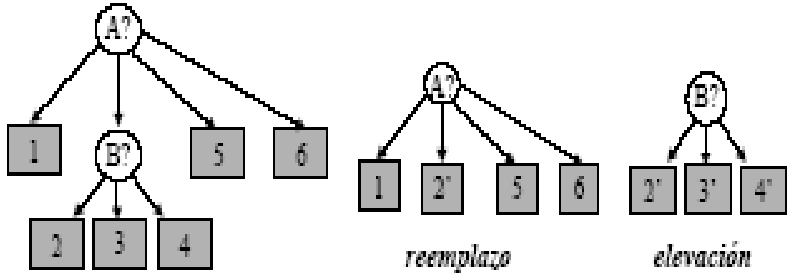
• no se divide un nodo si se tiene poca confianza en él (no es significativa la diferencia de clases), o

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



Hay dos estrategias de post- poda en C4.5

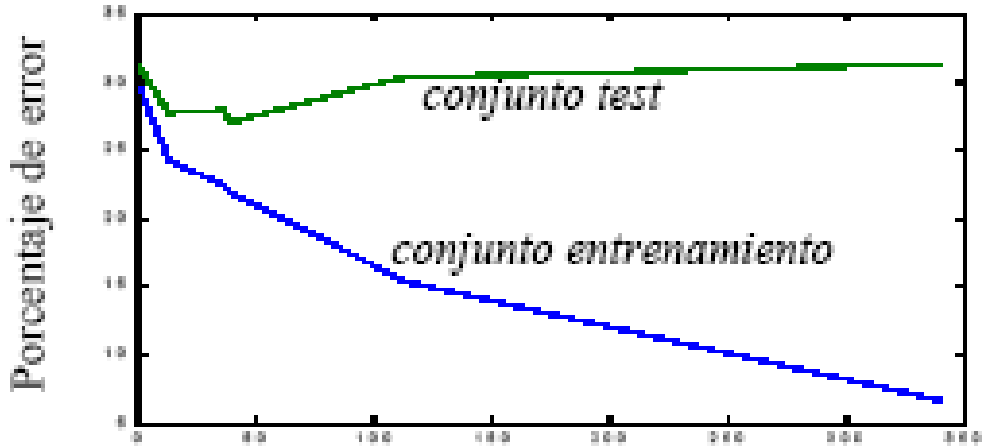
- Reemplazamiento de subárboles: Se reemplaza un subárbol por una hoja si al hacerlo el error es similar al original
- Elevación de subárbol: Reemplaza un subárbol por su subárbol más utilizado (un subárbol se mueve de su localización a un nodo superior en el árbol)



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

Efecto del tamaño del árbol

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



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

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

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

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

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

En el ejemplo anterior:

Name	Gender	Height	Output1
Kristina	F	1.6m	Short
Maggie	F	1.9m	Medium
Martha	F	1.88m	Medium
Stephanie	F	1.7m	Short
Kathy	F	1.6m	Short
Debbie	F	1.8m	Medium
Kim	F	1.9m	Medium
Amy	F	1.8m	Medium
Wynette	F	1.75m	Medium

Name	Gender	Height	Output1
Jim	M	2m	Tall
Bob	M	1.85m	Medium
Dave	M	1.7m	Short
Worth	M	2.2m	Tall
Steven	M	2.1m	Tall
Todd	M	1.95m	Medium

Para evaluar *gender*

H(9/15, 6/15)= 9/15 log (15/9) + 6/15 log (15/6) = 0.292

GainRatio (S,gender) = 0.09688/0.292 = 0.332

Para evaluar height

H(2/13, 2/13, 3/13, 4/13, 2/13) =

2/13 log (13/2) + 2/13 log (13/2) +

3/13 log (13/3) + 4/13 log (13/4) +

2/13 log (13/3)

Name	Gender	Height	Output1
Kristina	F	1	Short
Kathy	F	1	Short

Name	Gender	Height	Output1
Stephanie	F	2	Short
Dave	M	2	Short

Name	Gender	Height	Output1
Debbie	F	3	Medium
Amy	F	3	Medium
Wynette	F	3	Medium

Name	Gender	Height	Output1
Maggie	F	4	Medium
Martha	F	4	Medium
Bob	M	4	Medium
Kim	F	4	Medium

Name	Gender	Height	Output1
Jim	M	5	Tall
Todd	M	5	Medium

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

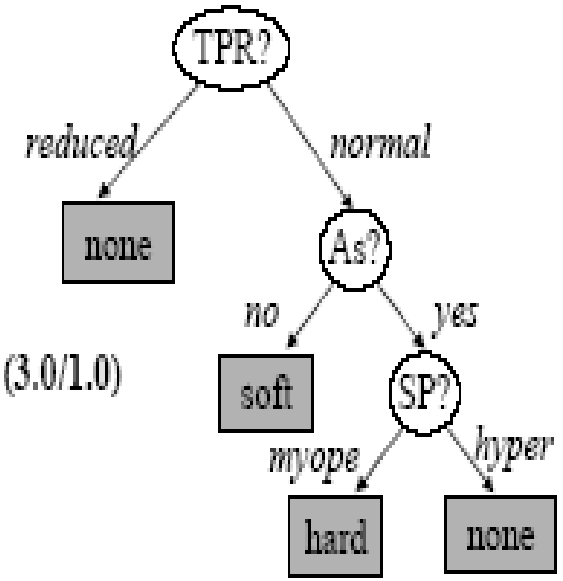
- Ejemplo de poda por número de ejemplos. Árbol original

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

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

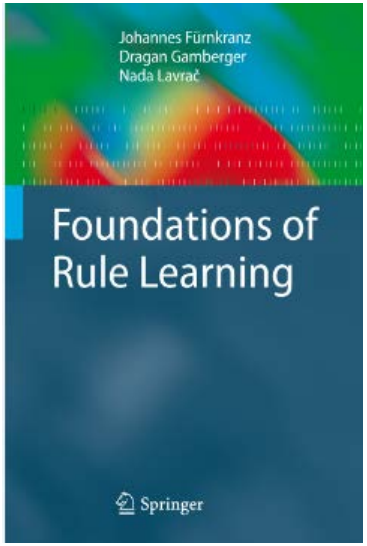
2 ejemplos por hoja

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



Clasificación con árboles y reglas

1. Árboles de decisión
2. **Clasificadores basados en reglas**
   - **2.1. Algoritmos de minería de datos basados en reglas**
   - 2.2. Generación de reglas mediante árboles de decisión
   - 2.3. Generación de reglas mediante cobertura



1. Algoritmos de minería de datos basados en reglas

- Una regla de clasificación está formada por
  - un antecedente, que contiene un predicado que se evaluará como verdadero o falso con respecto a cada ejemplo de la base de ejemplos
  - Un consecuente, que contiene una etiqueta de clase

- Se puede utilizar un árbol de decisión para generar reglas pero no son equivalentes:
  - El árbol tiene implícito un orden en el que se van comprobando los atributos. Con las reglas, en general, no hay orden
  - El árbol se crea mirando todas las clases. Cuando se genera una regla sólo se examina una clase

- Líneas principales en los algoritmos de minería de datos para la obtención de reglas:
  - Generación de reglas mediante árboles de decisión
  - Generación de reglas mediante cobertura

Clasificación con árboles y reglas

1. Árboles de decisión
2. **Clasificadores basados en reglas**
   - 2.1. Algoritmos de minería de datos basados en reglas
   - **2.2. Generación de reglas mediante árboles de decisión**
   - 2.3. Generación de reglas mediante cobertura

2. Generación de reglas mediante árboles de decisión

- Podemos extraer las reglas de un árbol de decisión y usarlas como un SBR (p.e. para el primer árbol de crédito):

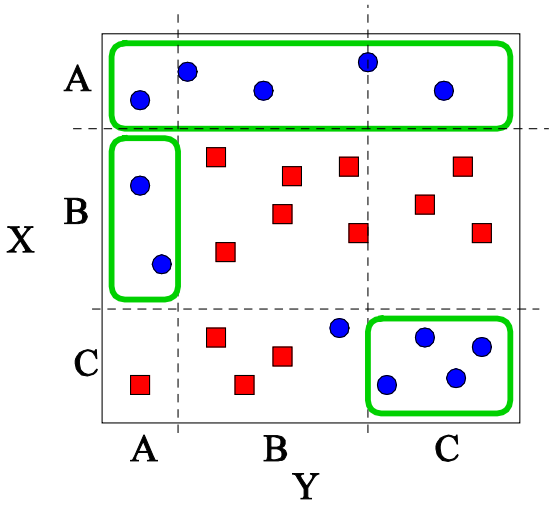
  - SI gastos=altos Y ingresos=bajos ENTONCES NO
  - SI gastos=altos Y ingresos=medios ENTONCES NO
  - Si gastos=altos Y ingresos=altos Y propietario=no ENTONCES NO
  - SI gastos=altos Y ingresos=altos Y propietario=si ENTONCES SI
  - SI gastos=bajos Y propietario=no ENTONCES NO
  - SI gastos=bajos Y propietario=si ENTONCES SI

- El algoritmo para obtener el SBR, para cada camino de la raíz del árbol a un nodo hoja, genera una regla y la añade al conjunto final (SBR)

2. Generación de reglas mediante árboles de decisión

Listas de decisión

- Las reglas están ordenadas
  - SI gastos=altos Y ingresos=altos y propietario=si ENTONCES SI
  - SI gastos=bajos y propietario=si ENTONCES SI EN OTRO CASO NO
- En estos clasificadores lo que se intenta es buscar las reglas que cubran el mayor número de ejemplos positivos (o el menor de ejemplos negativos)
- No se realiza un particionamiento exhaustivo del espacio de soluciones:



SI X=A ENTONCES círculo
SI X=B e Y=A ENTONCES círculo
Si X=C e Y=C ENTONCES círculo
EN OTRO CASO cuadrado

Clasificación con árboles y reglas

1. Árboles de decisión
2. **Clasificadores basados en reglas**
   - 2.1. Algoritmos de minería de datos basados en reglas
   - 2.2. Generación de reglas mediante árboles de decisión
   - **2.2. Generación de reglas mediante cobertura**

3. Generación de reglas mediante cobertura

- Los algoritmos de cobertura intentan generar reglas que cubran exactamente una clase
- Habitualmente generan la mejor regla posible optimizando la probabilidad de clasificación deseada
- A diferencia de los algoritmos de generación de árboles (seleccionan el mejor atributo), suelen elegir el mejor par (atributo,valor)

Ejemplo: Si se quiere generar una regla para clasificar personas altas, se busca una regla

Si ???? Entonces clase = alto

El objetivo para algoritmos de cobertura es reemplazar ??? por el predicado que obtenga la mejor probabilidad de que la clase sea alto

- Ejemplo básico de algoritmo de cobertura: 1R (genera un conjunto de reglas equivalente a un árbol de decisión de primer nivel). Elige el mejor atributo para realizar la clasificación basándose en datos de entrenamiento
- Otro ejemplo PRISM: genera reglas para cada clase mirando en el conjunto de entrenamiento y añadiendo reglas que describan todos los ejemplos de dicha clase

3. Generación de reglas mediante cobertura

Esquema algoritmo PRISM

```
R←Φ
Para cada valor c de clase hacer
    E←BD
    Mientras E contenga registros con clase c hacer
        Regla← CrearRegla (antecedentes(Φ),consecuente(c))
        RI ← Φ
        Hasta que Regla sea perfecta* hacer
            Para todo atributo X no incluido en la Regla hacer
                Para todo valor v de X hacer
                    RI ←RI U AñadirAntecedente(X=v)
            Regla ← SeleccionarMejorRegla (RI)**
        EC ← registros cubiertos por Regla
        R ← R U {Regla}
        E ← E – EC
```

* O no haya más atributos para utilizar

** Selecciona en base a la proporción de ejemplos positivos de dicha regla

3. Ejemplo: algoritmo basado en cobertura

- Aplicación sobre la BD crédito:

Crédito	Ingresos	Propietario	Gastos-mensuales
NO	Bajos	No	Altos
NO	Altos	No	Altos
NO	Bajos	No	Bajos
NO	Medios	No	Bajos
NO	Altos	No	Bajos
NO	Medios	No	Altos
NO	Bajos	Si	Altos
NO	Medios	Si	Altos
SI	Medios	Si	Bajos
SI	Altos	Si	Bajos

- Para seleccionar la mejor regla utilizaremos la exactitud s o el tanto por ciento de acierto

3. Ejemplo: algoritmo basado en cobertura

- Comenzamos con la clase crédito=si
  regla: Φ → crédito = SI
- En la primera iteración crearíamos las siguientes reglas en RI

antecedente	consecuente	s
Ingresos=altos	Si	1/3
Ingresos=medios	Si	1/4
Ingresos=bajos	Si	0/3
Propietario=si	Si	2/4
Propietario=no	Si	0/6
Gastos=altos	Si	0/5
Gastos=bajos	Si	2/5

- Seleccionamos la regla propietario=si → credito=SI
  Como NO es perfecta seguimos refinando

Crédito	Ingresos	Propietario	Gastos-mensuales
NO	Bajos	No	Altos
NO	Altos	No	Altos
NO	Bajos	No	Bajos
NO	Medios	No	Bajos
NO	Altos	No	Bajos
NO	Medios	No	Altos
NO	Bajos	Si	Altos
NO	Medios	Si	Altos
SI	Medios	Si	Bajos
SI	Altos	Si	Bajos

3. Ejemplo: algoritmo basado en cobertura

- Nuestra regla base ahora es: (regla: propietario=si → crédito=SI)
- Se añaden las siguientes reglas a RI:

antecedente	consecuente	s
propietario=si Y ingresos=altos	Si	1/1
propietario=si Y ingresos=medios	Si	1/2
propietario=si Y ingresos=bajos	Si	0/1
propietario=si Y gastos=altos	Si	0/2
propietario=si Y gastos=bajos	Si	2/2

Crédito	Ingresos	Propietario	Gastos-mensuales
NO	Bajos	No	Altos
NO	Altos	No	Altos
NO	Bajos	No	Bajos
NO	Medios	No	Bajos
NO	Altos	No	Bajos
NO	Medios	No	Altos
NO	Bajos	Si	Altos
NO	Medios	Si	Altos
SI	Medios	Si	Bajos
SI	Altos	Si	Bajos

- Seleccionamos la regla

propietario=si Y ingresos=altos
→ crédito=SI
Como es perfecta añadimos la regla a R

- Aún quedan casos con

clase SI sin tratar

Crédito	Ingresos	Propietario	Gastos-mensuales
NO	Bajos	No	Altos
NO	Altos	No	Altos
NO	Bajos	No	Bajos
NO	Medios	No	Bajos
NO	Altos	No	Bajos
NO	Medios	No	Altos
NO	Bajos	Si	Altos
NO	Medios	Si	Altos
SI	Medios	Si	Bajos

3. Ejemplo: algoritmo basado en cobertura

- Procedemos para crédito=SI (regla: Φ → crédito=si)
- En la primera iteración crearíamos las siguientes reglas en RI

antecedente	consecuente	s
Ingresos=altos	Si	0/2
Ingresos=medios	Si	1/4
Ingresos=bajos	Si	0/3
Propietario=si	Si	1/3
Propietario=no	Si	0/6
Gastos=altos	Si	0/5
Gastos=bajos	Si	1/4

Crédito	Ingresos	Propietario	Gastos-mensuales
NO	Bajos	No	Altos
NO	Altos	No	Altos
NO	Bajos	No	Bajos
NO	Medios	No	Bajos
NO	Altos	No	Bajos
NO	Medios	No	Altos
NO	Bajos	Si	Altos
NO	Medios	Si	Altos
SI	Medios	Si	Bajos

- Seleccionamos la regla propietario=si→credito=si

Como NO es perfecta seguimos refinando

3. Ejemplo: algoritmo basado en cobertura

- Regla base: (propietario=si → crédito=SI)
- Se añaden las siguientes reglas a RI:

Antecedente	consecuente	s
Propietario=si Y ingresos=altos	Si	0/0
Propietario=si Y ingresos=medios	Si	1/2
Propietario=si Y ingresos=bajos	Si	0/1
Propietario=si Y gastos=altos	Si	0/2
Propietario=Si Y gastos=bajos	si	1/1

Crédito	Ingresos	Propietario	Gastos-mensuales
NO	Bajos	No	Altos
NO	Altos	No	Altos
NO	Bajos	No	Bajos
NO	Medios	No	Bajos
NO	Altos	No	Bajos
NO	Medios	No	Altos
NO	Bajos	Si	Altos
NO	Medios	Si	Altos
SI	Medios	Si	Bajos

- Seleccionamos la regla propietario=si Y gastos=bajos → credito = SI. Como es perfecta añadimos la regla a R
- Ya no quedan casos con clase Si, por tanto, pasaríamos a la siguiente categoría

3. Ejemplo: algoritmo basado en cobertura

- Si seguimos iterando obtendremos el siguiente conjunto de reglas:

Si propietario = si & ingresos=altos	Crédito=SI
Si propietario = SI & gastos=bajos	Crédito=SI
Si propietario = NO	Crédito=NO
Si ingresos = bajos	Crédito=NO
Si ingresos = medios & gastos = altos	Crédito=NO

- Podríamos aplicar el algoritmo para todas las categorías de la variable clase menos una (la más probable a priori) y dejar ésta como regla por defecto:

Si propietario = si & ingresos=altos	Crédito=SI
Si propietario = SI & gastos=bajos	Crédito=SI
En otro caso	Crédito=NO

3. Ejemplo: algoritmo basado en cobertura

- Genera reglas para todas las categorías (puede modificarse para incluir regla por defecto)
- El conjunto óptimo de reglas puede no obtenerse, debido a que actúa de forma voraz incluyendo una condición en cada paso
- A no ser que haya inconsistencias en la BD consigue una clasificación perfecta → Sobreajusta a los datos

  Por ejemplo: si tenemos
  - Regla 1: con s=1/1, y
  - Regla 2: con s=19/20

  El algoritmo elegiría r1 cuando r2 es mucho más general

  Alternativa: utilizar otros criterios en lugar de la exactitud(s) como puede ser la medida F

$$F = \frac{2 \cdot VP}{2 \cdot VP + FP + FN}$$

- ¿Qué ocurre con los atributos numéricos? Condiciones del tipo {<,≤,≥,>}
- Existen algoritmos basados en cobertura/recubrimiento mucho más avanzados que el descrito

Inteligencia de Negocio

TEMA 3. Modelos de Predicción: Clasificación, regresión y series temporales

Clasificación

1. El problema de clasificación
2. Clasificación con árboles y reglas
3. **Clasificación con otras técnicas**
4. Multiclasificadores

Clasificación con otras técnicas

1. **Clasificadores basados en métodos bayesianos**
   - **1.1. Teorema de Bayes e hipótesis MAP**
   - 1.2. Clasificador Naïve Bayes
2. Clasificadores basados en instancias
3. Clasificadores basados en redes neuronales
4. Clasificadores basados en Máquina Soporte Vectorial

1. Clasificadores basados en métodos bayesianos

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

  → Una gran ventaja sobre otros métodos

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

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

    (P1, invertir) (P2, invertir)

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

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

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

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

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

1.1. Teorema de Bayes e hipótesis MAP

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

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

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

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

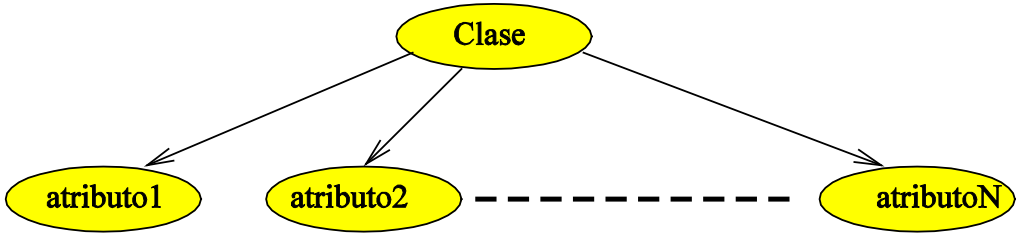
- Problema: Hay que trabajar con la distribución conjunta y eso normalmente es inmanejable

Clasificación con otras técnicas

1. **Clasificadores basados en métodos bayesianos**
   - 1.1. Teorema de Bayes e hipótesis MAP
   - **1.2. Clasificador Naïve Bayes**
2. Clasificadores basados en instancias
3. Clasificadores basados en redes neuronales
4. Clasificadores basados en Máquina Soporte Vectorial

1.2. Clasificador Naïve Bayes

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



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

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

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

1.2. Clasificador Naïve Bayes

¿Cómo se estiman estas probabilidades? Estimación de párametros

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

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

  - Suavizando por la corrección de Laplace:

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

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

1.2. Clasificador Naïve Bayes

Ejemplo:

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

- Estimación para *PLAY, outlook y windy*

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

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

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

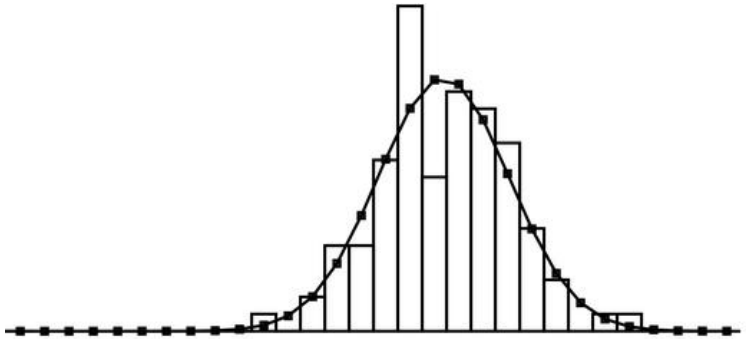
1.2. Clasificador Naïve Bayes

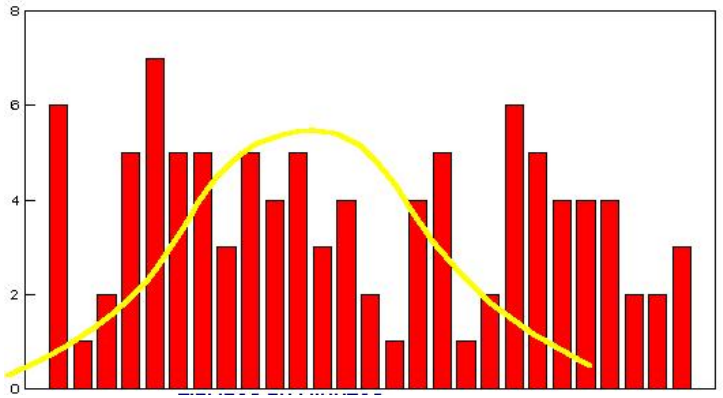
- Variables numéricas
  - $P(x|c_i)$ se estima mediante una función de densidad gaussiana. Es decir, se asume que los valores numéricos siguen una distribución normal

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

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

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





1.2. Clasificador Naïve Bayes

Ejemplo:

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

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

1.2. Clasificador Naïve Bayes

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

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

  - PLAY=yes

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

- Normalizando:

$P(NO) = 0.8$ $\qquad$ $P(YES) = 0.2$

1.2. Clasificador Naïve Bayes

Ejemplo de Naïve Bayes en WEKA

- Base de datos iris.arff (150 registros)
  - 4 atributos: sepalLength(numérico), sepalWidth (numérico), petalLength(numérico), petalWidth
  - Objetivo: a partir de las medidas anteriores, clasificar el objeto como un tipo de flor: iris-setosa, iris-versicolour, iris-virginica
  - Utilizamos Naïve Bayes (con opciones por defecto) en WEKA (10cv)

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

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

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

1.2. Clasificador Naïve Bayes

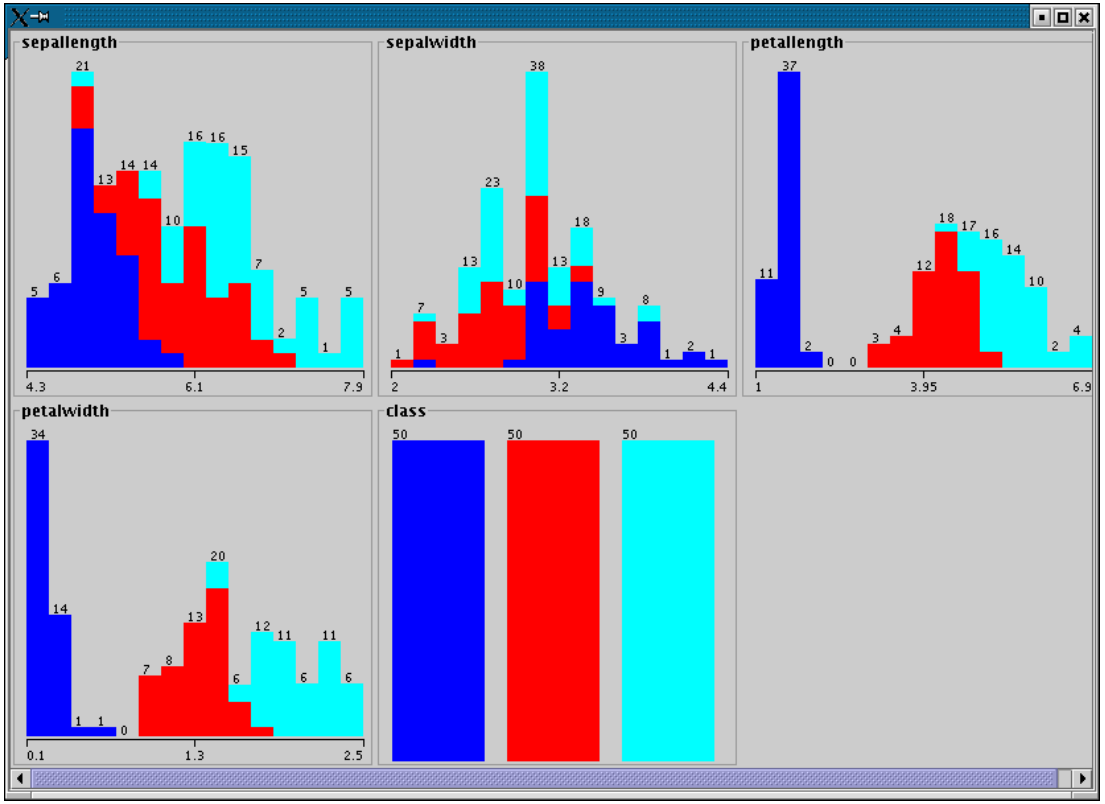
- Supongamos que discretizamos la BD anterior, utilizando la discretización en 2 intervalos de igual anchura

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

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

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

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

Correctly Classified Instances         116     77.3333%
```

1.2. Clasificador Naïve Bayes

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



1.2. Clasificador Naïve Bayes

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

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

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

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

Correctly Classified Instances 141 94 %

1.2. Clasificador Naïve Bayes

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

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

Clasificación con otras técnicas

1. Clasificadores basados en métodos bayesianos
   1.1. Teorema de Bayes e hipótesis MAP
   1.2. Clasificador Naïve Bayes

2. **Clasificadores basados en instancias**
3. Clasificadores basados en redes neuronales
4. Clasificadores basados en Máquina Soporte Vectorial

2. Clasificadores basados en instancias

- Están basados en el aprendizaje por analogía
- Se encuadran en el paradigma perezoso de aprendizaje, frente al voraz al que pertenecen los paradigmas anteriores
  - Perezoso: El trabajo se retrasa todo lo posible
    - No se construye ningún modelo, el modelo es la propia BD o conjunto de entrenamiento
    - Se trabaja cuando llega un nuevo caso a clasificar: Se buscan los casos más parecidos y la clasificación se construye en función de la clase a la que dichos casos pertenecen
- Los algoritmos más conocidos están basados en la regla del vecino más próximo

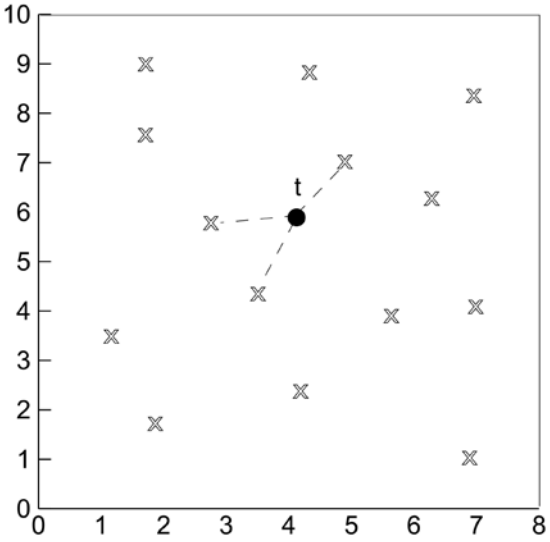
(ya presentado en la sección inicial)

2. Clasificadores basados en instancias

Regla del vecino más próximo o *Nearest neighbour* (1-NN)

- Si tenemos *m* instancias $\{e_1, \ldots, e_m\}$ en nuestra base de datos, para clasificar un nuevo ejemplo e' se hará lo siguiente:

1. c_{min} = clase (e_1)
2. d_{min} = d (e_1, e')
3. Para $i=2$ hasta m hacer
   $d=d(e_i,e')$
   Si $(d<d_{min})$
   Entonces c_{min} = clase (e_i), $d_{min} = d$
4. Devolver c_{min} como clasificación de e'



- d(·,·) es una función de distancia
- En el caso de variables nominales se utiliza la distancia de *Hamming:*

$$d_h(a,b) = \begin{cases} 0, & si\ a = b \\ 1, & si\ a \neq b \end{cases}$$

2. Clasificadores basados en instancias

Distancias para las variables numéricas

- Las variables numéricas se suelen normalizar al intervalo [0,1]
- Si e_i^j es el valor de la variable j en e_i, es decir $e_i=(e_i^1,\ldots,e_i^n)$ entonces algunas de las distancias más utilizadas son
  - Euclídea $$d_e(e_1,e_2)=\sqrt{\sum_{i=1}^{n}(e_1^i-e_2^i)}$$
  - Manhattan: $$d_m(e_1,e_2)=\sum_{i=1}^{n}\left|e_1^i-e_2^i\right|$$
  - Minkowski $$d_m^k(e_1,e_2)=\left(\sum_{i=1}^{n}\left|e_1^i-e_2^i\right|^k\right)^{1/k}$$

Como se puede observar, $d_m^1=d_m$ y $d_m^2=d_e$

2. Clasificadores basados en instancias

- Por tanto, la distancia entre dos instancias e_1 y e_2, utilizando p.e. d_e para las variables numéricas sería

$$d_e(e_1, e_2) = \sqrt{\sum_i (e_1^i - e_2^i)^2 + \sum_j d_h(e_1^j, e_2^j)}$$

siendo *i* el índice que se utiliza para recorrer las variables numéricas y *j* el índice que se utiliza para recorrer las variables nominales

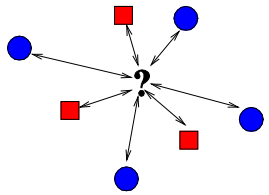
- Tratamiento de valores desconocidos: si ej_1=? Y ej_2=? Entonces la distancia asociada a la j-ésima componente es la máxima, es decir, 1

- Una crítica a esta regla es que todas las variables se consideran con igual importancia. La solución es asignar pesos a los atributos de forma que se pondere su importancia dentro del contexto

$$d_e(e_1, e_2) = \sqrt{\sum_i w_i \cdot (e_1^i - e_2^i)^2 + \sum_j w_j \cdot d_h(e_1^j, e_2^j)}$$

2. Clasificadores basados en instancias

- La extensión a la regla del vecino más próximo, es considerar los k vecinos más próximos: k-NN (NN = 1-NN)
- Funcionamiento: Dado *e* el ejemplo a clasificar
  1. Seleccionar los k ejemplos con $K = \{e_1,\ldots, e_k\}$ tal que, no existe ningún ejemplo *e'* fuera de *K* con $d(e,e') < d(e,e_i),\ i=1,\ldots,k$
  2. Devolver la clase que más se repite en el conjunto $\{clase(e_1), \ldots, clase(e_k)\}$ (la clase mayoritaria)
- Por ejemplo, si k=7 el siguiente caso (?) se clasificaría como ●



- Se podría tratar de forma diferente a los k-vecinos, p.e., dependiendo de la distancia al objeto a clasificar. De esta forma tendríamos: Clasificación
  - Voto por la mayoría ●
  - Voto con pesos en función de la distancia □

2. Clasificadores basados en instancias

- El algoritmo k-NN es robusto frente al ruido cuando se utilizan valores de k moderados (k>1)
- Es bastante eficaz, puesto que utiliza varias funciones lineales locales para aproximar la función objetivo
- Es válido para clasificación y para predicción numérica (devolviendo la media o la media ponderada por la distancia)
- Es muy ineficiente en memoria ya que hay que almacenar toda la BD
- La distancia entre vecinos podría estar dominada por variables irrelevantes
  - Selección previa de características
- Su complejidad temporal (para evaluar un ejemplo) es $O(dn^2)$ siendo $O(d)$ la complejidad de la distancia utilizada
  - Una forma de reducir esta complejidad es mediante el uso de prototipos
- El algoritmo *k*-NN está disponible en WEKA bajo el nombre de ibk. Permite voto por mayoría o voto ponderado por la distancia *(1/d y 1-d)*. No permite ponderar la variables

Clasificación con otras técnicas

1. Clasificadores basados en métodos bayesianos
   1.1. Teorema de Bayes e hipótesis MAP
   1.2. Clasificador Naïve Bayes
2. Clasificadores basados en instancias

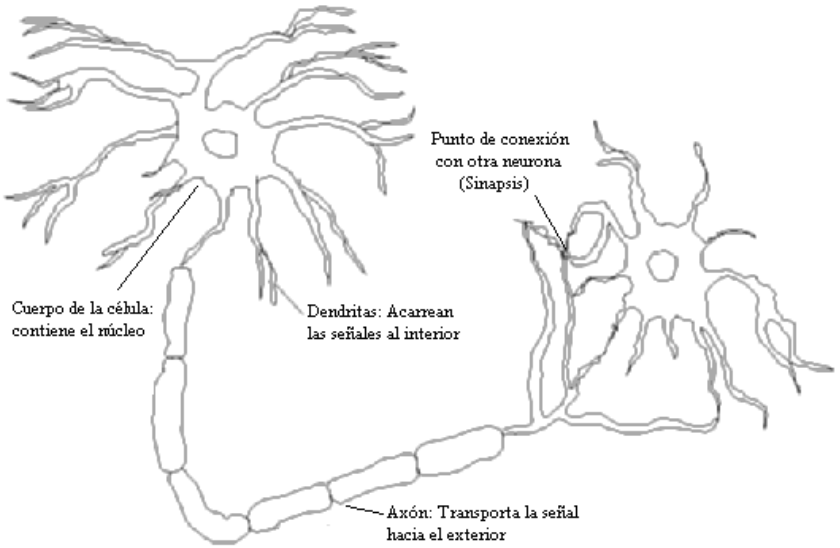
3. Clasificadores basados en redes neuronales

4. Clasificadores basados en Máquina Soporte Vectorial

3. Clasificadores basados en redes neuronales

Redes neuronales

- Surgieron como un intento de emulación de los sistemas nerviosos biológicos
- Actualmente: computación modular distribuida mediante la interconexión de una serie de procesadores (neuronas) elementales

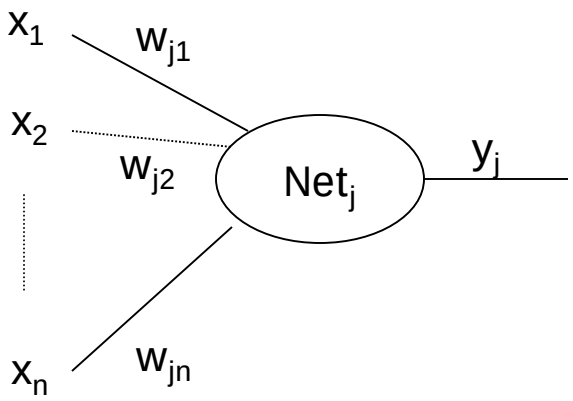


3. Clasificadores basados en redes neuronales

Redes neuronales

- Ventajas:
  - Habitualmente gran tasa de acierto en la predicción
  - Son más robustas que los árboles de decisión por los pesos
  - Robustez ante la presencia de errores (ruido, outliers,…)
  - Gran capacidad de salida: nominal, numérica, vectores,…
  - Eficiencia (rapidez) en la evaluación de nuevos casos
  - Mejoran su rendimiento mediante aprendizaje y éste puede continuar después de que se haya aplicado al conjunto de entrenamiento
- Desventajas:
  - Necesitan mucho tiempo para el entrenamiento
  - Entrenamiento: gran parte es ensayo y error
  - Poca (o ninguna) interpretabilidad del modelo (caja negra)
  - Difícil de incorporar conocimiento del dominio
  - Los atributos de entrada deben ser numéricos
  - Generar reglas a partir de redes neuronales no es inmediato
  - Pueden tener problemas de sobreaprendizaje
- ¿Cuándo utilizarlas?
  - Cuando la entrada tiene una dimensión alta
  - La salida es un valor discreto o real o un vector de valores
  - Posibilidad de datos con ruido
  - La forma de la función objetivo es desconocida
  - La interpretabilidad de los resultados no es importante

3. Clasificadores basados en redes neuronales



- Las señales que llegan a las dendritas se representan como $x_1, x_2, \ldots, x_n$
- Las conexiones sinápticas se representan por unos pesos w_{j1}, w_{j2}, w_{jn} que ponderan (multiplican) a las entradas. Si el peso entre las neuronas *j* e *i* es:
  a) positivo, representa una sinapsis excitadora
  b) negativo, representa una sinapsis inhibidora
  c) cero, no hay conexión

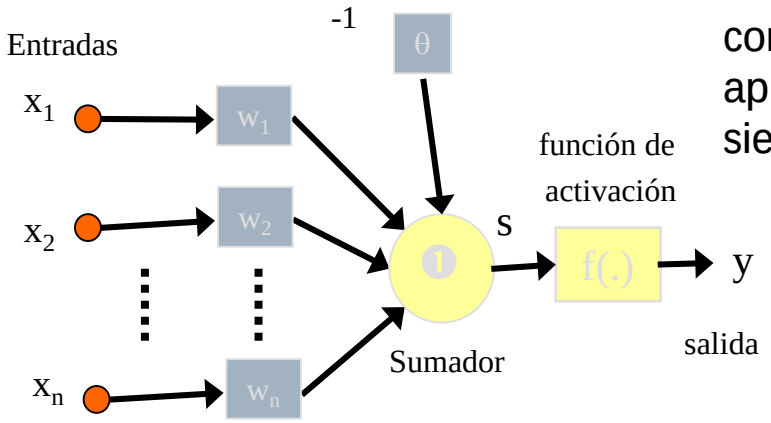
- La acción integradora del cuerpo celular (o actividad interna de cada célula) se presenta por

$$Net_j = w_{j1} \cdot x_1 + w_{j2} \cdot x_2 + \ldots + w_{jn} \cdot x_n = \sum_{i=1}^{n} w_{ji} \cdot x_i$$

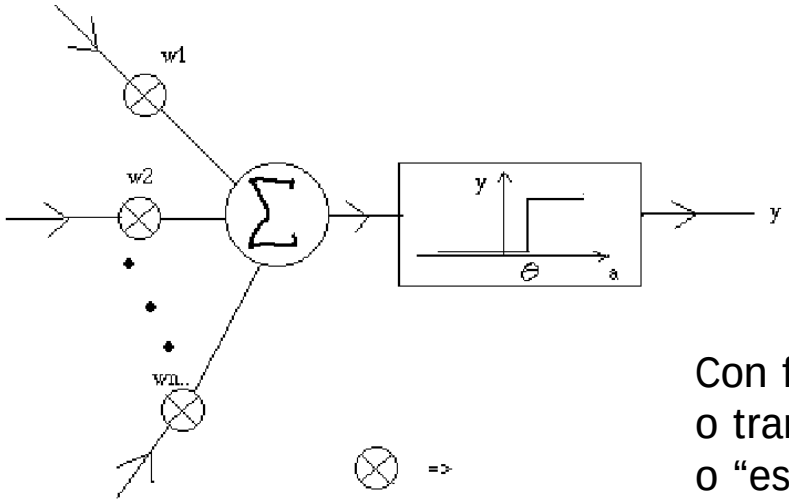
- La salida de la neurona se representa por y_j. Se obtiene mediante una función que, en general, se denomina función de salida, de transferencia o de activación. Esta función depende de Net_j y de un parámetro θ_j que representa el umbral de activación de la neurona

$$y_j = f(Net_j - \theta_j) = f\left(\sum w_{ji} \cdot x_i - \theta\right)$$

3. Clasificadores basados en redes neuronales



El umbral se puede interpretar como un peso sináptico que se aplica a una entrada que vale siempre -1



Con función de activación o transferencia tipo salto o “escalón”

3. Clasificadores basados en redes neuronales

- **Función de escalón**.
  Representa una neurona con sólo dos estados de activación: activada (1) y inhibida (0 ó -1)

$$y_j = H(Net_j - \theta_j) = \begin{cases} 1, & si\ Net_j >= \theta_j \\ -1, & si\ Net_j < \theta_j \end{cases}$$

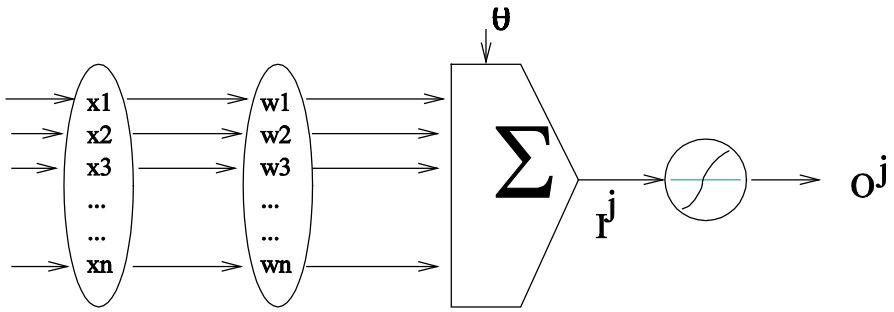
- **Función lineal**: $y_j = Net_j - \theta_j$

- **Función lineal a tramos**: $y_j = \begin{cases} 1, & si\ Net_j >= \theta_j + a \\ Net_j - \theta_j, & si\ |Net_j - \theta_j| < a \\ -1, & si\ Net_j < \theta_j - a \end{cases}$

- **Función sigmoidal**: $y_j = \dfrac{1}{1+e^{-\lambda(Net_j-\theta_j)}}$ $\qquad$ $y_j = \dfrac{2}{1+e^{-\lambda(Net_j-\theta_j)}} - 1$

- **Función base radial**: $y_j = e^{-\left(\frac{Net_j-\theta_j}{\sigma}\right)^2}$

3. Clasificadores basados en redes neuronales



- En definitiva, los componentes básicos de una red neuronal son:
  - Vector de pesos (w_{ij}), uno por cada entrada desde a la neurona *j*
  - Un sesgo (Θ_j) asociado a la neurona
  - Los datos de entrada $\{x_1,\ldots,x_n\}$ a una neurona N_j se hacen corresponder con un número real utilizando los componentes anteriores
  - Una función de activación f. Puede ser muy sencilla (función escalón) aunque normalmente es una función sigmoidal

3. Clasificadores basados en redes neuronales

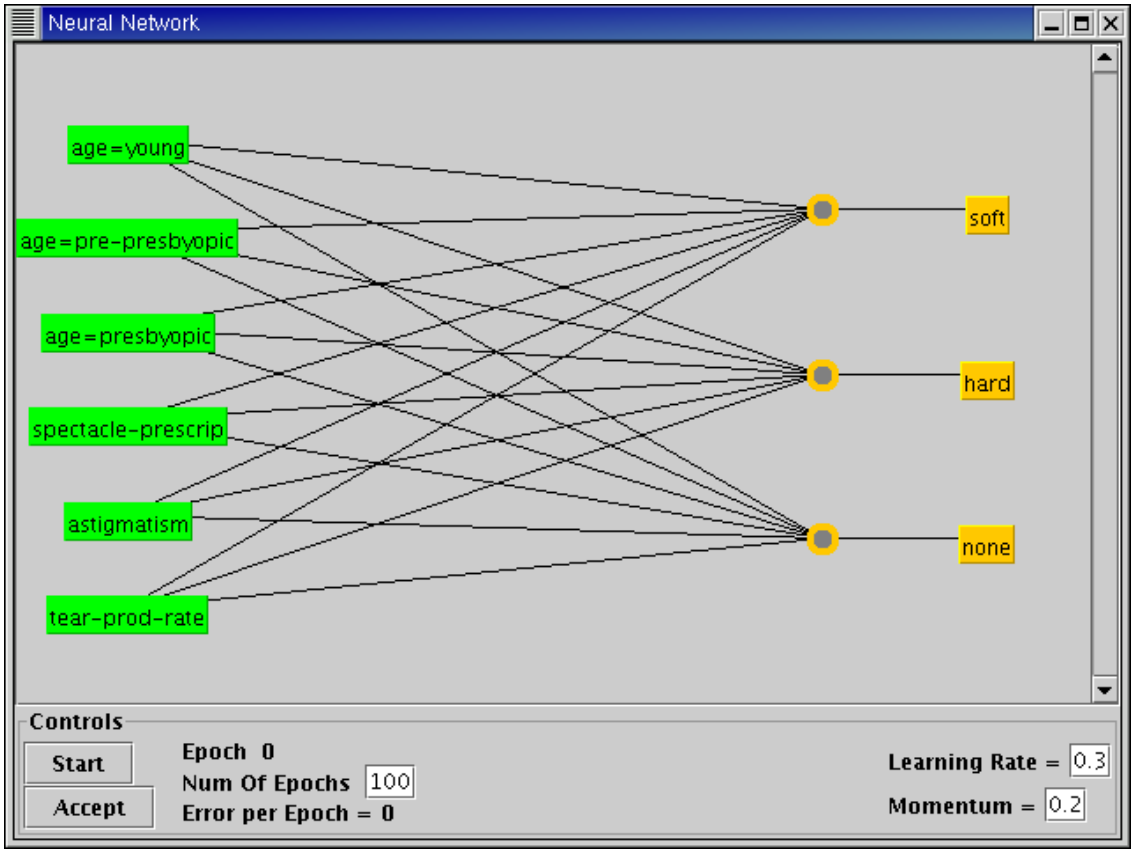
- Resolver un problema de clasificación con una red neuronal implica
  - Determinar (habitualmente con conocimiento experto)
    - el número de nodos de salida,
    - el número de nodos de entrada (y los atributos correspondientes,
    - el número de capas ocultas
  - Determinar pesos y funciones a utilizar
  - Para cada tupla en el conjunto de entrenamiento propagarlo en la red y evaluar la predicción de salida con el resultado actual.
    - Si la predicción es precisa, ajustar los pesos para asegurar que esa predicción tendrá un peso de salida más alto la siguiente vez
    - Si la predicción no es precisa, ajustar los pesos para que la siguiente ocasión se obtenga un valor menor para dicha clase
  - Para cada tupla en el conjunto de test propagarla por la red para realizar la clasificación

3. Clasificadores basados en redes neuronales

- El aprendizaje de la estructura de una red neuronal requiere experiencia, aunque existen algunas guías
- Entradas: Por cada variable numérica o binaria/booleana se pone una neurona de entrada. Por cada variable nominal (con más de dos estados) se pone una neurona de entrada por cada estado posible.
- Salidas: Si es para predicción se pone una única neurona de salida por cada valor de la variable clase
- Capas ocultas. Hay que indicar cuántas capas y cuantas neuronas hay que poner en cada capa. En Weka,
  - 0: No se pone capa oculta (¡sólo particiones lineales!)
  - Números enteros separados por comas: cada número representa la cantidad de neuronas de esa capa. P.e. 4,3,5: representa una red con tres capas ocultas de 4, 3 y 5 neuronas respectivamente
  - Algunos comodines:
    - i/o: neuronas en la entrada/salida
    - t: i+o
    - a: t/2 (es el valor por defecto)
- Cuando la red neuronal está entrenada, para clasificar una instancia, se introducen los valores de la misma que corresponden a las variables de los nodos de entrada.
  - La salida de cada nodo de salida indica la probabilidad de que la instancia pertenezca a esa clase
  - La instancia se asigna a la clase con mayor probabilidad

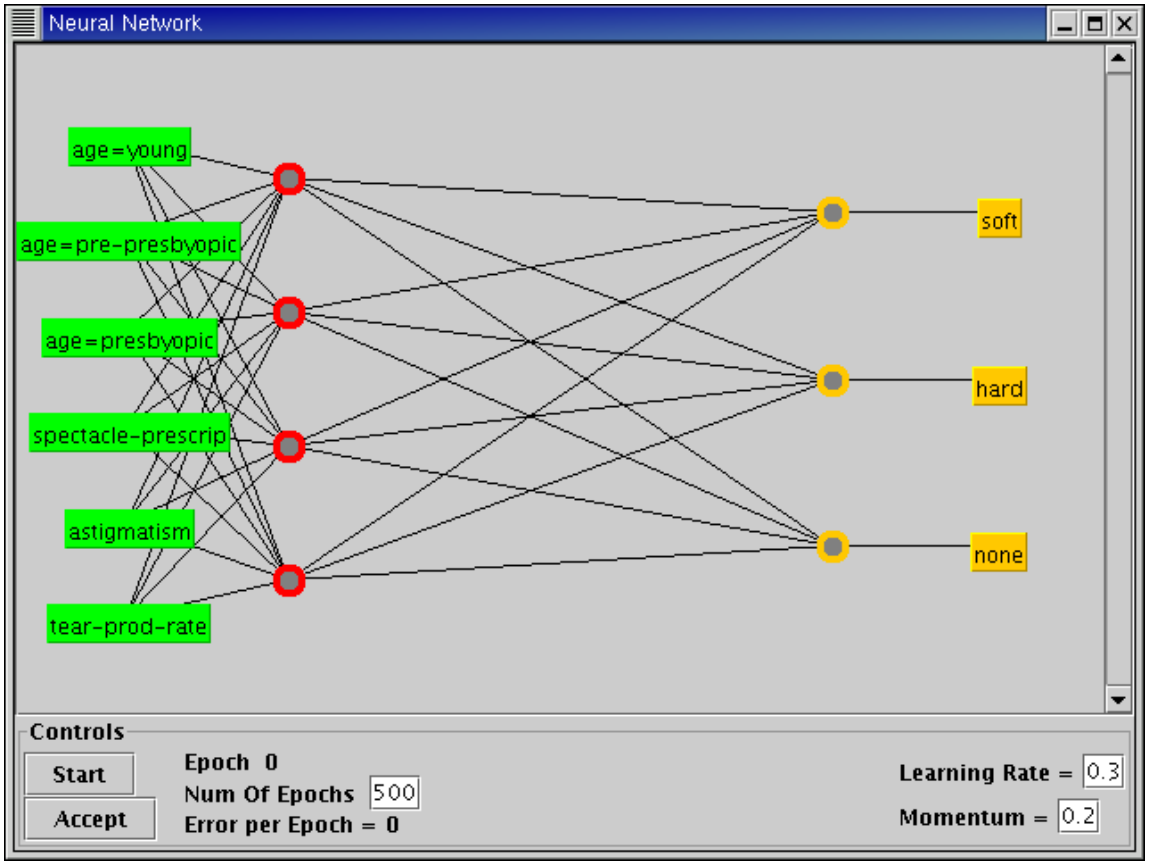
3. Clasificadores basados en redes neuronales

- Ejemplos con la BD lentes
  - Capa oculta = 0



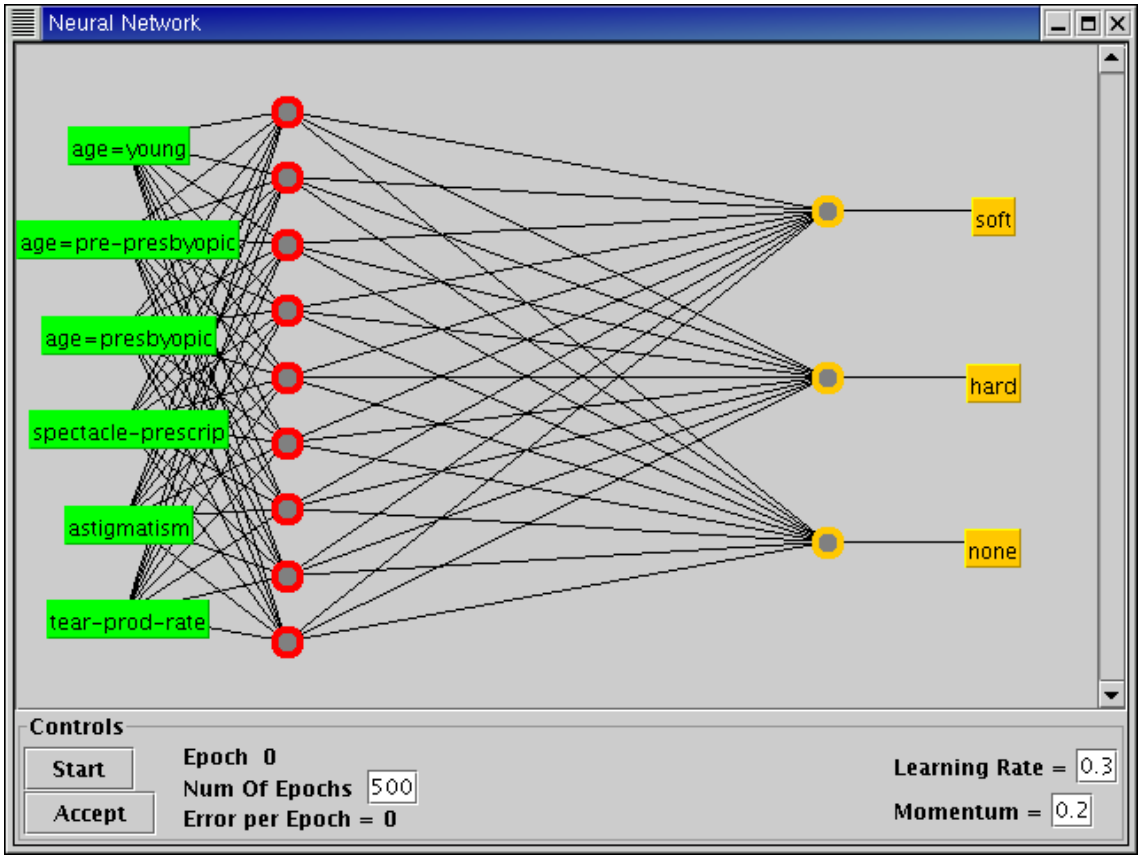
3. Clasificadores basados en redes neuronales

- Ejemplos con la BD lentes
  - Capa oculta = a



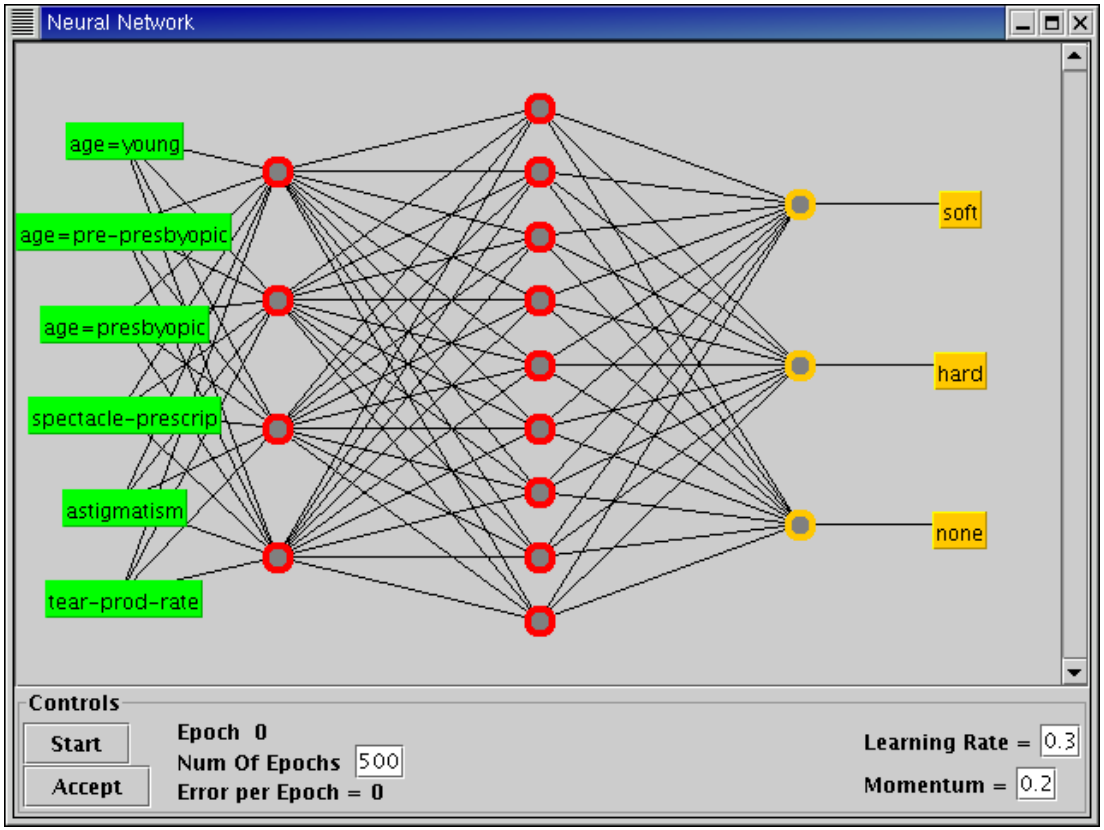
3. Clasificadores basados en redes neuronales

- Ejemplos con la BD lentes
  - Capa oculta = t



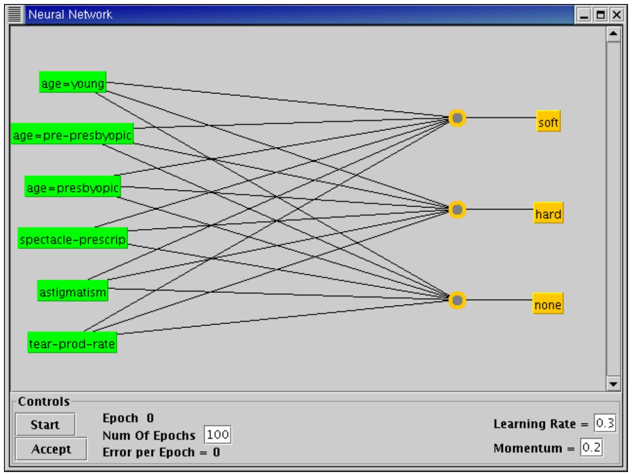
3. Clasificadores basados en redes neuronales

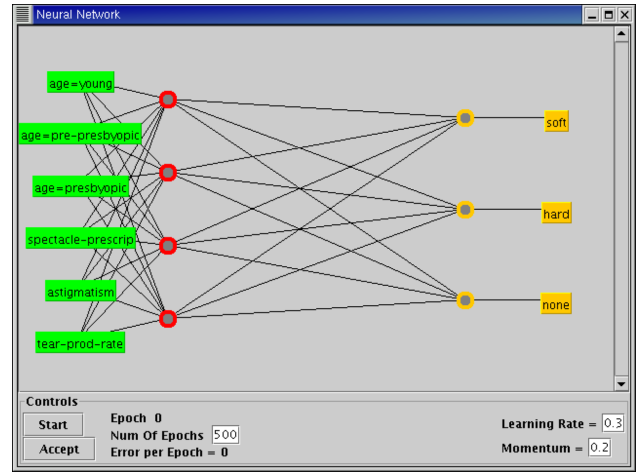
- Ejemplos con la BD lentes
  - Capa oculta = a, t

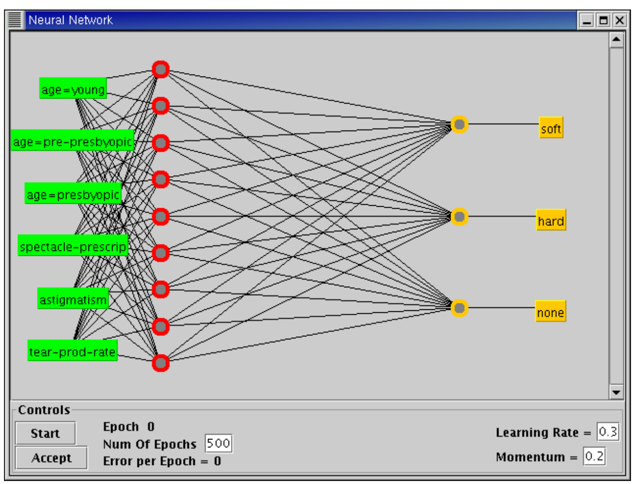


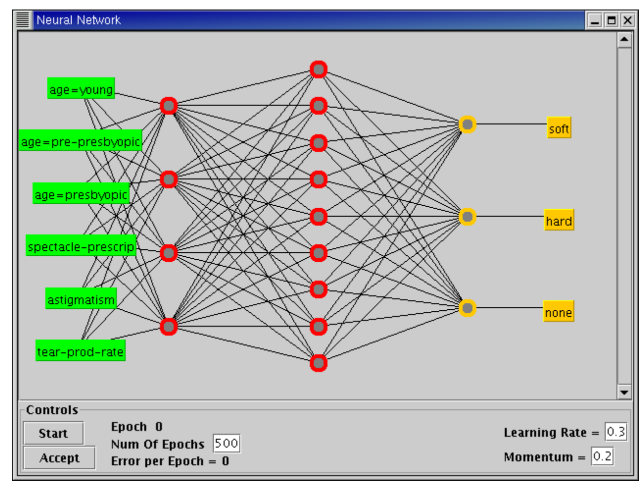
3. Clasificadores basados en redes neuronales

- Ejemplos con la BD lentes









3. Clasificadores basados en redes neuronales

- Todas las variables numéricas se normalizan [-1,1]

- Algoritmo de *Backprogation* o retropropagación:
  - Basado en la técnica del gradiente descendiente
  - No permite conexiones hacia atrás (retroalimentación) en la red

- Esquema del algoritmo de retropropagación
  1. Inicializar los pesos y sesgos aleatoriamente
  2. Para r=1 hasta número de *epoch* hacer
     a) Para cada ejemplo *e* de la BD hacer
     b) Lanzar un proceso *forward* de propagación en la red neuronal para obtener la salida asociada a *e* usando las expresiones vistas anteriormente
     c) Almacenar el valor O_j producido en cada neurona N_j
     d) Lanzar un proceso *backward* para recalcular los pesos y los sesgos asociados a cada neurona

  1 *epoch* = procesamiento de todos los ejemplos de la BD

3. Clasificadores basados en redes neuronales

- Fase de retropagación:
  - Para cada neurona de salida N_s hacer BP(N_s)
  - BP(N_j):
    1. Si N_j es una neurona de entrada, finalizar
    2. Si N_j es una neurona de salida
       entonces $Err_j = O_j \cdot (1-O_j) \cdot (T_j - O_j)$
       si no $Err_j = O_j(1-O_j) \sum_{k \text{ de salida}} Err_k \cdot w_{jk}$
       con T_j el valor predicho en la neurona N_j
    3. Actualizar los pesos $w_{ij} = w_{ij} + \alpha \cdot Err_j \cdot O_i$
    4. Actualizar los sesgos: $\Theta_j = \Theta_j + a \cdot Err_j$
    5. Para cada neurona N_i tal que $N_i \rightarrow N_j$ hacer BP(N_i)

3. Clasificadores basados en redes neuronales

En definitiva,

- Inicializar todos los pesos a números aleatorios pequeños
- Hasta que se verifique la condición de parada hacer
  - Para cada ejemplo de entrenamiento hacer
    - Introducir el ejemplo en la red y propagarla para obtener la salida (O_k)
    - Para cada unidad de salida k
      - $Error_k \leftarrow O_k(1-O_k)(t_k-O_k)$
    - Para cada unidad oculta
      - $Error_k \leftarrow O_k(1-O_k)\sum_j salida\ w_{k,j}\ error_j$
    - Modificar cada peso de la red
      - $w_{i,j} \leftarrow w_{i,j} + \Delta w_{ij}$
      - $\Delta w_{i,j} = \eta\ Error_j\ x_{i,j}$

3. Clasificadores basados en redes neuronales

- Existen muchos otros modelos de redes neuronales (recurrentes, memorias asociativas, redes neuronales de base radial, redes ART, ...)

- El término de Deep Learning (procesamiento masivo de datos que utiliza redes neuronales) ha centrado la atención en modelos escalables de redes neuronales

- Google cuenta, desde hace cinco años con el departamento conocido como ' **Google Brain'** (dirigido por Andrew Ng, Univ. Stanford, responsable de un curso de coursera de machine learning) dedicado a esta técnica entre otras. En 2013, Google adquirió la compañía DNNresearch Inc de uno de los padres del Deep Learning (Geoffrey Hinton). En enero de 2014 se hizo con el control de la 'startup' Deepmind Technologies una pequeña empresa londinense en la trabajaban algunos de los mayores expertos en 'deep learning'.

Clasificación con otras técnicas

1. Clasificadores basados en métodos bayesianos
   1.1. Teorema de Bayes e hipótesis MAP
   1.2. Clasificador Naïve Bayes
2. Clasificadores basados en instancias

3. Clasificadores basados en redes neuronales
4. **Clasificadores basados en Máquina Soporte Vectorial (SVM)**

4. Clasificadores basados en máquinas de soporte vectorial (SVM)

"Una **SVM(support vector machine)** es un modelo de aprendizaje que se fundamenta en la *Teoría de Aprendizaje Estadístico*. La idea básica es encontrar un hiperplano canónico que maximice el margen del conjunto de datos de entrenamiento, esto nos garantiza una buena capacidad de generalización."

Representación dual de un problema

+

Funciones Kernel

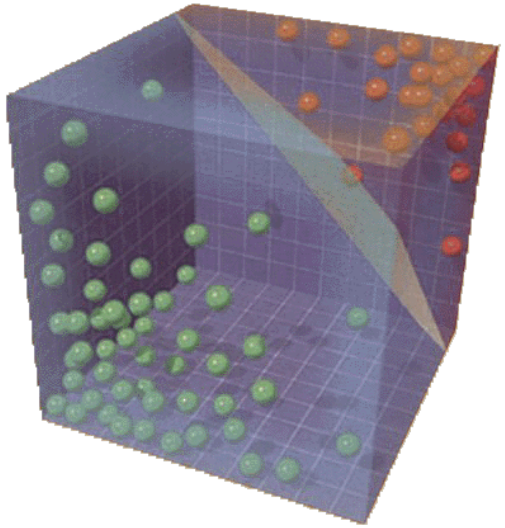
+

Teoría de Aprendizaje Estadística

+

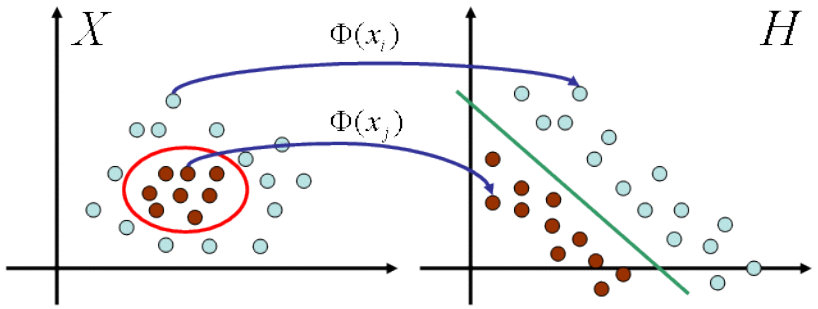
Teoría Optimización Lagrange

=



4. Clasificadores basados en máquinas de soporte vectorial (SVM)

- Una **SVM** es un **máquina de aprendizaje lineal** (requiere que los datos sean linealmente separables).
- Estos métodos explotan la información que proporciona el **producto interno (escalar)** entre los datos disponibles.
- La idea básica es:



4. Clasificadores basados en máquinas de soporte vectorial (SVM)

- **Problemas de esta aproximación:**
  - ¿Cómo encontrar la función ϕ?
  - El espacio de características inducido **H** es de **alta dimensión.**
    - **Problemas de cómputo y memoria.**

4. Clasificadores basados en máquinas de soporte vectorial (SVM)

- **Solución:**
  - Uso de funciones kernel.
  - **Función kernel =** producto interno de dos elementos en algún espacio de características inducido (**potencialmente de gran dimensionalidad**).
  - Si usamos una función kernel no hay necesidad de especificar la función ϕ**.**

4. Clasificadores basados en máquinas de soporte vectorial (SVM)

- Polinomial: $K(x,y) = \langle x,y \rangle^d$
- Gausiano: $K(x,y) = e^{-\|x-y\|^2 / 2\sigma}$
- Sigmoide: $K(x,y) = \tanh(\alpha \langle x,y \rangle + \beta)$

4. Clasificadores basados en máquinas de soporte vectorial (SVM)

- Para resolver el problema de optimización planteado se usa la **teoría de Lagrange**.

Minimizar

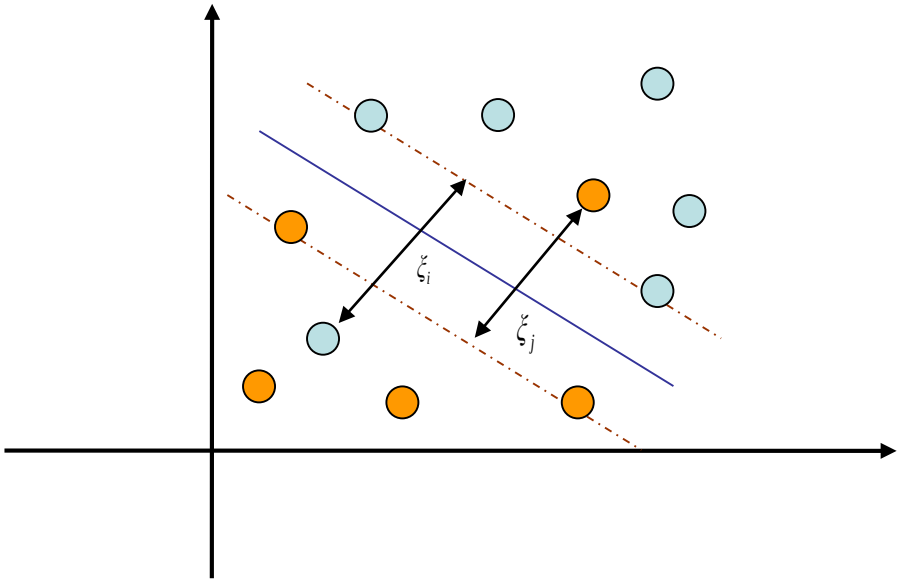
$$L(w,b) = \frac{1}{2}\langle w,w\rangle - \sum_{i=1}^{l} \alpha_i \left[y_i \left(\langle w, x_i \rangle + b \right) - 1 \right]$$

Condicionado a

$$\alpha_i \geq 0$$

- Cada una de las variables α_i es un multiplicador de Lagrange y existe una variable por cada uno de los datos de entrada.

4. Clasificadores basados en máquinas de soporte vectorial (SVM)



$$y_i\left[\left\langle w, x_i\right\rangle + b\right] \geq 1 - \xi_i$$

4. Clasificadores basados en máquinas de soporte vectorial (SVM)

- Originariamente el **modelo de aprendizaje basado en SVMs** fue diseñado para **problemas de clasificación binaria**.

- La extensión a multi-clases se realiza mediante combinación de clasificadores binarios (se estudia en la siguiente sección, modelos One vs One).

Inteligencia de Negocio

TEMA 3. Modelos de Predicción: Clasificación, regresión y series temporales

Clasificación

1. El problema de clasificación
2. Clasificación con árboles y reglas
3. Clasificación con otras técnicas
4. **Multiclasificadores**

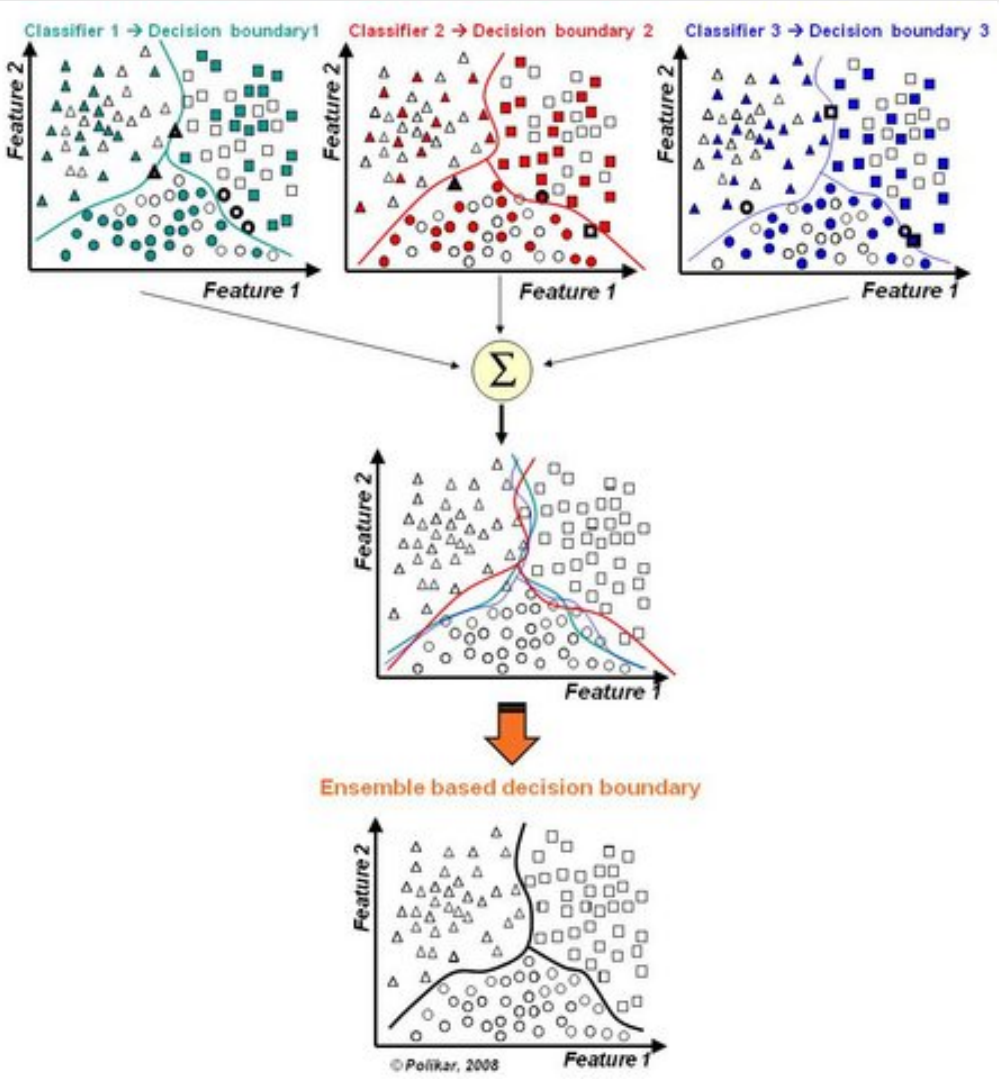
Multiclasificadores

1. **Combinación de clasificadores**
2. Bagging
3. Boosting
4. Descomposicion de Problemas de Multiclase. Binarización (Pairwise learning)

1. Combinación de clasificadores

- La idea es inducir *n* clasificadores en lugar de uno solo
- Para clasificar se utilizará una combinación de la salida que proporciona cada clasificador
- Los clasificadores pueden estar basados en distintas técnicas (p.e. árboles, reglas, instancias,...)
- Se puede aplicar sobre el mismo clasificador o con diferentes.
- Modelos específicos:
  - *Bagging*: cada clasificador se induce independientemente incluyendo una fase de diversificación sobre los datos
  - *Boosting*: cada clasificador tiene en cuenta los fallos del anterior

1. Combinación de clasificadores



Multiclasificadores

1. Combinación de clasificadores
2. **Bagging**
3. Boosting
4. Descomposicion de Problemas de Multiclase. Binarización (Pairwise learning)

2. Bagging

- Bagging definido a partir de "**b**ootstrap **agg**regat**ing**".

- Un método para combinar múltiples predictores/clasificadores.

- Bagging funciona bien para los algoritmos de aprendizaje "inestables", aquellos para los que un pequeño cambio en el conjunto de entrenamiento puede provocar grandes cambios en la predicción (redes neuronales, árboles de decisión, ...) (No es el caso de k-NN que es un algoritmo estable)

2. Bagging

Bagging.[Breiman,94] Repeat for $t = 1, \ldots, T$:

- Select, at random *with replacement*, N training examples.
- Train learner on selected samples to generate h_t

Final hypothesis is simple vote:

$$H(x) = MAJ(h_1(\mathbf{x}), \ldots, h_T(\mathbf{x}))$$

2. Bagging

- Fase 1: Generación de modelos
  1. Sea *n* el número de ejemplos en la BD y *m* el número de los modelos a utilizar
  2. Para *i*=1,…,*m* hacer
     - Muestrear con reposición *n* ejemplos de la BD (Boosting)
     - Aprender un modelo con ese conjunto de entrenamiento
     - Almacenarlo en modelos[*i*]

- Fase 2: Clasificación
  1. Para i=1,…,m hacer
     - Predecir la clase utilizando modelos[i]
  2. Devolver la clase predicha con mayor frecuencia

2. Bagging

Algorithm 1 Bagging

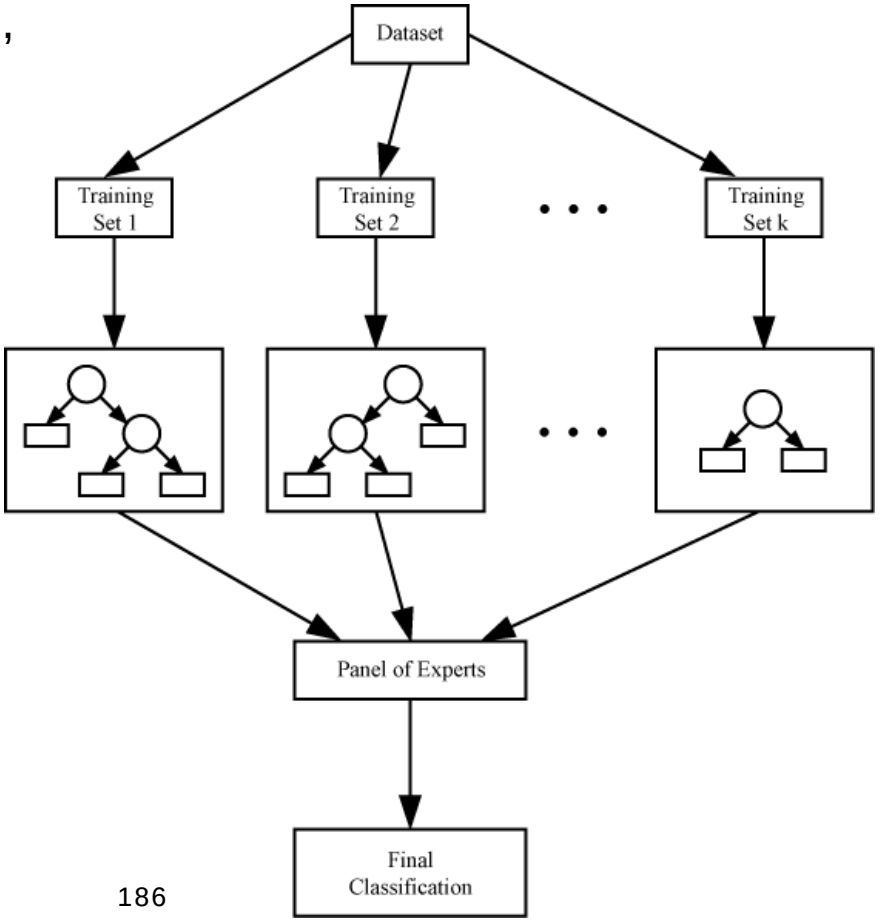
Input: S: Training set; T: Number of iterations;
n: Bootstrap size; I: Weak learner

Output: Bagged classifier: $H(x) = sign\left(\sum_{t=1}^{T} h_t(x)\right)$ where $h_t \in [-1, 1]$ are the induced classifiers

1: **for** $t = 1$ to T **do**
2: $S_t \leftarrow$ RandomSampleReplacement(n,S)
3: $h_t \leftarrow$ I(S_t)
4: **end for**

2. Bagging

Random Forest (Leo Breiman, Adele Cutler): Boostraping (muestreo con reemplazamiento, 66%), Selección aleatoria de un conjunto muy pequeño de variables (m<<M) (ej. log M + 1) para elegir entre ellas el atributo que construye el árbol (medida de Gini), sin poda (combinación de clasificadores débiles)



Breiman, Leo (2001). «Random Forests». *Machine Learning* **45** (1): pp. 5–32. doi:10.1023/A:1010933404324

Multiclasificadores

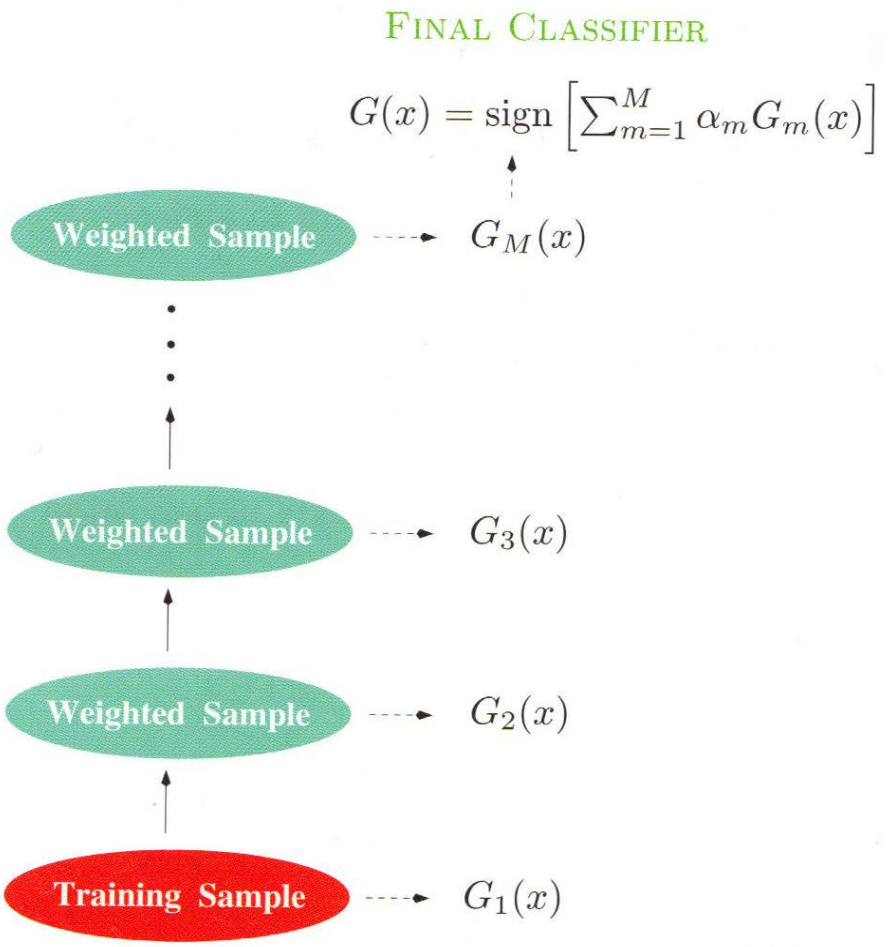
1. Combinación de clasificadores
2. Bagging
3. **Boosting**
4. Descomposicion de Problemas de Multiclase. Binarización (Pairwise learning)

3. Boosting

- **Muestreo ponderado (ejemplos):**
  - En lugar de hacer un muestreo aleatorio de los datos de entrenamiento, se ponderan las muestras para concentrar el aprendizaje en los ejemplos más difíciles.
  - Intuitivamente, los ejemplos más cercanos a la frontera de decisión son más difíciles de clasificar, y recibirán pesos más altos.
- **Votos ponderados (clasificadores):**
  - En lugar de combinar los clasificadores con el mismo peso en el voto, se usa un voto ponderado.
  - Esta es la regla de combinación para el conjunto de clasificadores débiles.
  - En conjunción con la estrategia de muestreo anterior, esto produce un clasificador más fuerte.

3. Boosting

Idea



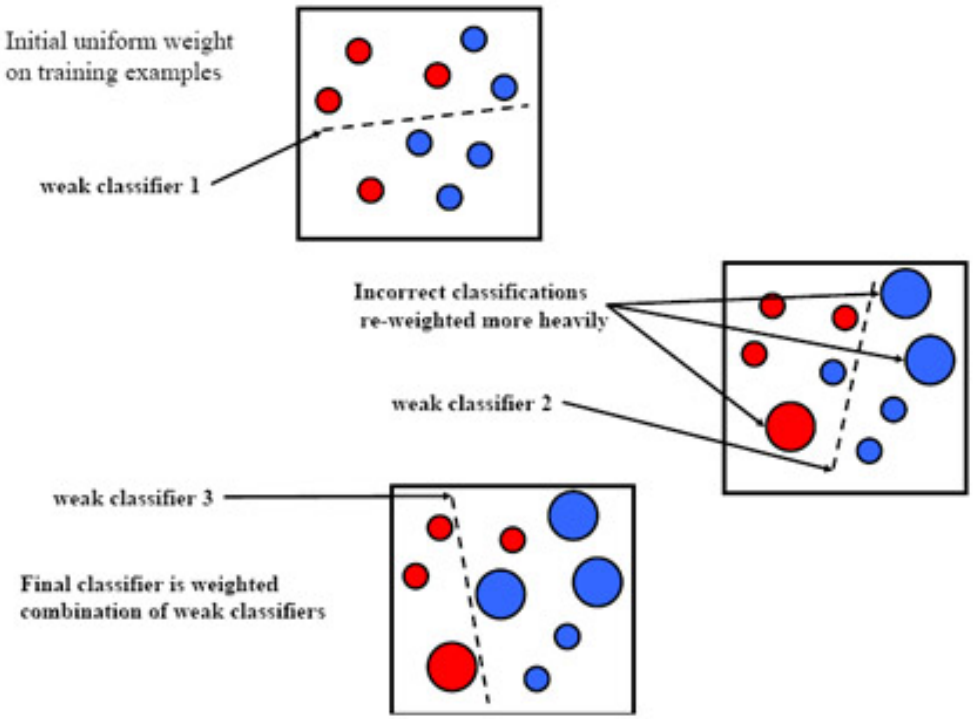
3. Boosting

- Los modelos no se construyen independientemente, sino que el modelo i-ésimo está influenciado por los anteriores
- La idea es prestar más atención a los ejemplos mal clasificados por los modelos anteriores
- Fase 1: Generación de modelos
  1. Asignar a cada ejemplo el mismo peso ($1/n$)
  2. Para $i=1,\ldots,m$ hacer
     - Aprender un modelo a partir de la BD con pesos
     - Almacenarlo en modelos[i]
     - Calcular el error ε_i sobre el conjunto de ejemplos
     - Si $\varepsilon_i=0$ o $\varepsilon_i>=0.5$ terminar
     - Para cada ejemplo bien clasificado multiplicar su peso por $\varepsilon_i/(1-\varepsilon_i)$
     - Normalizar los pesos de todos los ejemplos
- Fase 2: Clasificación
  1. Asignar peso cero a todas las categorías de la variable clase
  2. Para $i=1,\ldots,m$ hacer
     - Sumar $-log(\varepsilon_i/(1-\varepsilon_i))$ a la categoría predicha por modelos[i]
  3. Devolver la categoría con mayor peso

3. Boosting

AdaBoost, abreviatura de "Adaptive Boosting", es un algoritmo de aprendizaje meta-algoritmo formulado por Yoav Freund y Robert Schapire que ganó el prestigioso "Premio Gödel" en 2003 por su trabajo. Se puede utilizar en conjunción con muchos otros tipos de algoritmos de aprendizaje para mejorar su rendimiento.

La salida de los otros algoritmos de aprendizaje ("algoritmos débiles") se combina en una suma ponderada que representa la salida final del clasificador impulsado.



$$H(x) = sign(\alpha_1 h_1(x) + \alpha_2 h_2(x) + \alpha_3 h_3(x))$$

https://en.wikipedia.org/wiki/G%C3%B6del_Prize

3. Boosting

Algorithm 1: Adaboost Algorithm (Freund and Schapire)

Input : A weak learning algorithm $WeakLearn$, an integer T specifying number of iterations, and N labelled training data $\{(x_1, y_1), \ldots, (x_N, y_N)\}$.

Output : A strong classifier F.

Initialize the weight vector $w_i^1 = \frac{1}{N}$, *for* $i = 1, \ldots, N$.

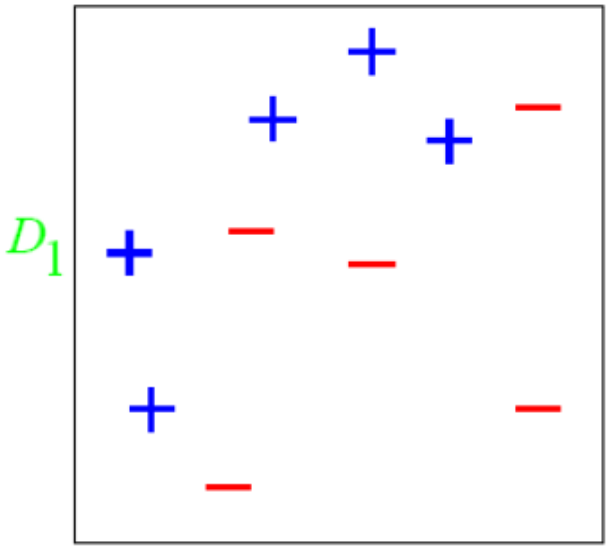
for $t \leftarrow 1, 2, \ldots, T$ **do**

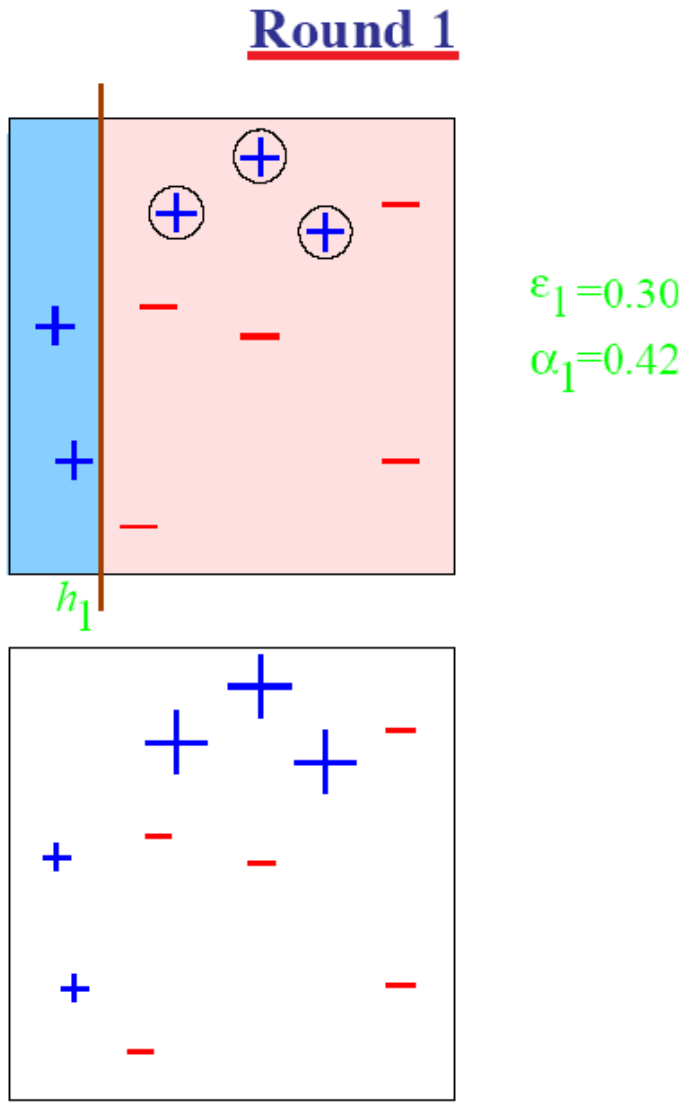
1. $\mathbf{p}^t \leftarrow \mathbf{w}^t / \sum_{i=1}^{N} w_i^t$.
2. Call $WeakLearn$, providing it with the distribution on $\mathbf{p}^t$; get back a weak learner $h_t : X \rightarrow \pm 1$.
3. Calculate the weight error of h_t: $\epsilon_t = \sum_{i=1}^{N} p_i^t \frac{1}{2} |h_t(x_i) - y_i|$.
4. $\alpha_t \leftarrow \log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$.
5. $w_i^{t+1} \leftarrow w_i^t \exp\left(\alpha_t \frac{1}{2} |h_t(x_i) - y_i|\right)$, for $i = 1, 2, \ldots, T$.

Output the final strong classifier:

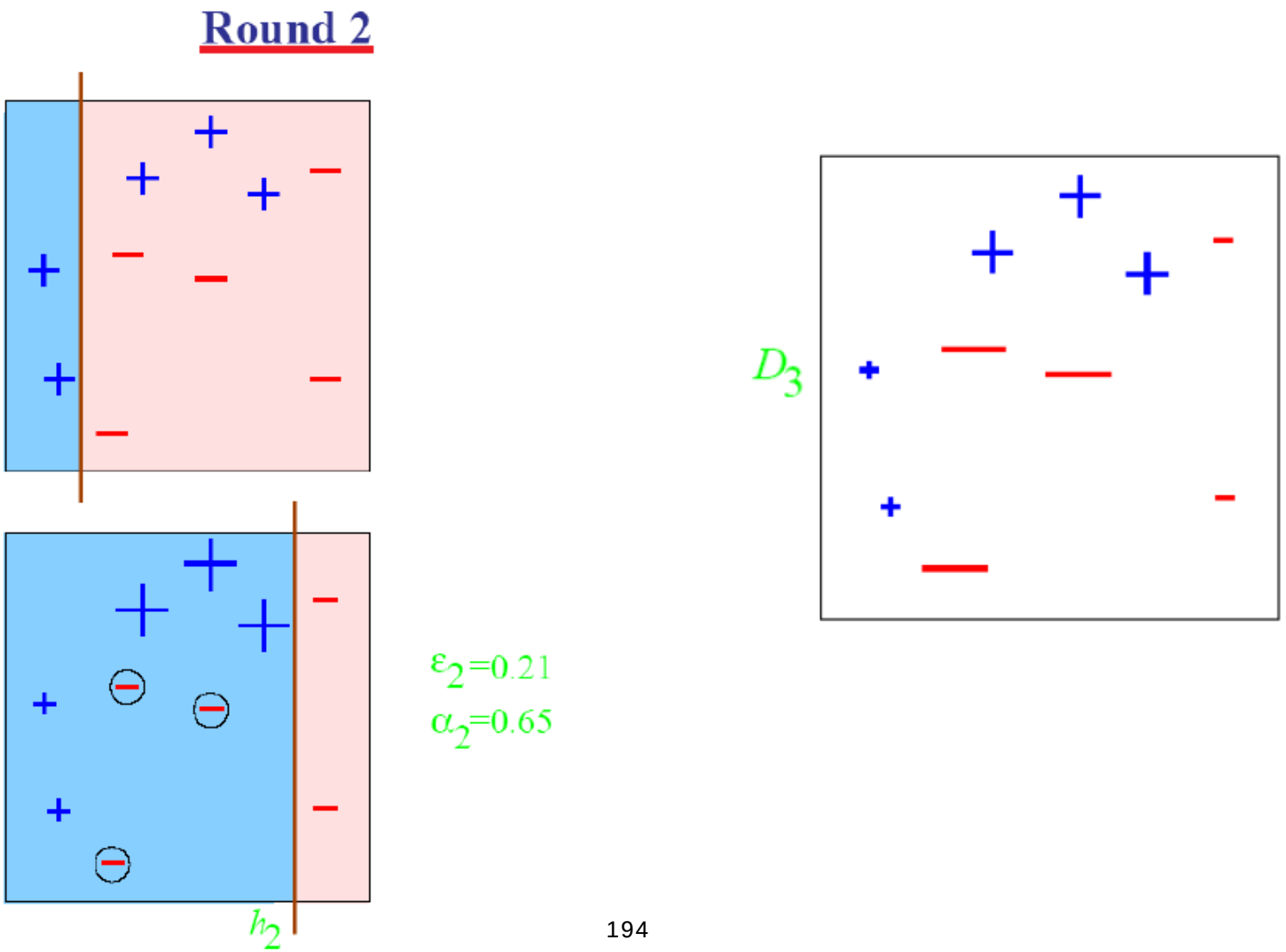
$$F(x) = \begin{cases} 1, & \text{if } \sum_{i=1}^{T} \alpha_t h_t(x) \geq 0, \\ -1, & \text{otherwise.} \end{cases}$$

3. Boosting. Ejemplo

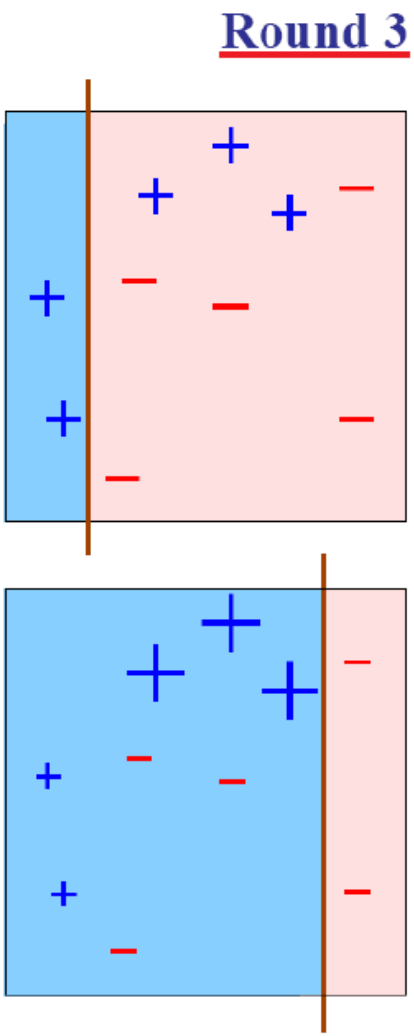


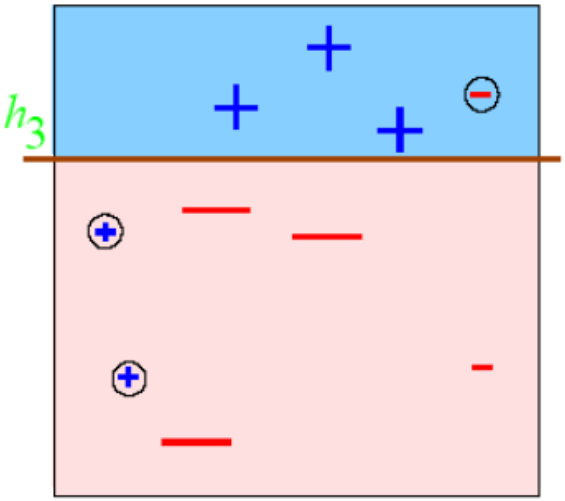


3. Boosting. Ejemplo



3. Boosting. Ejemplo





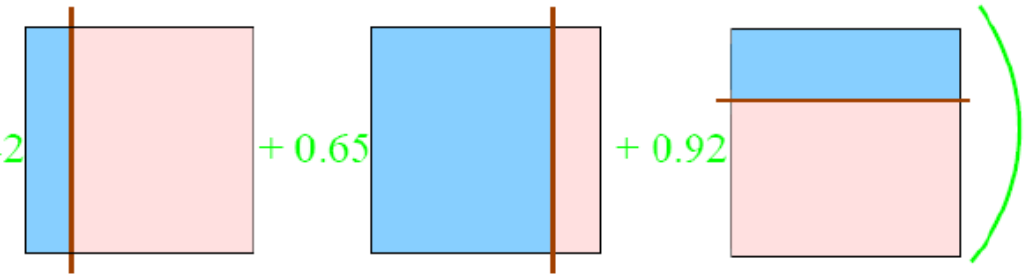
$\varepsilon_3 = 0.14$

$\alpha_3 = 0.92$

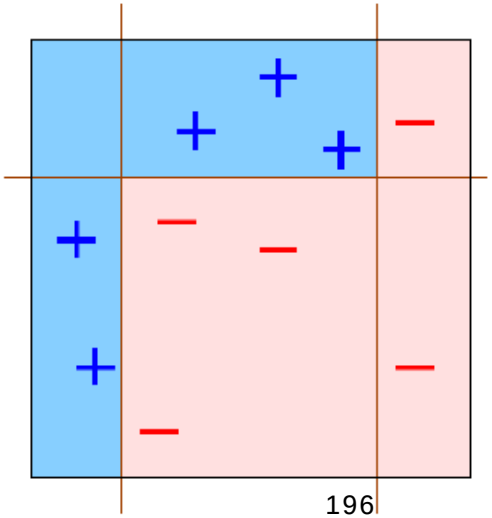
3. Boosting. Ejemplo

Final Hypothesis

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

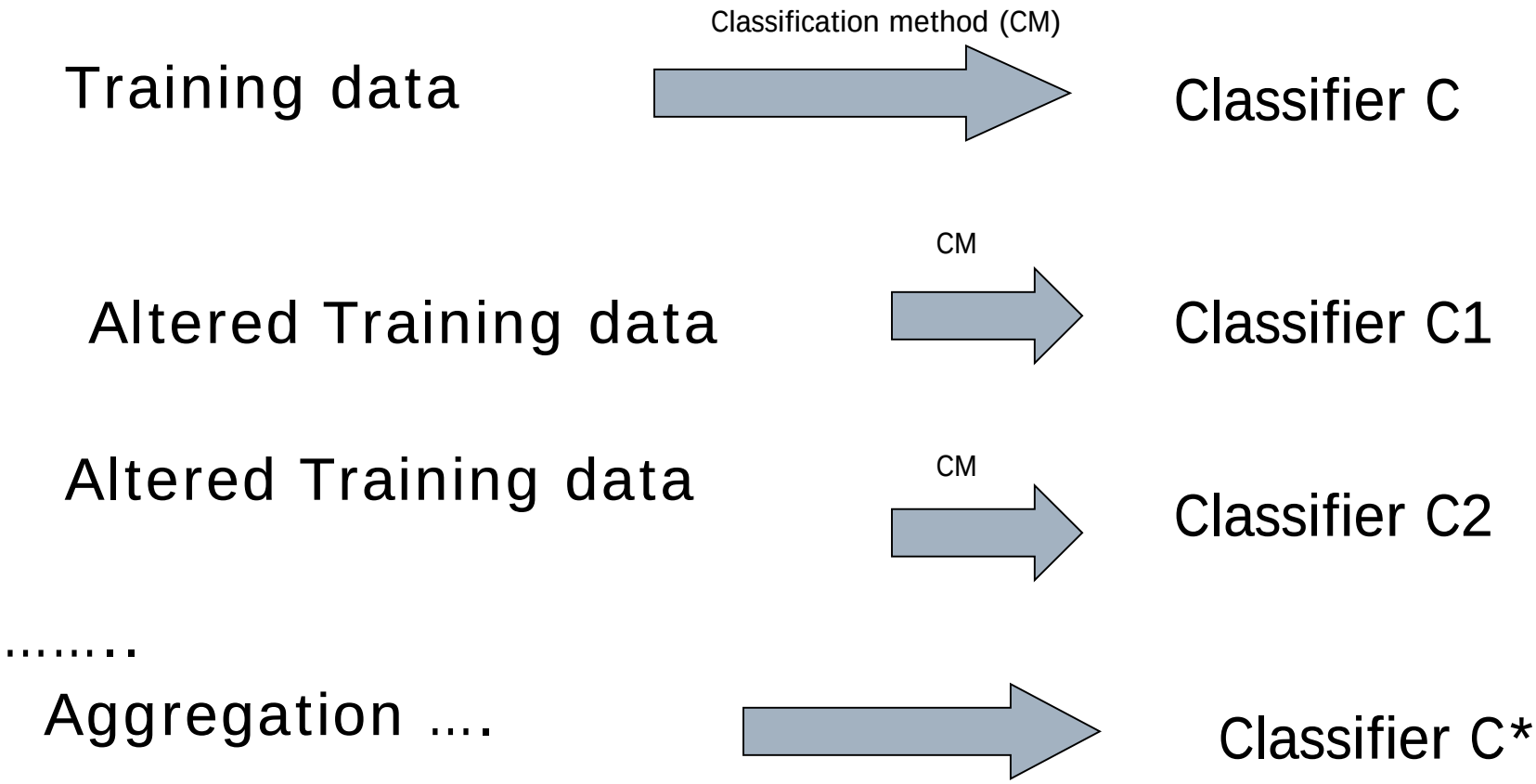


$=$



Bagging and Boosting

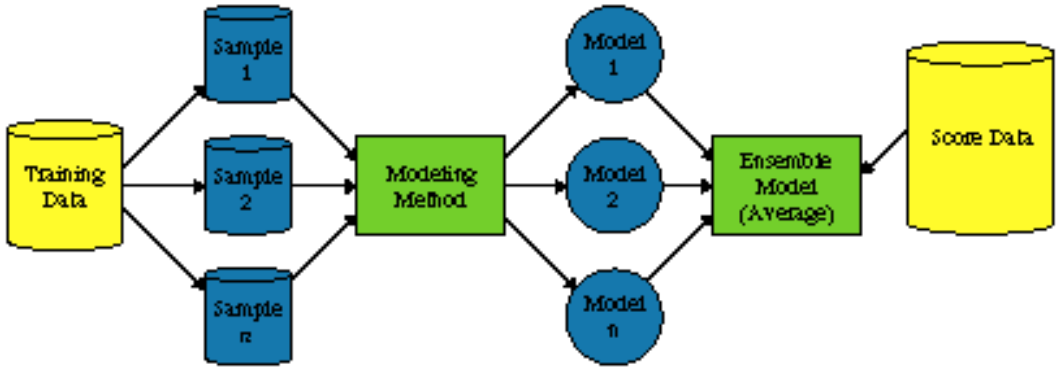
Resumen: Idea General



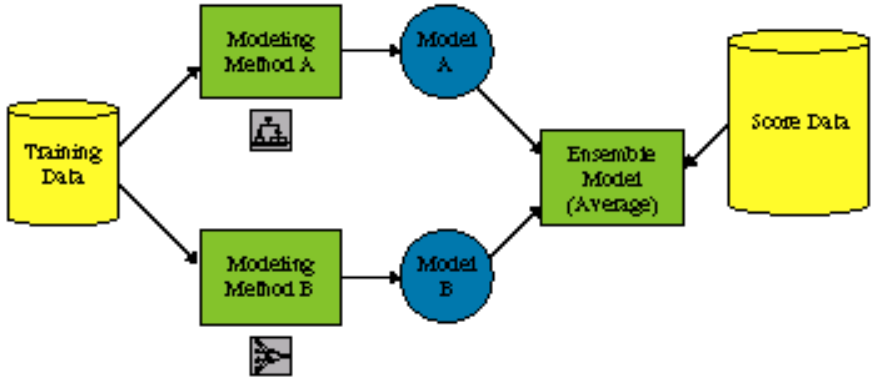
Bagging and Boosting

Resumen:

- Bagging= *Manipulation with data set*



- Boosting = *Manipulation with model*



Bagging and Boosting

Bagging and Boosting, **Poda de los clasificadores**

IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 31, NO. 2, FEBRUARY 2009 245

An Analysis of Ensemble Pruning Techniques Based on Ordered Aggregation

Gonzalo Martínez-Muñoz, Daniel Hernández-Lobato, and Alberto Suárez, *Member, IEEE*

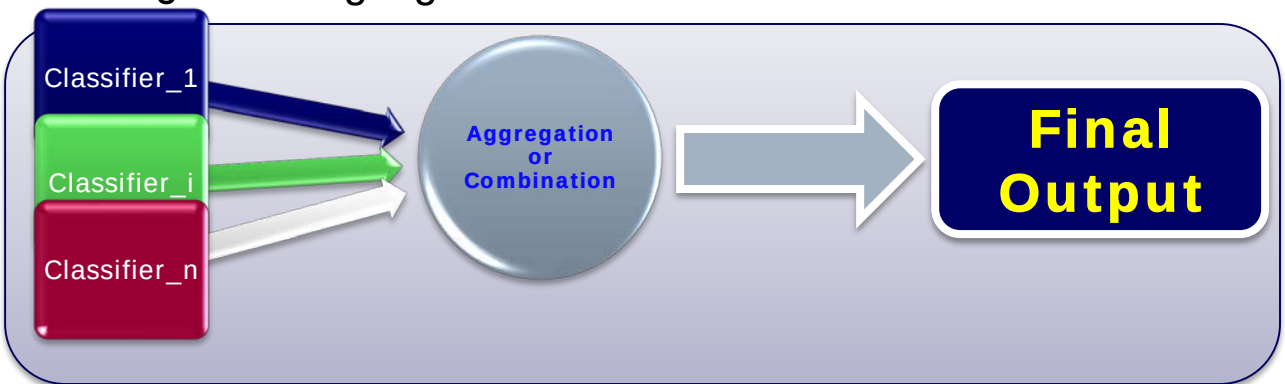
La idea general consiste en crear un pool de clasificadores más grande de lo habitual y luego reducirlo para quedarnos con los que forman el ensemble más preciso. Esta forma de selección de clasificadores se denomina "ensemble pruning".
Los modelos basados en ordenamiento generalmente parten de añadir 1 clasificador al modelo final. Posteriormente, en cada iteración añaden un nuevo clasificador del pool de los no seleccionados en base a una medida establecida. Y generalmente lo hacen hasta llegar a un número de clasificadores establecido. Según este artículo, en Bagging, teniendo un pool de 100 clasificadores, es suficiente con usar 21.
Boosting-based (**BB**): Hace un boosting a posteriori (muy interesante!). Coge el clasificador que minimiza más el coste respecto a boosting, pero no los entrena respecto a esos costes porque los clasificadores ya están en el pool.

Multiclasificadores

1. Combinación de clasificadores
2. Bagging
3. Boosting
4. **Descomposicion de Problemas de Multiclase. Binarización (Pairwise learning)**

4. Descomposición de problemas multiclase. Binarización

- Descomposición de un problema con múltiples clases
  - Estrategia Divide y Vencerás
  - Multi-clase → Es más fácil resolver problemas binarios
    - Para cada problema binario
      - 1 clasificador binario = clasificador base
    - Problem
      - ¿Cómo hacer la descomposición?
      - ¿Cómo agregar las salidas?



3. Descomposición de problemas multiclase. Binarización

Una aplicación real en KAGGLE de Problema Multiclase

Classify products into the correct category

The Otto Group is one of the world's biggest e-commerce companies, with subsidiaries in more than 20 countries, including Crate & Barrel (USA), Otto.de (Germany) and 3 Suisses (France). We are selling millions of products worldwide every day, with several thousand products being added to our product line.

A consistent analysis of the performance of our products is crucial. However, due to our diverse global infrastructure, many identical products get classified differently. Therefore, the quality of our product analysis depends heavily on the ability to accurately cluster similar products. The better the classification, the more insights we can generate about our product range.

3. Descomposición de problemas multiclase. Binarización

Una aplicación real en KAGGLE de Problema Multiclase

For this competition, we have provided a dataset with 93 features for more than 200,000 products. The objective is to build a predictive model which is able to distinguish between our main product categories. The winning models will be open sourced.

3. Descomposición de problemas multiclase. Binarización

Una aplicación real en KAGGLE de Problema Multiclase

Submission Format

You must submit a csv file with the product id, all candidate class names, and a probability for each class. The order of the rows does not matter. The file must have a header and should look like the following:

```
id,Class_1,Class_2,Class_3,Class_4,Class_5,Class_6,Class_7,Class_8,Class_9
1,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0
2,0.0,0.2,0.3,0.3,0.0,0.0,0.1,0.1,0.0
...
etc.
```

1987 teams

2117 players

15502 entries

Started: 3:56 pm, Tuesday 17 March 2015 UTC
Ends: 11:59 pm, Monday 18 May 2015 UTC (62 total days)
Points: this competition awards standard ranking points
Tiers: this competition counts towards tiers

3. Descomposición de problemas multiclase. Binarización

Una aplicación real en KAGGLE de Problema Multiclase

Evaluation

Submissions are evaluated using the multi-class logarithmic loss. Each product has been labeled with one true category. For each product, you must submit a set of predicted probabilities (one for every category). The formula is then,

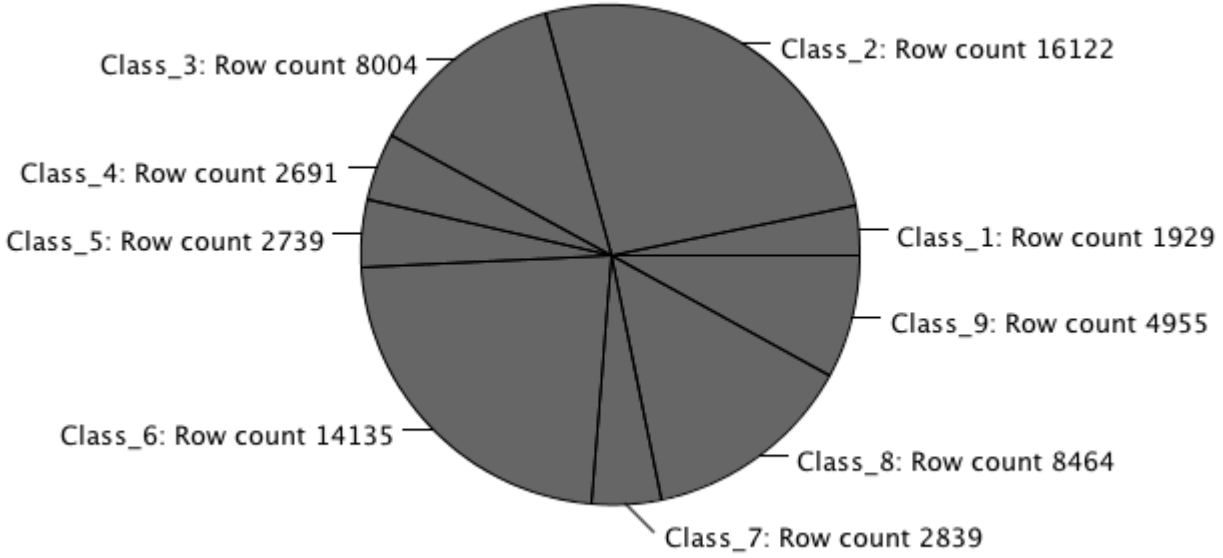
$$logloss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}\log(p_{ij}),$$

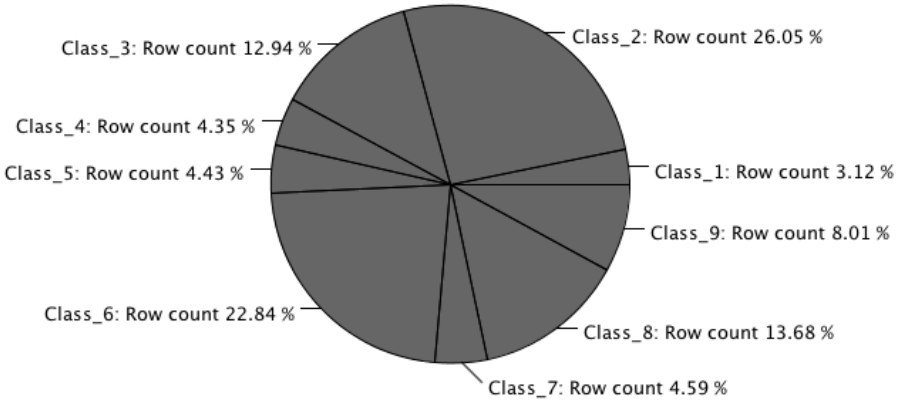
where N is the number of products in the test set, M is the number of class labels, log is the natural logarithm, y_{ij} is 1 if observation i is in class j and 0 otherwise, and p_{ij} is the predicted probability that observation i belongs to class j.

The submitted probabilities for a given product are not required to sum to one because they are rescaled prior to being scored (each row is divided by the row sum). In order to avoid the extremes of the log function, predicted probabilities are replaced with $max(min(p, 1-10^{-15}), 10^{-15})$.

3. Descomposición de problemas multiclase. Binarización

Una aplicación real en KAGGLE de Problema Multiclase





3. Descomposición de problemas multiclase. Binarización

Una aplicación real en KAGGLE de Problema Multiclase

otto group

$10,000 • 2,296 teams

Otto Group Product Classification Challenge

Tue 17 Mar 2015 — Enter/Merge by — Mon 18 May 2015 (32 days to go)

Dashboard

Public Leaderboard - Otto Group Product Classification Challenge

This leaderboard is calculated on approximately 70% of the test data.
The final results will be based on the other 30%, so the final standings may be different.

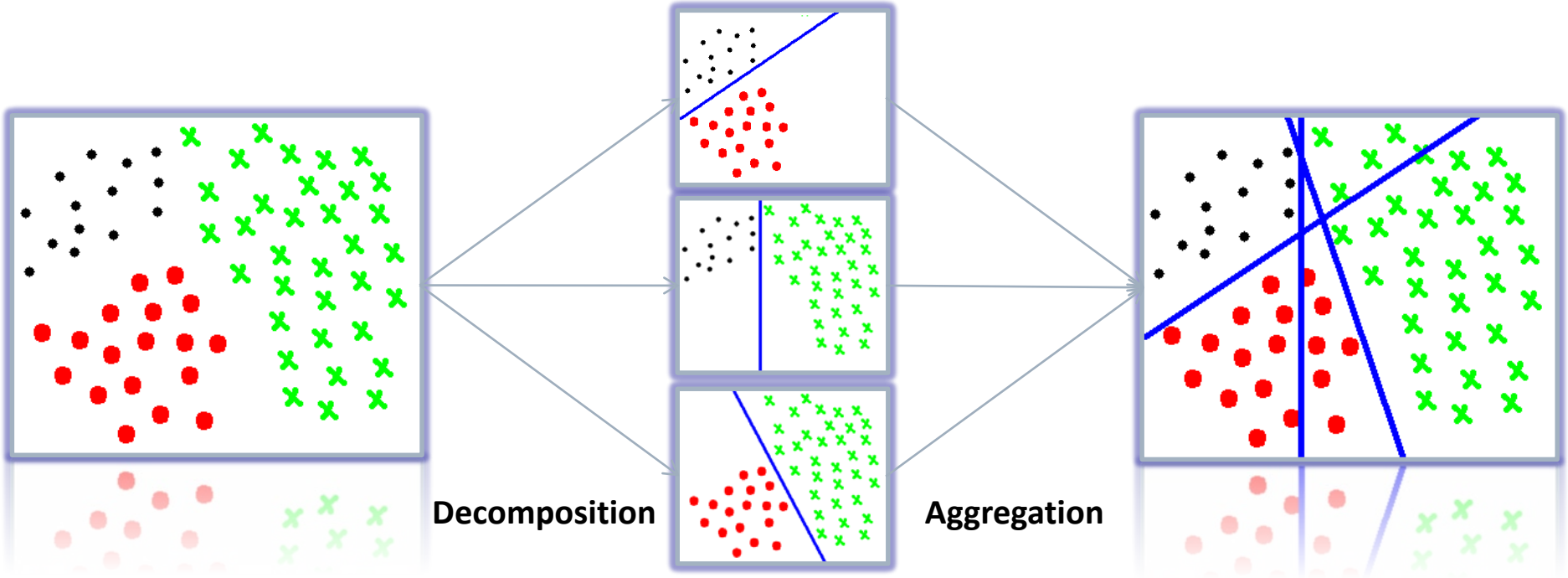
See someone using multiple accou
Let us kn

#	Δ1w	Team Name * in the money	Score	Entries	Last Submission UTC (Best – Last Submissio
1	—	i dont know *	0.39067	67	Thu, 16 Apr 2015 21:37:26
2	—	team *	0.40017	20	Thu, 16 Apr 2015 14:45:41
3	new	tks *	0.40110	1	Thu, 16 Apr 2015 16:54:01
4	↓1	IzuiT	0.40311	43	Thu, 16 Apr 2015 05:29:26
5	↓1	Hoang Duong	0.40382	32	Thu, 16 Apr 2015 05:44:21 (-8.8d)
6	↓1	Nicholas Guttenberg	0.40857	55	Thu, 16 Apr 2015 15:46:43 (-2.6d)

4. Descomposición de problemas multiclase. Binarización

Estrategias de descomposición: “One-vs-One” (OVO)

- 1 problema binario para cada par de clases
  - Nombres en la literatura: Pairwise Learning, Round Robin, All-vs-All…
  - *Total = m(m-1) / 2 classificadores*



4. Descomposición de problemas multiclase. Binarización

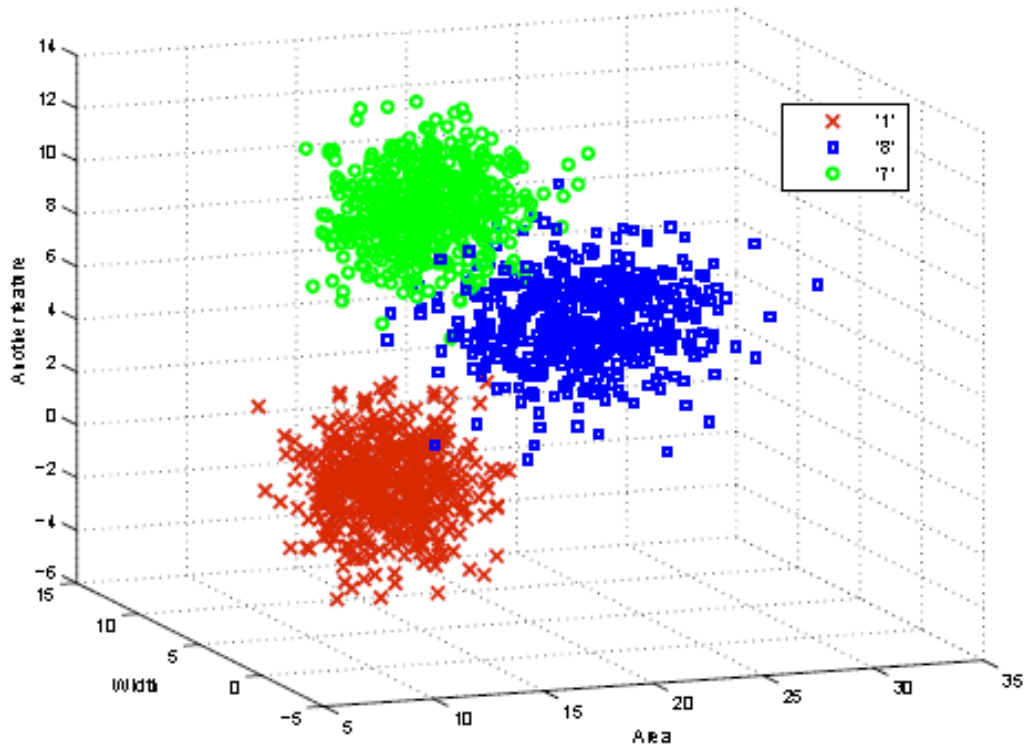


FIGURE : A multi-class problem, a new feature is needed

4. Descomposición de problemas multiclase. Binarización

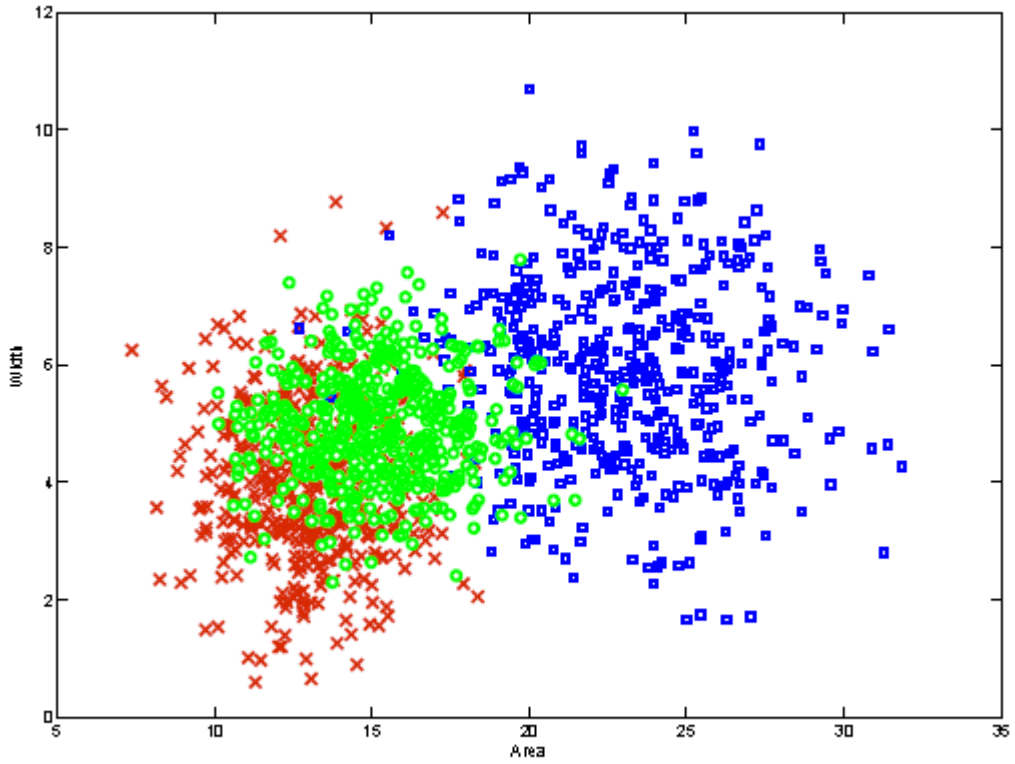
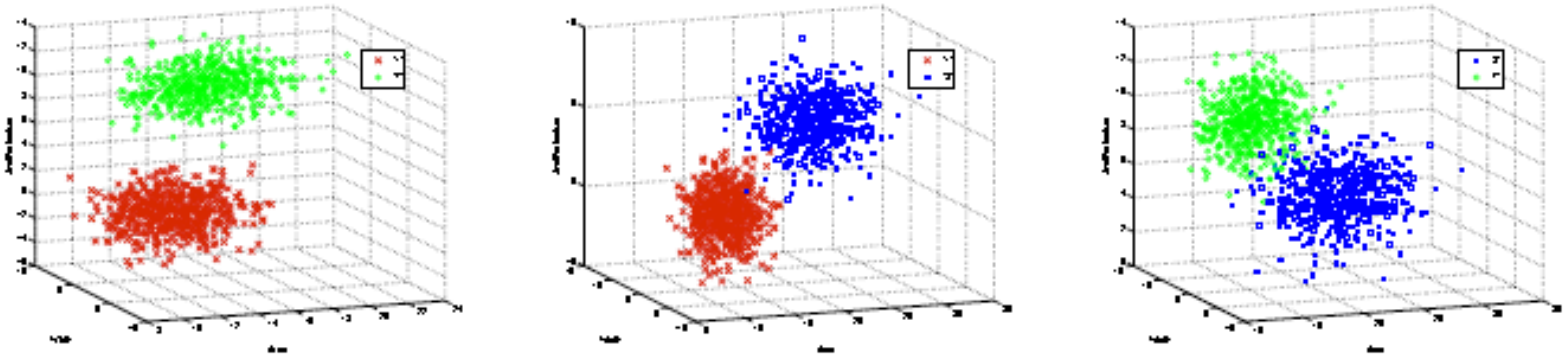


FIGURE : A multi-class problem

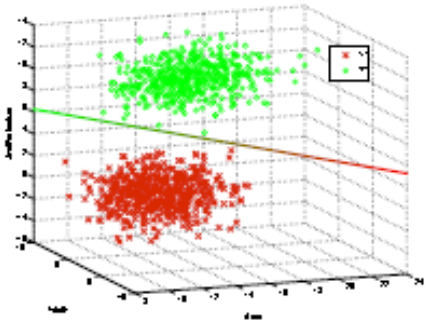
4. Descomposición de problemas multiclase. Binarización



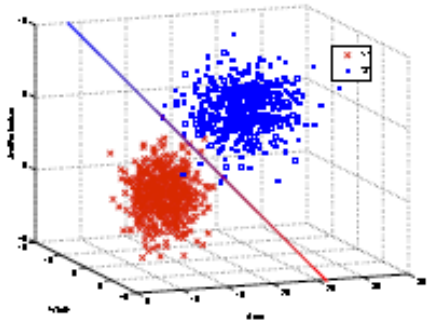
(a) '1' vs. '7' (b) '1' vs. '8' (c) '8' vs. '7'

FIGURE : One-vs-One scheme

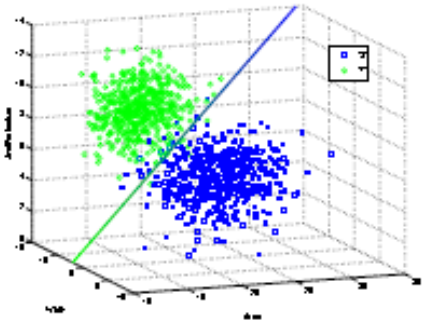
4. Descomposición de problemas multiclase. Binarización



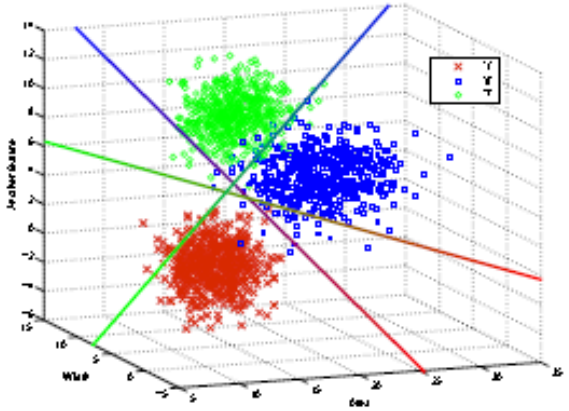
(a) '1' vs. '7'



(b) '1' vs. '8'



(c) '8' vs. '7'



(d) Aggregation

FIGURE : One-vs-One scheme

4. Descomposición de problemas multiclase. Binarización

Pairwise Learning: Combinación de las salidas

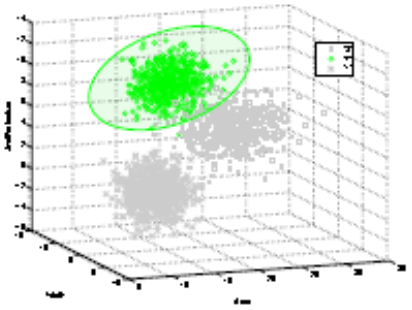
- Fase de agregación
  - *Se proponen diferentes vías para combinar los clasificadores base y obtener la salida final (clase).*
- Se comienza por una matriz de ratios de clasif.

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

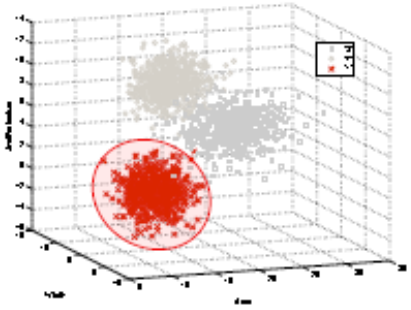
  - r_{ij} = confianza del clasificador en favor de la clase i
  - r_{ji} = confianza del clasificador en favor de la clase j
    - Usualmente: $r_{ji} = 1 - r_{ij}$
  - Se pueden plantear diferentes formas de combinar estos ratios.

4. Descomposición de problemas multiclase. Binarización

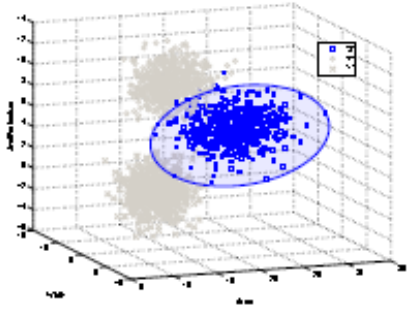
Otra estrategia de descomposición: One vs All (OVA)



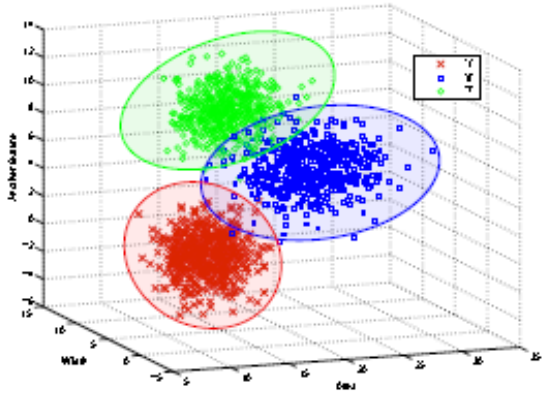
(a) '7' vs. '1' and '8'



(b) '1' vs. '7' and '8'



(c) '8' vs. '1' and '7'

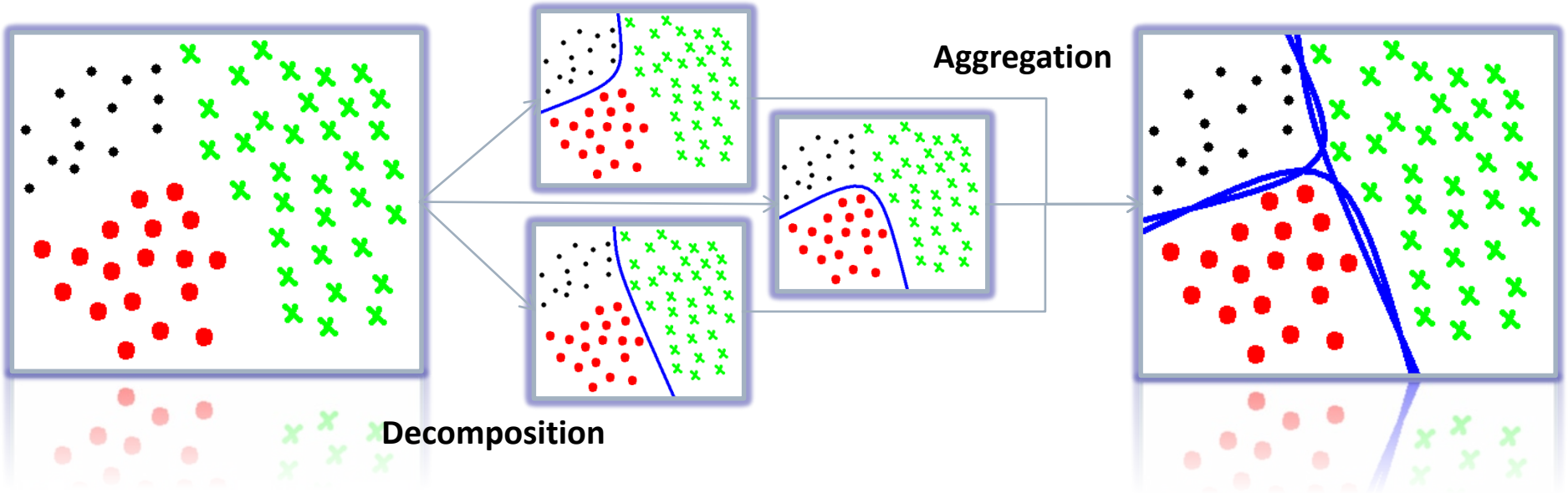


(d) Aggregation

FIGURE : One-vs-All scheme

4. Descomposición de problemas multiclase. Binarización

Otra estrategia de descomposición: One vs All (OVA)



4. Descomposición de problemas multiclase. Binarización

- Other approaches
  - ECOC (Error Correcting Output Code) [Allwein00]
    - Unify (generalize) OVO and OVA approach
    - Code-Matrix representing the decomposition
      - The outputs forms a code-word
      - An ECOC is used to decode the code-word
        - The class is given by the decodification

Class	Classifier								
	C1	C2	C3	C4	C5	C5	C7	C8	C9
Class1	1	1	1	0	0	0	1	1	1
Class2	0	0	-1	1	1	0	1	-1	-1
Class3	-1	0	0	-1	0	1	-1	1	-1
Class4	0	-1	0	0	-1	-1	-1	-1	1

Class	Code Word														
	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}
0	1	1	0	0	0	0	1	0	1	0	0	1	1	0	1
1	0	0	1	1	1	1	0	1	0	1	1	0	0	1	0
2	1	0	0	1	0	0	0	1	1	1	1	0	1	0	1
3	0	0	1	1	0	1	1	1	0	0	0	0	1	0	1
4	1	1	1	0	1	0	1	1	0	0	1	0	0	0	1
5	0	1	0	0	1	1	0	1	1	1	0	0	0	0	1
6	1	0	1	1	1	0	0	0	0	1	0	1	0	0	1
7	0	0	0	1	1	1	1	0	1	0	1	1	0	0	1
8	1	1	0	1	0	1	1	0	0	1	0	0	0	1	1
9	0	1	1	1	0	0	0	0	1	0	1	0	0	1	1

[Allwein00] E. L. Allwein, R. E. Schapire, Y. Singer, Reducing multiclass to binary: A unifying approach for margin classifiers, Journal of Machine Learning Research 1 (2000) 113–141.

4. Descomposición de problemas multiclase. Binarización

Ventajas:

- Clasificadores más pequeños (menor número de instancias)
- Fronteras de decisión más simples

Un algoritmo que está en el “estado del arte” en comportamiento y utiliza la estrategia OVO para múltiples clases: SVM (Support Vector Machine).

Estudios sobre la descomposición binaria:

M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, F. Herrera, An Overview of Ensemble Methods for Binary Classifiers in Multi-class Problems: Experimental Study on One-vs-One and One-vs-All Schemes. *Pattern Recognition 44:8 (2011) 1761-1776,* doi: 10.1016/j.patcog.2011.01.017

Anderson Rocha and Siome Goldenstein. Multiclass from Binary: Expanding One-vs-All, One-vs-One and ECOC-based Approaches. IEEE Transactions on Neural Networks and Learning Systems Vol. 25, Num. 2, pgs. 289–302, 2014 doi:10.1109/TNNLS.2013.2274735

Inteligencia de Negocio

TEMA 4. Modelos de Predicción: Clasificación, regresión y series temporales

Clasificación

1. **El problema de clasificación**
2. Clasificación con árboles y reglas
3. Clasificación con otras técnicas
4. Multiclasificadores

Bibliografía:

G. Shmueli, N.R. Patel, P.C. Bruce
Data mining for business intelligence (Part IV)
Wiley 2010 (2nd. edition)

M. Zaki and W. Meira Jr.
Data Mining and Analysis: Fundamental Concepts and Algorithms (Part 4, Cap. 18 - 22)
Cambridge University Press, 2014.
http://www.dataminingbook.info/pmwiki.php

V. Cherkassky, F.M. Mulier
Learning from Data: Concepts, Theory, and Methods (Sections 8 and 9)
2nd Edition, Wiley-IEE Prees, 2007

Conclusiones

Clasificación: Conclusiones

Clasificación es el tipo de problema más estudiado en el ámbito de Minería de Datos.

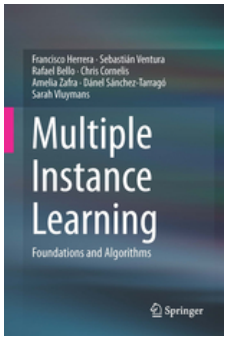
Existen múltiples aproximaciones algorítmicas que presentan buenos resultados y que deben se consideradas para su uso en función de lo que queremos obtener en cuanto a modelos:

- ✓ sistemas basados en reglas (interpretables),
- ✓ prototipos de clasificación (algoritmos basados en instancias),
- ✓ algoritmos de caja negra (no interpretables) cuya finalidad es la aproximación (redes neuronales, SVM ...)

Clasificación: Conclusiones

Existen muchos tipos concretos de problemas de clasificación que están siendo objeto de estudio en la actualidad, que dependen de la tipología de los datos.

- Multi-label Classification (MLC)
- Multi-Instance Learning (MIL)
- Semi-supervised Learning (SSL)
- Monotonic Classification
- Label Ranking



Adicionalmente existen muchos otros estudios asociados a la calidad de los datos, escalabilidad, … Algunos de ellos son:

- Imperfect Data (noise, missing values ….)
- Data Complexity Analysis of Data
- Big Data Algorithms for Classification

Clasificación: Conclusiones. Ejemplo de Clasificación No Equilibrada

Se estudiará en el Tema 7

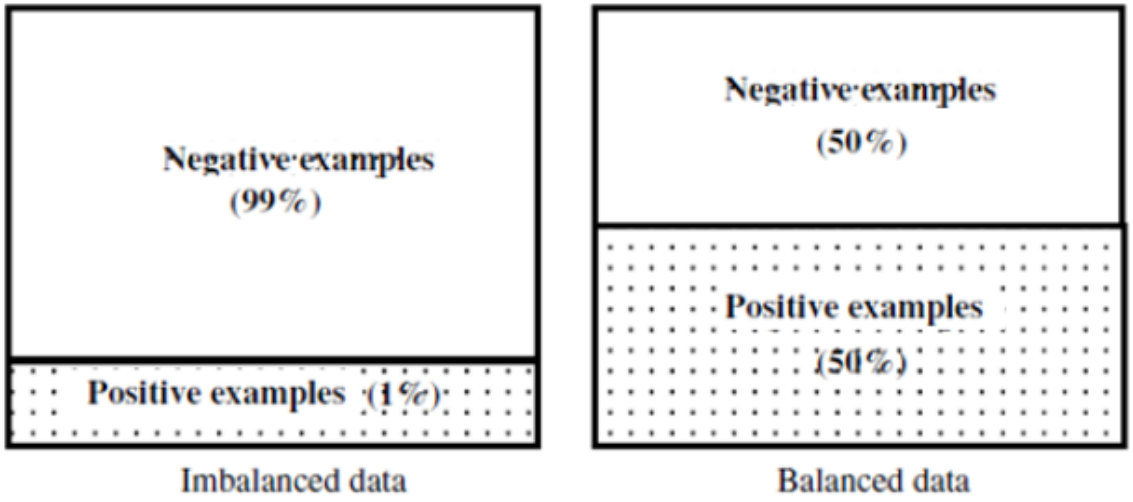


Fig. 1. Imbalanced and balanced data sets.

Clasificación no balanceada: Los clasificadores estandar tienden a sobreaprender la clase mayoritaria, ignorando la minoritaria.

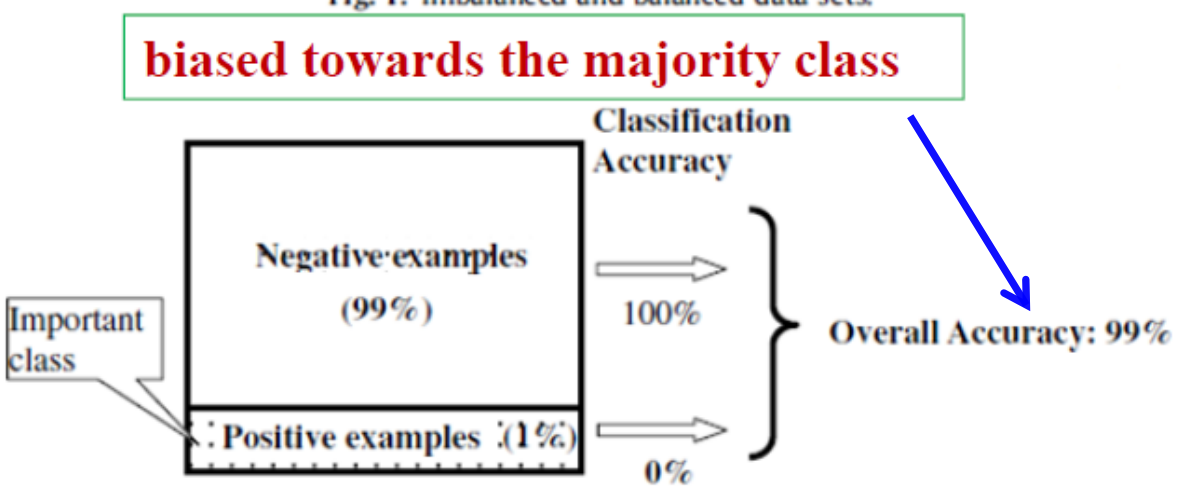


Fig. 2. The illustration of class imbalance problems.

¡Necesitamos cambiar la forma de evaluar el comportamiento de los algoritmos y modificar éstos!