

# Árboles de decisión

# Guión

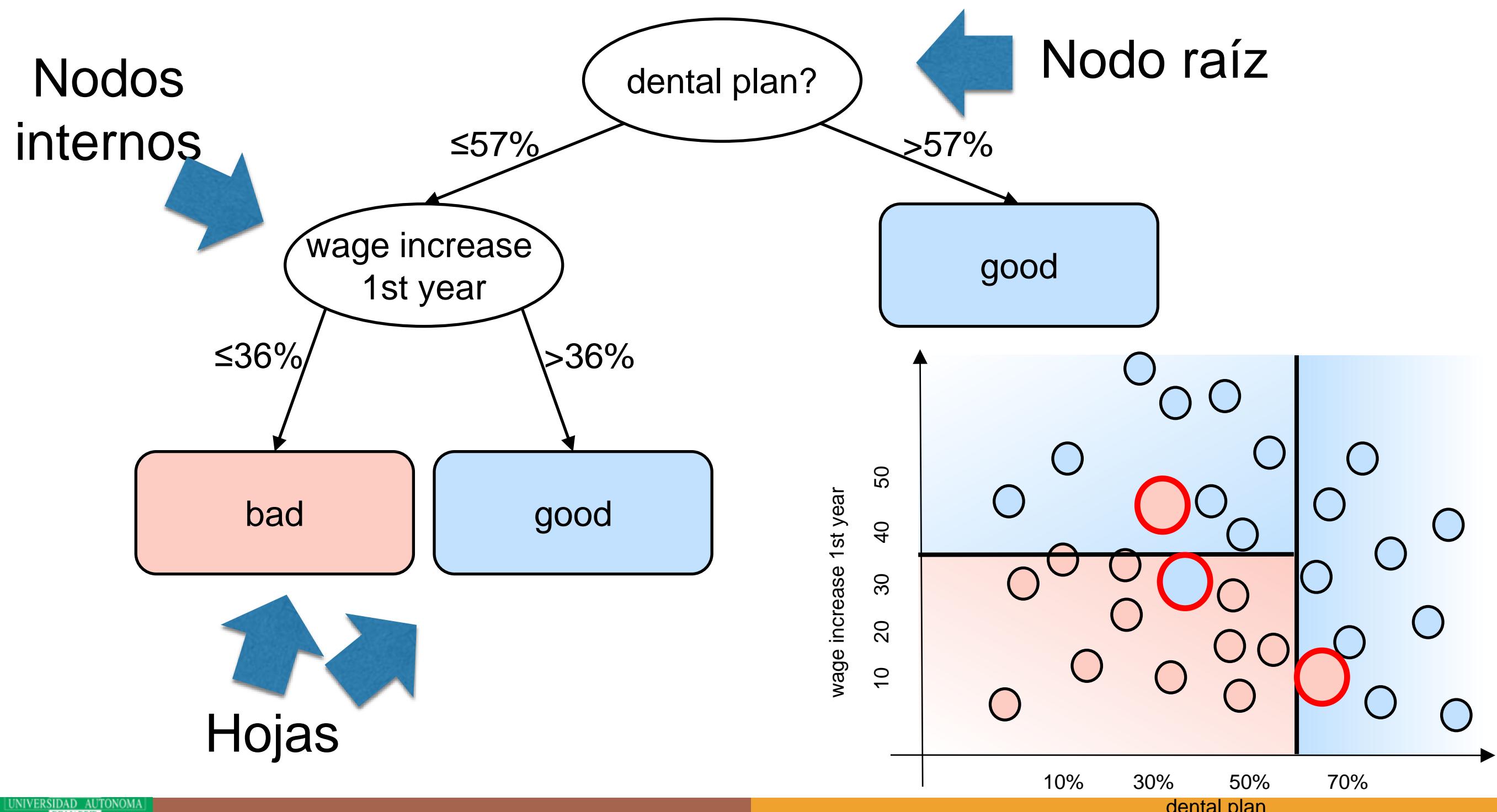
- ¿Qué es un árbol de decisión?
- Historia
- Algoritmos para creación de árboles de decisión
  - Crecer el árbol
  - Podar el árbol
- Capacidades

# ¿Qué es un árbol de decisión?

- Modelo de aprendizaje jerárquico que divide el espacio de usando reglas de decisión
- Es válido para regresión y clasificación
- Vale, pero ¿Qué es un árbol de decisión?

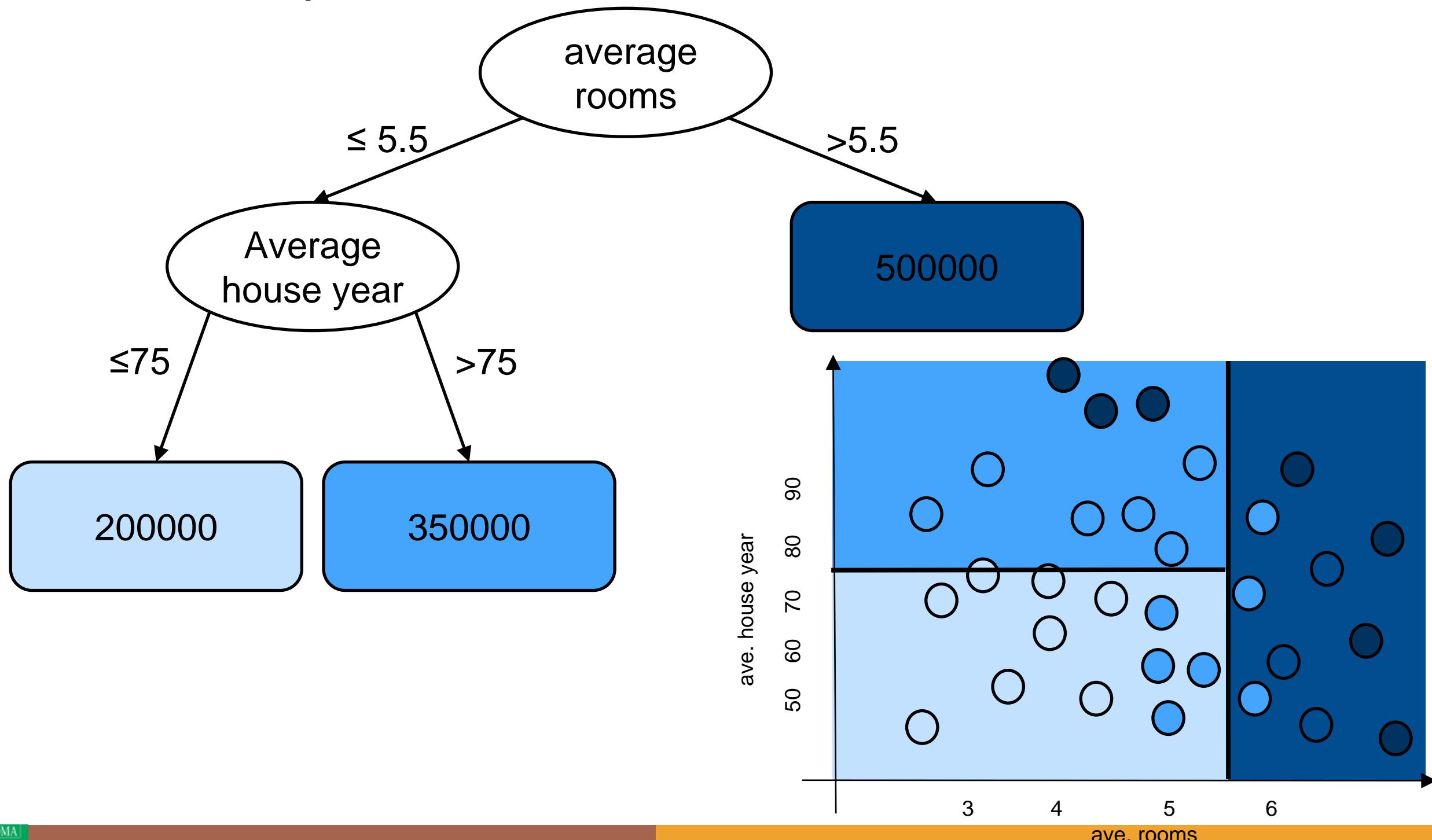
# Ejemplo para clasificación

- Negociaciones laborales: predecir if un convenio colectivo es bueno o malo



# Ejemplo para regresión

- Vivienda en Boston: Predicción precios de viviendas en Boston por barrio



# Historia

- Precursores: Sistemas expertos (SE)

SE = BBDD de conocimiento + Motor de inferencia

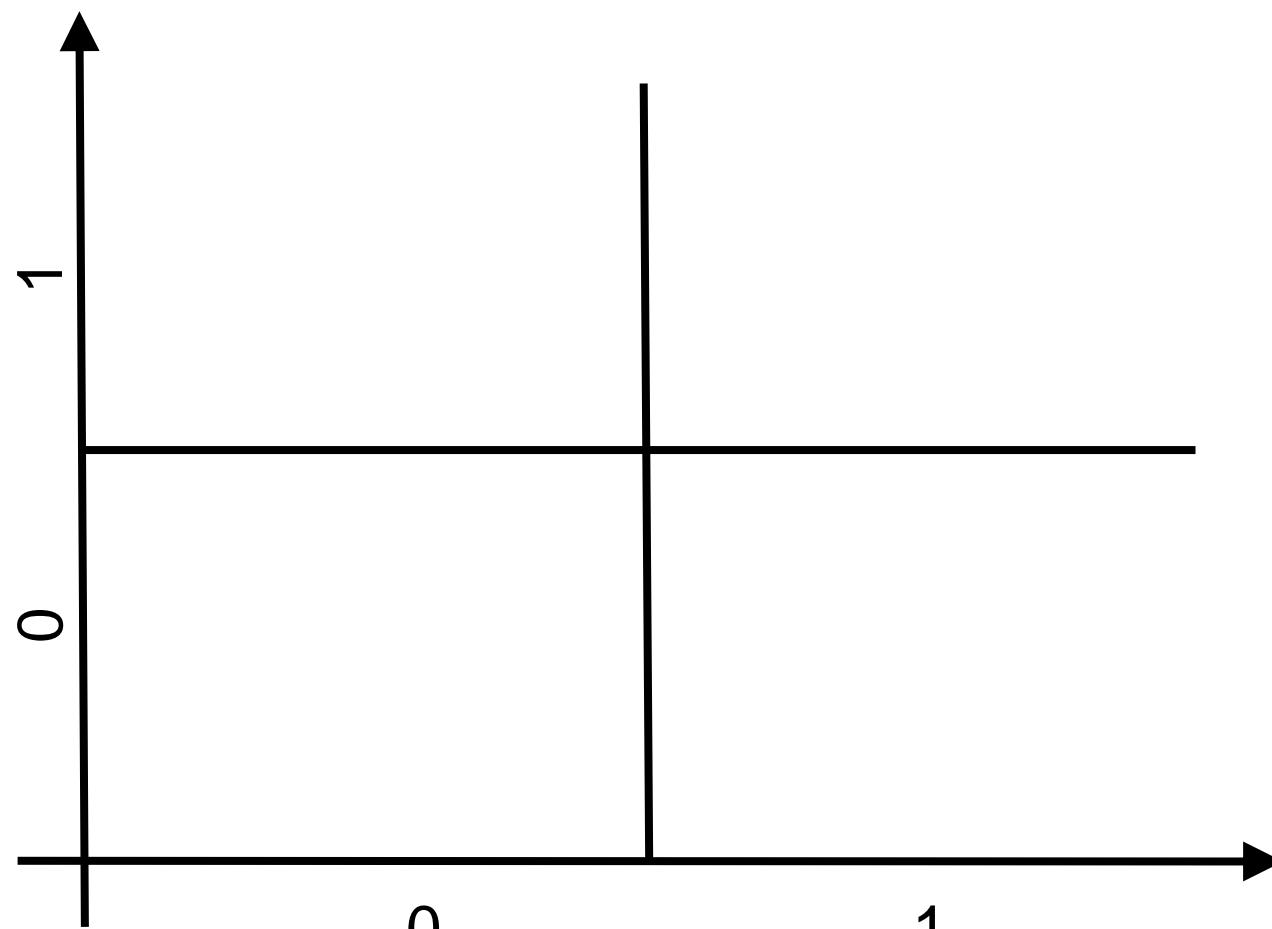
- MYCIN: Diagnóstico medico basado en 600 reglas
- XCON: Sistema para configurar ordenadores VAX, 2500 reglas (1982)
- Las reglas se creaban por expertos a mano!!
- La adquisición de conocimiento debe automatizarse
- Sustituir al **Experto** por su **archivo de casos resueltos**

# Historia

- CHAID (CHi-squared Automatic Interaction Detector) Gordon V. Kass ,1980
- CART (Classification and Regression Trees), Breiman, Friedman, Olsen y Stone, 1984
- ID3 (Iterative Dichotomiser 3), Quinlan, 1986
- C4.5, Quinlan 1993: Basado en ID3

# Computacional

- Considera dos variables binarias ¿De cuántas formas posibles podemos dividir el espacio con un árbol de decisión?



- Dos posibles divisiones y dos posibles asignaciones a las hojas → Al menos 8 árboles

# Computacional

- ¿Bajo qué condiciones esperamos en un restaurante para cenar?

Example	Attributes										Goal <i>WillWait</i>
	<i>Alt</i>	<i>Bar</i>	<i>Fri</i>	<i>Hun</i>	<i>Pat</i>	<i>Price</i>	<i>Rain</i>	<i>Res</i>	<i>Type</i>	<i>Est</i>	
$X_1$	Yes	No	No	Yes	Some	\$\$\$	No	Yes	French	0–10	Yes
$X_2$	Yes	No	No	Yes	Full	\$	No	No	Thai	30–60	No
$X_3$	No	Yes	No	No	Some	\$	No	No	Burger	0–10	Yes
$X_4$	Yes	No	Yes	Yes	Full	\$	No	No	Thai	10–30	Yes
$X_5$	Yes	No	Yes	No	Full	\$\$\$	No	Yes	French	>60	No
$X_6$	No	Yes	No	Yes	Some	\$\$	Yes	Yes	Italian	0–10	Yes
$X_7$	No	Yes	No	No	None	\$	Yes	No	Burger	0–10	No
$X_8$	No	No	No	Yes	Some	\$\$	Yes	Yes	Thai	0–10	Yes
$X_9$	No	Yes	Yes	No	Full	\$	Yes	No	Burger	>60	No
$X_{10}$	Yes	Yes	Yes	Yes	Full	\$\$\$	No	Yes	Italian	10–30	No
$X_{11}$	No	No	No	No	None	\$	No	No	Thai	0–10	No
$X_{12}$	Yes	Yes	Yes	Yes	Full	\$	No	No	Burger	30–60	Yes

Hay  $2 \times 2 \times 2 \times 2 \times 3 \times 3 \times 2 \times 2 \times 4 \times 4 = 9216$  casos

y dos clases  $\rightarrow 2^{9216}$  posibles hipótesis y muchos más árboles !!!

# Computacional

- Es simplemente inviable encontrar la solución óptima
- Debemos tener un sesgo (preferencia) a la hora de construir el modelo.
- Este es un problema general en aprendizaje automático.

# Computacional

Para árboles normalmente se une un sesgo codicioso (o cortoplacista):

- Construir el árbol paso a paso en vez de globalmente
- En cada paso seleccionar la mejor división con respecto a los datos de entrenamiento (siguiendo un **criterio de división**).
- El árbol se crece hasta que se cumple un **criterio de parada**.
- El árbol se poda (siguiendo un **criterio de poda**) para evitar sobre ajuste.

# Algoritmo básico de construcción de árboles de decisión

**trainTree (Dataset L)**

1.  $T = \text{growTree}(L)$
2.  $\text{pruneTree}(T, L)$
3. return  $T$

La poda:  
elimina  
subárboles  
de validez  
incierta.

**growTree (Dataset L)**

1.  $T.s = \text{findBestSplit}(L)$
2. if  $T.s == \text{null}$  return null
3.  $(L_1, L_2) = \text{splitData}(L, T.s)$
4.  $T.left = \text{growTree}(L_1)$
5.  $T.right = \text{growTree}(L_2)$
6. return  $T$

# Encontrar la mejor división

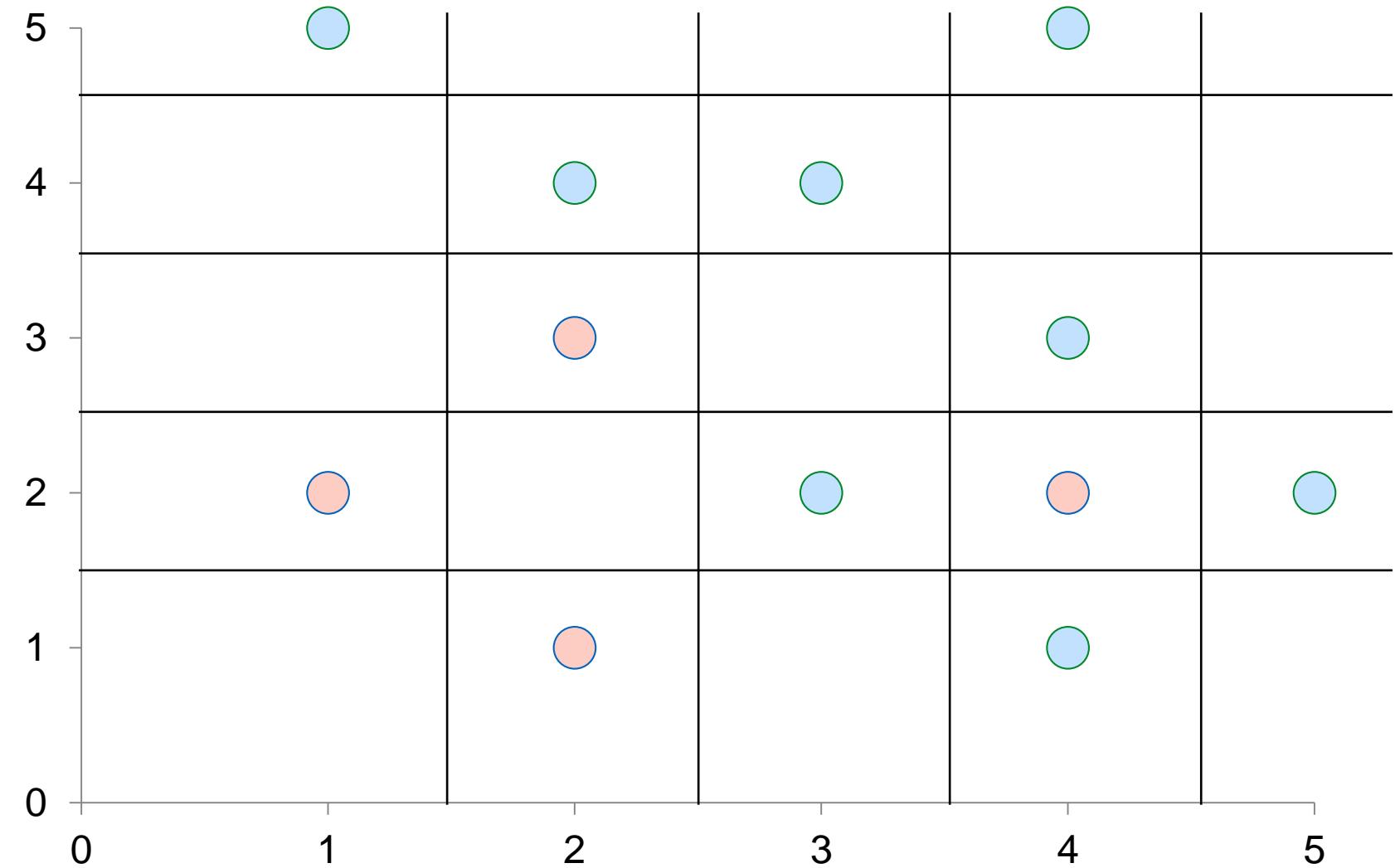
**findBestSplit(Dataset L)**

1. Try all possible splits
2. return best

Pero ¿Cuál es  
la mejor  
división?

No hay  
demasiadas  
divisiones  
→ Viable  
computacional-  
mente

$O(\text{no. atribs.} \times \text{no. puntos})$



# Criterio de división

- Debe medir la impureza del nodo:

$$i(t) = \sum_{i=1}^m f_i(1 - f_i) \quad \text{impureza Gini (CART)}$$

donde  $f_i$  es la fracción de ejemplos de clase  $i$  en el nodo  $t$  y  $m$  es el número de clases

- La mejora que da una división es la variación de la impureza antes y después de la división

$$\Delta i(t, s) = i(t) - p_L i(t_L) - p_R i(t_R)$$

donde  $p_L$  ( $p_R$ ) es la proporción de ejemplos que van al nodo izquierdo (derecho) tras la división

# Criterio de división

$$\Delta i(t, s) = i(t) - p_L i(t_L) - p_R i(t_R)$$

$$i(t) = 2 \times \frac{8}{12} \times \frac{4}{12} = 0,44$$

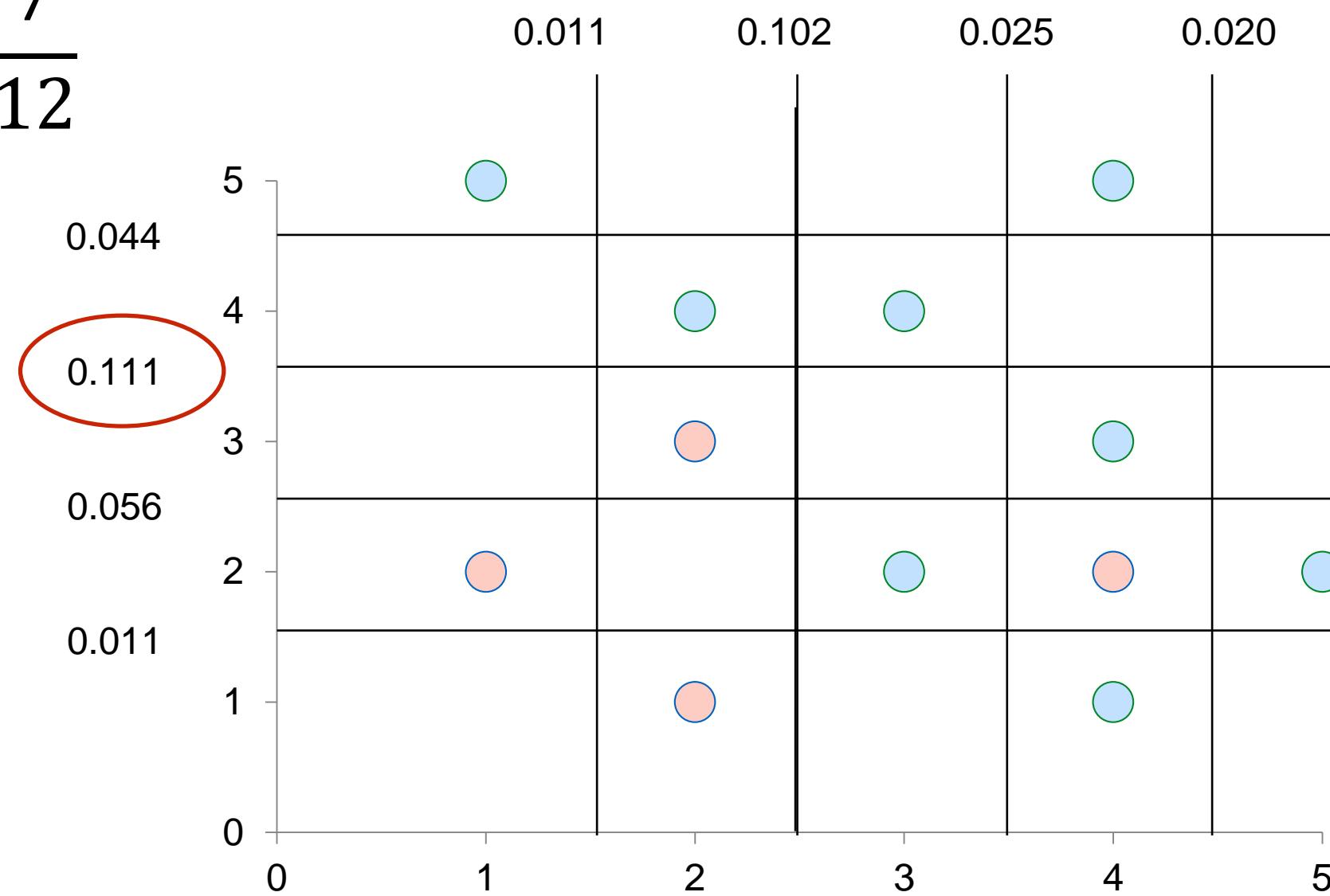
$$p_L = \frac{5}{12} ; p_R = \frac{7}{12}$$

$$i(t_L) = 2 \times \frac{2}{5} \times \frac{3}{5}$$

$$i(t_R) = 2 \times \frac{6}{7} \times \frac{1}{7}$$

$$\Delta i(t, s) = 0.102$$

¿Cuál es la mejor división?



# Otros criterios

- Basados en entropía

$$H(t) = - \sum_{i=1}^m f_i \log_2 f_i$$

- Ganancia de información (IG), usado en ID3 y C4.5

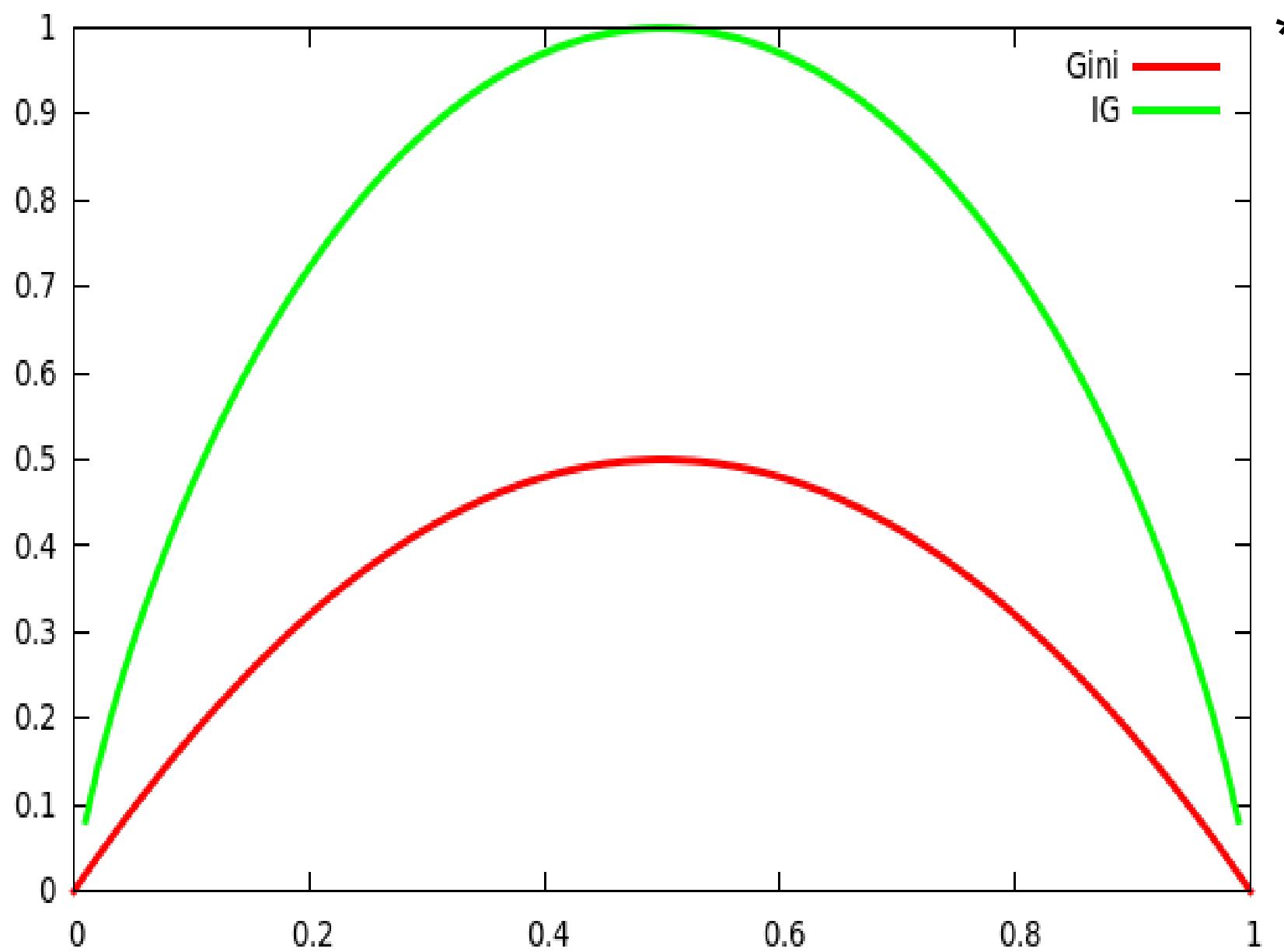
$$IG(t, s) = H(t) - p_L H(t_L) - p_R H(t_R)$$

- Ratio de ganancia de información (IGR) en C4.5

$$IGR(t, s) = IG(t, s)/H(s)$$

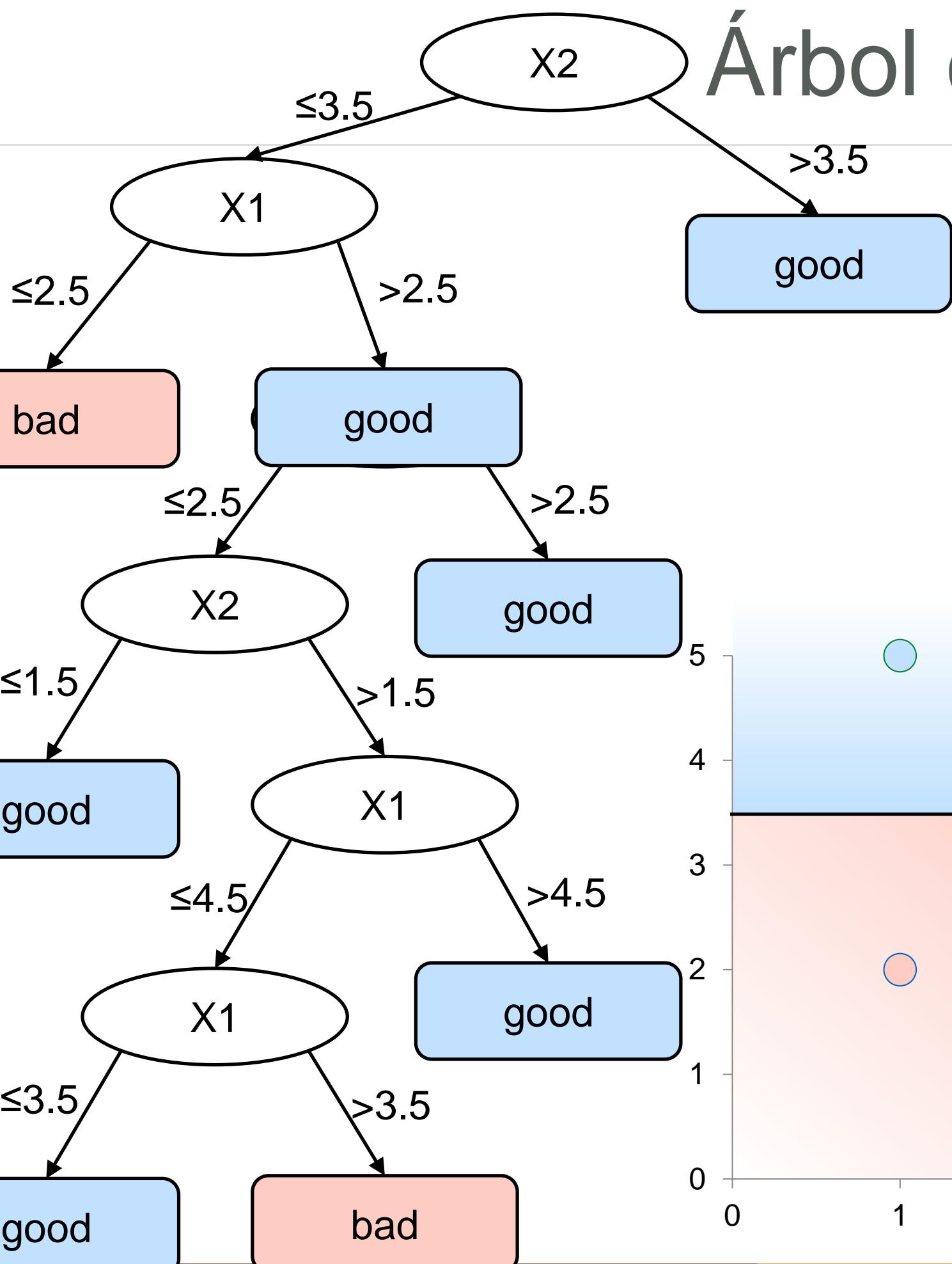
# Criterio de división

- En general, cualquier criterio con esta forma es bueno

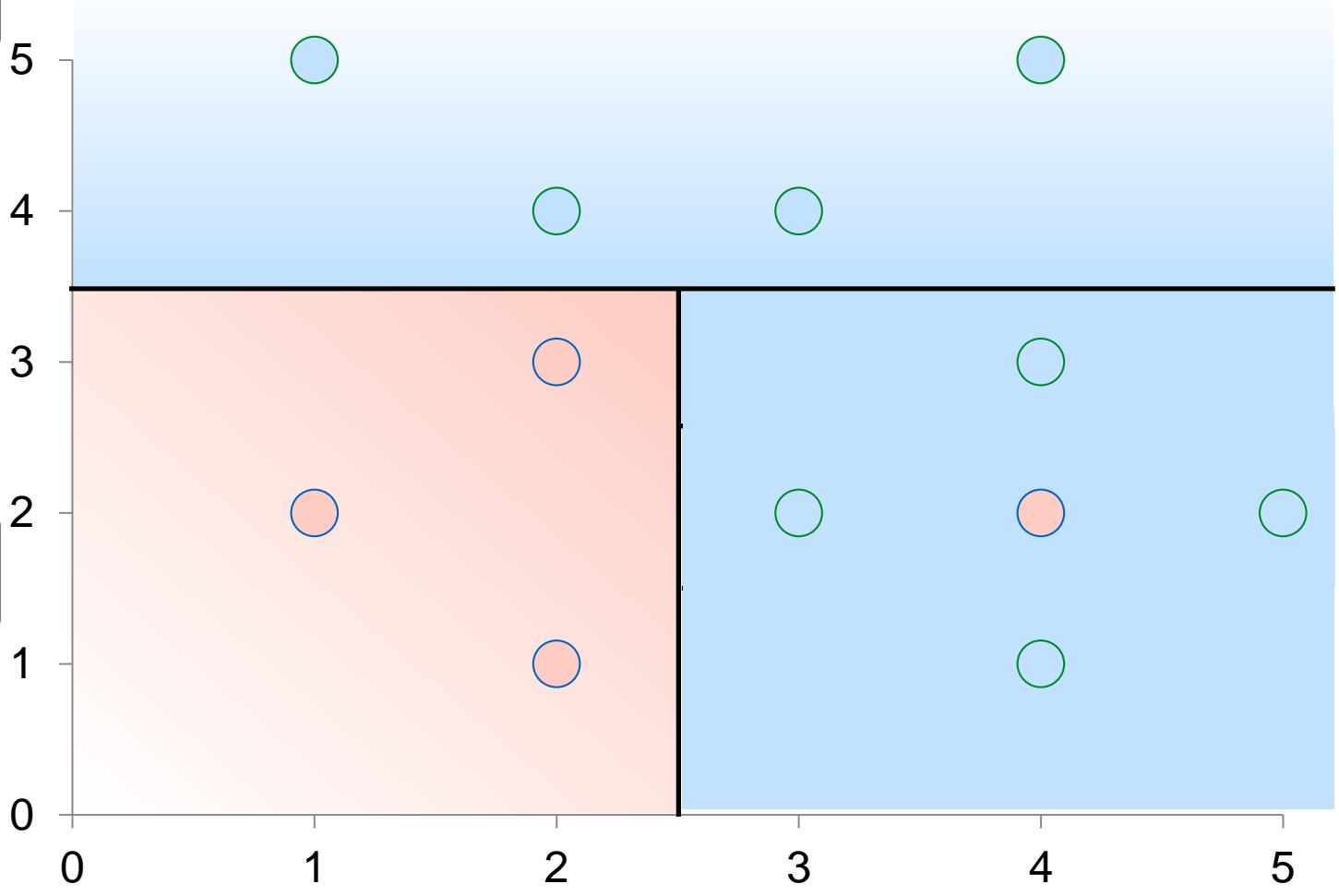


\* Esto es para problemas binarios, es decir de dos clases

# Árbol crecido al límite



¿Qué tal este otro?



# Criterio de parada

- Todos los ejemplos asignados al nodo son de la misma clase.
- No se encuentra ninguna división para partir los datos.
- El número de ejemplos en el nodo es inferior a un número predefinido (p.e. si hay menos de 10 ejemplos no se buscan más divisiones)
- La ganancia de impureza en un nodo no está por debajo de un umbral predefinido.
- El árbol alcanza la profundidad máxima.

Estos tres últimos elementos se llaman también pre-poda

# Poda

- Otra opción comúnmente utilizada es post-poda (o poda). Consiste en:
  - Crecer el árbol lo más posible.
  - Podar posteriormente sustituyendo sub-arboles por un único nodo a hoja si el error no empeora demasiado.
  - Este proceso continua hasta que no se puede podar más.
  - Realmente estamos yendo hacia atrás pero por un camino distinto al usado para crecer el árbol
  - La idea de la poda es evitar el sobre ajuste

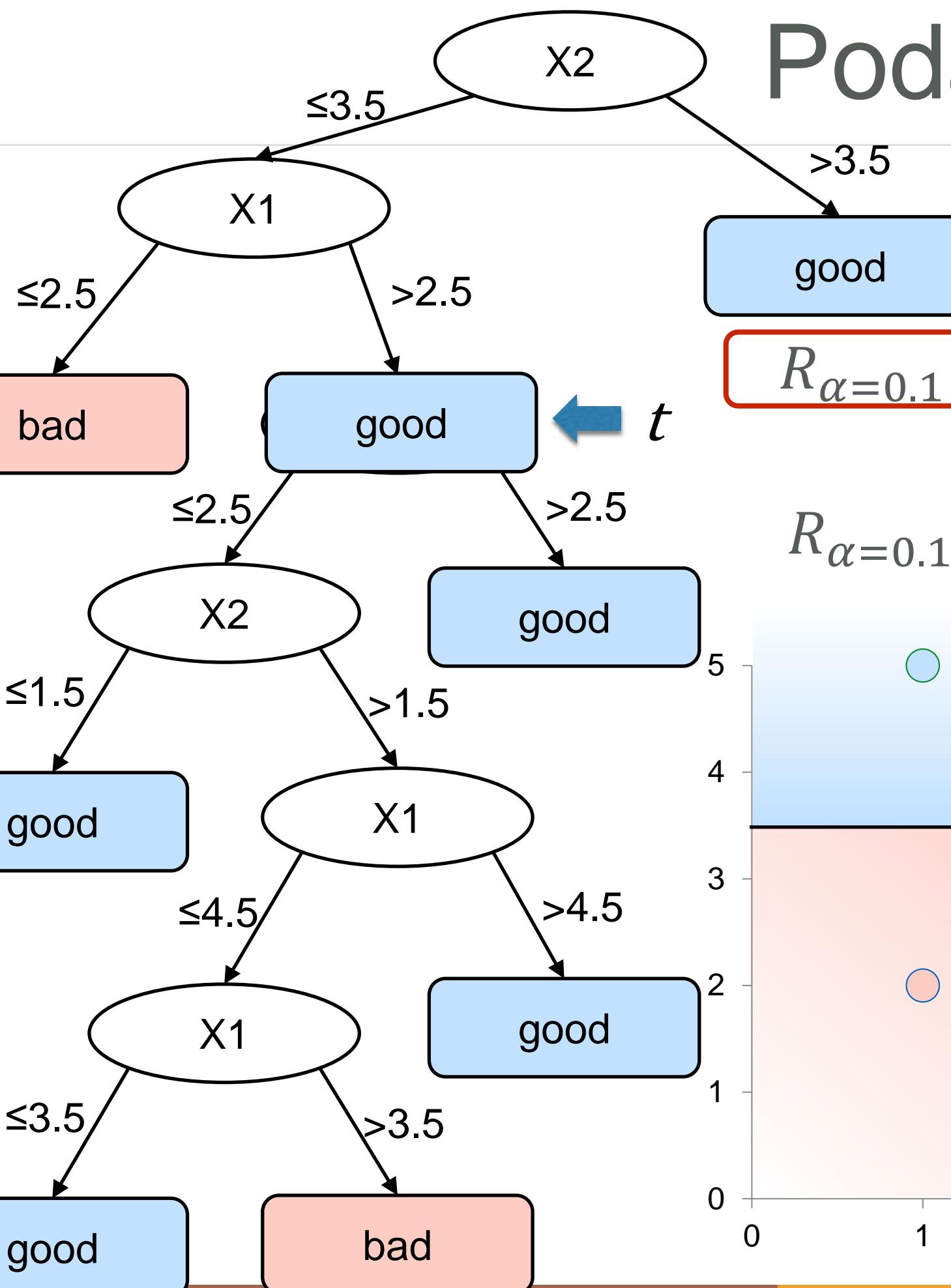
# Cost-complexity pruning (CART)

- Poda de coste-complejidad:

$$R_\alpha(t) = R(t) + \alpha \cdot C(t)$$

- $R(t)$  es el error del árbol con raíz en el nodo  $t$
- $C(t)$  es el número de hojas desde el nodo  $t$
- El parámetro  $\alpha$  especifica el peso relativo entre la precisión y la complejidad del árbol

# Poda con CART



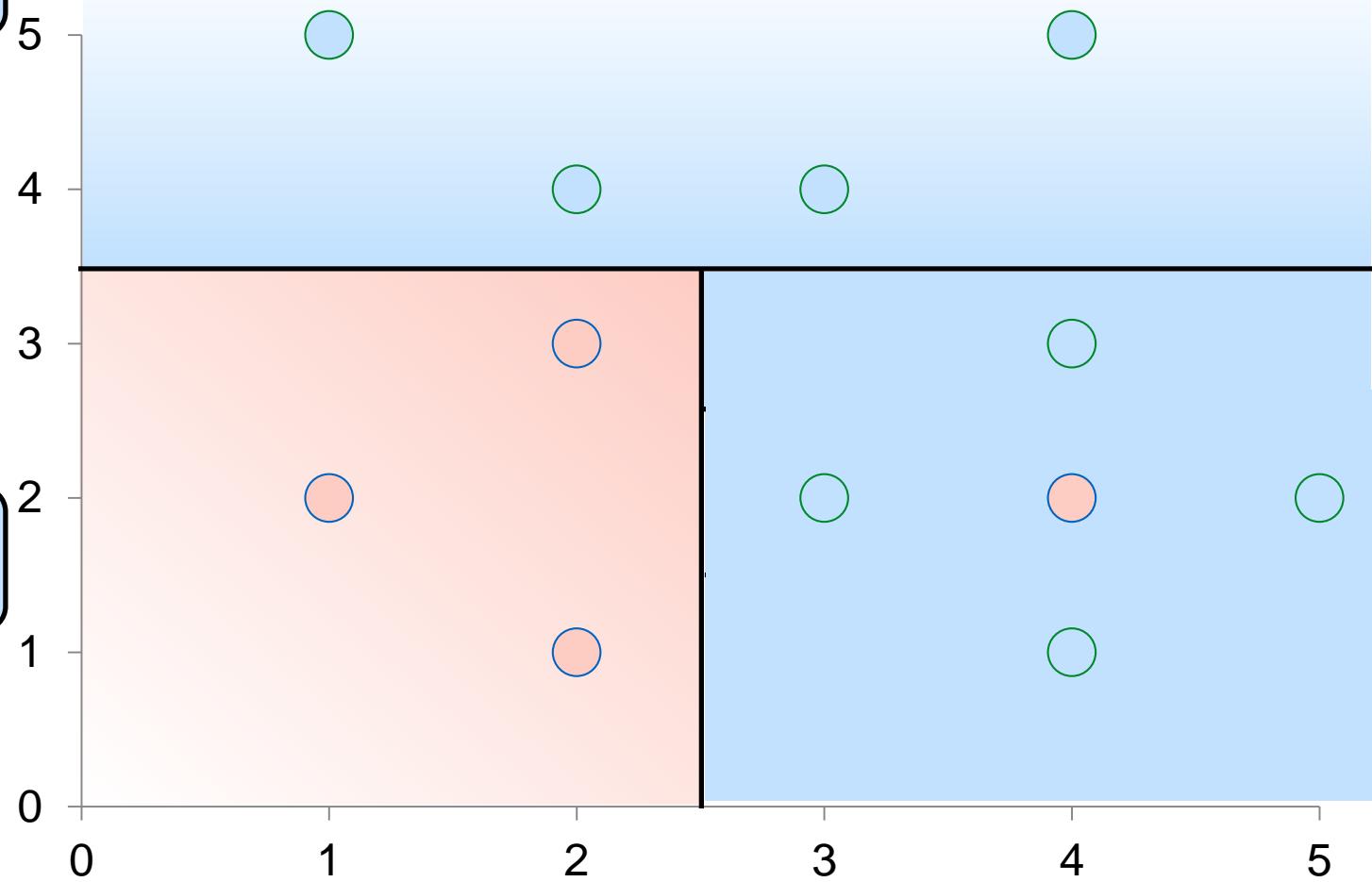
Ponemos  $\alpha = 0.1$

Podado:

$$R_{\alpha=0.1}(t) = \frac{1}{5} + 0.1 \cdot 1 = 0.3$$

No podado

$$R_{\alpha=0.1}(t) = 0 + 0.1 \cdot 5 = 0.5$$



# Cost-complexity pruning (CART)

- CART usa validación cruzada en 10 grupos en los datos de entrenamiento para estimar alfa. De forma iterativa 9 grupos se usan para entrenar un árbol y uno para probarlo.
- Un árbol es entrenado usando 9 grupos y se poda usando todos los posibles alfas (que son finitos).
- Cada árbol podado se prueba en el conjunto restante.
- El proceso se repite 10 veces y el valor de alfa que mejor error de generalización devuelve se utiliza para construir el árbol final usando todos los datos

# Statistical pruning (C4.5)

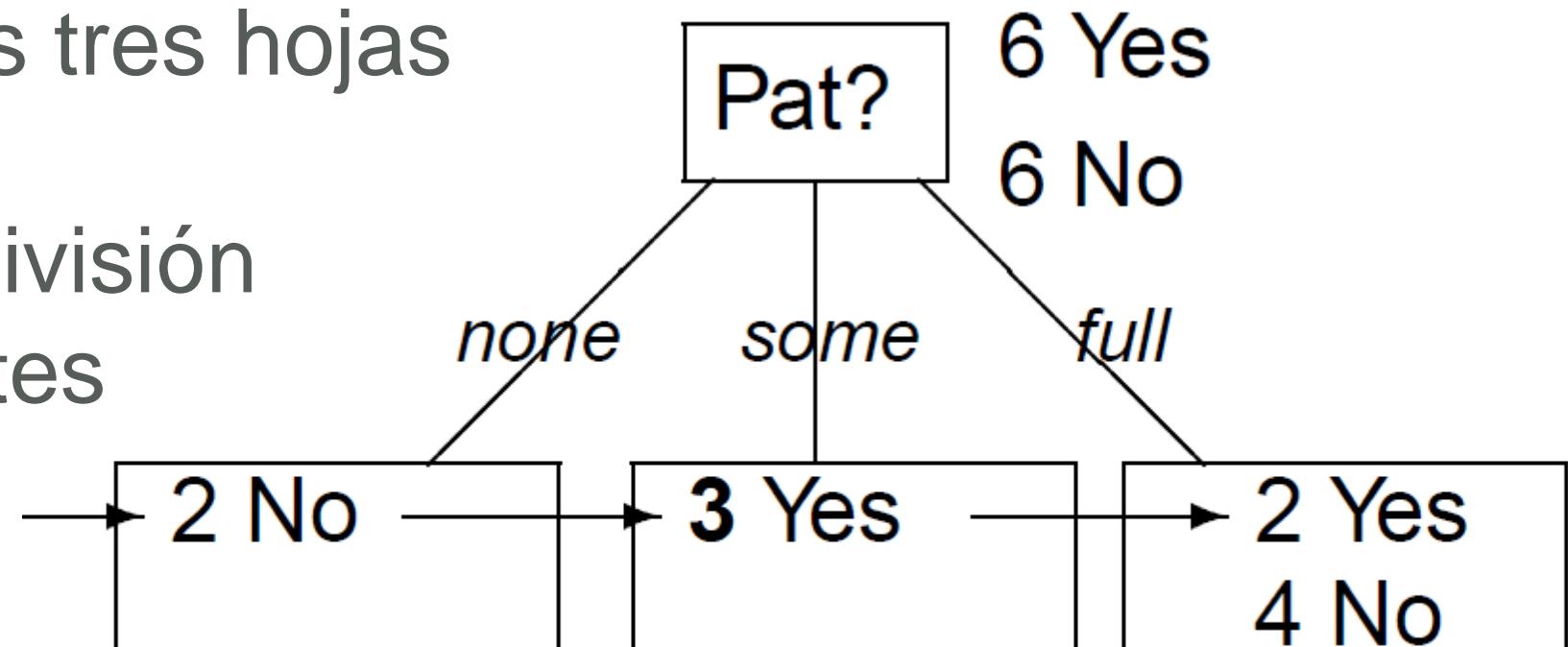
- C4.5 estima el error en las hojas usando el valor superior de confianza (es un parámetro) de una distribución normal en vez de usar el error directamente.
- El error de un sub-árbol es la suma ponderada de los errores de cada una de sus hojas
- Este error tiende a ser mayor cuanto menos ejemplos haya en una hoja.
- Por tanto, las hojas con pocas instancias tienden a ser podadas.

# Poda (CART vs. C4.5)

- Poda de CART (cost-complexity) es más lenta ya que hay que construir 10 árboles extra para estimar alfa.
- La poda de C4.5 es más rápida. Sin embargo el algoritmo no propone cómo estimar el valor umbral para la confianza
- La base estadística de la poda en C4.5 es cuestionable.
- Utilizar validación cruzada es más seguro

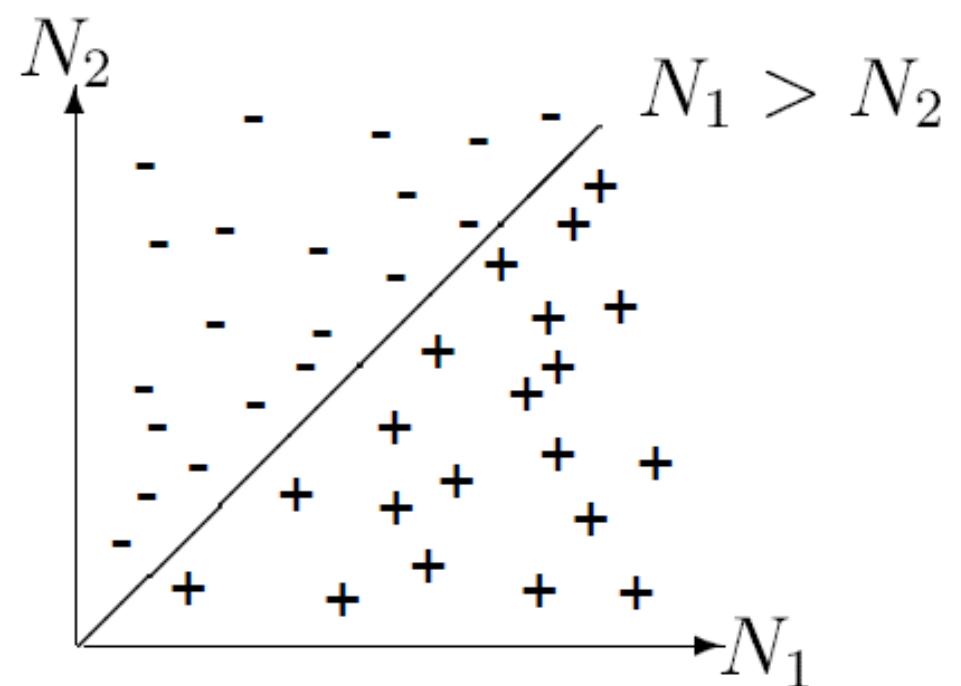
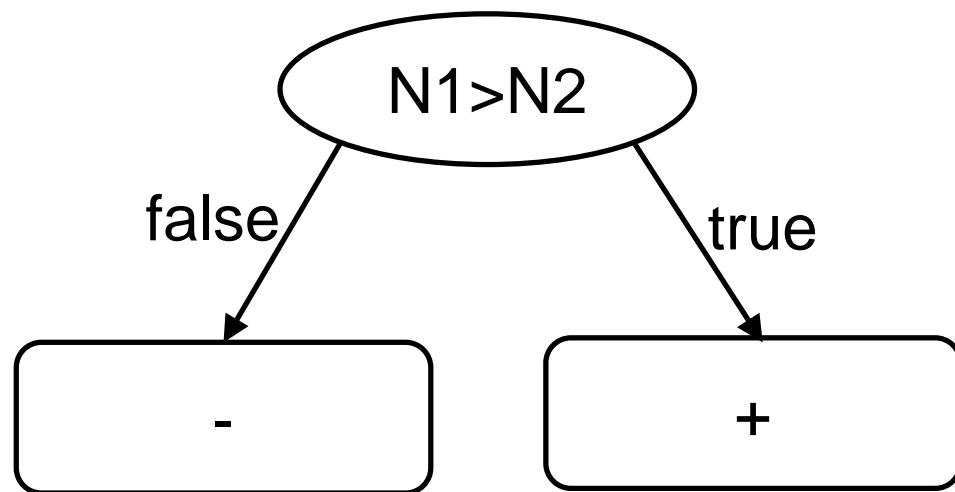
# Valores desconocidos (*Missing values*)

- ¿Cómo gestionan los árboles los *missing values*?
- Supongamos que el valor para “Pat” para una instancia es desconocido.
- Solución: el ejemplo con el valor desconocido baja ponderado por las tres hojas
- La validez de la división se calcula como antes



# Divisiones oblicuas

- El algoritmo CART permite hacer divisiones oblicuas, es decir, divisiones no ortogonales a los ejes de atributos.
- El algoritmo busca planos con buena reducción de impureza
- El crecimiento del árbol se ralentiza
- Pero los árboles son más expres



# Parámetros

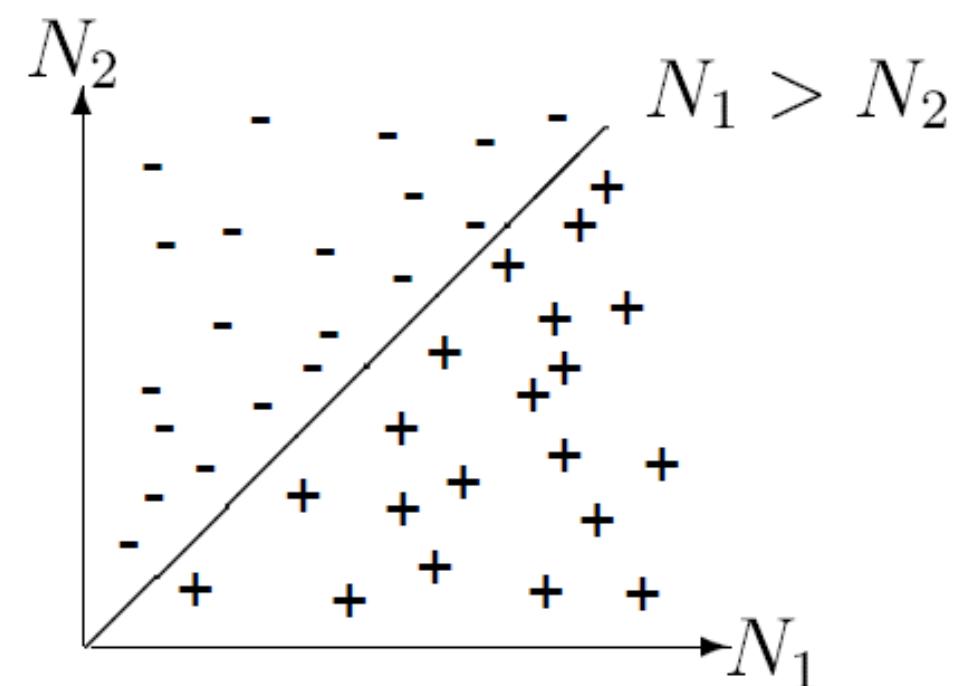
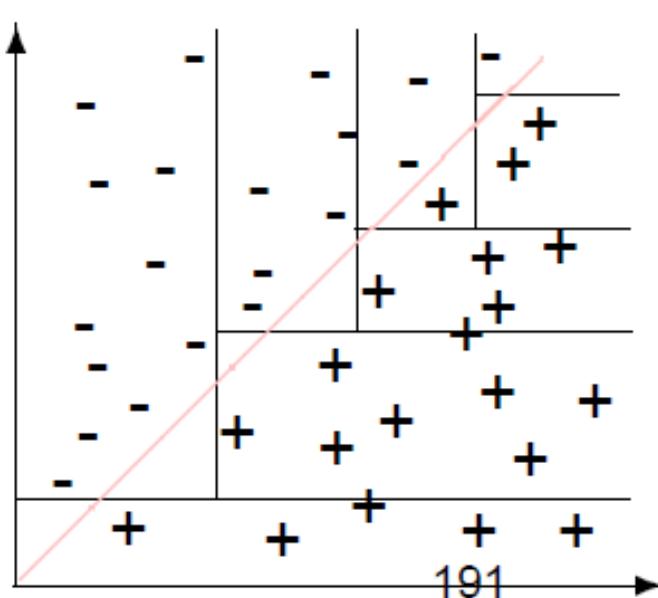
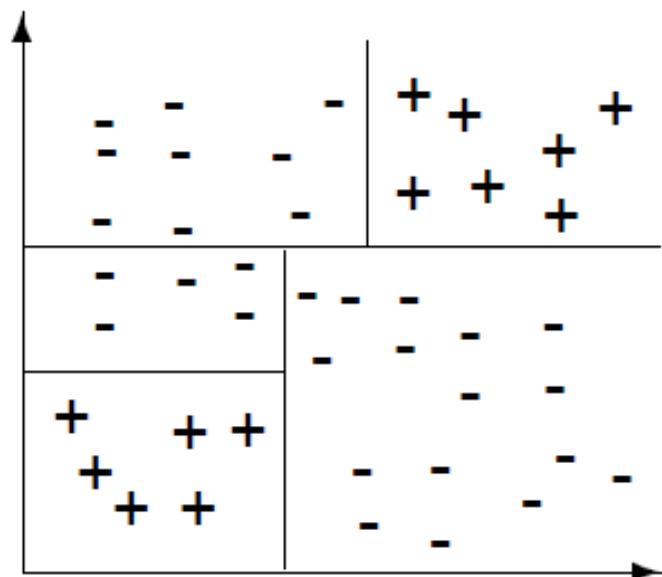
- Número mínimo de ejemplos necesarios para dividir una hoja.
- Aplicar o no poda
- Confianza en la poda ¿Cuándo podar?
- Si hay limitaciones de computacionales se puede limitar la profundidad o el número de nodos del árbol

# Detalles de los algoritmos

	Criterio de división	Criterio de poda	Otras características
CART	<ul style="list-style-type: none"><li>• Gini</li><li>• Twoing</li></ul>	Cost-complexity post-poda	<ul style="list-style-type: none"><li>• Regresión/Clasif.</li><li>• Atributos: categóricos /numéricos</li><li>• <i>Missing values</i></li><li>• Divisiones oblicuas</li><li>• Divisiones de atribs. categóricos por agrupaciones</li></ul>
ID3	Information Gain (IG)	Pre-poda	<ul style="list-style-type: none"><li>• Clasificación</li><li>• Atributos categóricos</li></ul>
C4.5	<ul style="list-style-type: none"><li>• Information Gain (IG)</li><li>• Information Gain Ratio (IGR)</li></ul>	Post-poda estadística	<ul style="list-style-type: none"><li>• Clasificación</li><li>• Atributos: categóricos /numéricos</li><li>• <i>Missing values</i></li><li>• Generador de reglas</li><li>• Divisiones múltiples.</li></ul>

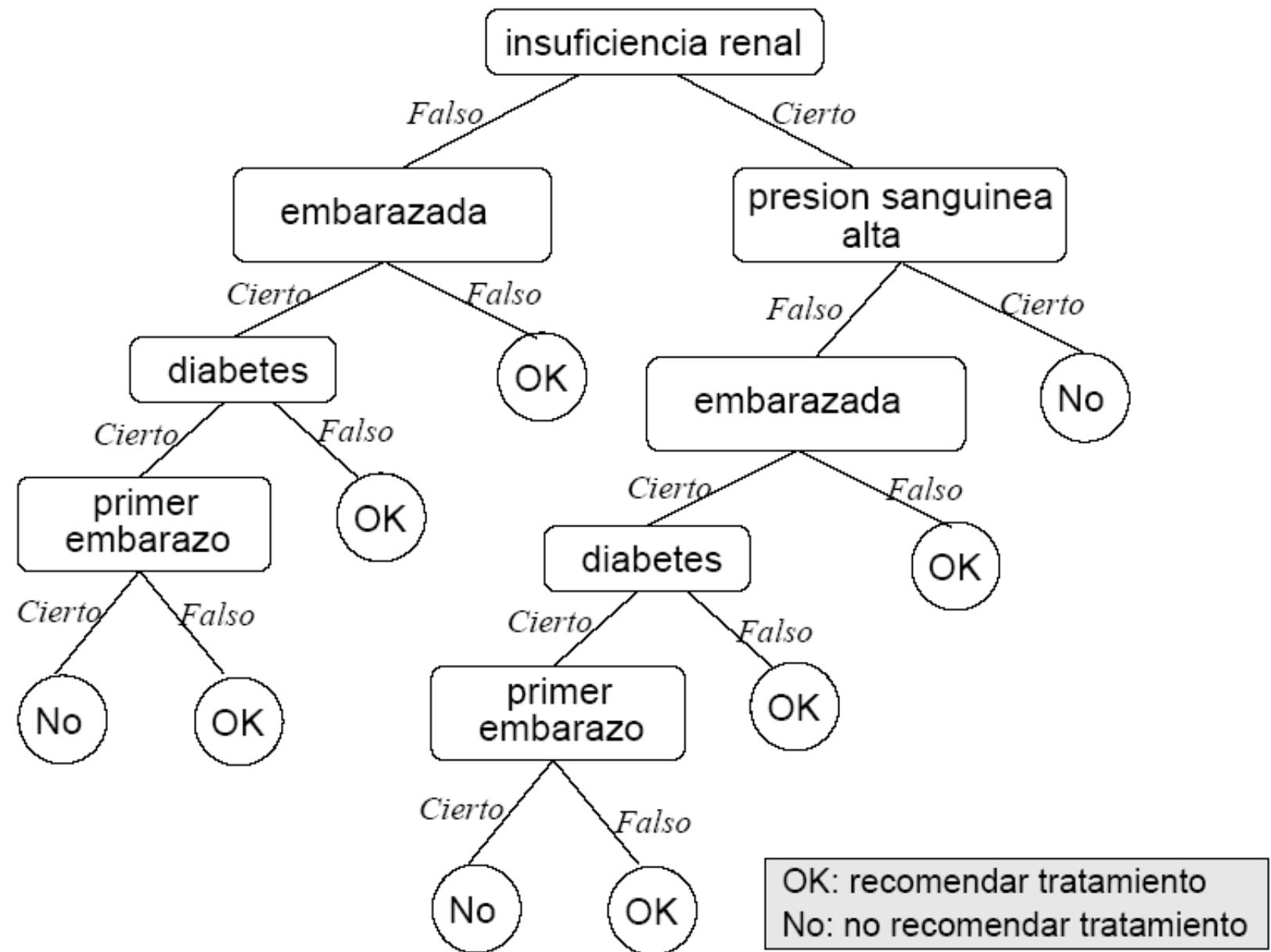
# Desventajas de usar árboles

- ¡Ninguna! Bueno alguna sí...
- No manejan bien interacciones complejas entre atributos. Falta de poder expresivo



# Desventajas de usar árboles

- Problema de replicación. Se puede acabar con árboles similares es regiones excluyentes



# Ventajas de usar árboles

- Interpretables. Fáciles de entender para no expertos. Se pueden convertir a reglas.
- Manejan atributos nominales y numéricos.
- Gestionan bien atributos no informativos o redundantes.
- Pueden trabajar con *missing values*.
- Método no paramétrico. No hay una idea predefinida sobre el concepto a aprender
- Fácil de hacer el ajuste de parámetros. Tienen muy pocos parámetros

# Árboles en sklearn I

## Modelos:

```
DecisionTreeClassifier(criterion='gini', splitter='best', max_depth=None,  
min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0,  
max_features=None, random_state=None, max_leaf_nodes=None,  
min_impurity_split=1e-07, class_weight=None, presort=False)
```

```
DecisionTreeRegressor(criterion='mse', splitter='best', max_depth=None,  
min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0,  
max_features=None, random_state=None, max_leaf_nodes=None,  
min_impurity_split=1e-07, presort=False)
```

## Librería:

```
from sklearn import tree
```

## Ejemplo:

```
from sklearn import tree  
  
# Crear un árbol con los parámetros por defecto  
clf = tree.DecisionTreeClassifier()
```

# Árboles en sklearn II

## Parámetros de tree:

- **criterion**: Criterio de división.
  - Clasificación (por omisión="gini"): Puede ser "gini" o "entropy"
  - Regresión (por omisión="mse"): Puede ser "mse" (error cuadrático medio) o "mae" (error absoluto medio)
- **min\_samples\_leaf** (=1): Número mínimo de ejemplos que debe haber en cada hoja creada.
- **max\_depth** (=None): Límite a la profundidad máxima del árbol.
- **min\_samples\_split** (=2), **min\_weight\_fraction\_leaf** (=0.0), **max\_leaf\_nodes** (=None), **min\_impurity\_split** (=1e-07): Otros criterios de prenda

Criterios de prenda

# Árboles en sklearn III

## Parámetros de tree:

- **max\_features** (=None): Número de atributos sobre el que se extrae la mejor división. Otros valores pueden ser: ‘sqrt’, ‘auto’, ‘log’. Esto se usa junto con Random Forest generalmente
- **random\_state** (=None): Semilla aleatoria que determina las operaciones aleatorias. Para una misma semilla y datos sale en mismo árbol.
- **[Solo clasificación] class\_weight** (=None): Modificar la importancia de cada clase. Útil cuando tenemos conjuntos desequilibrados. Se usa poniendo ‘balanced’

Ejemplo: Crear un modelo usando IG y un mínimo de 10 ejemplos por hoja

```
clf = tree.DecisionTreeClassifier(criterion="entropy", min_samples_leaf=10)
```

# Ejemplo de clasificación en sklearn

```
# Cargar de datos
fP = 'pimaND.csv'
d = pd.read_csv(fP, sep=', ')

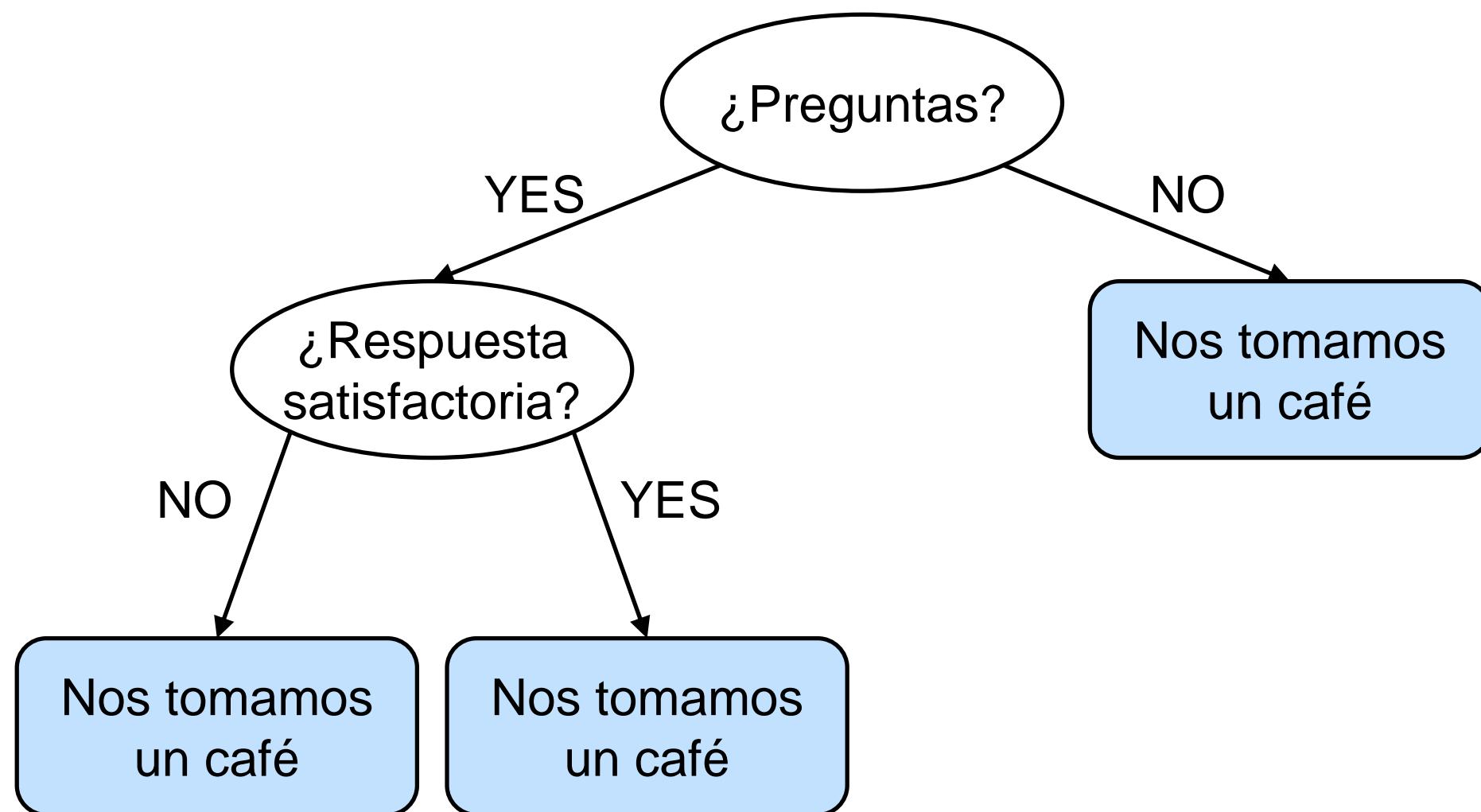
# Creamos las particiones en entrenamiento y test para probar el modelo
# (pe 10-fold cross validation) (version 0.18 de sklearn)
indexFolds = StratifiedKFold(n_folds=10, shuffle=True, random_state=11)

# Creamos clasificador
clf = tree.DecisionTreeClassifier()

aciertos= []
# Bucle que recorre las particiones
for idxTr, idxTs in indexFolds.split(d.values):
    # Entrenar modelo en datos de train
    clf.fit(d.values[idxTr,:-1],d.values[idxTr,-1])
    # Validar modelo en datos de test
    acierto = clf.score(d.values[idxTs,:-1],d.values[idxTs,-1])
    aciertos.append(acierto)

# Salida
aciertos = np.array(aciertos)
print "%0.3g%%" % (100*aciertos.mean()) + " +- %.3g" % (100*aciertos.std())
```

# Fin



# Conjunto de clasificadores

# Guión

- ¿Qué es un conjunto de clasificadores? ¿Cómo construirlo?
- Bagging, Boosting, Random forests, class-switching, xgBoost
- Combinar salida
- *Stacking*
- Otras técnicas
- ¿Por qué funcionan? Historias de éxito

# Teorema del jurado de Condorcet

- Alcanzar decisiones mediante la discusión de opiniones forma parte del ser humano
- Se formaliza en el teorema del jurado de Condorcet que dice que dado un jurado y suponiendo:
  1. que comenten errores independientes
  2. que la probabilidad de cada miembro del jurado de acertar es superior al 50%

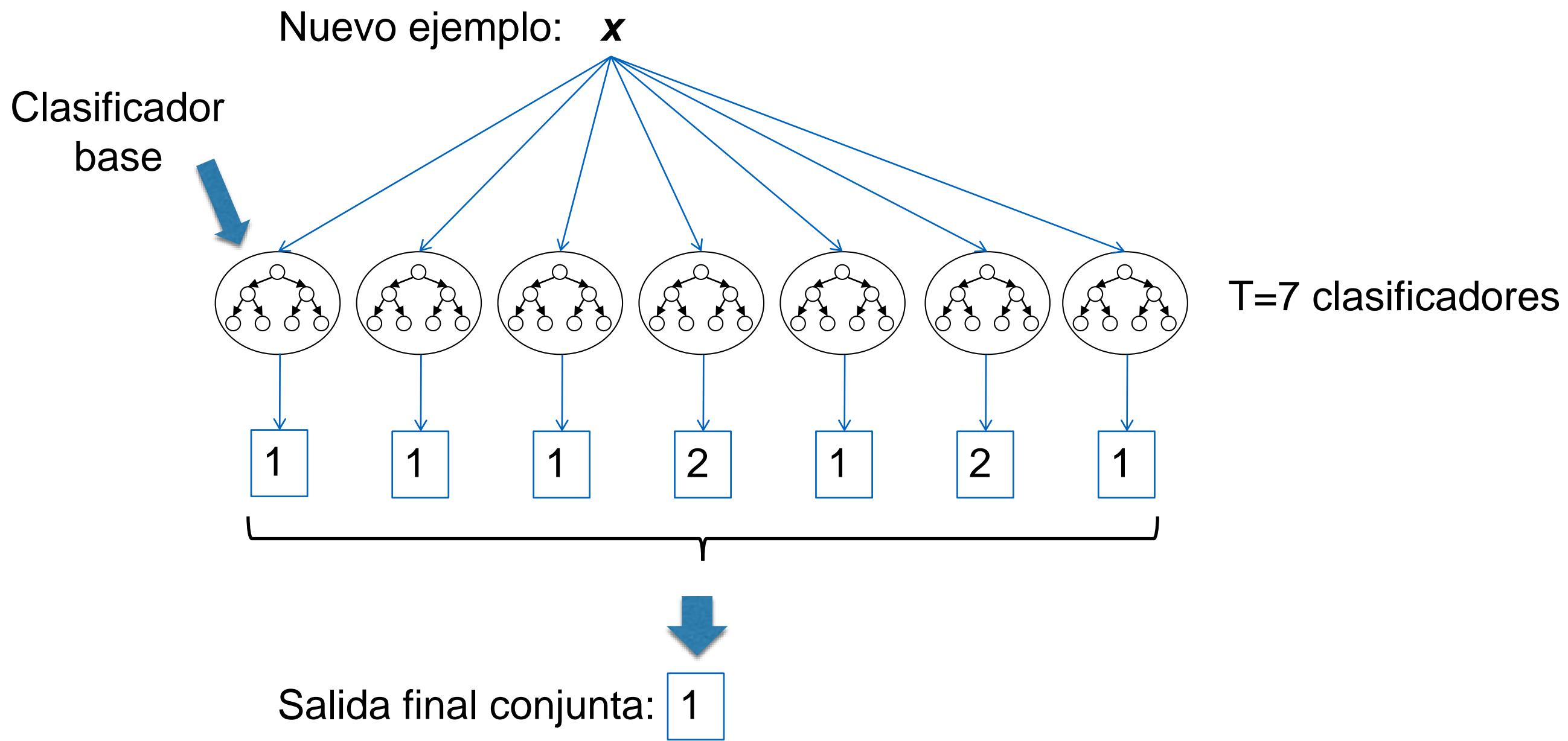
La probabilidad del jurado de acertar tiende a 100% al aumentar el numero de miembros del jurado



Nicolas de Condorcet (1743-1794),  
French mathematician

# ¿Qué es un conjunto de clasificadores?

- Es una combinación de clasificadores que dan una salida final conjunta.



# Idea general

- Generar muchos clasificadores y combinarlos para dar una clasificación final
- Funcionan muy bien. En general, mejor que cualquiera de los modelos que los componen
- Los clasificadores a combinar deben ser distintos
- La clave es la generación de clasificadores diversos a partir de los datos disponibles

# ¿Cómo construirlos?

- Hay varias técnicas para construir un conjunto de clasificadores diversos:
  - Utilizar distintas versiones modificadas aleatoriamente de los datos de entrenamiento para crear cada clasificador base
  - Inyectar cambios en los algoritmos de aprendizaje que introduzcan aleatorización de los modelos
- Estas estrategias se pueden usar en combinación.
- Generalmente, cuanto mayor sea la aleatorización, mejores serán los resultados

# ¿Cómo construirlos?

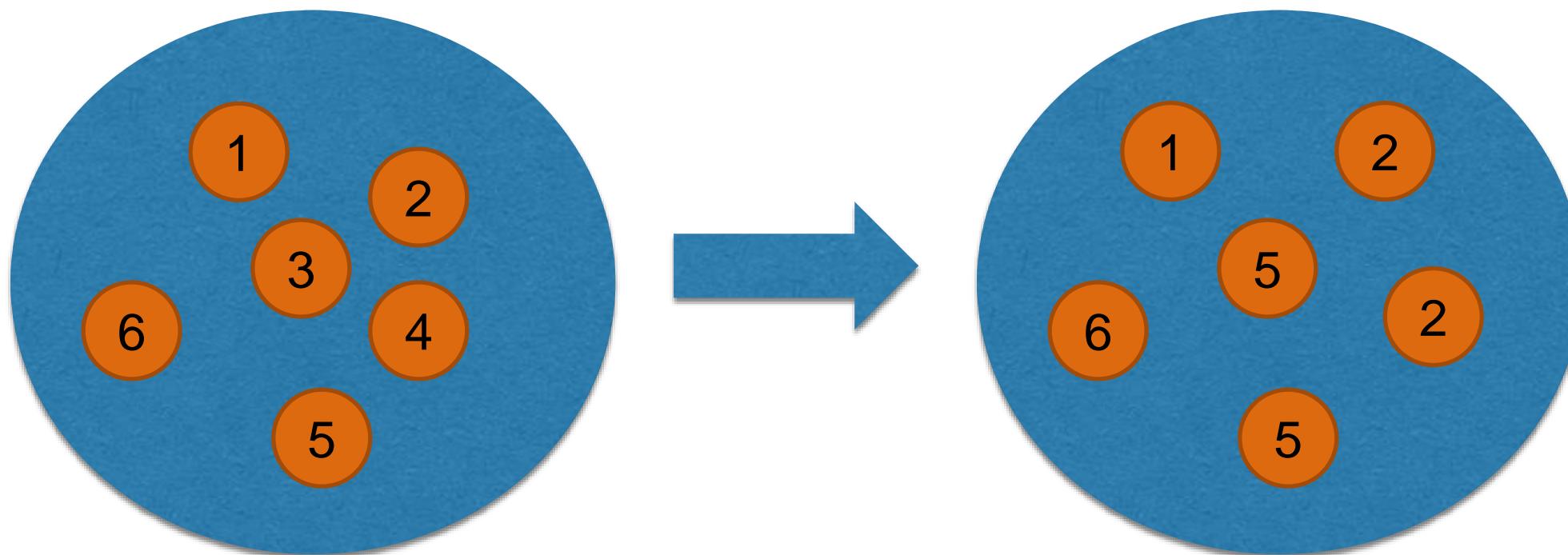
- Modificaciones aleatorias de los datos se pueden generar mediante:
  - Remuestreo de los datos. Por ejemplo por bootstrap (p.e. en bagging), muestreo ponderado (p.e en boosting).
  - Alteración de atributos: Los clasificadores base se entrena n usando distintos subconjuntos de atributos (p.e. en Random subspaces)
  - Modificando las clases: Agrupando clases en dos nuevas superclases aleatoriamente (p.e. ECOC) o modificando las etiquetas aleatoriamente (p.e. Class-switching)

# ¿Cómo construirlos?

- Aleatorizando los algoritmos de aprendizaje
  - Introducir cierta aleatorización en los algoritmos de aprendizaje, de forma que dos ejecuciones consecutivas den clasificadores distintos
  - Ejecutando clasificadores base distintos o con distintos parámetros, etc.

# Muestra *bootstrap* (inciso)

- Dado un conjunto de  $N$  elementos, una muestra *bootstrap* consiste en extraer  $N$  elementos con reemplazamiento.



6 extracciones con repetición

# Bagging

**Input:**

Dataset  $D = (\mathbf{x}_i, y_i) \ i=1 \dots N$

Ensemble size  $T$

```
1. for  $t=1$  to  $T$ :  
2.     sample = BootstrapSample( $D$ )  
3.      $h_t$  = TrainClassifier(sample)
```

*Bootstrap*

+

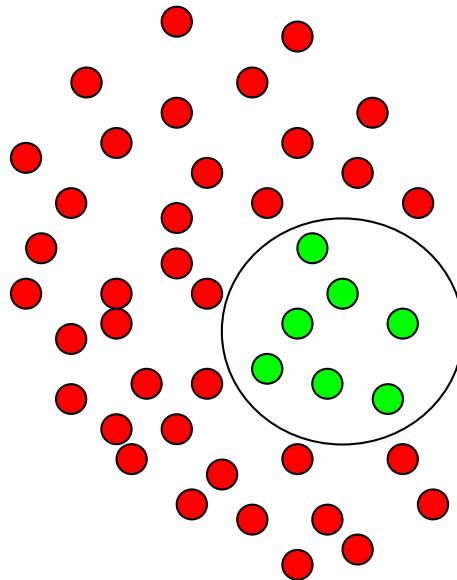
**Output:**

$$H(\mathbf{x}) = \operatorname{argmax}_j \left( \sum_{t=1}^T I(h_t(\mathbf{x}) = j) \right)$$

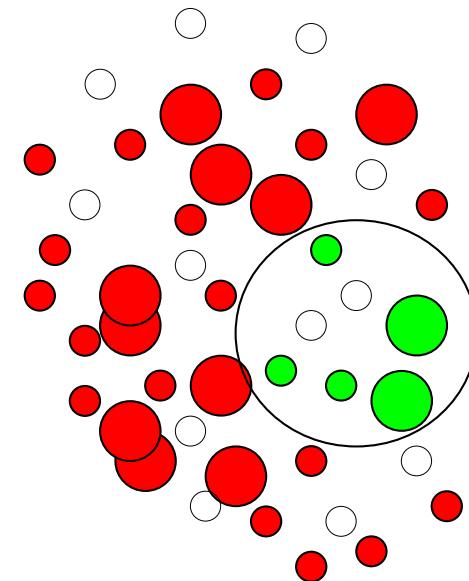
*Agregación  
(por voto)*

# Bagging

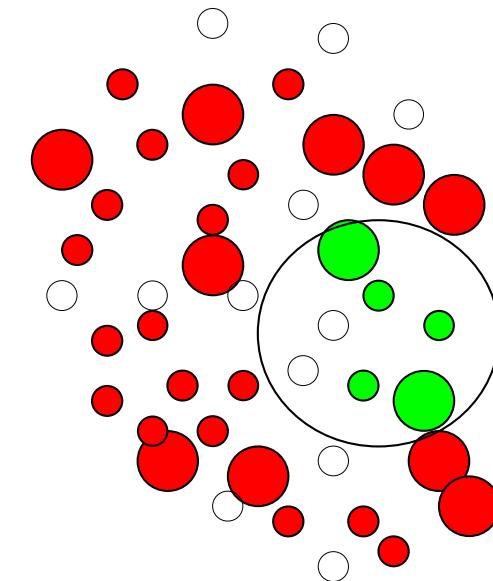
Original dataset



Bootstrap sample 1



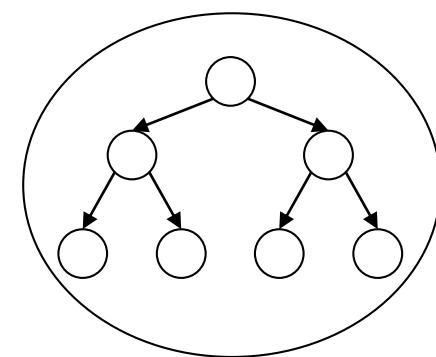
Bootstrap sample T



Repeated example



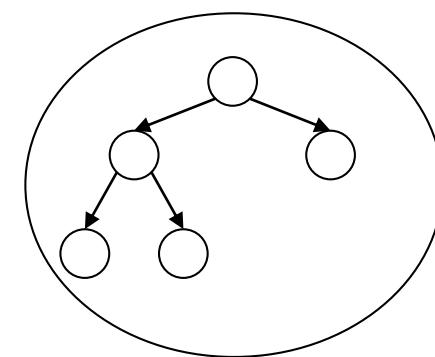
Removed example



...



...



# Consideraciones sobre bagging

- Use un 63,2% de los datos de entrenamiento en media para construir cada clasificador.
- Muy robusto frente a ruido en las etiquetas de clase.
- En general, mejora el rendimiento del clasificador base.
- Se puede paralelizar fácilmente

# Random forest

- Breiman define los bosques aleatorios (Random forests) como un conjunto de clasificadores tal que:
  - Tiene árboles de decisión como algoritmo base
  - Introduce aleatorización en el proceso de entrenamiento
  - Bajo esta definición Bagging con árboles de decisión es un random forest y de hecho lo es. Sin embargo...

# Random forest

- En la práctica, se considera que un random forest es:
  - Cada árbol se genera, como en bagging, usando muestras *bootstrap*
  - Los árboles se construyen de forma que cada división se calcula usando:
    - Un subconjunto aleatorio de los atributos
    - Se selecciona la mejor división de ese subconjunto de atributos
    - Se usan árboles sin podar

# Consideraciones sobre random forests

- Su rendimiento es mejor que boosting en media. Y de hecho es uno de los mejores clasificadores.
- Es muy robusto al ruido (¡¡¡no sobre-ajusta!!!)
- Random forest introduce un mecanismo adicional de aleatorización con respecto a *bagging*
- Fácilmente paralelizable
- Los árboles aleatorios nos muy rápidos de construir

## Abstract

---

We evaluate **179 classifiers** arising from **17 families** (discriminant analysis, Bayesian, neural networks, support vector machines, decision trees, rule-based classifiers, boosting, bagging, stacking, random forests and other ensembles, generalized linear models, nearest-neighbors, partial least squares and principal component regression, logistic and multinomial regression, multiple adaptive regression splines and other methods), implemented in Weka, R (with and without the caret package), C and Matlab, including all the relevant classifiers available today. We use **121 data sets**, which represent **the whole UCI** data base (excluding the large-scale problems) and other own real problems, in order to achieve significant conclusions about the classifier behavior, not dependent on the data set collection. **The classifiers most likely to be the bests are the random forest (RF)** versions, the best of which (implemented in R and accessed via caret) achieves 94.1% of the maximum accuracy overcoming 90% in the 84.3% of the data sets. However, the difference is not statistically significant with the second best, the SVM with Gaussian kernel implemented in C using LibSVM, which achieves 92.3% of the maximum accuracy. A few models are clearly better than the remaining ones: random forest, SVM with Gaussian and polynomial kernels, extreme learning machine with Gaussian kernel, C5.0 and avNNet (a committee of multi-layer perceptrons implemented in R with the caret package). The random forest is clearly the best family of classifiers (3 out of 5 bests classifiers are RF), followed by SVM (4 classifiers in the top-10), neural networks and boosting ensembles (5 and 3 members in the top-20, respectively).

Do we Need Hundreds of Classifiers to Solve Real World Classification Problems? Manuel Fernández-Delgado, Eva Cernadas, Senén Barro, Dinani Amorim; 15(Oct):3133–3181, 2014.

# Boosting

## Entrada:

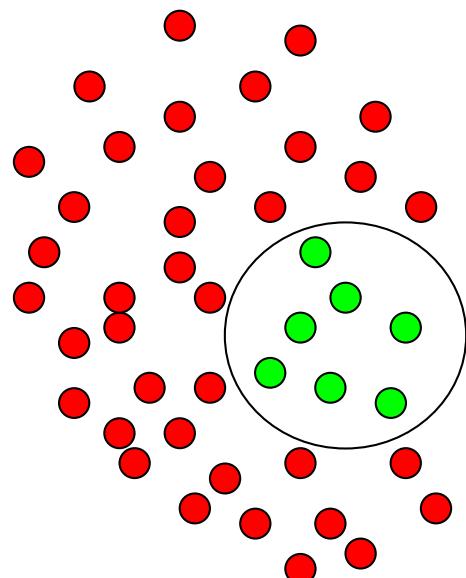
Datos  $D = (\mathbf{x}_i, y_i)$   $i=1\dots N$

Tamaño del conjunto  $T$

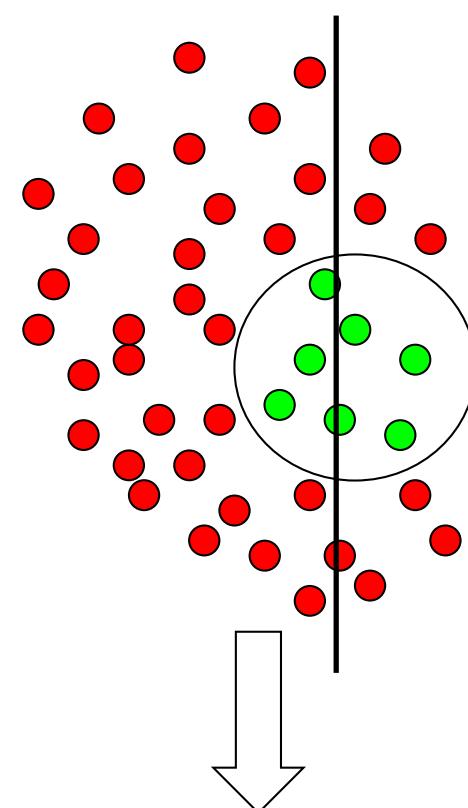
1. Asignar pesos a los ejemplos a  $1/N$
2. **for**  $t=1$  **to**  $T$ :
3.      $h_t = \text{BuildClassifier}(D, \text{pesos})$
4.      $e_t = \text{WeightedError}(D, \text{pesos})$
5.     **if**  $e_t == 0$  **or**  $e_t \geq 0.5$  **break**
6.     Multiplicar los pesos de los ejemplos  
             mal clasificados  $h_t$  por  $e_t / (1 - e_t)$
7.     Normalizar pesos

# Boosting

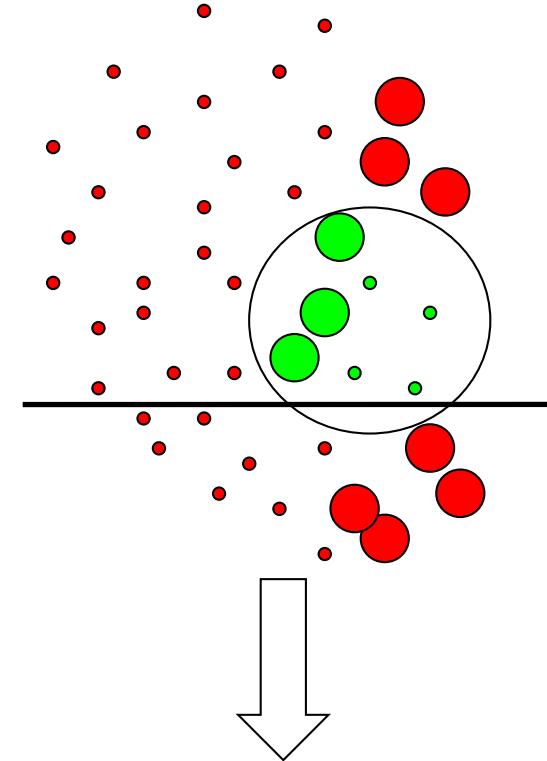
Original dataset



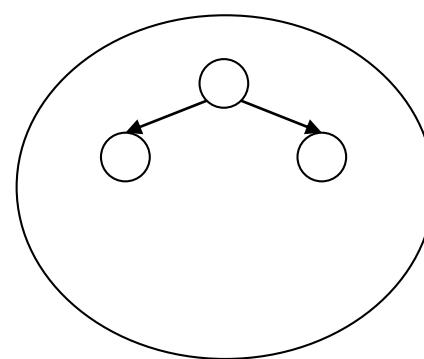
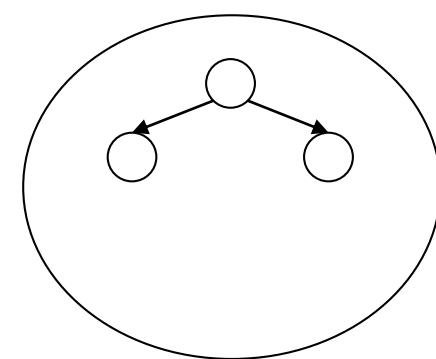
Iteration 1



Iteration 2



...



...

# Consideraciones sobre boosting

- Obtiene muy buenos resultados en general
- No es robusto frente a ruido en las etiquetas de clase
- Puede incrementar el error del clasificador base
- No se puede implementar fácilmente en paralelo

# GradientBoosting

## Entrada:

Datos  $D = \{(\mathbf{x}_i, y_i)\} \ i=1\dots N$

Tamaño del conjunto  $T$

Función de pérdida  $L$

1.  $F_0(\mathbf{x}) = \operatorname{argmin}_p \sum_{i=1}^N L(y_i, p)$
2. **for**  $t=1$  **to**  $T$ :
3.  $r_i = \text{CalcularResiduos}(F_{t-1}(\mathbf{x}_i), y_i)$
4.  $h_t = \text{TrainRegressor}(\{(\mathbf{x}_i, r_i)\}_{i=1}^N)$
5.  $\rho_t = \operatorname{argmin}_\rho \sum_{i=1}^N L(y_i, F_{t-1}(\mathbf{x}_i) + \rho h_t(\mathbf{x}_i))$
6.  $F_t(\mathbf{x}) = F_{t-1}(\mathbf{x}) + \rho_t h_t(\mathbf{x})$
7. **return**  $F_T(\mathbf{x})$

# GradientBoosting para pérdida cuadrática

## Entrada:

Datos  $D = \{(\mathbf{x}_i, y_i)\} \ i=1\dots N$

Tamaño del conjunto  $T$

Función de pérdida  $L = (F(x) - y)^2$

1.  $F_0 = \text{average}(\mathbf{y})$  // media del objetivo

2. **for**  $t=1$  **to**  $T$ :

3.  $r_i = y_i - F_{t-1}(\mathbf{x}_i)$  para  $i=1\dots N$

4.  $h_t = \text{TrainRegressor}(\{(\mathbf{x}_i, r_i)\}_{i=1}^N)$

5.  $\rho_t = 1$

5.  $F_t(x) = F_{t-1}(x) + \rho_t h_t(x)$

6. **return**  $F_T(x)$

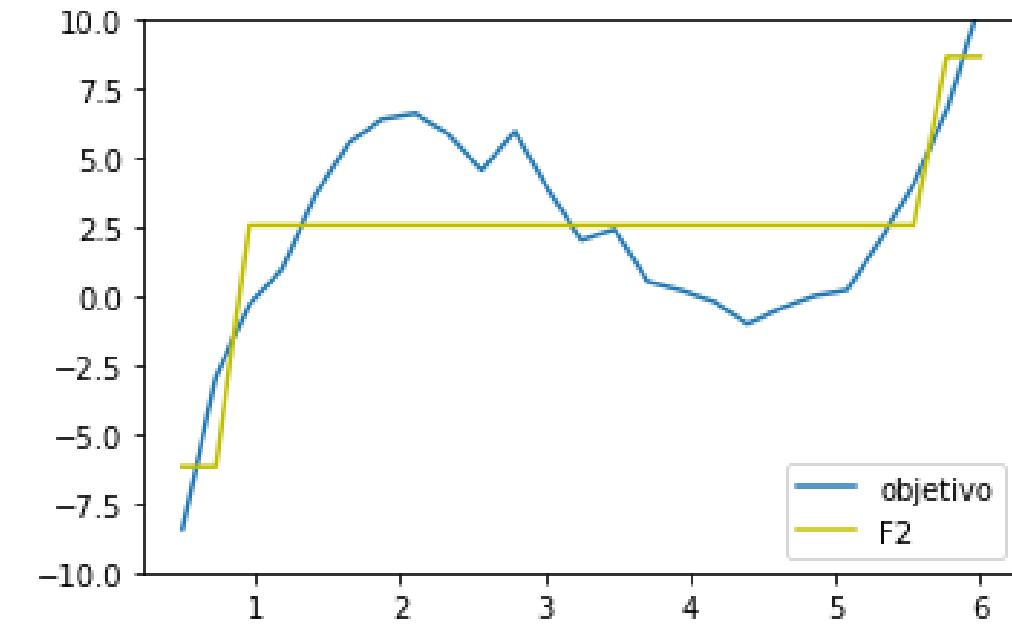
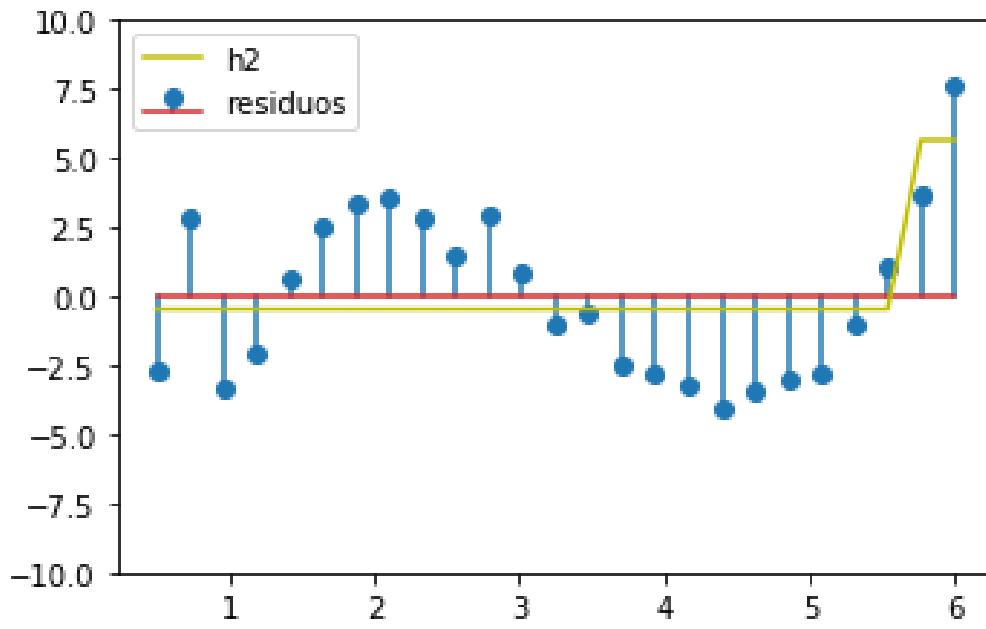
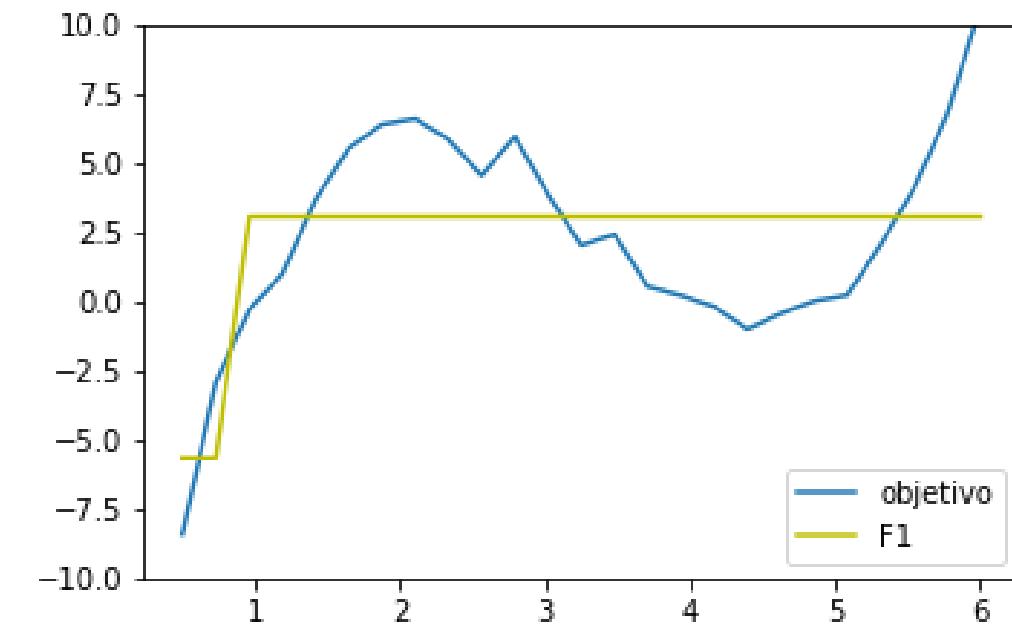
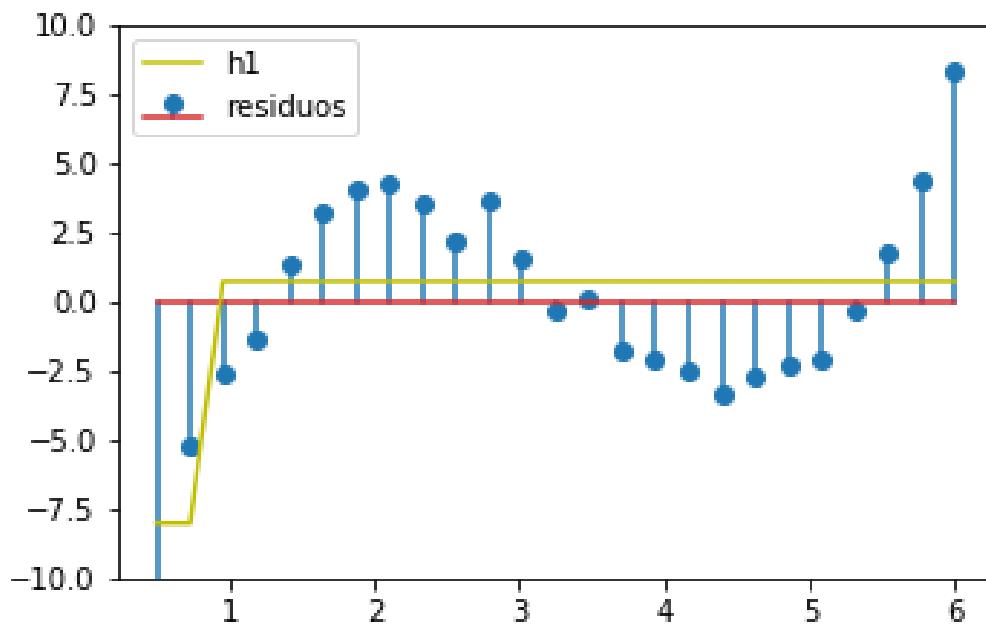
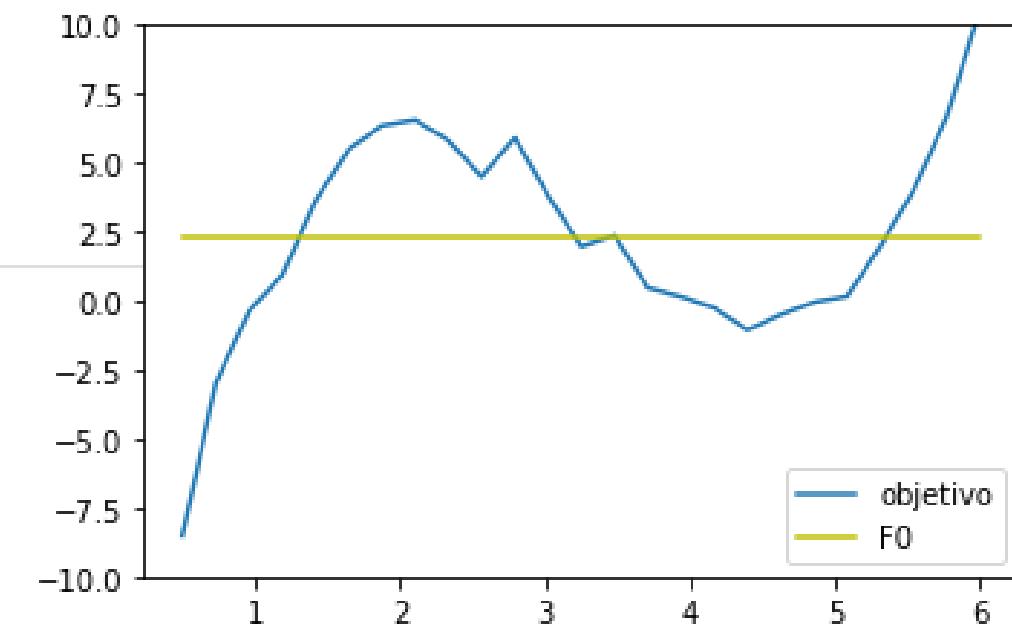
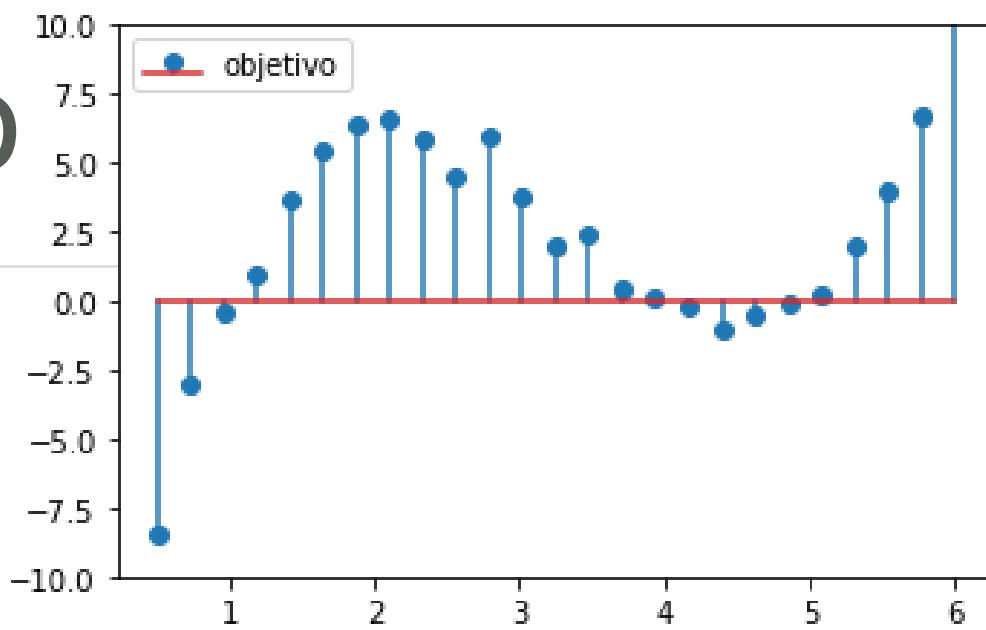
# Variantes de Gradient boosting

- Tasa de aprendizaje: la regla de adición cambia

$$F_m(x) = F_{m-1}(x) + \nu \cdot \gamma_m h_m(x), \quad 0 < \nu \leq 1.$$

- Controlar la complejidad de los árboles: número de hojas, profundidad, etc.
- Stochastic gradient boosting: Usar muestras bootstrap sin replazamiento para entrenar los modelos

# Ejemplo GB



# Consideraciones sobre GradientBoosting

- Obtiene muy buenos resultados en general
- Casi siempre se combina con árboles de decisión sencillos (de poca profundidad)
- Muchos parámetros a ajustar lo que permite que funcione de forma más estable que boosting

# XGBoost

- Un algoritmo basado en Gradient Boosting optimizado para ser mucho más rápido:
  - Permite trabajar con datos dispersos
  - Guarda los datos en bloques en un formato que evita tener que reordenar los atributos continuamente
- Incluye un término que penaliza la complejidad en la función de pérdida:

$$\mathbb{E}_{x,y} [L(y, F(x))] + \sum_{m=1}^M \Omega(f_m)$$

# XGBoost

- Funcionalidades adicionales:
  - Función de pérdida con un término de penalización de la complejidad.
  - Como Gradient boosting tiene los parámetros:
    - Tasa de aprendizaje
    - Muestreo bootstrap
    - Límites en profundidas, número de hojas, etc.
  - Como en random forest tiene muestreo por columnas

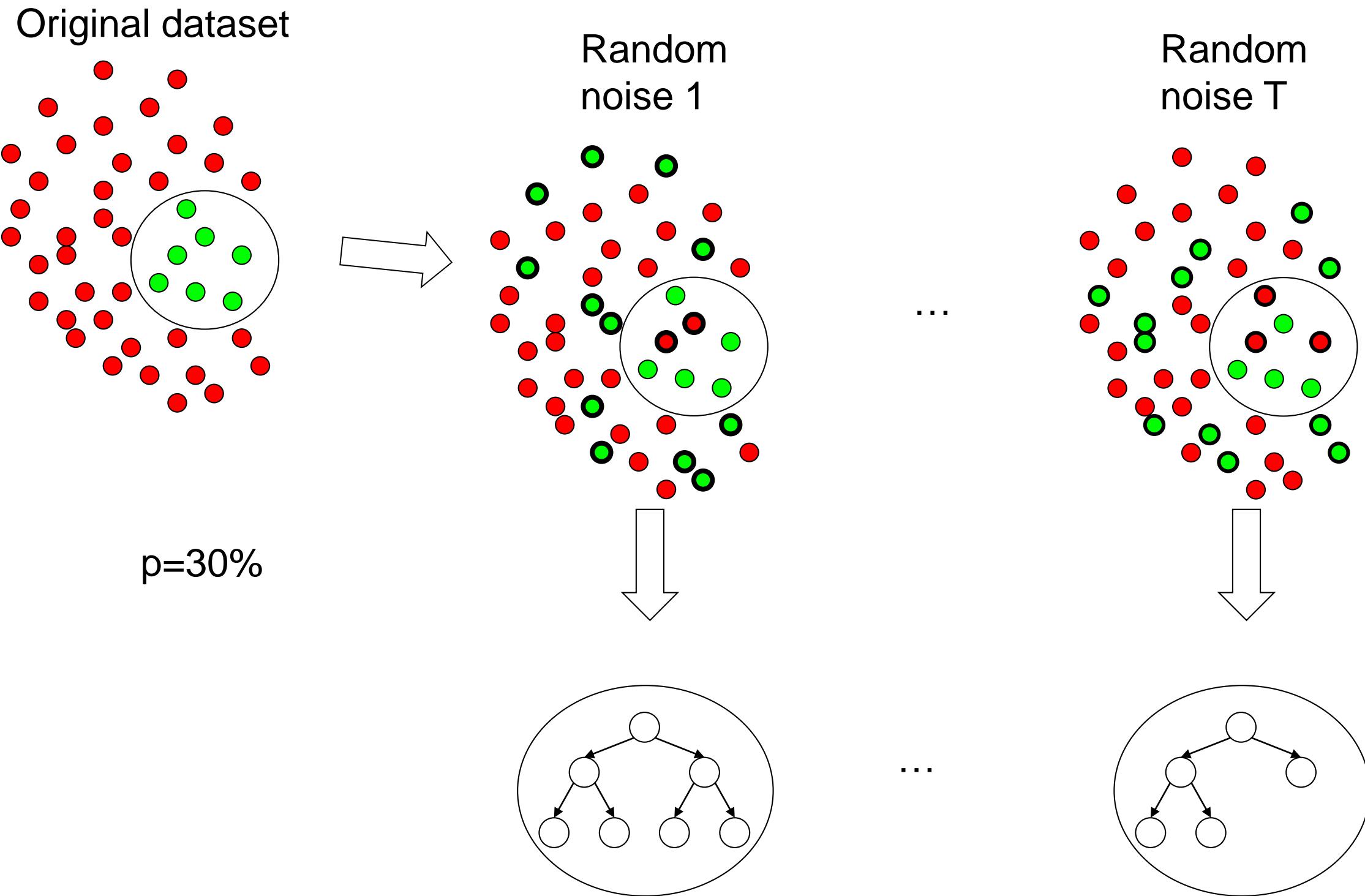
# Consideraciones sobre XGBoost

- Obtiene muy buenos resultados en general
- Tiene decenas de parámetros que hay que ajustar
- XGBoost está logrando los mejores resultados en competiciones de Kaggle.
- Fuente XGBoost: <https://github.com/dmlc/xgboost>

# Class switching

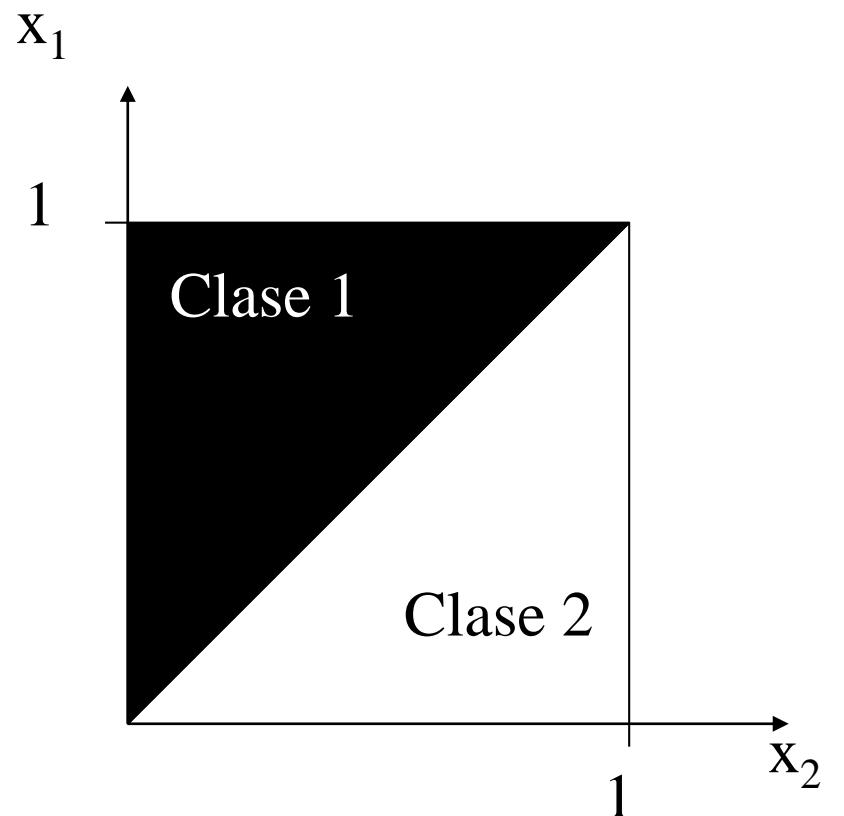
- Class switching es un conjunto donde la diversidad se obtiene usando versiones del conjunto de entrenamiento contaminadas con ruido aleatorio en las etiquetas de clase.
- En concreto, los conjuntos de entrenamiento necesarios para construir cada clasificador base, se crean modificando la clase de cada punto por otra con probabilidad  $p$ .

# Class switching

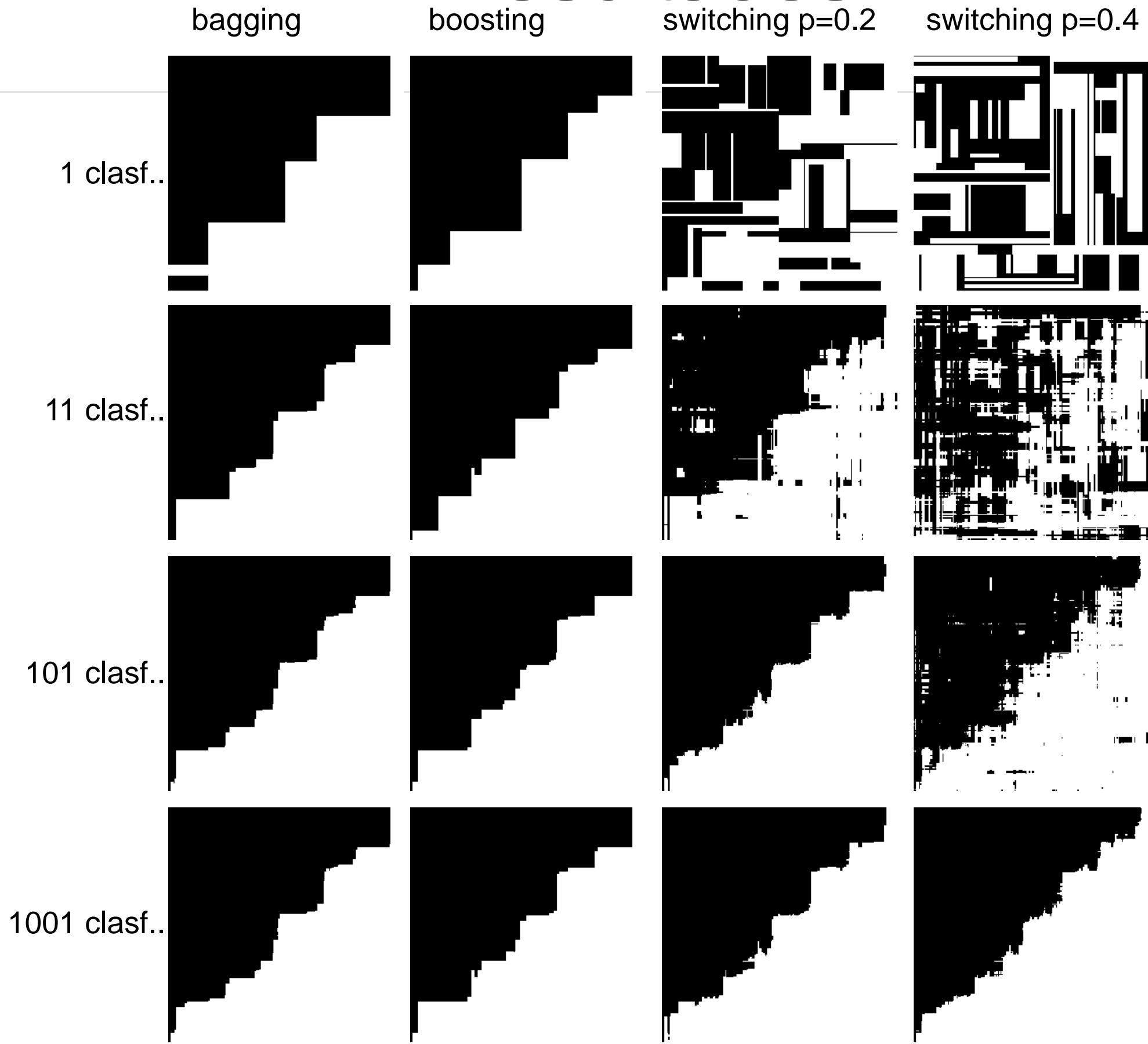


# Ejemplo

- 2D ejemplo
- Frontera is  $x_1=x_2$
- $x_1 \sim U[0, 1]$   $x_2 \sim U[0, 1]$
- No es fácil para un árbol de decisión
- Probamos *bagging*, *boosting* y *class-switching* con  $p=0.2$  y  $p=0.4$



# Resultados



# Parametrización

## Parámetros utilizados

	Clasificadores base	Tamaño del ensemble T	Otros parámetros /opciones
Bagging	Árboles no podados	$O(200)$ aunque cuantos más mejor	Usar submuestreo
Boosting	Árboles podados o Aprendices débiles	$O(100)$	
Random forest	Árboles no podados aleatorios	$O(200)$ aunque cuantos más mejor	No. de atributos aleatorios para las divisiones = $\log(\#features)$ o $\sqrt{\#features}$
Class-switching	Árboles no podados	$O(1000)$	% de ruido en las etiquetas, $p \sim 30\%$

# Parametrización

## Parámetros utilizados

	Clasificadores base	Tamaño del ensemble T	Otros parámetros /opciones
GradientBoosting	Árboles poco profundos	O(50-100)	<ul style="list-style-type: none"><li>• Submuestreo</li><li>• Varios parámetros de complejidad de los árboles</li><li>• Tasa de aprendizaje</li></ul>
XGBoost	Árboles poco profundos	O(50-100)	<ul style="list-style-type: none"><li>• Submuestreo</li><li>• Varios parámetros de complejidad de los árboles</li><li>• Tasa de aprendizaje</li><li>• Muestreo de columnas</li><li>• Y muchos más...</li></ul>

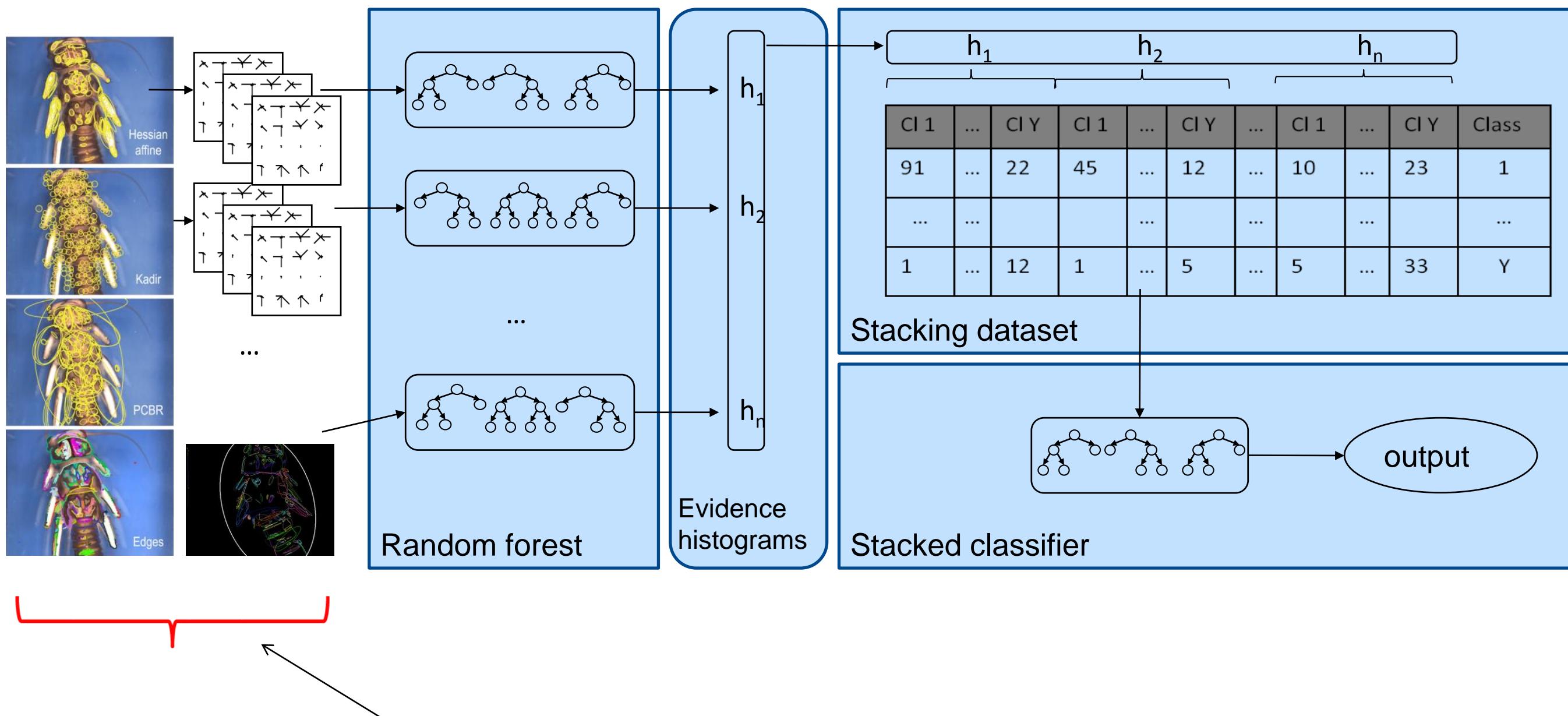
# Combinar la salida

- Las técnicas de combinación se pueden dividir en dos grupos:
  - Estrategias de voto: La salida final del conjunto es la clase más votada por los clasificadores base.  
También puede ser voto ponderado
  - Otras estrategias: Operadores como máximo, mínimo, producto, mediana y media se pueden utilizar para combinar las salidas de los clasificadores base.
- No hay una estrategia ganadora. Depende de muchos factores

# *Stacking*

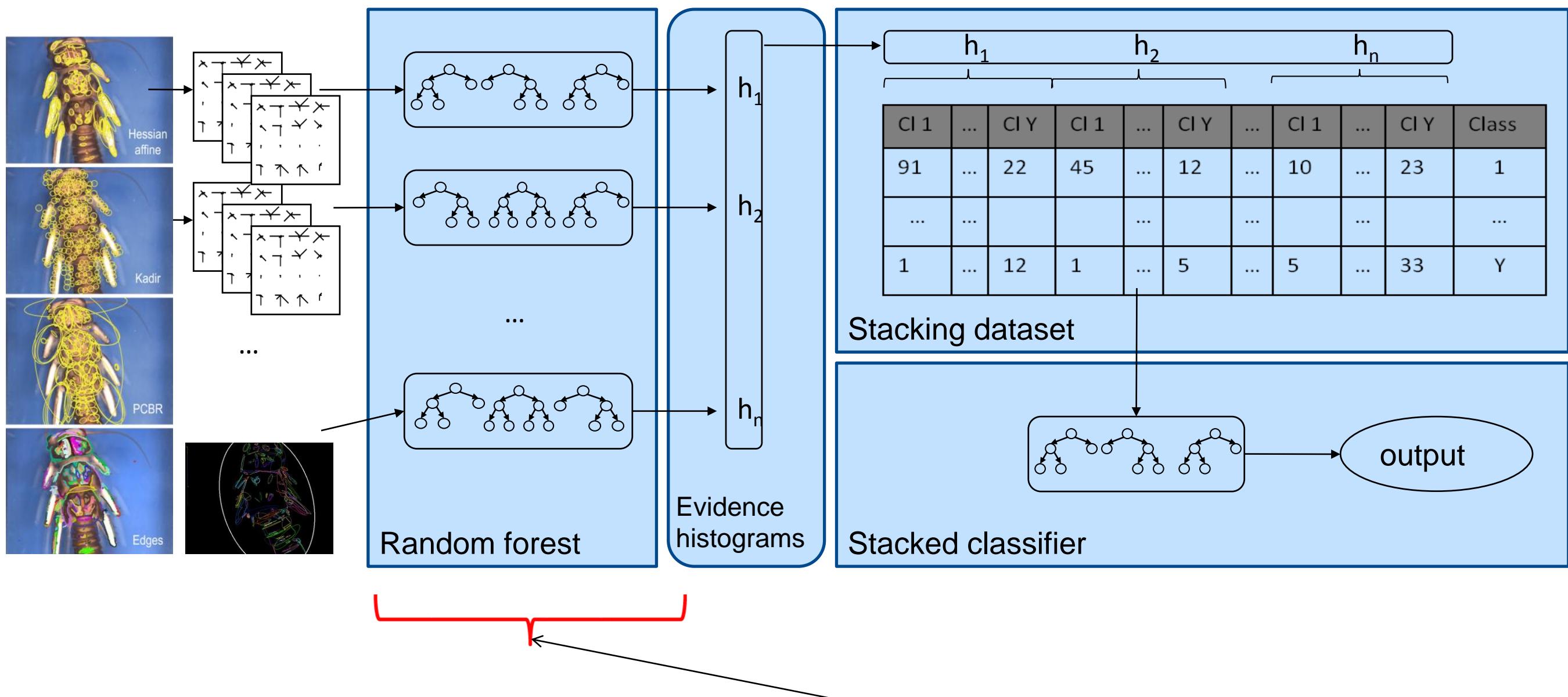
- En *stacking* la fase de combinación también se aprende.
- Primero construimos los clasificadores base usando unos datos de entrenamiento
- Después, las predicciones de estos clasificadores (usando otros datos) se usan como atributos para otro clasificador que aprende a combinarlas (meta-clasificador).

# Ejemplo de stacking



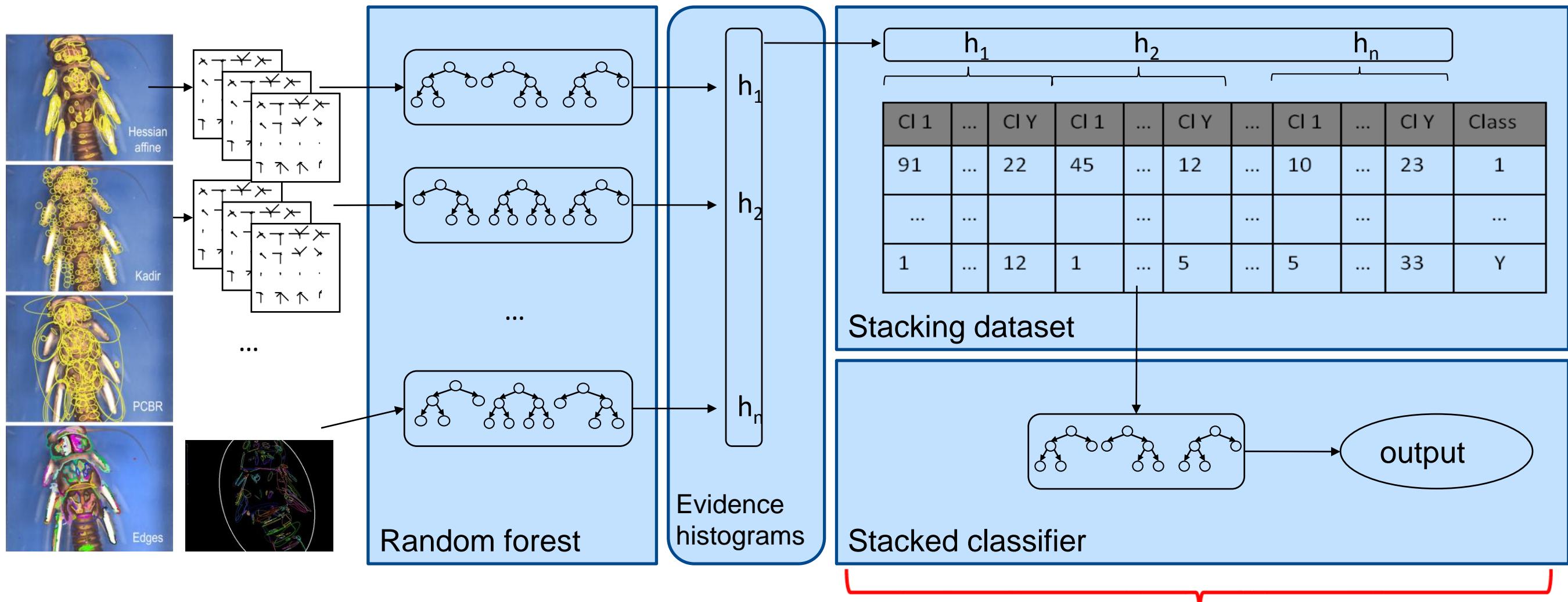
Extraer descriptores

# Ejemplo de stacking



1. Se construyen N Random forest en los descriptores

# Ejemplo de stacking



- En la segunda fase se aplica *stacking*:
  - Las salidas de los conjuntos de la primera fase se concatenan
  - Se aplica *boosting* a estos nuevos vectores para sacar la clasificación final.

# Poda de conjuntos

1.- El orden aleatorio dado por *bagging*

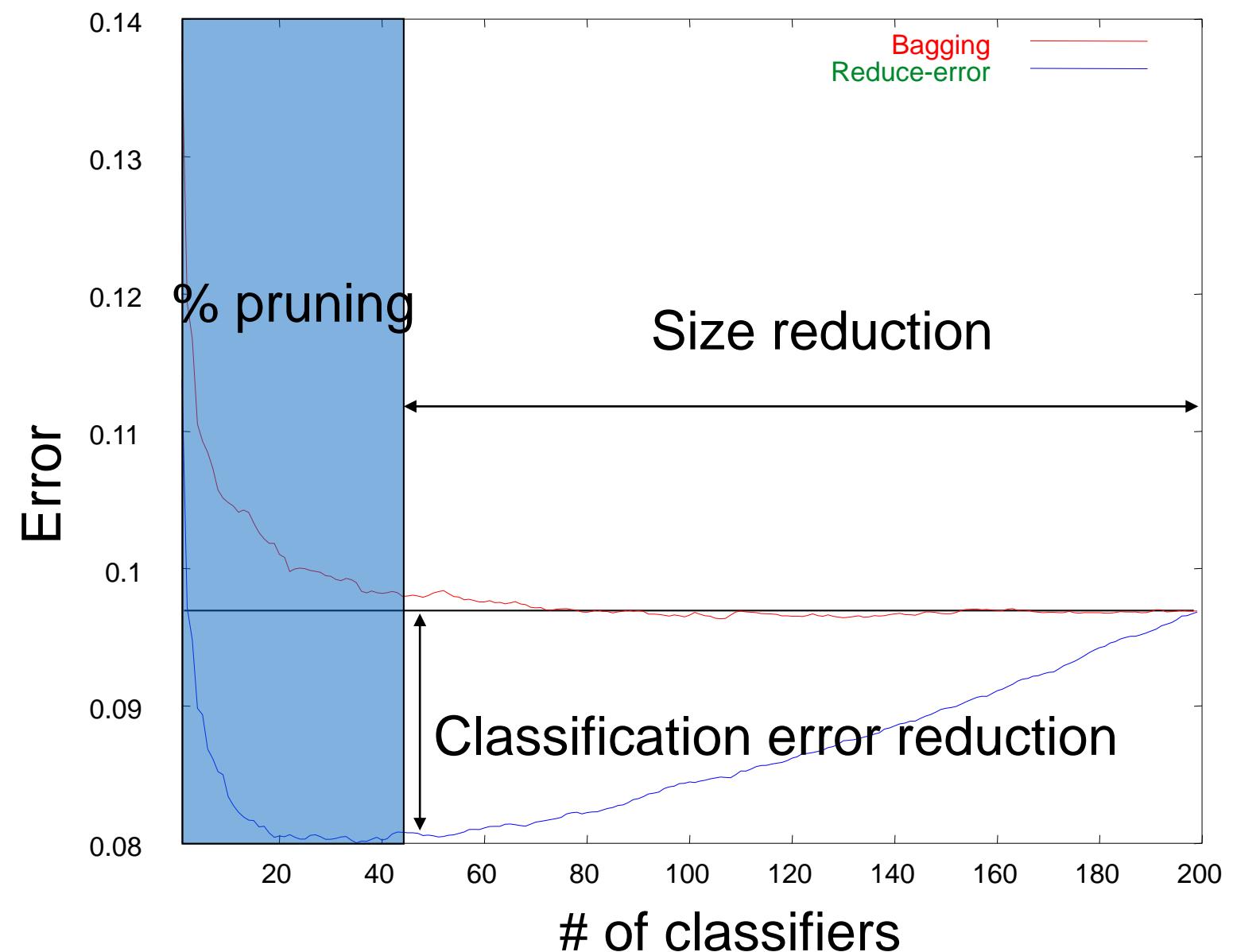
$h_1, h_2, h_3, \dots, h_T$

2.- Nueva ordenación

$h_{s1}, h_{s2}, \dots, h_{sT}$

3.- Poda

$h_{s1}, \dots, h_{sM}$

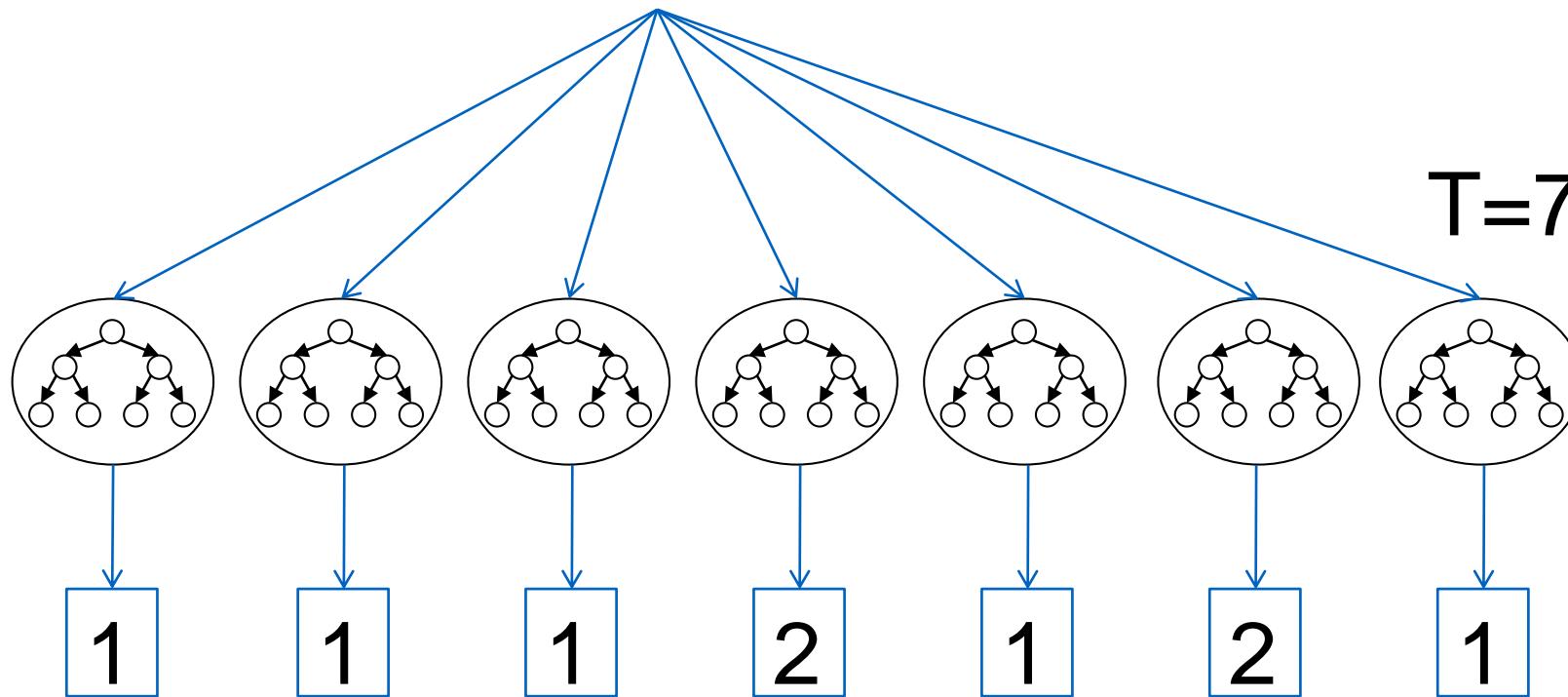


# Poda dinámica de conjuntos

Nuevo ejemplo:

**X**

T=7 clasificadores



Votos acumulados:  $t \rightarrow \begin{matrix} 5 & 2 \\ t_1 & t_2 \end{matrix} \longrightarrow \text{Clase final } 1$

¿Necesitamos preguntar a todos los clasificadores?

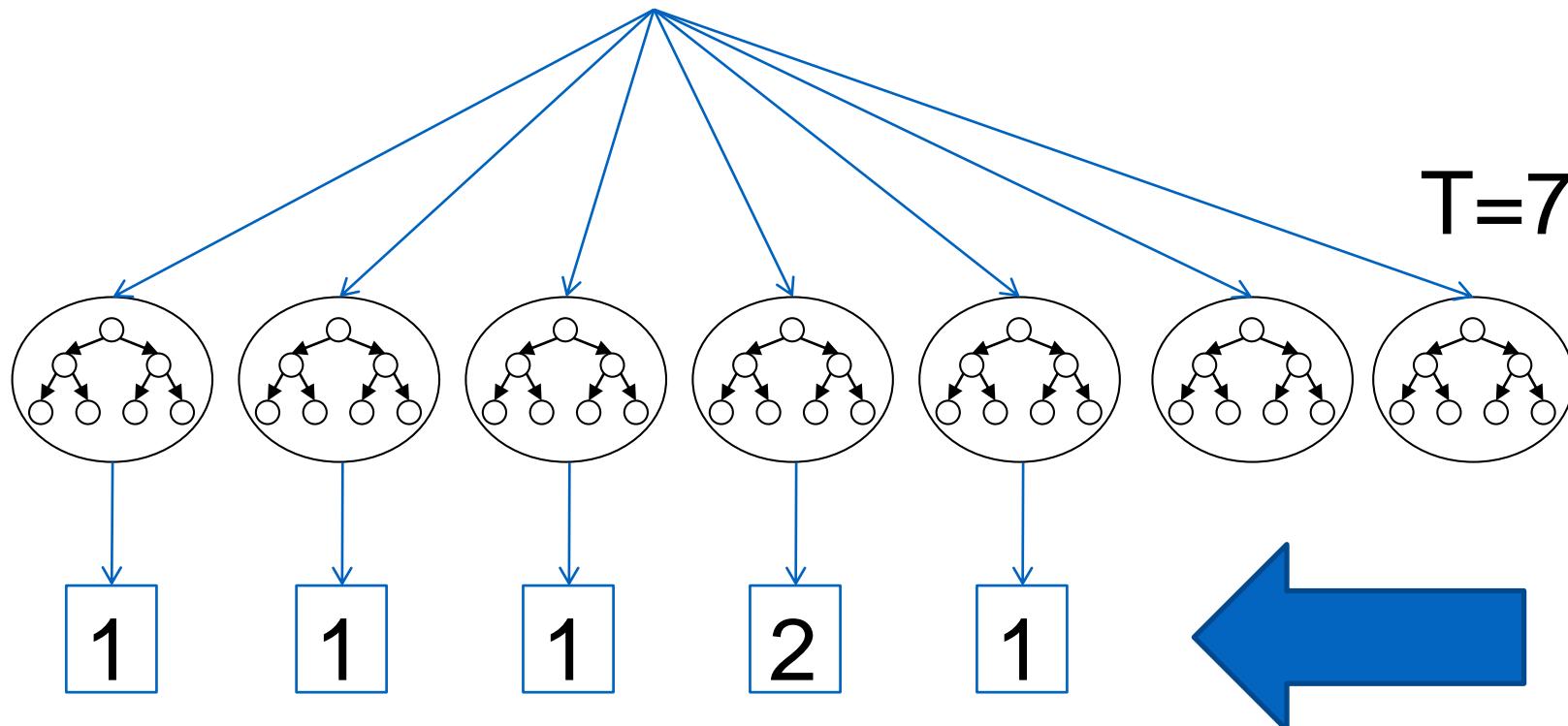
NO

# Poda dinámica de conjuntos

Nuevo ejemplo:

**X**

T=7 clasificadores



Votos acumulados:  $t \rightarrow \begin{matrix} 4 & 1 \\ t_1 & t_2 \end{matrix}$   $\longrightarrow$  Clase final  $\boxed{1}$

¿Necesitamos preguntar a todos los clasificadores?

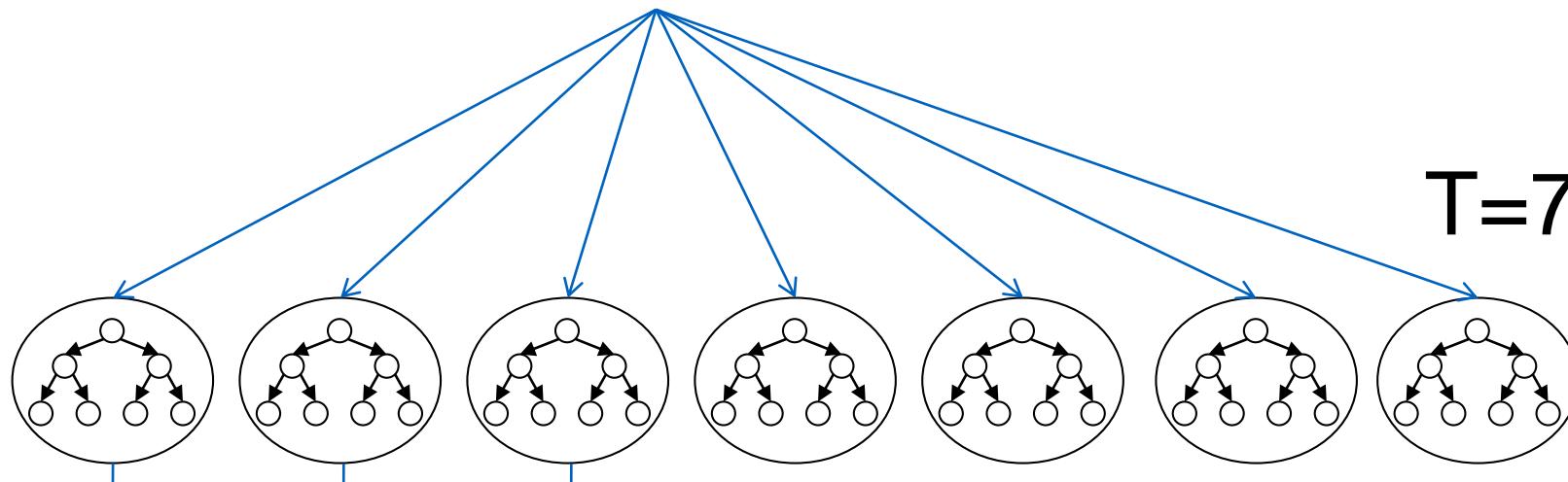
NO

# Poda dinámica de conjuntos

Nuevo ejemplo:

**X**

T=7 clasificadores



Votos acumulados:

$t \rightarrow \begin{matrix} 3 & 0 \\ t_1 & t_2 \end{matrix}$

Clase final **1**

¿Necesitamos preguntar a todos los clasificadores?

NO

# Importancia de atributos

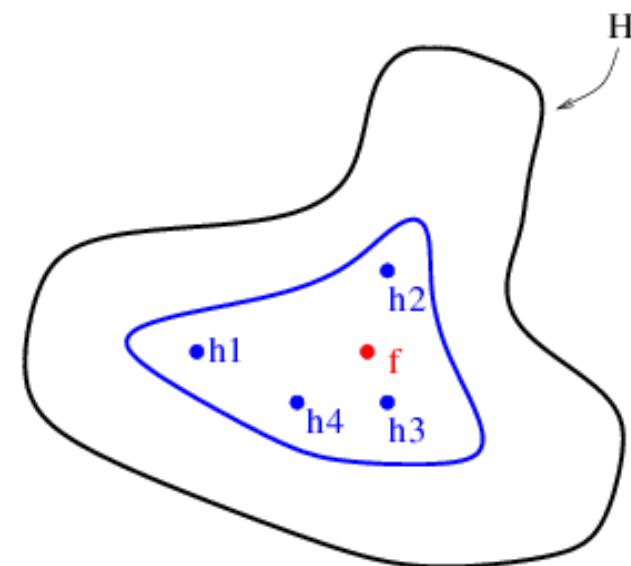
- Los conjuntos obtienen % de error mejores que los árboles de decisión. Sin embargo, se pierde su interpretabilidad. Aunque no del todo
  - El número de veces que aparece cada atributo en los árboles y cómo de arriba estén nos da una idea de lo importante que es un atributo.
  - Breiman lo sistematiza con el siguiente algoritmo. Para cada atributo desordenar los datos y ver cómo se deteriora la clasificación. Cuánto más se deteriora más importante es el atributo.

# ¿Por qué funcionan?

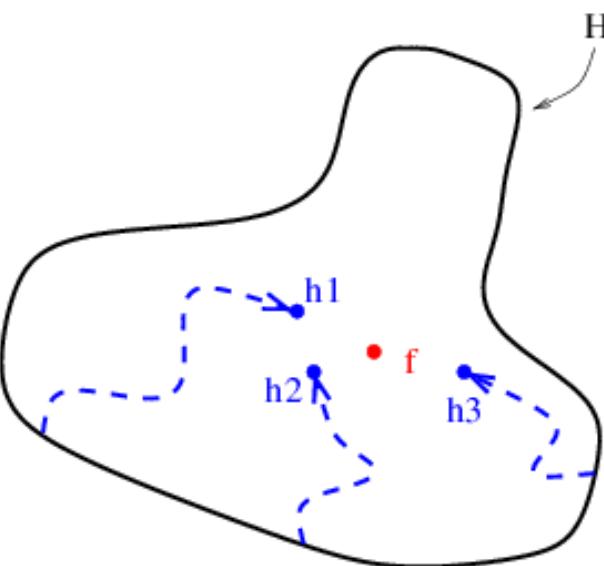
- Algunos motivos:
  - Motivos estadísticos: No hay suficientes datos para que el algoritmo de clasificación obtenga una solución óptima.
  - Motivos computacionales: El algoritmo no puede alcanzar la solución óptima.
  - Motivos expresivos: La solución está fuera del espacio de hipótesis que estamos explorando.

# ¿Por qué funcionan?

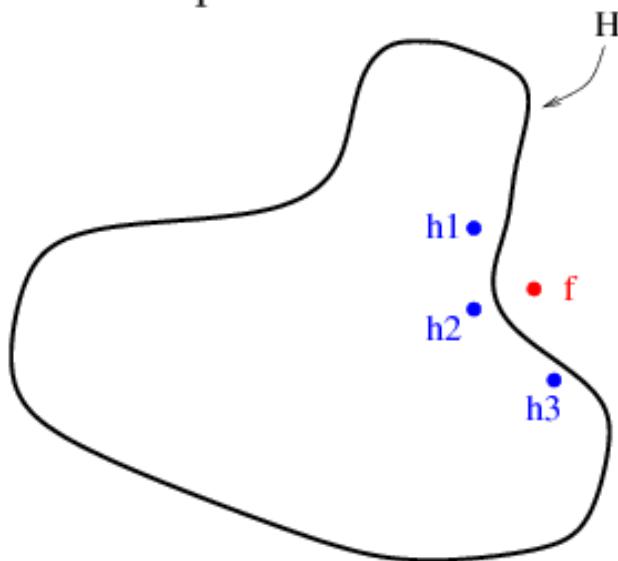
Statistical



Computational



Representational

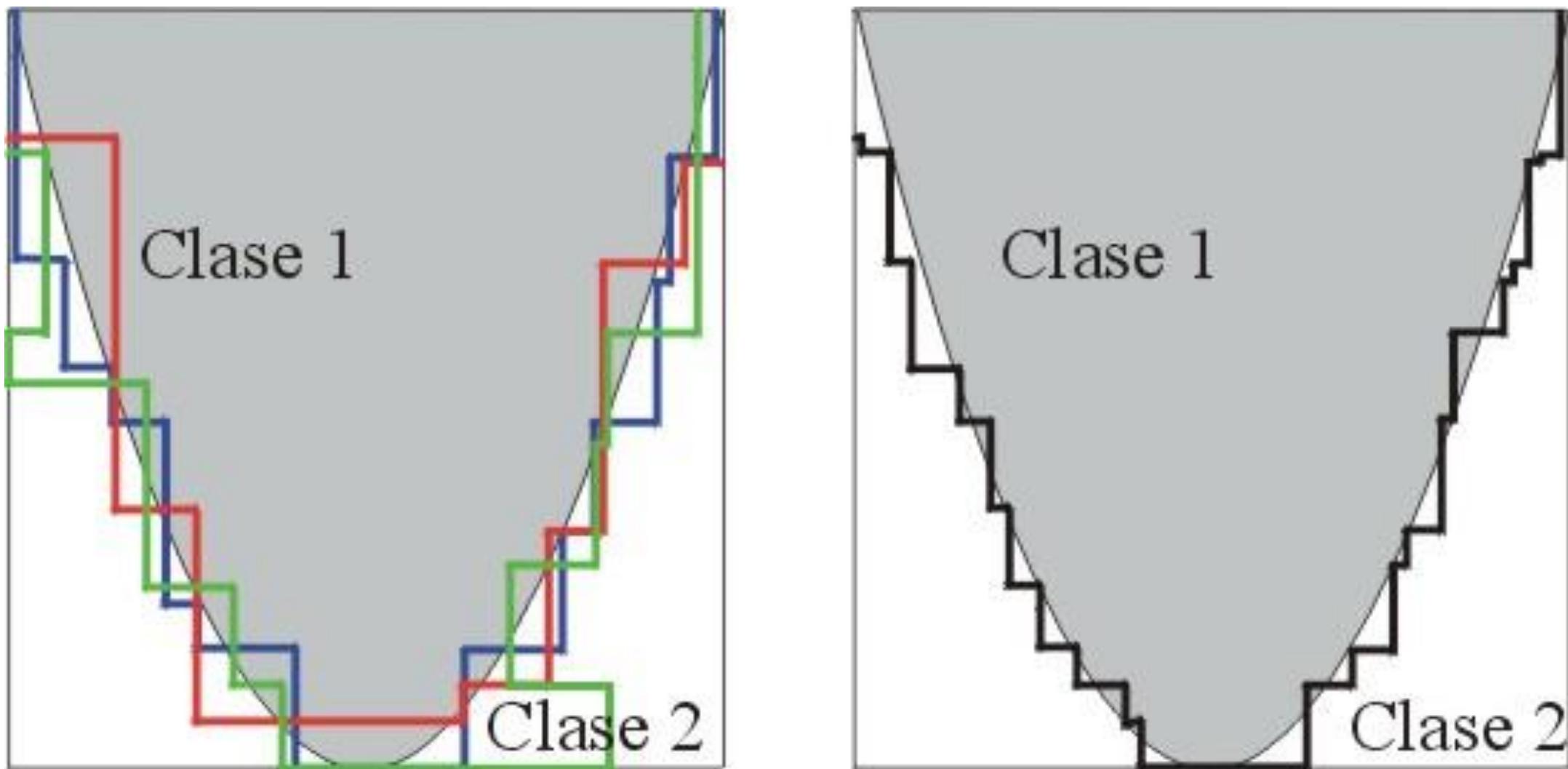


Thomas Dietterich

**Fig. 2.** Three fundamental reasons why an ensemble may work better than a single classifier

# ¿Por qué funcionan?

Un conjunto de soluciones subóptimas se puede combinar para compensar sus limitaciones.



# Historia de éxito 1: Netflix prize challenge

- Dataset: ratings de 17770 películas y 480189 usuarios

## Leaderboard

Showing Test Score. [Click here to show quiz score](#)

Display top **20** leaders.

Combina cientos  
de modelos de tres  
equipos

Variante de stacking

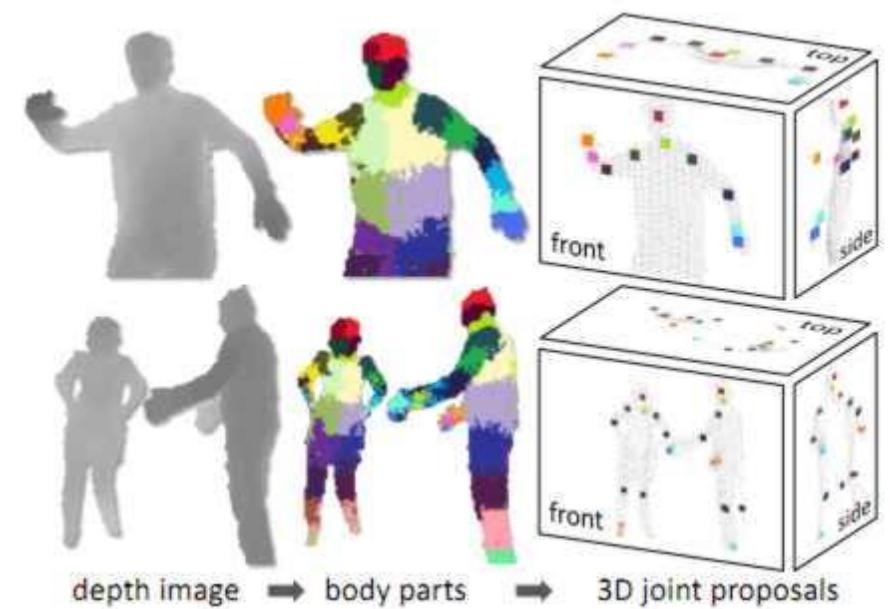
Rank	Team Name	Best Test Score	% Improvement	Best Submit Time
<b>Grand Prize - RMSE = 0.8567 - Winning Team: BellKor's Pragmatic Chaos</b>				
1	<a href="#">BellKor's Pragmatic Chaos</a>	0.8567	10.06	2009-07-26 18:18:28
2	<a href="#">The Ensemble</a>	0.8567	10.06	2009-07-26 18:38:22
3	<a href="#">Grand Prize Team</a>	0.8582	9.90	2009-07-10 21:24:40
4	<a href="#">Opera Solutions and Vandelay United</a>	0.8588	9.84	2009-07-10 01:12:31
5	<a href="#">Vandelay Industries !</a>	0.8591	9.81	2009-07-10 00:32:20
6	<a href="#">Pragmatic Theory</a>	0.8594	9.77	2009-06-24 12:06:56
7	<a href="#">BellKor in BigChaos</a>	0.8601	9.70	2009-05-13 08:14:09
8	<a href="#">Dace</a>	0.8612	9.59	2009-07-24 17:18:43
9	<a href="#">Feeds2</a>	0.8622	9.48	2009-07-12 13:11:51
10	<a href="#">BigChaos</a>	0.8623	9.47	2009-04-07 12:33:59
11	<a href="#">Opera Solutions</a>	0.8623	9.47	2009-07-24 00:34:07
12	<a href="#">BellKor</a>	0.8624	9.46	2009-07-26 17:19:11

# Historia de éxito 2: KDD cup

- KDD cup 2013: Predecir que artículos están escritos por qué autor.
  - El equipo ganador usaba Random Forest y Boosting among other models combined with regularized linear regression.
- KDD cup 2014: Predict funding requests that deserve an A+ in [donorschoose.org](http://donorschoose.org)
  - Multistage ensemble
- KDD cup 2015: Predict dropouts in MOOC
  - Multistage ensemble

# Historia de éxito 3: Kinect

- Visión por ordenador
- Clasificar píxeles en partes del cuerpo (mano, cabeza, etc)
- Usa *Random Forests*



# Desventajas de utilizar conjuntos

- ¡Ninguna! Bueno puede que alguna...
- Más lento que un clasificador único ya que hay que crear cientos o miles de clasificadores.
  - Se puede mitiga mediante la poda
  - Se pierde parte de la interpretabilidad de los árboles

# Ventajas

- Una familia de algoritmos con un rendimiento de entre los mejores actualmente, (especialmente random forest). Comparable o mejor que SVMs
- Son algoritmos sin prácticamente parámetros que ajustar.
- Si los algoritmos base son árboles, se pueden crear muy rápido y clasifican muy rápido.

# Información no estructurada

## Textos

Manuel Sánchez-Montaños

# Contenidos

- Etapas de preprocessamiento fundamentales en un proyecto de analítica de textos
- ¿Qué es un embedding?
- Estrategias de embedding más comunes: Latent Semantic Analysis (LSA), Word2Vec, capas de embedding de redes profundas, transfer learning
- Creación en Python de scripts sencillos para realizar analítica de textos usando librerías como NLTK o Scikit-Learn. Ejemplo con una base de datos real

## Analítica de Textos

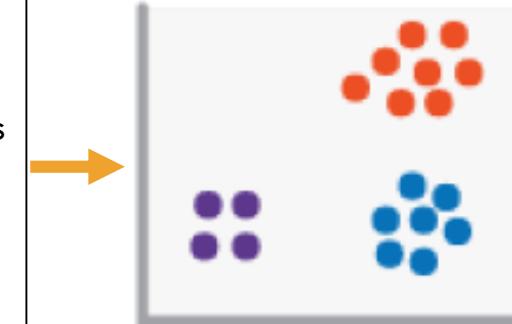
There is a credit card showing up on my credit report which is not mine.  
I have never taken out a Capital One credit card. Please help

I have been paying extra money each month on my mortgage. This month I received a check for {\$420.00} as a reimbursement for escrow. My escrow amount was also increased which makes no sense at all. I believe that charging this much escrow for no reason is a ridiculous use of my money. Please help.

I have had issues in the past with overdrafts to my business checking accounts. My bank's online banking does not allow transfers after XXXX to go through until the next business day ( two days later ), even if I have funds directly in the bank.

...

¿Conocimiento cuantitativo?



# Etapas de preprocesamiento fundamentales en un proyecto de analítica de textos

- 1. Preprocesado de textos**
- 2. Word / Paragraph / Document vector encoding**
- 3. Procesamiento de Lenguaje Natural de Alto Nivel**  
Clustering, construcción de modelos predictivos, etc.

## Parte 1. Preprocesado de los textos

- **Lematizador**

“cables”  $\Rightarrow$  “cable”

- **Eliminación de stop words**

Stop word: cualquier palabra que no esperemos que contenga contenido semántico importante. Podemos cargar una lista de stopwords predefinida en NLTK, cambiarla, o crear nuestra propia lista de cero.

[“hello”, “john”, “bought”, “two”, “cable”, “and”, “lcd”, “monitor”]

$\Rightarrow$  [“hello”, “john”, “bought”, “cable”, “lcd”, “monitor”]

## Parte 2. Vector encoding

### Estrategia 1: BOW (Bag of Words)

```
[ ["bought", "car", "frequent", "failure"]           # documento 0
  ["cost", "bought", "car", "low", "energy", "low", "cost"] # documento 1
  ["failure", "bought", "motorcycle", "low", "quality"]    # documento 2
  ["frequent", "low", "quality", "bought", "car"] ]        # documento 3
```

Nuestra codificación de los documentos es la matriz “term frequency” (tf) donde:

$tf[i,j]$  = número de apariciones del término  $j$  en el documento  $i$

Cada documento está representado por un vector de tantas componentes como palabras en el vocabulario (puede ser enorme!!)

## Parte 2. Vector encoding

### Estrategia 1. BOW (Bag of Words)

```
[ ["bought", "car", "frequent", "failure"]           # documento 0
  ["cost", "bought", "car", "low", "energy", "low", "cost"] # documento 1
  ["failure", "bought", "motorcycle", "low", "quality"]    # documento 2
  ["frequent", "low", "quality", "bought", "car"] ]        # documento 3
```

**vocabulario** = ["bought", "car", "cost", "energy", "failure", "frequent", "low", "motorcycle", "quality"]

$$tf = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 2 & 1 & 0 & 0 & 2 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix} \quad \text{número de documentos}$$

## Parte 2. Vector encoding

### Estrategia 2. TF-IDF (1)

**Idea:** Un término que aparece en todos los documentos no es muy informativo. Un término que aparece en un subconjunto pequeño (cuidado: demasiado pequeño es también malo) es mucho más informativo.

- **df[t]** (“document frequency”): número de documentos donde el término (palabra)  $t$  aparece al menos una vez
- **idf[t]** (“inverse of document frequency”): función de  $df[t]$  que decrece monotónicamente si  $df$  aumenta.  
Por ejemplo:

$$idf[t] = 1 + \log \left[ \frac{D + 1}{df[t] + 1} \right]$$

**D:** número total de documentos en el corpus. El “+1” en el numerador y en el denominador evita problemas numéricos:  $\log(0/\text{número})$  o  $\log(\text{número}/0)$

Otras variantes muy similares a esta ecuación existen en la literatura y también se conocen como “idf”

## Parte 2. Vector encoding

### Estrategia 2. TF-IDF (2)

Del ejemplo anterior:

$$tf = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 2 & 1 & 0 & 0 & 2 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

- **vocabulario** = [“bought”, “car”, “cost”, “energy”, “failure”, “frequent”, “low”, “motorcycle”, “quality”]
- D=4. df = [4, 3, 1, 1, 2, 2, 3, 1, 2]
- $idf[t] = 1 + \log \left[ \frac{D+1}{df[t]+1} \right] = [1, 1.22, 1.92, 1.92, 1.51, 1.51, 1.22, 1.92, 1.51]$
- La matriz TF-IDF se computa multiplicando cada columna t de TF por  $idf[t]$

## Parte 2. Vector encoding

### Estrategia 2. TF-IDF (3)

$$tf = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 2 & 1 & 0 & 0 & 2 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

- **vocabulario** = ["bought", "car", "cost", "energy", "failure", "frequent", "low", "motorcycle", "quality"]
- $idf[t] = 1 + \log \left[ \frac{D+1}{df[t]+1} \right] = [1, 1.22, 1.92, 1.92, 1.51, 1.51, 1.22, 1.92, 1.51]$

$$tfidf = \begin{bmatrix} 1 & 1.22 & 0 & 0 & 1.51 & 1.51 & 0 & 0 & 0 \\ 1 & 1.22 & 3.84 & 1.92 & 0 & 0 & 2.44 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1.51 & 0 & 1.22 & 1.92 & 1.51 \\ 1 & 1.22 & 0 & 0 & 0 & 1.51 & 1.22 & 0 & 1.51 \end{bmatrix}$$

- Cada fila de la matriz tf-idf puede ser normalizada individualmente de acuerdo por ejemplo a la norma L2  $\Rightarrow$  los documentos muy grandes se pueden comparar “en igualdad de condiciones” a los documentos muy pequeños

## Parte 2. Vector encoding

### Estrategia 3: Latent Semantic Analysis (LSA), 1

- Singular Value Decomposition (SVD): cualquier matriz  $M$  de dimensiones  $D \times W$  ( $D$ : número de documentos;  $W$ : número de palabras) puede descomponerse **exactamente** como

$$M = U \cdot \Sigma \cdot V^T$$

donde:

- $\Sigma$  es una matriz diagonal con componentes no negativos y dimensiones  $L \times L$
- $L$  (“dimensiones del espacio latente”) =  $\min(D, W)$
- $U$  es una matriz de  $D \times L$  dimensiones donde cada columna está normalizada y es ortogonal a las otras
- $V$  es una matriz de  $W \times L$  dimensiones donde cada columna está normalizada y es ortogonal a las otras

## Parte 2. Vector encoding

### Estrategia 3: Latent Semantic Analysis (LSA), 2

- Ejemplo:

$$M = \begin{bmatrix} [0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1] \\ [0 & 1 & 0 & 1 & 0 & 2 & 1 & 0 & 1] \\ [1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0] \\ [0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1] \end{bmatrix} = U \cdot \Sigma \cdot V^T$$

con

$$U = \begin{bmatrix} [-0.53609237 & 0.03922979 & 0.4594192 & -0.70710678] \\ [-0.63342998 & -0.29925753 & -0.71359049 & 0. ] \\ [-0.15484316 & 0.952555812 & -0.26202407 & 0. ] \\ [-0.53609237 & 0.03922979 & 0.4594192 & 0.70710678] \end{bmatrix}$$

$$\Sigma = \begin{bmatrix} [3.87493377 & 0. & 0. & 0. ] \\ [0. & 1.94118656 & 0. & 0. ] \\ [0. & 0. & 1.79351137 & 0. ] \\ [0. & 0. & 0. & 0. ] \end{bmatrix}$$

$$V = \begin{bmatrix} [-0.03996021 & 0.49070921 & -0.14609557 & -0.16990314] \\ [-0.44016616 & -0.11374381 & 0.11443914 & -0.31889378] \\ [-0.27669756 & 0.04041836 & 0.51231256 & -0.30320033] \\ [-0.44016616 & -0.11374381 & 0.11443914 & 0.8370474 ] \\ [-0.03996021 & 0.49070921 & -0.14609557 & 0.11095191] \\ [-0.32693719 & -0.30832433 & -0.79574683 & -0.15160017] \\ [-0.48012637 & 0.3769654 & -0.03165643 & -0.05200069] \\ [-0.03996021 & 0.49070921 & -0.14609557 & 0.11095191] \\ [-0.44016616 & -0.11374381 & 0.11443914 & -0.1629526 ] \end{bmatrix}$$

## Parte 2. Vector encoding

### Estrategia 3: Latent Semantic Analysis (LSA), 3

- Las componentes de  $\Sigma$  se interpretan como las “importancias” de cada dimensión latente y se ordenan en orden decreciente:

$$\Sigma = \begin{bmatrix} [3.87493377 & 0. & 0. & 0.] \\ [0. & 1.94118656 & 0. & 0.] \\ [0. & 0. & 1.79351137 & 0.] \\ [0. & 0. & 0. & 0.] \end{bmatrix}$$

- En este caso la cuarta dimensión latente tiene “importancia” nula (0)
- Eliminarla es equivalente a usar las siguientes matrices:

## Parte 2. Vector encoding

### Estrategia 3: Latent Semantic Analysis (LSA), 4

- Si “eliminamos” la cuarta dimensión latente:

$$U = \begin{bmatrix} [-0.53609237 & 0.03922979 & 0.4594192 & -0.76710778] \\ [-0.63342998 & -0.29925753 & -0.71359049 & 0.] \\ [-0.15484316 & 0.95255812 & -0.26202407 & 0.] \\ [-0.53609237 & 0.03922979 & 0.4594192 & 0.76710778] \end{bmatrix}$$

~~[[3.87493377 0. 0. 0.]  
[0. 1.94118656 0. 0.]  
[0. 0. 1.79351137 0.]  
[0. 0. 0. 0.]]~~

$$\Sigma = \begin{bmatrix} [-0.03996021 & 0.49070921 & -0.14609557 & -0.16990314] \\ [-0.44016616 & -0.11374381 & 0.11443914 & -0.18890778] \\ [-0.27669756 & 0.04041836 & 0.51231256 & -0.3132033] \\ [-0.44016616 & -0.11374381 & 0.11443914 & 0.830474] \\ [-0.03996021 & 0.49070921 & -0.14609557 & 0.11045191] \\ [-0.32693719 & -0.30832433 & -0.79574683 & -0.1500017] \\ [-0.48012637 & 0.3769654 & -0.03165643 & -0.0202069] \\ [-0.03996021 & 0.49070921 & -0.14609557 & 0.1095091] \\ [-0.44016616 & -0.11374381 & 0.11443914 & -0.1629515] \end{bmatrix}$$

$$V = \begin{bmatrix} [0 1 1 1 0 0 1 0 1] \\ [0 1 0 1 0 2 1 0 1] \\ [1 0 0 0 1 0 1 1 0] \\ [0 1 1 1 0 0 1 0 1] \end{bmatrix}$$

(esto es, reducimos el número de columnas en las matrices)

Si ahora calculamos  $U \cdot \Sigma \cdot V^T$  obtenemos de nuevo la matriz M original:

$$\begin{bmatrix} [0 1 1 1 0 0 1 0 1] \\ [0 1 0 1 0 2 1 0 1] \\ [1 0 0 0 1 0 1 1 0] \\ [0 1 1 1 0 0 1 0 1] \end{bmatrix}$$

## Parte 2. Vector encoding

### Estrategia 3: Latent Semantic Analysis (LSA), 5

- Uso práctico de LSA:

Computar la descomposición SVD de la matriz tf-idf

Eliminar las dimensiones latentes con menos información

Eliminar las correspondientes columnas en  $U$ ,  $V$ ,  $\Sigma$

Ahora la fila  $d$  en  $U \cdot \Sigma$  se interpreta como la vectorización del documento  $d$

La fila  $w$  en  $V \cdot \Sigma$  se interpreta como la vectorización de la palabra  $d$

- Por tanto **hemos asignado un vector de dimensión  $L'$  a cada documento y palabra**
- A partir de ahí se construirán modelos de alto nivel con esa vectorización

## Parte 2. Vector encoding

BOW / TF-IDF / LSA:

**Es conveniente normalizar cada documento (por lo que la "longitud" de todos los vectores del documento es 1)**

Esto permite comparar documentos pequeños con documentos grandes

- Hacer esto es equivalente a no normalizar pero usando la "distancia del coseno"

$$\text{Cosine distance}(\mathbf{v1}, \mathbf{v2}) = 1 - \frac{\mathbf{v1} \cdot \mathbf{v2}}{\|\mathbf{v1}\| \|\mathbf{v2}\|}$$

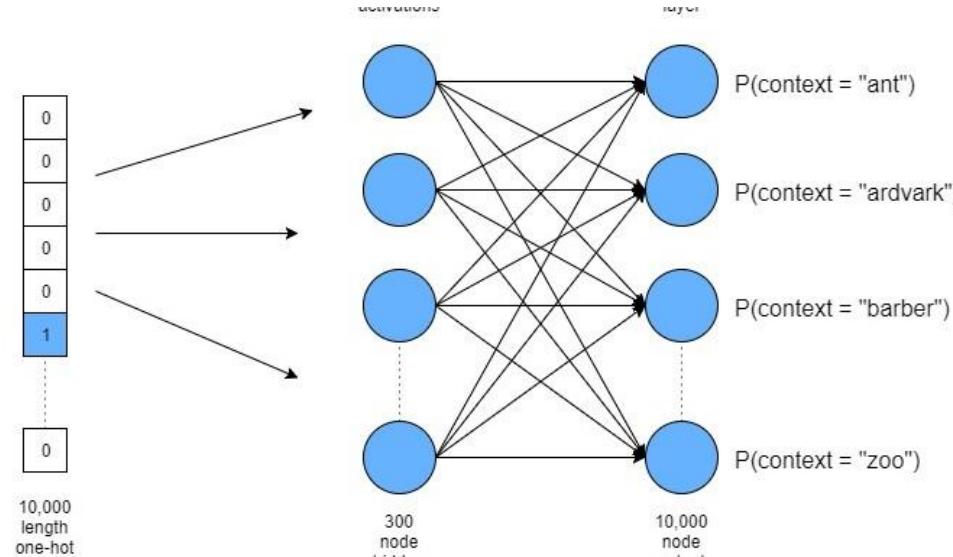
- Distancia cuadrática euclíadiana entre vectores de documentos normalizados:

$$d^2(\mathbf{v1}_{norm}, \mathbf{v2}_{norm}) = \left\| \frac{\mathbf{v1}}{\|\mathbf{v1}\|} - \frac{\mathbf{v2}}{\|\mathbf{v2}\|} \right\|^2 = 2 \left[ 1 - \frac{\mathbf{v1} \cdot \mathbf{v2}}{\|\mathbf{v1}\| \|\mathbf{v2}\|} \right]$$

## Parte 2. Vector encoding

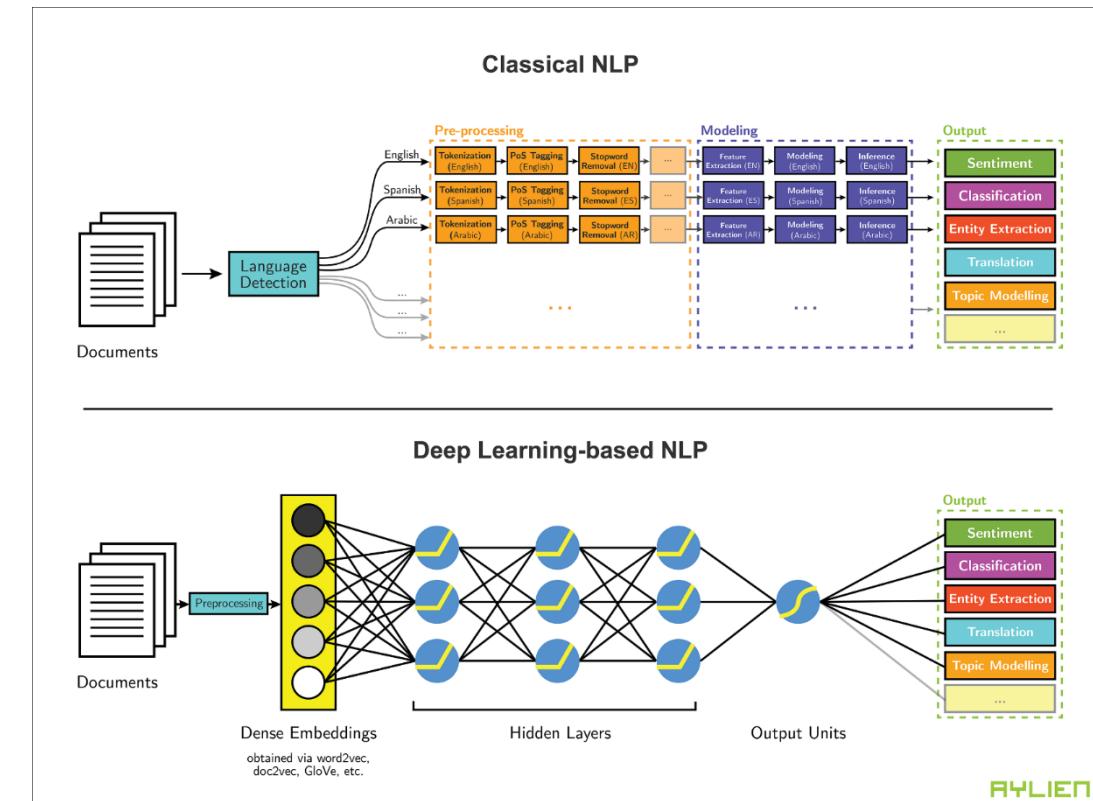
### Estrategia 4. Word2Vec

<https://radimrehurek.com/gensim/models/word2vec.html>



## Parte 2. Vector encoding

### Estrategia 5. Deep Learning Embeddings



## Parte 2. Vector encoding

### Estrategia 6. Transfer Learning

**Embeddings pre-entrenados con bases de datos masivas**

**GloVe:**

<https://nlp.stanford.edu/projects/glove/>

**LexVec:**

<https://github.com/alexandres/lexvec>

**FastText:**

<https://github.com/icoxfog417/fastTextJapaneseTutorial>

## Parte 3. Creación de modelos de Machine Learning



### Clasificación:

- Chequear primero baseline con modelo Dummy (clasificación por mayoría)
- Probar primero modelos sencillos como Naïve Bayes o k-NN ya que pueden dar buenos resultados en clasificación de textos
- Regresión logística regularizada
- Support Vector Machines
- Ensembles (random forest, XG-Boost)



### Clustering:

- k-means con vectores de documentos normalizados (típicamente con L2) suele dar buenos resultados

# Nuestra estrategia para procesar el lenguaje natural

- 1. Preprocesado de texto**
- 2. Conversión a representación Bag-of-Words**
- 3. TF-IDF**
- 4. Latent Semantic Analysis (LSA)**
- 5. Creación de modelos descriptivos/predictivos con técnicas de Machine Learning estándar**

Clustering, construcción de modelos predictivos, etc.

# Software

- Python
- Librería NLTK:
  - <http://www.nltk.org/>
  - <http://www.nltk.org/book/>
- Scikit-Learn, procesado de los textos:
  - [http://scikit-learn.org/stable/modules/classes.html#module-sklearn.feature\\_extraction.text](http://scikit-learn.org/stable/modules/classes.html#module-sklearn.feature_extraction.text)
  - [http://scikit-learn.org/stable/tutorial/text\\_analytics/working\\_with\\_text\\_data.html](http://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html)

# Clustering

## **1** *Introduction*

- *Definition*
- *Components*
- *Goals in the design of clustering algorithms*

## **2** *Distance Measures*

## **3** *Taxonomy of clustering algorithms*

*K-means*

## **4** *Comparison*

## **5** *Bibliography*

TRATADOS DE LÓGICA  
(ORGANON)

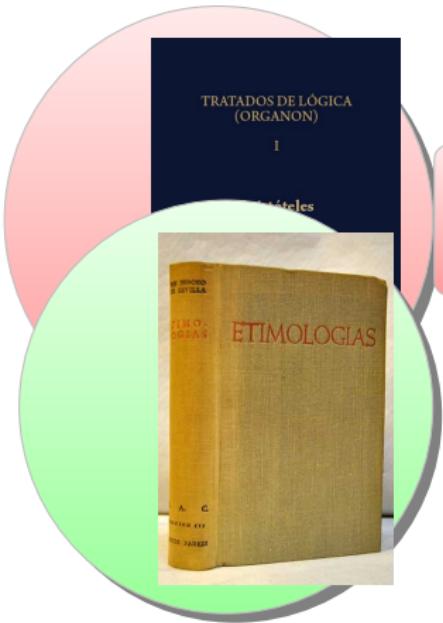
I

Aristóteles

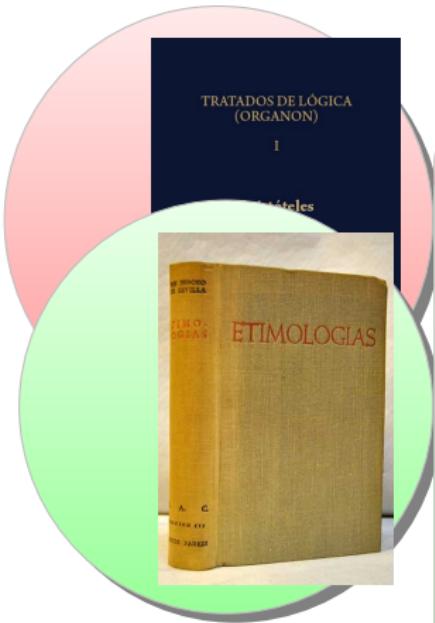
BIBLIOTECA CLÁSICA GREDOS



Categories: all the possible things that can be the subject or the predicate of a proposition



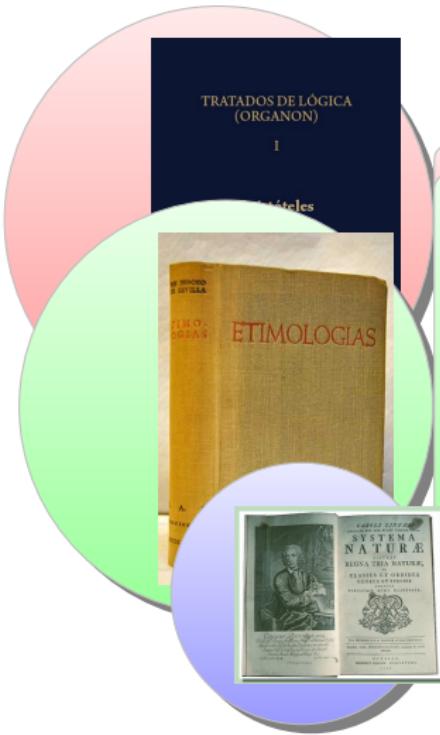
Categories: all the possible things that can be the subject or the predicate of a proposition



Categories: all the possible

summa of universal knowledge:

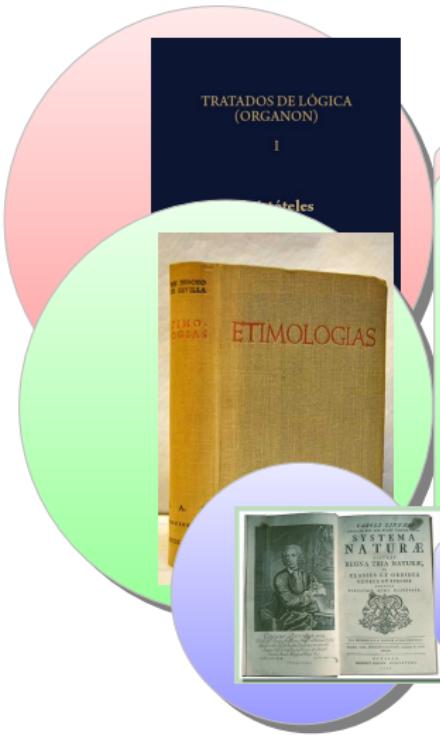
“The knowledge of a word’s etymology often has an indispensable usefulness for interpreting the word, for when you have seen whence a word has originated, you understand its force more quickly. Indeed, one’s insight into anything is clearer when its etymology is known.”



### Categories: all the possible

summa of universal knowledge:

"The knowledge of a word's etymology often has an indispensable usefulness for interpreting the word, for when you have seen whence a word has originated, you understand its force more quickly. Indeed, one's insight into anything is clearer when its etymology is known."

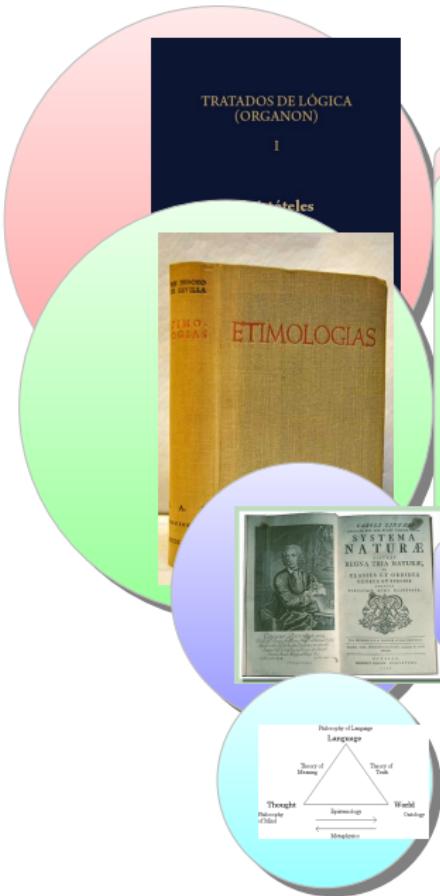


### Categories: all the possible

summa of universal knowledge:

"The knowledge of a word's etymology often has an indispensable usefulness for interpreting the word, for when you have seen whence a word has originated, you understand its force more quickly. Indeed, one's

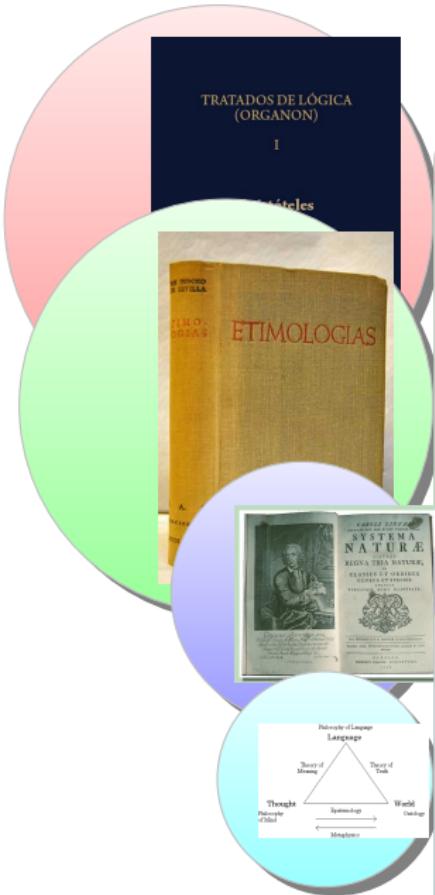
*Systema naturae* ⇔ binomial nomenclature: genus + species



**Categories: all the possible summa of universal knowledge:**

“The knowledge of a word’s etymology often has an indispensable usefulness for interpreting the word, for when you have seen whence a word has originated, you understand its force more quickly. Indeed, one’s *Systema naturae*  $\Leftrightarrow$  binomial nomenclature: genus + species





### Categories: all the possible

summa of universal knowledge:

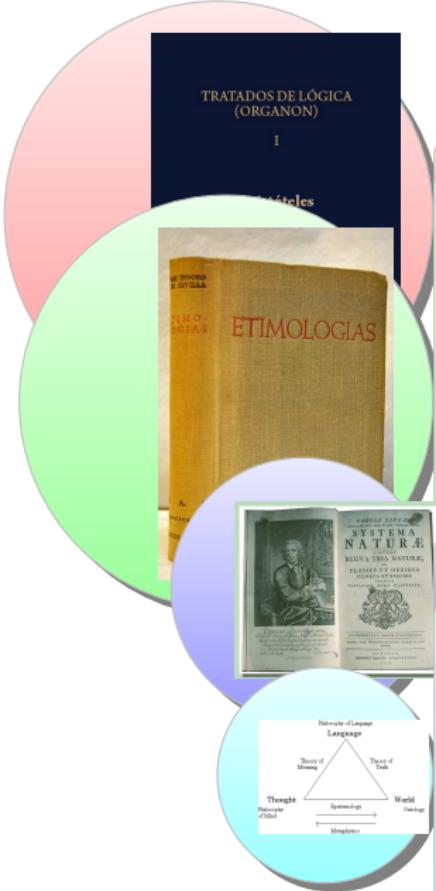
"The knowledge of a word's etymology often has an indispensable usefulness for interpreting the word, for when you have seen whence a word has originated, you understand its force more quickly. Indeed, one's

*Systema naturae*  $\Leftrightarrow$  binomial



"Are thinking and referring identical with computational states of the brain?"

Hilary Putnam



### Categories: all the possible

summa of universal knowledge:

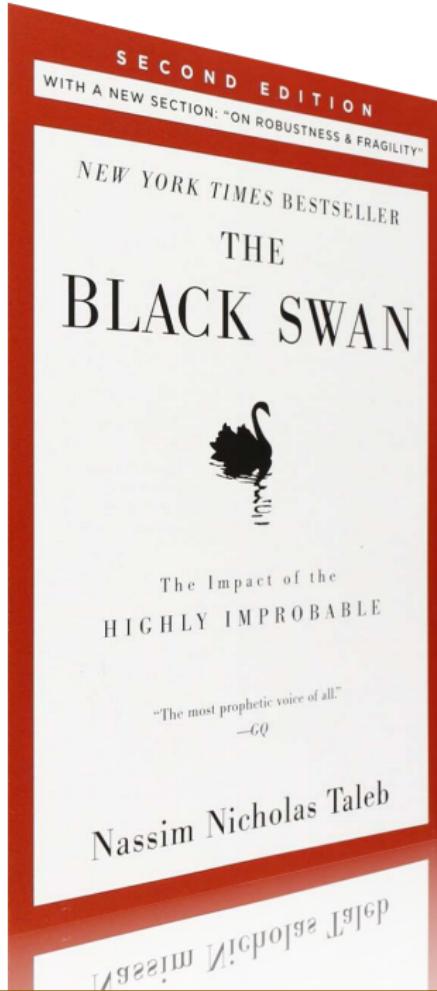
"The knowledge of a word's etymology often has an indispensable usefulness for interpreting the word, for when you have seen whence a word has originated, you understand its force more quickly. Indeed, one's

*Systema naturae*  $\Leftrightarrow$  binomial



"Are thinking and referring identical with computational states of the brain?"

Hilary Putnam



Judea Pearl

Behind any causal conclusion there must be some causal assumption,  
untested in observational studies<sup>a</sup>

---

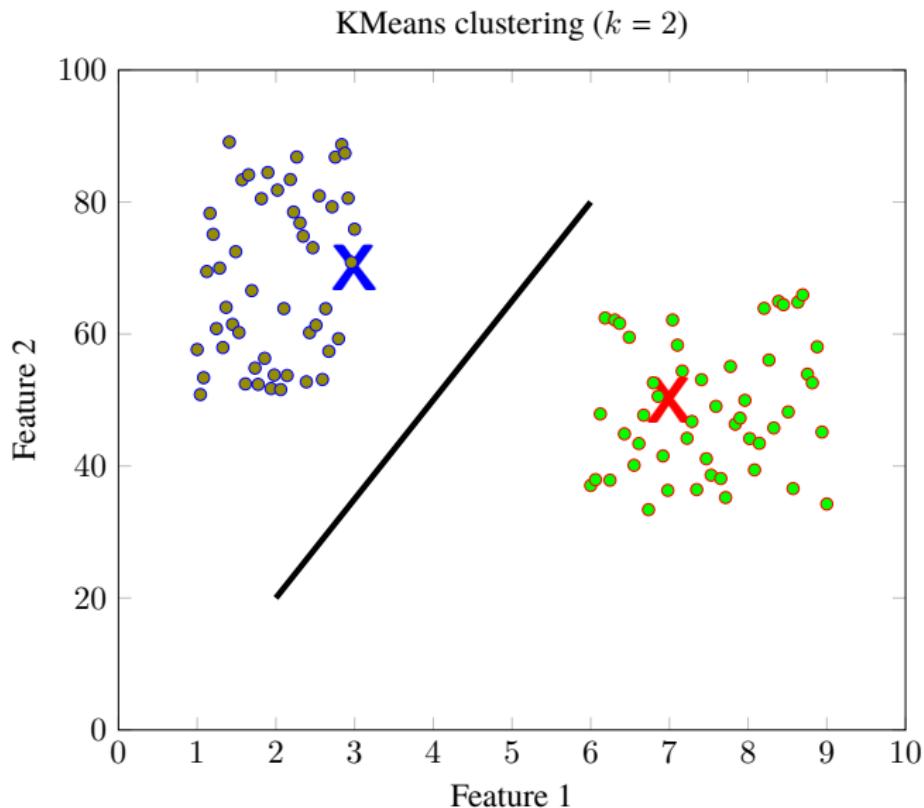
<sup>a</sup>Pearl 2009.

---

# Clustering from a computational point of view

1. Proximity between items
2. What do we mean by proximity?
3. Are they really related?
4. Can we perform such a methodology in a automatic way?

# Clustering vs. Classification



## Classification

- ✓ There exist **LABELS** for some points
- ✓ Rule such that new points are assigned labels properly
- ✓ Supervised learning

## Clustering

- ✓ No labels
- ✓ Points assign to clusters according to **HOW CLOSE** they are to one another
- ✓ Identify **STRUCTURE** in data
- ✓ Unsupervised learning

# Clustering definition

- ✓ Cluster analysis  $\equiv$  organization of a “collection of patterns into clusters based on similarity” (Jain, Murty, et al. 1999)
- ✓ The definition and the scope of “cluster” in the data set are not easy to define
- ✓ According to (Jain and Dubes 1988), a cluster
  1. set of similar objects
  2. set of points aggregated in the testing environment | the distance between two points in a cluster is less than the distance between any point in the cluster and any point of other clusters
  3. clusters can be densely connected regions in a multi-dimensional space separated by loosely connected points

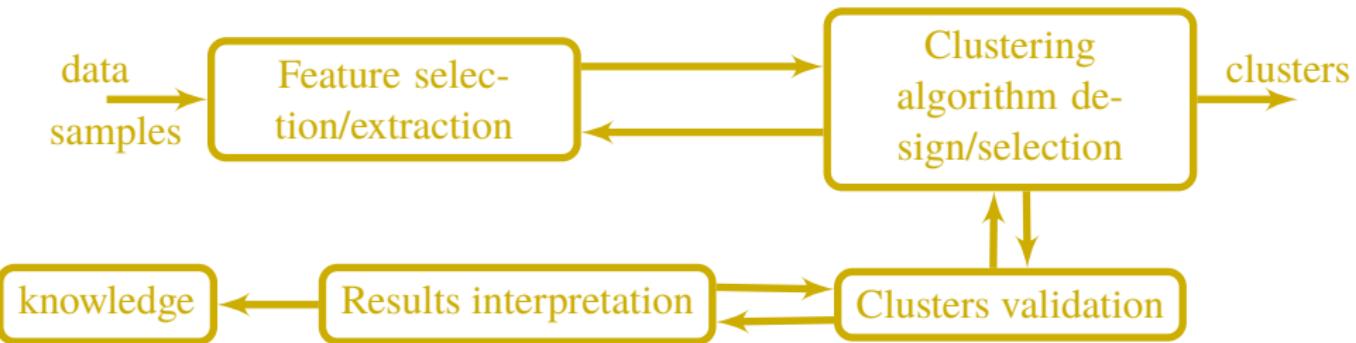
# Clustering definition

- ✓ Cluster analysis  $\equiv$  organization of a “collection of patterns into clusters based on similarity” (Jain, Murty, et al. 1999)
- ✓ According to (Jain and Dubes 1988), a cluster
  1. set of similar objects  $\Leftarrow$  (minimum intra-cluster)
  2. set of points aggregated in the testing environment | the distance between two points in a cluster is less than the distance between any point in the cluster and any point of other clusters  $\Leftarrow$  (minimum inter-/intra-cluster)
  3. clusters can be densely connected regions in a multi-dimensional space separated by loosely connected points  $\Leftarrow$  (graph-/dense-based clustering)

# Clustering ≠ unsupervised predictive learning

- ✓ Unsupervised predictive learning
  - ✗ Vector quantization (Gersho and Gray 2012), probability density estimation, expectation maximization (Bishop 2006)
  - ✗ Accurate characterization of unobserved samples generated from the same probability distribution
- ✓ Clustering analysis
  - ✗ Unsupervised “non-predictive” learning
  - ✗ Split data sets into subsets (based on specific distance/similarity measures)
  - ✗ Not based on the “trained characterization”

# Components of the clustering methodology



# Goals in the design of clustering algorithms

- <sup>1</sup> Scalability Temporal and spatial complexity should be bounded even in large datasets
- Robustness Data outliers should be accurately detected
- Order independence Different input data should not lead to different final results
- Minimum user-defined parameters Configurability should be at least as possible: reduce configuration burden
- Mixed data type Data representation should comprise numeric and/or binary and/or categorical codes
- Variability/flexibility in clusters shape Clusters shape should be fixed
- Point proportion admissibility Duplicating dataset + re-clustering  $\Rightarrow$  changes in the final results

---

<sup>1</sup> Andreopoulos et al. 2009.

# Distance measures

## Definition $(dx_i, x_j)$

*The distance between two instances  $x_i$  and  $x_j$ , which is a metric distance measure if it satisfies the following properties:*

- ✓ Triangle inequality

$$dx_i, x_j \leq dx_i, x_k + dx_k, x_j, \forall x_i, x_j, x_k \in \mathcal{S}$$

- ✓  $dx_i, x_j = 0 \rightarrow x_i = x_j \quad \forall x_i, x_j \in \mathcal{S}$

## Correlation Based

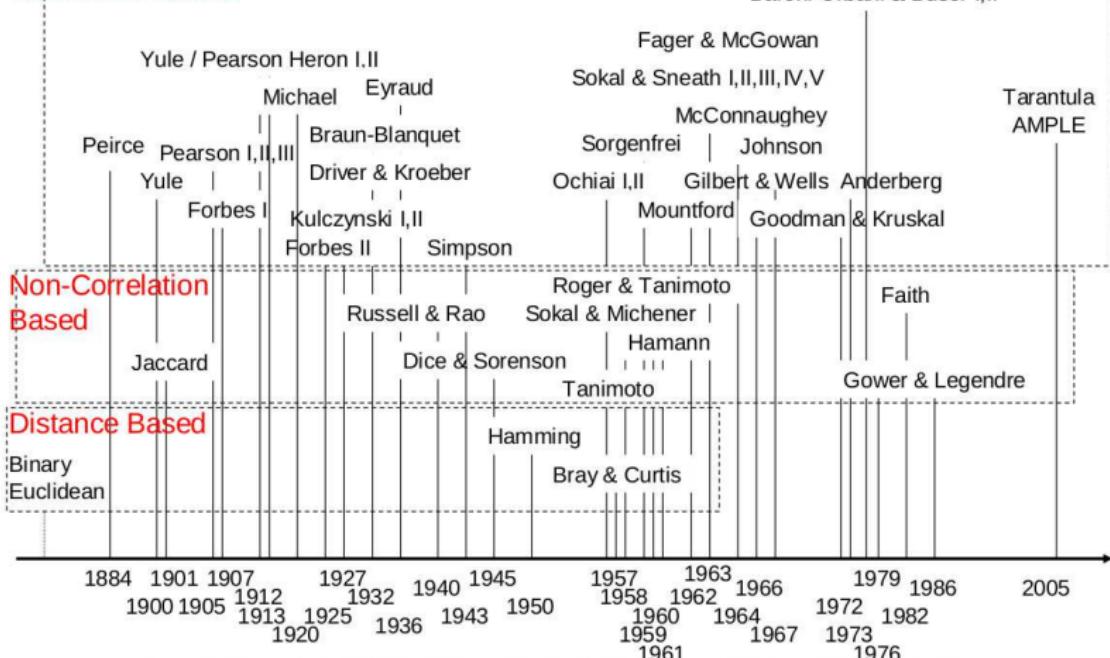


Figure 1 Chronological Table of Binary Similarity Measures and Distance Measures by Year

<sup>2</sup>Image extracted from Choi et al. 2010

# Mathematical definition of items proximity

## Similarity measures

$$\zeta x_i, x_i = 0$$

$$\zeta x_i, x_j \neq 0, \forall i \neq j$$

# Minkowski distance

⇒ Set of distance measures for numeric attributes

## Definition (Minkowski metric)

$$d_{x_i, x_j} = |x_{i,1} - x_{j,1}|^g + |x_{i,2} - x_{j,2}|^g + \cdots + |x_{i,p} - x_{j,p}|^{g1g}$$
$$x_{i,k} \in a, b \subset \mathcal{R}$$

- ✓  $g = 2 \Rightarrow$  Euclidean distance
- ✓  $g = 1 \Rightarrow$  Manhattan distance
- ✓  $g = \infty \Rightarrow$  The greatest of the paraxial distances, i.e., the Chebychev metric

## **1** *Introduction*

## **2** *Distance Measures*

- *Minkowski distance*
- *Distances for binary attributes*

## **3** *Taxonomy of clustering algorithms*

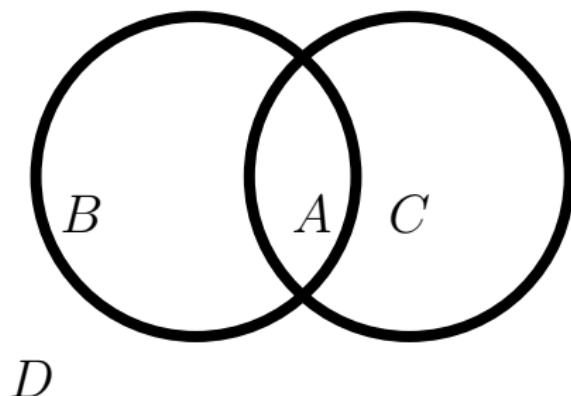
## **4** *Comparison*

## **5** *Bibliography*

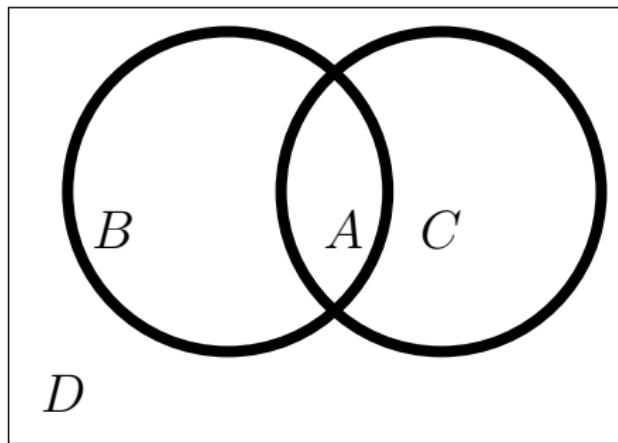
# Distances for binary attributes

$B$  stands for the attributes only contained in the original set

$C$  stands for the attributes only contained in the test set



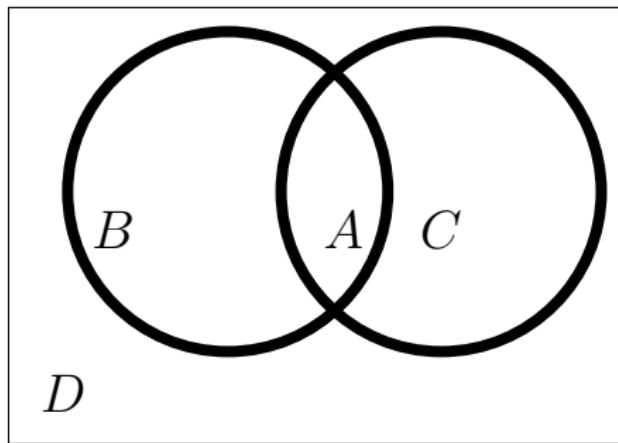
# Distances for binary attributes



Definition (Jaccard coefficient)

$$\frac{A}{A + B + C}$$

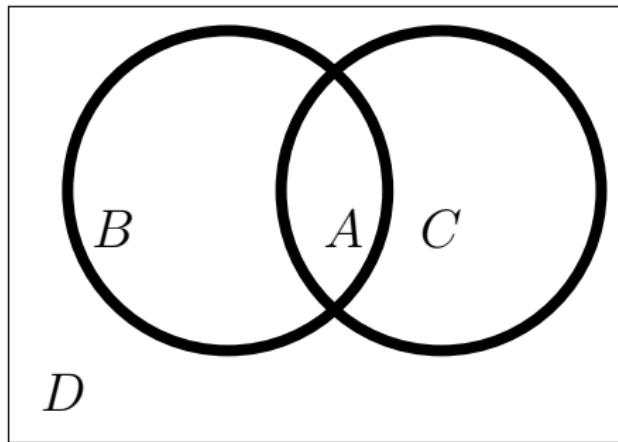
# Distances for binary attributes



Definition (Russell and Rao coefficient)

$$\frac{A}{A + B + C + D}$$

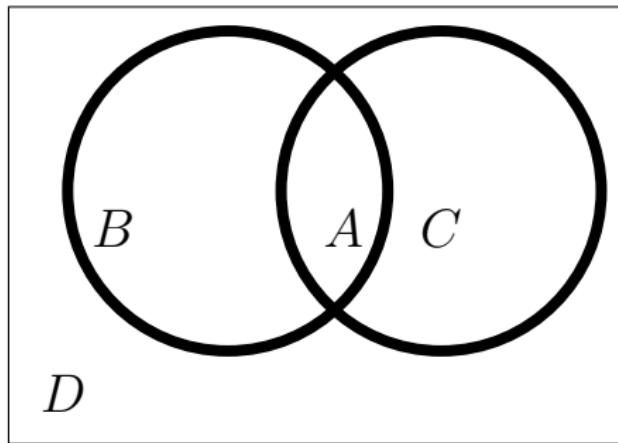
# Distances for binary attributes



Definition (Dice coefficient)

$$\frac{2A}{A + B + C}$$

# Distances for binary attributes



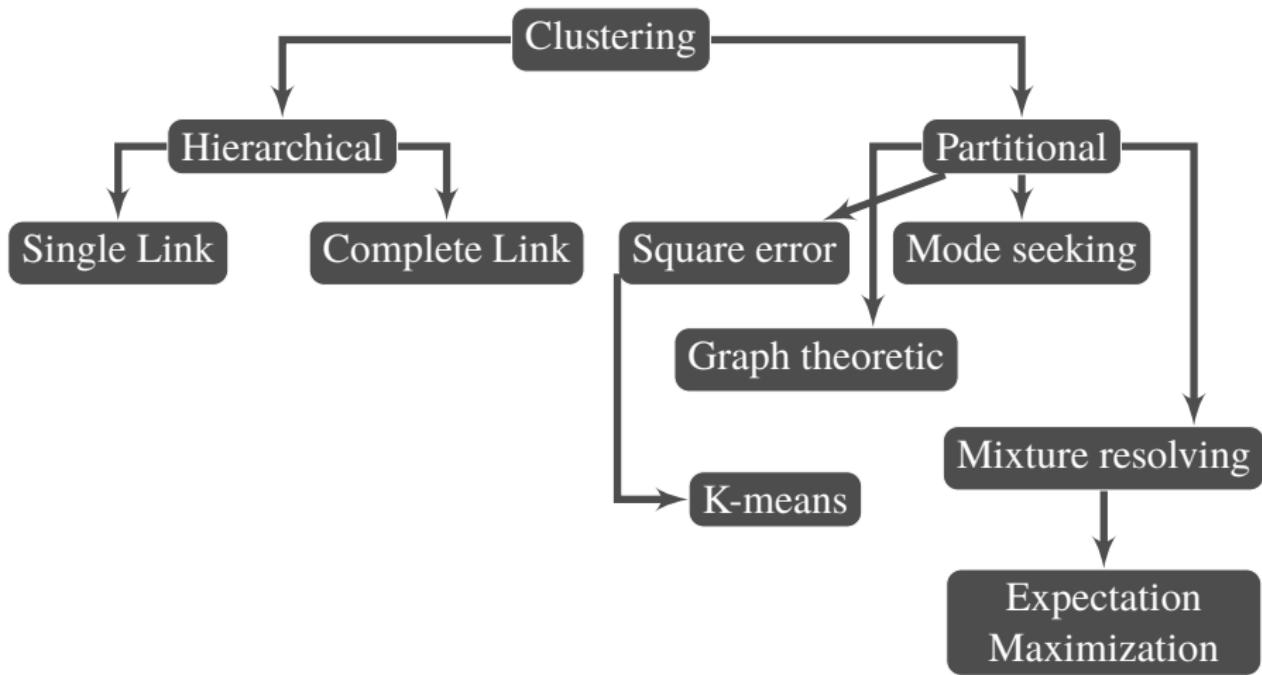
Definition (Roger and Tanimoto coefficient)

$$\frac{2A}{A + B + C}$$

# Music dataset for the example

- ✓  prueba.tsv
- ✓ ejemplo\_metricas\_distancias\_final.ipynb

# Clustering of clustering: a taxonomy



# The bottom-line of the previous taxonomy

**Agglomerative vs. Divisive** Algorithmic structure and operation ⇒

agglomerative approach: bottom-up construction;  
divisive approach: top-down (⇒ Hierarchical clustering)

**Monothetic vs. Polythetic** Use sequential or simultaneous of features in the process (most algorithms are polythetic)

**Hard vs. Fuzzy** In fuzzy schemes a point can pertain to several clusters

**Deterministic vs. Stochastic** Deterministic objective function or random search technique

**Incremental vs. Non-incremental** If the size of the data is incremental or not

## **1** *Introduction*

## **2** *Distance Measures*

## **3** *Taxonomy of clustering algorithms*

- *Partitioning clustering algorithms*
- *Hierarchical methods*

## **4** *Comparison*

## **5** *Bibliography*

# Partitioning clustering algorithms

Fixed number of clusters

Numerical methods

K-means

Farthest First Traversal (FFT)  
k-center

K-medoids

Partition Around Medoids (PAM)

CLARA (Clustering  
Large Applications)

CLARANS (Clustering  
Large Applications Based  
Upon Randomized Search)

Fuzzy K-means

Discrete methods

K-modes

Fuzzy K-modes

non-negative matrix factorization (NMF) method (clustering of microarray data)

$N$  patterns

$x_i, i \in \{0, 1, \dots, M - 1\}$

$\{x_i\} \subset \mathcal{S}$

$\mathcal{S} = \mathcal{S}_1 \ \mathcal{S}_2 \ \mathcal{S}_3 \cdots \mathcal{S}_k$

$\mathcal{S}_i \ \mathcal{S}_j = \emptyset, \forall i \neq j$

$x_i \in \mathcal{S}_j$

# K-means

**Input:**  $\mathcal{S}$  set of all possible objects (with  $N$  attributes, and  $\#\mathcal{S} = M$ )

**Input:**  $K$  number of clusters (user-defined parameter)

**Output:**  $K$  clusters

- 1 **while** *Termination condition is not satisfied* **do**
- 2     Assign instances to the closest cluster center
- 3     Update cluster centers according to the previous operation
- 4 **end**

# K-means

**Input:**  $\mathcal{S}$  set of all possible objects (with  $N$  attributes, and  $\#\mathcal{S} = M$ )

**Input:**  $K$  number of clusters (user-defined parameter)

**Output:**  $K$  clusters

```
1 while Termination condition is not satisfied do
2     | Assign instances to the closest cluster center
3     | Update cluster centers according to the previous operation
4 end
```

- ✓ Proof of the finite convergence (Selim and Ismail 1984)
- ✓ Complexity per iteration  $\sim \mathcal{O}(K \cdot M \cdot N)$

# K-means

- 😊 Low complexity, ease of interpretation and implementation, adaptability to sparse data
- 😢 ↑↑ sensitivity to the initial partition
- ✓ Works well only on data sets having isotropic clusters: it is not as flexible as single link algorithms
- ✓ ↑↑ sensitivity to noisy data and outliers
- ✓ It can be only computed if the mean is properly defined (i.e., for numeric attributes)
- ✓ It requires in advance the number of clusters: no trivial when no prior knowledge is available

# K-means examples

1. ejemplo\_kmeans\_fundamentos.zip
2. ejemplo\_kmeans\_fundamentos.ipynb

# K-prototypes

- ✓ Based on K-means but it can applied to categorical attributes
- ✓ Similarity measure is based on the number of mismatches instead of the Euclidean distance

# PAM algorithm

- ✓ Very close to K-means
- ✓ Each cluster is represented by the most centric object in the cluster (instead of the mean, which may not belong to the cluster)
- ✓ The centers of the clusters are always datapoints

✓ ejemplo\_medoids.ipynb

# Hierarchical methods

- ✓ The result of hierarchical methods is a dendrogram
- ✓ A clustering of the data objects is obtained by cutting the dendrogram at the desired similarity level
- ✓ The merging (agglomerative) or division (divisive) of clusters is ruled by some similarity measure, e.g., a sum of squares

# Regarding the similarity measure. . .

**Single-link clustering** The distance between two clusters is assumed to be equal to the shortest distance from any member of one cluster to any member of the other cluster

**Complete-link clustering** The distance between two clusters is assumed to be equal to the longest distance from any member of one cluster to any member of the other cluster

**Average-link clustering** The distance between two clusters is equal to the average distance from any member of one cluster to any member of the other cluster

# Regarding the similarity measure...

**Single-link clustering** The distance between two clusters is assumed to be equal to the shortest distance from any member of one cluster to any member of the other cluster

- (?) “chaining effect”: a few points that form a bridge between two clusters cause the single-link clustering to unify these two clusters into one

**Complete-link clustering** The distance between two clusters is assumed to be equal to the longest distance from any member of one cluster to any member of the other cluster

**Average-link clustering** The distance between two clusters is equal to the average distance from any member of one cluster to any member of the other cluster

# Regarding the similarity measure...

**Single-link clustering** The distance between two clusters is assumed to be equal to the shortest distance from any member of one cluster to any member of the other cluster

**Complete-link clustering** The distance between two clusters is assumed to be equal to the longest distance from any member of one cluster to any member of the other cluster

**Average-link clustering** The distance between two clusters is equal to the average distance from any member of one cluster to any member of the other cluster

- (?) May cause elongated clusters to split and for portions of neighboring elongated clusters to merge

# Examples hierarchical methods

- ✓ example\_distance\_ngrams.ipynb
- ✓ [http://scikit-learn.org/0.15/auto\\_examples/cluster/plot\\_lena\\_ward\\_segmentation.html](http://scikit-learn.org/0.15/auto_examples/cluster/plot_lena_ward_segmentation.html)

# Main advantages of hierarchical methods

**Versatility** For example, the single-link methods: good performance on datasets containing non-isotropic clusters, including well-separated, chain-link and concentric clusters

**Multiple partitions** These methods produce not one partition, but multiple nested partitions, which allow different users to choose different partitions

# Main disadvantages of hierarchical methods

**Inability to scale well** The time complexity of hierarchical algorithms is at least  $\mathcal{O}M^2$  (where  $M$  stands for the number of instances).

**No back-tracking capability** It is possible to go back and undo previous steps

# Comparison between different clustering methods

Algorithm	Loss function	# clusters	Cluster shape	Parameter Estimation Algorithm
K-means	Within-class squared distance from mean	Pre-determined	Isotropic	K-means
Single-link clustering	Maximum distance between a point and its nearest neighbor within a cluster	Data-dependent	Anisotropic	Greedy agglomerative clustering
Gaussian Mixture Models	$-\log P(X)$ , (equivalent to within-class squared distance from mean)	Isotropic	Pre-determined	EM
Spectral Clustering	Balanced cut	Pre-determined	Anisotropic	Laplacian Eigenmaps + Kmeans/ thresholding eigenvector signs

Cluster algorithm	Complexity	High dimensional data
k-means	$\mathcal{O}(K \cdot M \cdot N)$ time $\mathcal{O}(M + K)$ space	No
Hierarchical clustering	$\mathcal{O}(M^2)$ time $\mathcal{O}(M^2)$ space	No
DBSCAN (Density Based Spatial Clustering)	$\mathcal{O}(M \log M)$ time	No
DENCLUE (Density Based Clustering)	$\mathcal{O}(M \log M)$ time	Yes

See `pruebas_hdbscan.ipynb`

# DBSCAN vs Kmeans examples

- ✓ example\_dbSCAN.ipynb

# Assignment

## 1. ejemplo\_metricas\_distancias\_final.ipynb

Create a pandas dataframe to compute cosine and smoothed cosine distances for a given set of artists

# Some references... |

-  Andreopoulos, Bill, Aijun An, Xiaogang Wang, and Michael Schroeder (2009). “A roadmap of clustering algorithms: finding a match for a biomedical application”. In: *Briefings in Bioinformatics* 10.3, pp. 297–314.
-  Bishop, Christopher M (2006). *Pattern recognition and machine learning*. Springer.
-  Choi, Seung-Seok, Sung-Hyuk Cha, and Charles C Tappert (2010). “A survey of binary similarity and distance measures”. In: *Journal of Systemics, Cybernetics and Informatics* 8.1, pp. 43–48.
-  Gersho, Allen and Robert M Gray (2012). *Vector quantization and signal compression*. Vol. 159. Springer Science & Business Media.
-  Jain, Anil K and Richard C Dubes (1988). *Algorithms for clustering data*. Prentice-Hall, Inc.

# Some references... II

-  Jain, Anil K, M Narasimha Murty, and Patrick J Flynn (1999). “Data clustering: a review”. In: *ACM computing surveys (CSUR)* 31.3, pp. 264–323.
-  Ling, Maurice HT (2010). “COPADS, I: Distance Coefficients between Two Lists or Sets.”. In: *Python Papers Source Codes* 2.
-  Pearl, Judea (2009). *Causality*. Cambridge university press.
-  Selim, Shokri Z and Mohamed A Ismail (1984). “K-means-type algorithms: a generalized convergence theorem and characterization of local optimality”. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 1, pp. 81–87.
-  Taleb, Nassim Nicholas (2010). *The black swan:: The impact of the highly improbable fragility*. Vol. 2. Random House.
-  Xu, Rui and Donald Wunsch (2005). “Survey of clustering algorithms”. In: *IEEE Transactions on neural networks* 16.3, pp. 645–678.

# Clustering II

- 1** *Introduction*
- 2** *Initialization problems in K-means*
- 3** *Evaluation Criteria Measures*
- 4** *Determining the number of clusters*
- 5** *EM algorithm*
- 6** *Assignment*
- 7** *Bibliography*

# Initialization problem in K-means

- ✓ Extremely 💀 sensitive to cluster center initialization
- ✓ Bad initialization
  - 1. Poor convergence rate
  - 2. Bad overall clustering
- ✓ Safeguarding measures
  - 1. Choose first center, second which is the farthest from the first, third which is the farthest from both, so on
  - 2. Choose all centers randomly
  - 3. Try multiple initializations and choose the best result
  - 4. Use other initialization procedures (Pena et al. 1999)  
⇒ In sklearn → **kmeans++**

# kmeans++

**Input:**  $\mathcal{S}$  set of all possible objects (with  $N$  attributes, and  $\#\mathcal{S} = M$ )

**Input:**  $K$  number of clusters (user-defined parameter)

**Output:**  $K$  centroids,  $T_i$ , with  $T = \bigcup_i T_i$  and  $i \in [0, K]$

- 1 INITIALIZATION: select  $x \in \mathcal{S}$  randomly, and do  $T \leftarrow \{x\}$
- 2 **while**  $|T| < K$  **do**
- 3     pick  $x \in \mathcal{S}$  at random, with probability proportional to  
$$cost(x, T) = \min_{z \in T} \|x - z\|^2$$
- 4      $T \leftarrow T \cup \{x\}$
- 5 **end**

# Examples

kmeans\_initialization.ipynb

# Comparing clusterings

- ✓ Is a clustering algorithm sensitive to small perturbations?
- ✓ Is the algorithm sensitive to the order of the data?
- ✓ How similar are the solutions of two different algorithms?
- ✓ In case there exists optimal solution, how far are we from that solution?

# Why comparison?

- Robustness** To combine and improve the results of different clustering algorithms
- Re-use** Old clusterings that cannot be reconstructed but can be useful
- Distribution computation/integration** Databases geographically split and centralization leads to ↑ computational, bandwidth, and storage costs
- Legal compliance** Legal restrictions impose several copies of data, each with a different feature set (think about anonymized data) ⇒ feature distributed clustering + integration into one *mean value* clustering
- Integration of different optimization criteria** In some scenarios, as in social sciences, we could have different clusterings obtained using several optimization criteria, e.g., different distance/similarity measures. A procedure to proper integration is required (Li et al. 2004)

# Measures to compare clusters

As underlined in (S. Wagner and D. Wagner 2007):

1. Measures based on counting pairs
2. Measures based on set overlaps
3. Measures based on mutual information

# Some definitions and notations

- ✓ Let  $X$  be a set of finite set with cardinality  $|X| = n$
- ✓ Clustering  $\mathcal{C}$  is a set  $\{\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_k\}$  with
  - $\mathcal{C}_i \cap \mathcal{C}_j = \emptyset \quad \forall i \neq j$ ,
  - $X = \bigcup_{i=1}^k \mathcal{C}_i$ , and assuming
  - $|\mathcal{C}_i| \neq 0 \quad \forall i \in \{1, 2, \dots, k\}$
- ✓  $\mathcal{C}(X) \equiv$  the set of all clusterings of  $X$
- ✓ Let  $\mathcal{C}' = \{\mathcal{C}'_1, \dots, \mathcal{C}'_l\} \in \mathcal{C}(X)$  be a second cluster of  $X$

Definition (The confusion matrix  $M = (m_{ij})$  or contingency table of the pair  $\mathcal{C}, \mathcal{C}'$ )

*It is a  $k \times l$  matrix whose  $ij$ -the entry is the number of elements in the intersection of the clusters  $\mathcal{C}_i$  and  $\mathcal{C}'_j$ , i.e.,*

$$m_{ij} = |\mathcal{C}_i \cap \mathcal{C}'_j|, \quad i \in \{1, \dots, k\}, j \in \{1, \dots, l\}$$

# Measures based on counting pairs

The set of all pairs of elements of  $X$  is the disjoint union of the following sets:

$$\mathcal{S}_{11} = \{\text{pairs included in the same cluster under } \mathcal{C} \text{ and } \mathcal{C}'\}$$

$$\mathcal{S}_{00} = \{\text{pairs included in different clusters under } \mathcal{C} \text{ and } \mathcal{C}'\}$$

$$\mathcal{S}_{10} = \{\text{pairs included in the same cluster under } \mathcal{C} \text{ but in different ones under } \mathcal{C}'\}$$

$$\mathcal{S}_{01} = \{\text{pairs included in different clusters under } \mathcal{C} \text{ but in the same one under } \mathcal{C}'\}$$

If  $n_{ab} \triangleq |\mathcal{S}_{ab}|, a, b \in 0, 1$  is the size of the respective set, then

$$n_{11} + n_{00} + n_{10} + n_{01} = \binom{n}{2}$$

# Rand index I

## Definition (General rand index)

*It defines the ratio between the number of elements correct and uncorrectly classified and the total number of elements*

$$\mathcal{R}(\mathcal{C}, \mathcal{C}') = \frac{2(n_{11} + n_{00})}{n(n - 1)}$$

This measure is very dependent upon the number of clusters and, in the unlikely case of independent clusterings, it converges to 1 as the number of clusters increases (no desirable at all)

# Rand index II

## Definition (Adjusted Rand Index)

*Assuming a generalized hypergeometric distribution as null hypothesis, it is given by*

$$\mathcal{R}_{adj}(\mathcal{C}, \mathcal{C}') = \frac{\sum_{i=1}^k \sum_{j=1}^l \binom{m_{ij}}{2} - t_3}{\frac{1}{2}(t_1 + t_2) - t_3}$$

where  $t_1 = \sum_{i=1}^k \binom{|\mathcal{C}_i|}{2}$ ,  $t_2 = \sum_{j=1}^l \binom{|\mathcal{C}'_j|}{2}$ , and  $t_3 = \frac{2t_1 t_2}{n(n-1)}$

## PROBLEMS

- ✓ Strong assumptions on the distribution
- ✓ Sensitivity to the number of clusters

# Rand index III

- ✓ The adjusted version can take negative values, but it should be in the interval  $[0, 1]$
- ✓ For  $n/k > 3$ , the base-line of  $R_{adj}$  varies too much (Meilă 2007)
- ✓ High scores for clusterings with large number of clusters, since eventually all instances end up alone in a cluster

# Measures to compare clusters

As underlined in (S. Wagner and D. Wagner 2007):

1. Measures based on counting pairs
2. **Measures based on set overlaps**
3. Measures based on mutual information

Now, clusterings that have a maximum absolute or relative overlap.  
Let us consider just one example:

### Definition ( $\mathcal{F}$ -measure)

The  $\mathcal{F}$ -measure for a cluster  $\mathcal{C}'_j$  with respect to a certain class  $\mathcal{C}_i$  indicates how good the cluster  $\mathcal{C}'_j$  describes the class  $\mathcal{C}_i$ . To do so, first it is calculated the harmonic mean of precision

$$p_{ij} = \frac{m_{ij}}{|\mathcal{C}'_j|}$$

and recall

$$r_{ij} = \frac{m_{ij}}{|\mathcal{C}_i|}$$

which leads to

$$\mathcal{F}(\mathcal{C}_i, \mathcal{C}'_j) = \frac{2 \cdot r_{ij} \cdot p_{ij}}{r_{ij} + p_{ij}}$$

The overall  $\mathcal{F}$ -measure is

$$\mathcal{F}(\mathcal{C}, \mathcal{C}') = \mathcal{F}(\mathcal{C}') = \frac{1}{n} \sum_{i=1}^n |\mathcal{C}_i| \max_{j=1}^l \mathcal{F}(\mathcal{C}_i, \mathcal{C}'_j)$$

# Problems of the measures based on overlaps

This kind of measures do not take into account unmatched parts of the clusters. For example

- ✓ Consider  $\mathcal{C}'$  obtained from  $\mathcal{C}$  just by shifting a fraction  $\alpha$  of the elements of each cluster  $C_i$  to the *next* cluster  $\mathcal{C}_{(i+1) \bmod k}$
- ✓ Let  $\mathcal{C}''$  be a clustering derived from  $\mathcal{C}$  by a reassigning a fraction  $\alpha$  of the elements in each cluster  $\mathcal{C}_i$  evenly between the others clusters
- ✓ If  $\alpha < 0.5 \Rightarrow \mathcal{F}(\mathcal{C}, \mathcal{C}') = \mathcal{F}(\mathcal{C}, \mathcal{C}'') \dots$  but  $\mathcal{C}'$  is a less modified version of  $\mathcal{C}$  than  $\mathcal{C}''$ !!!

# Measures to compare clusters

As underlined in (S. Wagner and D. Wagner 2007):

1. Measures based on counting pairs
2. Measures based on set overlaps
3. Measures based on mutual information

# Measures based on mutual information I

Definition (The entropy associated with clustering  $\mathcal{C}$ )

$$\mathcal{H}(\mathcal{C}) = - \sum_{i=1}^k P(i) \log_2(P(i))$$

with  $P(i) = \frac{|\mathcal{C}_i|}{n}$

# Measures based on mutual information II

Definition (The mutual information between two clusterings  $\mathcal{C}$ ,  $\mathcal{C}'$ )

$$\mathcal{I}(\mathcal{C}, \mathcal{C}') = \sum_{i=1}^k \sum_{j=1}^k P(i, j) \log_2 \frac{P(i, j)}{P(i)P(j)}$$

where  $P(i, j)$  is the probability of an element belonging to cluster  $\mathcal{C}_i$  in  $\mathcal{C}$  and to cluster  $\mathcal{C}'_j$  in  $\mathcal{C}'$

$$P(i, j) = \frac{|\mathcal{C}_i \cap \mathcal{C}'_j|}{n}$$

The mutual information is a metric on the space of all clusterings

- ✓ It is not bounded by a constant value  $\Rightarrow$  difficult to interpret
- ✓  $\mathcal{I}(\mathcal{C}, \mathcal{C}') \leq \min\{\mathcal{H}(\mathcal{C}), \mathcal{H}(\mathcal{C}')\}$

## Definition (Normalized mutual information by Strehl & Ghosh)

$$NMI_{SG}(\mathcal{C}, \mathcal{C}') = \frac{\mathcal{I}(\mathcal{C}, \mathcal{C}')}{\sqrt{\mathcal{H}(\mathcal{C})\mathcal{H}(\mathcal{C}')}}$$

1.  $0 \leq NMI_{SG}(\mathcal{C}, \mathcal{C}') \leq 1$
2.  $NMI_{SG}(\mathcal{C}, \mathcal{C}') = 1 \Rightarrow \mathcal{C} = \mathcal{C}'$
3.  $NMI_{SG}(\mathcal{C}, \mathcal{C}') = 0 \Rightarrow \forall i \in \{1, \dots, k\}, \text{ and } \forall j \in \{1, \dots, l\}, P(i, j) = 0 \text{ OR } P(i, j) = P(i) \cdot P(j)$

- 1** *Introduction*
- 2** *Initialization problems in K-means*
- 3** *Evaluation Criteria Measures*
- 4** *Determining the number of clusters*
  - *Methods based on intra-cluster scatter*
  - *Methods based on inter- and intra-cluster scatter*
- 5** *EM algorithm*
- 6** *Assignment*
- 7** *Bibliography*

# Determining the number of clusters

- ✓ Most criteria (as SSE -Smallest Square Error-) are monotonically decreasing in  $K$  (i.e., the number of clusters)  $\Rightarrow$  leads to the trivial cluster (one item per cluster)
- ✓ Alternatives mainly based on heuristic methodologies

# Methods based on intra-cluster scatter

For example:

$$W_K = \sum_{k=1}^K \frac{1}{2N_k} D_k$$

with  $D_K$  as the sum of the pairwise distances for all instances in cluster  $k$

$$D_k = \sum_{x_i, x_j \in \mathcal{C}_k} \|x_i - x_j\|$$

Usually as the number of clusters increases, the within-cluster **first decay**. From a **certain value of  $K$** , the curve flattens  $\Rightarrow$  this gives the proper value of  $K$

# Methods based on inter- and intra-cluster scatter

Definition (Mean Intra-Cluster Distance for the  $k$ -th cluster)

$$MICD_k = \sum_{x_j \in \mathcal{C}_k} \frac{\|x_i - \mu_k\|}{n_k}$$

- ✓ Data under-partitioned  $K < K^*$ , at least one cluster has large  $MICD$
- ✓ As the partition state moves towards over-partitioned ( $K > K^*$ ), the large  $MICD$  abruptly decreased

# Methods based on inter- and intra-cluster scatter

## Definition (Inter-Cluster Minimum Distance)

$$ICMD = \min_{i \neq j} ||\mu_i - \mu_j||$$

- ✓ The  $ICMD$  is large when the data are under-partitioned or optimally partitioned
- ✓ ↓↓ when the data enters the over-partitioned state

# Methods based on inter- and intra-cluster scatter

- ⇒ Several measures are derived from the previous ones to better capture the under-/over-partitioned nature of a given clustering
- ⇒ Criteria based on probabilistic measures: Bayesian Information Criterion (BIC), Minimum Message Length (Minimum Message Length), Minimum Description Length (MDL)

- ✓ example\_pca2.ipynb
- ✓ Time series Anomaly detection example:  
<http://amid.fish/anomaly-detection-with-k-means-clustering>

# Some useful examples from sklearn

- ✓ The Silhouette coefficient: [http://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette\\_score.html](http://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html)
- ✓ <http://scikit-learn.org/stable/modules/clustering.html#silhouette-coefficient>
- ✓ [http://scikit-learn.org/stable/auto\\_examples/cluster/plot\\_kmeans\\_silhouette\\_analysis.html](http://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html)

- 1** *Introduction*
- 2** *Initialization problems in K-means*
- 3** *Evaluation Criteria Measures*
  - *Counting pairs*
  - *Set overlaps*
  - *Mutual Information*
- 4** *Determining the number of clusters*
  - *Methods based on intra-cluster scatter*
  - *Methods based on inter- and intra-cluster scatter*
- 5** *EM algorithm*
  - *Trivial example*
  - *Formalization*
  - *Classification*
  - *EM clustering*
  - *Comparison with K-means*
- 6** *Assignment*
- 7** *Bibliography*

# EM trivial example

Let us consider the grades in a class:

- ✓  $a$  students get an  $A \Rightarrow P(A) = \frac{1}{2}$
- ✓  $b$  students get a  $B \Rightarrow P(B) = \mu$
- ✓  $c$  students get a  $C \Rightarrow P(C) = 2\mu$
- ✓  $d$  students get a  $D \Rightarrow P(D) = \frac{1}{2} - 3\mu$

with  $\mu \in [0, 1/6]$

GOAL: get an estimation of  $\mu$  from data b answering

⇒ What's the maximum likelihood estimate of  $\mu$  given  $a$ ,  $b$ ,  $c$ , and  $d$ ?

# Trivial example: solution

1.  $P(a, b, c, d|\mu) = (\frac{1}{2})^a \cdot \mu^b \cdot (2\mu)^c \cdot (\frac{1}{2} - 3\mu)^d$
2.  $\log(P(a, b, c, d|\mu)) =$   
 $a \cdot \log(1/2) + b \cdot \log \mu + c \log(2\mu) + d \log(1/2 - 3\mu)$
3. Max w.r.t.  $\mu \Rightarrow \frac{\partial \log P}{\partial \mu} = 0$
4.  $\frac{\partial \log P}{\partial \mu} = \frac{b}{\mu} + \frac{2c}{2\mu} - \frac{3d}{1/2 - 3\mu} = \frac{(b+c)(1-6\mu) - 6d\mu}{\mu(1-6\mu)} = 0$
5.  $\mu = \frac{b+c}{6(b+c+d)}$

In case  $a = 14$ ,  $b = 6$ ,  $c = 9$ ,  $d = 10 \Rightarrow \mu = \frac{1}{10}$

# In case hidden information?

Let us consider we just know:

1. Number of high grades ( $A$ 's and  $B$ 's) is  $h$
2. Number of  $C$ 's is  $c$
3. Number of  $D$ 's is  $d$

The ratio  $a : b$  should be the same as the ratio

$P(A) : P(B) \equiv 1/2 : \mu$ . In the **expectation phase**:

$$a = \frac{1/2}{1/2 + \mu} h$$

$$b = \frac{\mu}{1/2 + \mu} h$$

# Hidden information: maximization stage

If we know the values of  $a$  and  $b$  we can get  $\mu$  through the maximum likelihood

$$\mu = \frac{b + c}{6(b + c + d)}$$

# EM for our trivial problem

1. A first guess for  $\mu$
2. Iterate between expectation and maximization to improve the estimates of  $\mu$ ,  $a$ , and  $b$

Definitions:

- ✓  $\mu(t)$  the estimate of  $\mu$  in the  $t$ -th iteration
- ✓  $b(t)$  the estimate of  $b$  in the  $t$ -th iteration
- ✓  $\mu(0) \equiv$  initial guess
- ✓ **E-step**

$$b(t) = \frac{\mu(t)h}{1/2 + \mu(t)} = E[b|\mu(t)]$$

- ✓ **M-step**

$$\mu(t+1) = \frac{b(t) + c}{6(b(t) + c + d)}$$

# EM convergence

It can be proven converge to a local optimum

In our example, let us consider  $h = 20$ ,  $c = 10$ ,  $d = 10$ ,  $\mu(0) = 0$ :

$t$	$\mu(t)$	$b(t)$
0	0	0
1	0.08333	2.85714
2	0.09375	3.15789
3	0.09469	3.18452
4	0.09478	3.18706
5	0.09479	3.18734
6	0.09479	3.18734

# Normal Sample I

- ✓  $X \sim N(\mu, \sigma^2)$
- ✓ Given  $n$  samples  $\mathbf{x} = [x_1, x_2, \dots, x_n]^T$
- ✓ And assuming

$$f(x_i|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

- ✓  $\hat{\mu}, \hat{\sigma}^2??$
- ✓ Another assumption:  $x_i$  are i.i.d. (independent and identically distributed)

$$f(\mathbf{x}|\mu, \sigma^2) = \prod_{i=1}^n f(x_i|\mu, \sigma^2) = \left( \frac{1}{2\pi\sigma^2} \right)^{(n/2)} e^{-\sum_{i=1}^n \frac{(x_i-\mu)^2}{2\sigma^2}}$$

- ✓ We want to maximize

$$\mathcal{L}(\mu, \sigma^2 | \mathbf{x}) = f(\mathbf{x}|\mu, \sigma^2)$$

# Log-Likelihood function

Taken into account that:

$$x < y \Rightarrow \log(x) < \log(y)$$

instead of the likelihood, we are maximizing

$$\begin{aligned} I(\mu, \sigma^2 | \mathbf{x}) &= \log(\mathcal{L}(\mu, \sigma^2 | \mathbf{x})) = \\ &= -\frac{n}{2} \log \sigma^2 - \frac{n}{2} \log 2\pi - \frac{1}{2\sigma^2} \sum_{i=1}^n x_i^2 + \frac{\mu}{\sigma^2} \sum_{i=1}^n x_i - \frac{n\sigma^2}{2\sigma^2} \end{aligned}$$

and thus  $\mu$  and  $\sigma^2$  are derived by imposing

$$\frac{\partial I(\mu, \sigma^2 | \mathbf{x})}{\partial \mu} = 0, \quad \frac{\partial I(\mu, \sigma^2 | \mathbf{x})}{\partial \sigma^2} = 0$$

# Max. the Log-Likelihood function

$$\frac{\partial I(\mu, \sigma^2 | \mathbf{x})}{\partial \mu} \Bigg|_{\mu=\hat{\mu}} = \frac{1}{\sigma^2} \sum_{i=1}^n x_i - \frac{n\hat{\mu}}{\sigma^2} = 0 \Rightarrow \hat{\mu} = \frac{\sum_{i=1}^n x_i}{n}$$

$$\frac{\partial I(\mu, \sigma^2 | \mathbf{x})}{\partial \sigma^2} \Bigg|_{\mu=\hat{\mu}, \sigma^2=\hat{\sigma}^2} = -\frac{n}{2\hat{\sigma}^2} - \frac{1}{\hat{\sigma}^4} + \frac{\hat{\mu}}{\hat{\sigma}^4} \sum_{i=1}^n x_i - \frac{n\hat{\mu}^2}{2\hat{\sigma}^4} = 0$$

$$n\hat{\sigma}^2 = \sum_{i=1}^n x_i^2 - 2\hat{\mu} \sum_{i=1}^n x_i + n\hat{\mu}^2$$

$$= \sum_{i=1}^n x_i^2 - \frac{2}{n} \left( \sum_{i=1}^n x_i \right)^2 + \frac{1}{n} \left( \sum_{i=1}^n x_i \right)^2 \Rightarrow \hat{\sigma}^2 = \frac{1}{n} \sum_{i=1}^n x_i^2 - \hat{\mu}^2$$

# Consider EM for classification

- ✓ Given a training data set

$$\mathbf{x} = \{x_1, x_2, \dots, x_n\}$$

- ✓ Class labels

$$\mathbf{z} = \{z_i, z_2, \dots, z_n\}$$

- ✓ Data is modelled by a joint distribution

$$p(x_i, z_i) = p(x_i|z_i)p(z_i)$$

- ✓ Assumption:  $z_i \sim \text{multinomial}(\boldsymbol{\theta})$

$$\boldsymbol{\theta} = [\theta_1, \theta_2, \dots, \theta_k]^T, \quad \theta_j \geq 0, \quad \sum_{j=1}^k \theta_j = 1, \quad \theta_j = p(z_i = j)$$

$$x_i|z_i = j \sim \mathcal{N}(\mu_j, \sum_j)$$

- ✓ Known data:  $\mathbf{x}, \mathbf{z}$ ; Unknown parameters:  $\boldsymbol{\mu}, \boldsymbol{\theta}, \boldsymbol{\Sigma}$

# EM clustering algorithm

- ✓ Given a training data set

$$\mathbf{x} = \{x_1, x_2, \dots, x_n\}$$

- ✓ Class labels

$$\mathbf{z} = \{z_1, z_2, \dots, z_n\}$$

- ✓ In clustering  $\mathbf{x}$  is given and  $\mathbf{z}$  is unknown

- ✓ Expectation

✗ If the expected values of  $\mathbf{z}$  are known, it is possible to compute the maximum likelihood value of  $\boldsymbol{\mu}$ ,  $\boldsymbol{\theta}$ ,  $\boldsymbol{\Sigma}$

- ✓ Maximization

✗ If the values of  $\boldsymbol{\mu}$ ,  $\boldsymbol{\theta}$ ,  $\boldsymbol{\Sigma}$  are known, it is possible to compute the expected values of  $\mathbf{z}$

- ✓ We start with a guess for  $\boldsymbol{\mu}$ ,  $\boldsymbol{\theta}$ ,  $\boldsymbol{\Sigma}$ . Then iterate between EXPECTATION and MAXIMIZATION... until converge

# EM clustering: the procedure I

The starting point, the log-likelihood

$$I(\boldsymbol{\theta}) = \sum_{i=1}^n \log p(x_i | z_i; \boldsymbol{\mu}, \boldsymbol{\Sigma}) + \log p(z_i; \boldsymbol{\theta})$$

Second point, maximization of the log-likelihood with respect to  $\boldsymbol{\theta}$ ,  $\boldsymbol{\mu}$ ,  $\boldsymbol{\Sigma}$ . The result:

✓  $\theta_j = \frac{1}{n} \sum_{i=1}^n 1\{z_i = j\}$

✓  $\mu_j = \frac{\sum_{i=1}^n 1\{z_i=j\}x_i}{\sum_{i=1}^n 1\{z_i=j\}}$

✓  $\Sigma_j = \frac{\sum_{i=1}^n 1\{z_i=j\}(x_i - \mu_j)(x_i - \mu_j)^T}{\sum_{i=1}^n 1\{z_i=j\}}$

# EM clustering: the procedure II

Third step, repeat EXPECTATION and MAXIMIZATION until convergence:

**expectation** For each  $i, j$  set

$$w_{j,i} \triangleq p(z_i = j | x_i; \boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$$

**maximization** Update de parameters

$$\theta_j \triangleq \frac{1}{n} \sum_{i=1}^n w_{j,i}, \quad \mu_j \triangleq \frac{\sum_{i=1}^n w_{j,i} x_i}{\sum_{i=1}^n w_{j,i}}$$

$$\Sigma_j \triangleq \frac{\sum_{i=1}^n w_{j,i} (x_i - \mu_j)(x_i - \mu_j)^T}{\sum_{i=1}^n w_{j,i}}$$

# Comparison with K-means

- ✓ Given a training data set

$$\mathbf{x} = \{x_1, x_2, \dots, x_n\}$$

- ✓ Class labels

$$\mathbf{z} = \{z_1, z_2, \dots, z_n\}$$

- ✓ Assumption:  $z_i \sim \text{multinomial}(\boldsymbol{\theta})$

$$\boldsymbol{\theta} = [\theta_1, \theta_2, \dots, \theta_k]^T, \quad \theta_j \geq 0, \quad \sum_{j=1}^k \theta_j = 1, \quad \theta_j = p(z_i = j)$$

$$x_i | z_i = j \sim \mathcal{N}(\mu_j, \sum_j)$$

- ✓ K-means is a simplified EM

✗  $\theta_i = \frac{1}{k}, \forall i, \Sigma_i = \Sigma_j \forall i \neq j, i \in \{1, 2, \dots, k\}$

✗  $k$  is given by user

✗  $\mu_1, \mu_2, \dots, \mu_k \Rightarrow$  the means of clusters, are the only unknown parameters of the model

# Assignment

kmeans\_vs\_gmm-1617.ipynb

# Some references... I

-  Bishop, Christopher M (2006). *Pattern recognition and machine learning*. Springer.
-  Guttag, John V (2013). *Introduction to Computation and Programming Using Python*. MIT Press.
-  Li, Tao, Mitsunori Ogihara, and Sheng Ma (2004). "On combining multiple clusterings". In: *Proceedings of the thirteenth ACM international conference on Information and knowledge management*. ACM, pp. 294–303.
-  Meilă, Marina (2007). "Comparing clusterings—an information based distance". In: *Journal of multivariate analysis* 98.5, pp. 873–895.
-  Pena, José Manuel, Jose Antonio Lozano, and Pedro Larrañaga (1999). "An empirical comparison of four initialization methods for the k-means algorithm". In: *Pattern recognition letters* 20.10, pp. 1027–1040.

# Some references... II



- Wagner, Silke and Dorothea Wagner (2007). *Comparing clusterings: an overview*. Universität Karlsruhe, Fakultät für Informatik Karlsruhe.

# Feature Extraction (Extracción de características)

# Reducción de la Dimensionalidad

Dos estrategias:

**Selección de características:** se selecciona un subconjunto de los atributos originales.

- **Algoritmos de Filtrado:** la calidad de los atributos se mide usando alguna medida estadística general (Chi Cuadrado, Correlación, Información Mutua, etc.)
- **Algoritmos de Wrapping:** la calidad de un conjunto de atributos se evalúa calculando la calidad promedio de un modelo entrenado con esos atributos
- **Algoritmos embebidos:** la selección es parte del algoritmo de aprendizaje

**Extracción / construcción de características:** la idea es construir nuevas características que condensen / resuman la información relevante de los atributos originales.  
Por ejemplo PCA, LDA

# Reducción de la dimensionalidad con transformaciones de los atributos (1)

- Idea: encontrar una transformación  $y=f(x)$  que conserve la información acerca del problema, minimizando el número de componentes
- En general, la función óptima  $y=f(x)$  será no lineal
- Sin embargo, no hay una forma de generar sistemáticamente transformaciones no lineales:
  - La selección de un subconjunto particular de transformaciones depende del problema
  - Por esta razón, la limitación a transformaciones lineales ha sido ampliamente aceptada,  $y = W^T x$   
**→ y es una proyección lineal de x**

# Reducción de la dimensionalidad con transformaciones de los atributos (2)

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} \xrightarrow{\text{Transformación lineal}} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_M \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & \cdots \\ w_{21} & w_{22} & \cdots \\ \vdots & \vdots & \ddots \\ w_{M1} & w_{M2} & \cdots \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}$$

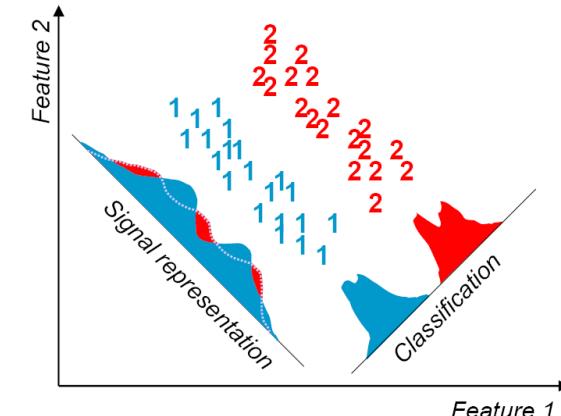
*M-dimensional*  
*M < N*

*N-dimensional*

- Por el momento nos centraremos en transformaciones lineales

## Representación de la señal versus clasificación (PCA vs. LDA)

- La selección de la transformación extractora de características,  $y=f(x)$ , está guiada por una función objetivo que buscamos maximizar (o minimizar)
- Dependiendo del criterio usado por la función objetivo, las técnicas de extracción de características se dividen en dos categorías:
  - **Clasificación:** El objetivo de la transformación extractora de características es resaltar en un espacio de menos dimensiones la información discriminante de clases
  - **Representación de la señal:** El objetivo de la transformación extractora de características es representar los vectores de atributos de manera precisa en un espacio de menos dimensiones
- Hay dos técnicas principales en la extracción lineal de características:
  - Análisis Discriminante Lineal (LDA), que utiliza el criterio de clasificación
  - Análisis de Componentes Principales (PCA), que usa el criterio de representación de la señal



## Análisis Discriminante Lineal (LDA)

- **Análisis Discriminante Lineal, dos clases**
- **Análisis Discriminante Lineal, C clases**
- **Limitaciones de LDA**
- **Variantes de LDA**
- **Otros métodos de reducción de la dimensionalidad basados en LDA**

## Análisis Discriminante Lineal, dos clases (1)

- El objetivo de **LDA** es realizar una **reducción de la dimensionalidad** preservando el máximo posible de **información sobre la clase**.

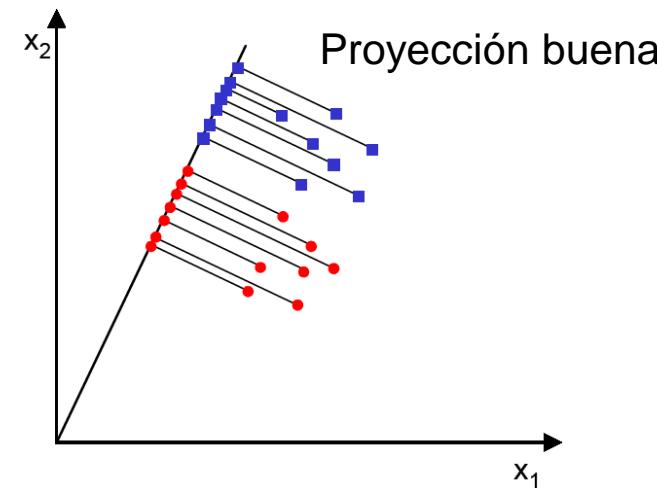
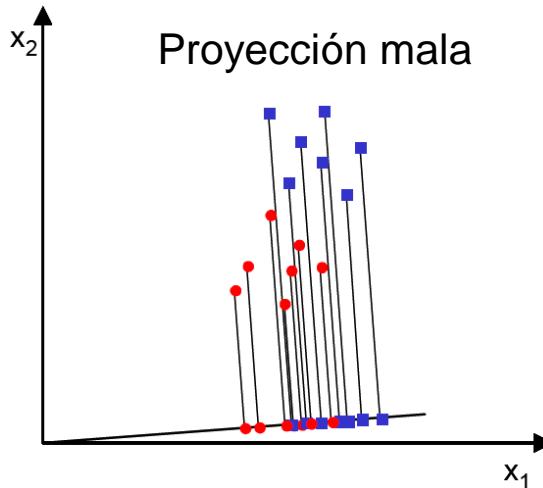
Tenemos un conjunto de vectores en **D** dimensiones  $\{x_1, x_2, \dots, x_N\}$ , donde **N<sub>1</sub>** son de clase **w<sub>1</sub>**, y **N<sub>2</sub>** de clase **w<sub>2</sub>**

Buscamos obtener un nuevo feature “y” proyectando los vectores **x** sobre un vector **w**:

$$y = w^T x$$

## Análisis Discriminante Lineal, dos clases (2)

- De todos los posibles  $w$ , nos gustaría seleccionar el que maximiza la separación de las clases en la proyección  $y = w^T x$
- Ilustramos a continuación esta idea para el caso de vectores  $x$  con 2 dimensiones:



## Análisis Discriminante Lineal, dos clases (3)

- Para poder encontrar un buen vector de proyección, necesitaremos **definir una medida de separación entre las proyecciones**
  - El vector promedio de cada clase en los espacios **x** e **y** es:

$$\mu_i = \frac{1}{N_i} \sum_{x \in \omega_i} x$$

Espacio original

$$\tilde{\mu}_i = \frac{1}{N_i} \sum_{y \in \omega_i} y = \frac{1}{N_i} \sum_{x \in \omega_i} w^T x = w^T \mu_i$$

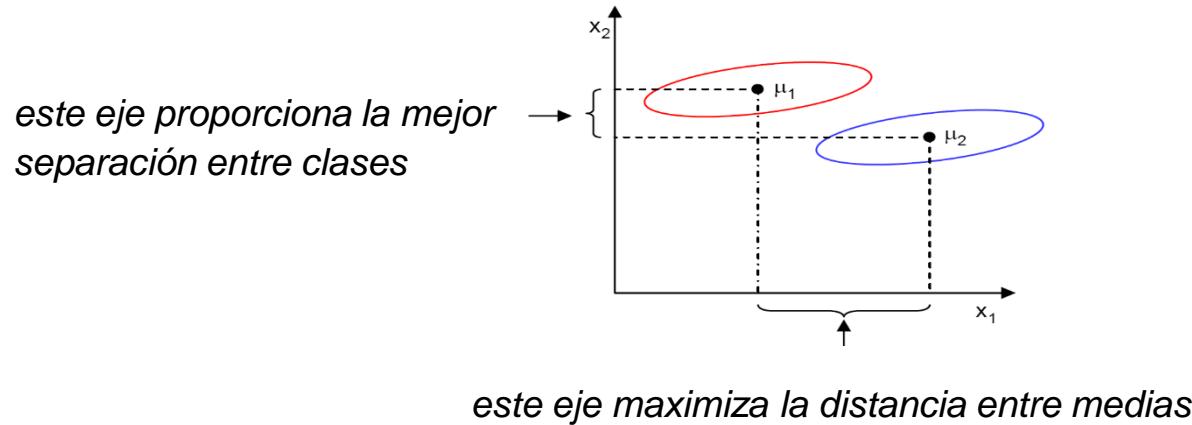
Espacio de la proyección

- Podríamos entonces elegir nuestra función objetivo como la distancia entre los promedios proyectados: clasificación por la distancia a las medias

$$J(w) = |\tilde{\mu}_1 - \tilde{\mu}_2| = |w^T(\mu_1 - \mu_2)|$$

## Análisis Discriminante Lineal, dos clases (4)

- Sin embargo, la distancia entre los promedios proyectados no es una buena medida ya que no tiene en cuenta la desviación standard dentro de las clases.



## Análisis Discriminante Lineal, dos clases (5)

- La solución propuesta por Fisher es maximizar una función que representa la diferencia entre las medias, normalizada por una medida de la dispersión dentro de las clases
  - Por cada clase definimos la dispersión, un equivalente a la varianza, como:

$$\tilde{s}_i^2 = \sum_{y \in \omega_i} (y - \tilde{\mu}_i)^2$$

- donde la cantidad  $(\tilde{s}_1^2 + \tilde{s}_2^2)$  es la dispersión intra clase de los ejemplos proyectados considerando igualdad en la probabilidad a priori de las clases →  $\tilde{S}_W$

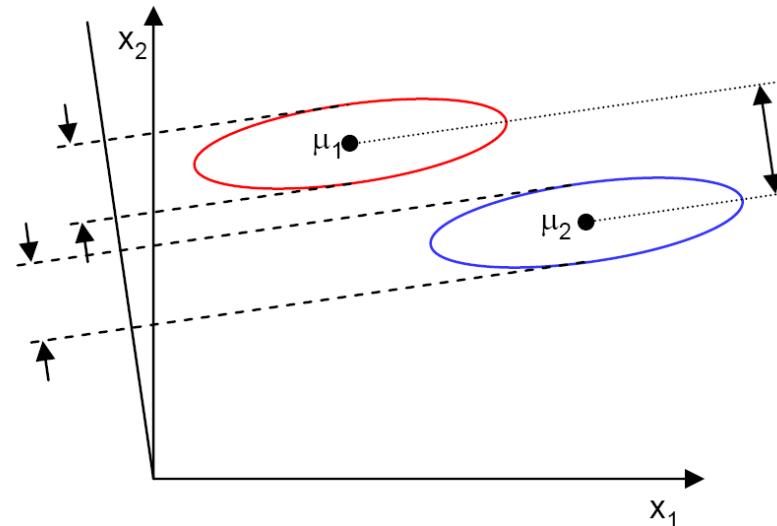
- El discriminante lineal de Fisher se define como la función lineal  $w^T x$  que maximiza la función objetivo:

$$J(w) = \frac{|\tilde{\mu}_1 - \tilde{\mu}_2|^2}{\tilde{s}_1^2 + \tilde{s}_2^2}$$


- 1. Diferencia entre medias proyectadas aumenta
- 2. Dispersión intra-clases en la proyección disminuye

## Análisis Discriminante Lineal, dos clases (6)

- De esta forma, estaremos buscando una proyección donde los ejemplos de la misma clase son proyectados muy cerca unos de otros (mínima dispersión), y al mismo tiempo, las medias proyectadas están lo más lejos posible.



## Análisis Discriminante Lineal, dos clases (7)

- Para poder encontrar la proyección óptima  $\mathbf{w}^*$ , necesitaremos expresar  $J(\mathbf{w})$  como una función explícita de  $\mathbf{w}$
- Primero definiremos las matrices de dispersión en el espacio original:

$$S_i = \sum_{x \in \omega_i} (x - \mu_i)(x - \mu_i)^T$$
$$S_w = \pi_1 S_1 + \pi_2 S_2$$

- donde  $S_w$  es la llamada “matriz de dispersión intra clase”

- La dispersión de la proyección y se puede expresar en función de la matriz de dispersión en el espacio original  $x$ :

$$\tilde{S}_i = \sum_{y \in \omega_i} (y - \tilde{\mu}_i)(y - \tilde{\mu}_i)^T = \sum_{x \in \omega_i} (w^T x - w^T \mu_i)(w^T x - w^T \mu_i)^T = \sum_{x \in \omega_i} w^T (x - \mu_i)(x - \mu_i)^T w = w^T S_i w$$

$$\tilde{S}_w = \pi_1 \tilde{S}_1 + \pi_2 \tilde{S}_2 = \pi_1 w^T S_1 w + \pi_2 w^T S_2 w = w^T (\pi_1 S_1 + \pi_2 S_2) w = w^T S_w w$$

## Análisis Discriminante Lineal, dos clases (8)

- De manera similar, podemos expresar la diferencia entre los promedios proyectados en función de las medias en el espacio original x:

$$(\tilde{\mu}_1 - \tilde{\mu}_2)^2 = (w^T \mu_1 - w^T \mu_2)^2 = w^T \underbrace{(\mu_1 - \mu_2)(\mu_1 - \mu_2)^T}_{S_B} w = w^T S_B w$$

$S_B$  es la “matriz de dispersión interclase”.

Como es el producto externo de un vector consigo mismo, tiene rango  $\leq 1$

## Análisis Discriminante Lineal, dos clases (9)

- Con lo que hemos visto, podemos expresar  $J(w)$  como una función explícita de  $w$ :

$$J(w) = \frac{|\tilde{\mu}_1 - \tilde{\mu}_2|^2}{\tilde{s}_1^2 + \tilde{s}_2^2} \longrightarrow J(w) = \frac{w^T S_B w}{w^T S_W w}$$

- Maximizar  $J(w)$  respecto a  $w$  tiene una solución analítica sencilla:

$$w^* = \underset{w}{\operatorname{argmax}} \left\{ \frac{w^T S_B w}{w^T S_W w} \right\} = S_W^{-1}(\mu_1 - \mu_2)$$

donde el módulo de  $w^*$  es indiferente

Esta solución es el famoso Discriminante Lineal de Fisher (1936), aunque en realidad no es un discriminante sino la elección de una dirección específica para la proyección de los datos a una dimensión

## Ejemplo de LDA

- Calcular la proyección LDA para el siguiente conjunto de datos en dos dimensiones:

$$X_1 = (x_1, x_2) = \{(4,1), (2,4), (2,3), (3,6), (4,4)\}$$

$$X_2 = (x_1, x_2) = \{(9,10), (6,8), (9,5), (8,7), (10,8)\}$$

- Solución (a mano):

- Las estadísticas de las clases son  $\mu_1 = \begin{pmatrix} 3.0 \\ 3.6 \end{pmatrix}, \mu_2 = \begin{pmatrix} 8.4 \\ 7.6 \end{pmatrix}$

- Las matrices de dispersión inter- e intra-clase son:

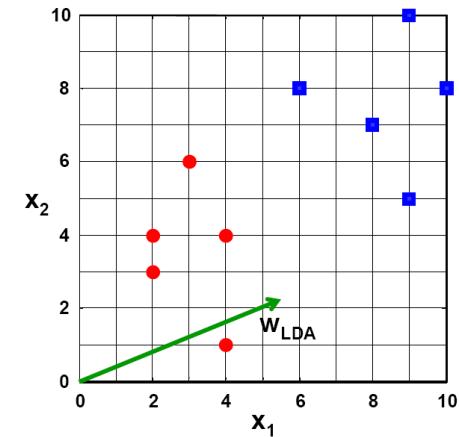
$$S_B = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T = \begin{pmatrix} 29.16 & 21.60 \\ 21.60 & 16.00 \end{pmatrix}$$

$$S_i = \sum_{x \in \omega_i} (x - \mu_i)(x - \mu_i)^T \quad S_1 = \begin{pmatrix} 4 & -2 \\ -2 & 13.2 \end{pmatrix}, S_2 = \begin{pmatrix} 4 & -2 \\ -2 & 13.2 \end{pmatrix}$$

$$S_W = \pi_1 S_1 + \pi_2 S_2 = \begin{pmatrix} 2.67 & 1.33 \\ 1.33 & 8 \end{pmatrix}$$

- La proyección óptima viene entonces dada por:

$$w^* = S_W^{-1}(\mu_1 - \mu_2) = [-0.91 \quad -0.39]^T$$



## Análisis Discriminante Lineal, C clases (1)

- El Discriminante de Fisher se puede generalizar a problemas con **C** clases (arbitrario)
- En vez de buscar una proyección **y** (escalar), buscamos **(C-1)** proyecciones **[y<sub>1</sub>, y<sub>2</sub>, ..., y<sub>C-1</sub>]** por medio de **(C-1)** vectores de proyección **w<sub>i</sub>**.
- Definimos por conveniencia la matriz de proyección **W** con **(C-1)** columnas:  
**W = [ w<sub>1</sub> | w<sub>2</sub> | ... | w<sub>C-1</sub> ]**

$$y_i = w_i^T x \Rightarrow y = W^T x$$

# Análisis Discriminante Lineal, C clases (2)

- Matriz de dispersión intra-clase

$$S_W = \sum_{i=1}^C \pi_i S_i$$

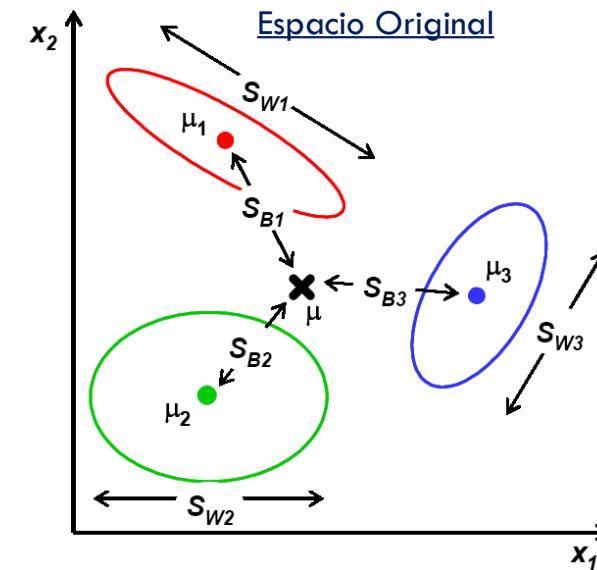
$$S_i = \sum_{x \in \omega_i} (x - \mu_i)(x - \mu_i)^T$$

$$\mu_i = \frac{1}{N_i} \sum_{x \in \omega_i} x$$

- Matriz de dispersión inter-clase

$$S_B = \sum_{i=1}^C \pi_i (\mu_i - \mu)(\mu_i - \mu)^T$$

$$\mu = \frac{1}{N} \sum_{\forall x} x = \frac{1}{N} \sum_{x \in \omega_i} N_i \mu_i$$



## Análisis Discriminante Lineal, C clases (3)

- De manera similar, definimos el vector promedio y las matrices de dispersión de los ejemplos **proyectados** como:

$$\tilde{\mu}_i = E[y \mid y \in \omega_i]$$

$$\tilde{S}_W = \sum_{i=1}^C \pi_i \tilde{S}_i$$

$$\tilde{\mu} = E[y] = \sum_{i=1}^C \pi_i \tilde{\mu}_i$$

$$\tilde{S}_B = \sum_{i=1}^C \pi_i (\tilde{\mu}_i - \tilde{\mu})(\tilde{\mu}_i - \tilde{\mu})^T$$

- De manera análoga a cuando teníamos 2 clases, podemos escribir:

$$\tilde{S}_W = W^T S_W W$$

$$\tilde{S}_B = W^T S_B W$$

## Análisis Discriminante Lineal, C clases (4)

- Estamos buscando una proyección que maximice la dispersión inter-clase y minimice la dispersión intra-clase.
- Ya que ahora la proyección no es un escalar (tiene C-1 dimensiones), usamos el determinante de las matrices de dispersión para obtener escalares:

$$J(W) = \frac{|\tilde{S}_B|}{|\tilde{S}_W|} = \frac{|W^T S_B W|}{|W^T S_W W|}$$

Función criterio de Fisher es una función escalar que es grande cuando:

- $S_B$  es grande
- $S_W$  es pequeña

- De esta forma, buscamos la matriz de proyecciones  $W^*$  que maximiza  $J(W)$ .

$$\tilde{S}_W = W^T S_W W$$

$$\tilde{S}_B = W^T S_B W$$

## Análisis Discriminante Lineal, C clases (5)

- Se puede demostrar analíticamente que la matriz óptima  $\mathbf{W}^*$  es la que en sus columnas contiene los ( $C-1$ ) autovectores de la matriz  $\mathbf{S}_W^{-1} \mathbf{S}_B$  correspondientes a los ( $C-1$ ) autovalores más grandes:

$$\mathbf{W}^* = [w_1^* | w_2^* | \cdots | w_{C-1}^*] = \operatorname{argmax} \left\{ \frac{|W^T S_B W|}{|W^T S_W W|} \right\} \Rightarrow (S_B - \lambda_i S_W) w_i^* = 0$$

$$(S_W^{-1} S_B) W^* = \lambda W^*$$

## Análisis Discriminante Lineal, C clases (5)

- Se puede demostrar analíticamente que la matriz óptima  $\mathbf{W}^*$  es la que en sus columnas contiene los ( $C-1$ ) autovectores de la matriz  $\mathbf{S}_W^{-1} \mathbf{S}_B$  correspondientes a los ( $C-1$ ) autovalores más grandes:

$$\mathbf{W}^* = [w_1^* | w_2^* | \cdots | w_{C-1}^*] = \operatorname{argmax} \left\{ \frac{|W^T S_B W|}{|W^T S_W W|} \right\} \Rightarrow (S_B - \lambda_i S_W) w_i^* = 0$$

$$(S_W^{-1} S_B) W^* = \lambda W^*$$

## Análisis Discriminante Lineal, C clases (6)

- ¿Por qué (C-1)?
  - $S_B$  es la suma de C matrices de orden 1 o menos

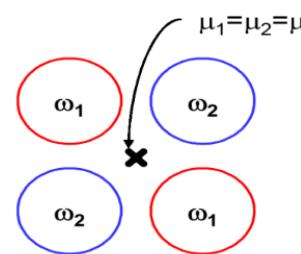
$$S_B = \sum_{i=1}^C \pi_i (\mu_i - \mu)(\mu_i - \mu)^T$$

y los vectores media están restringidos por  $\frac{1}{C} \sum_{i=1}^C \mu_i = \mu$

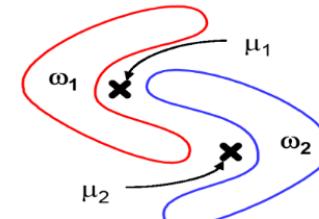
- De esta forma,  $S_B$  es de rango menor o igual que (C-1)
  - Esto significa que hay como mucho (C-1) autovalores  $\lambda_i$  que no son cero
- LDA se puede también derivar del método de Máxima Verosimilitud para el caso en el que las densidades condicionadas a la clase son gaussianas con las mismas matrices de covarianza.

## Análisis Discriminante Lineal, C clases (7)

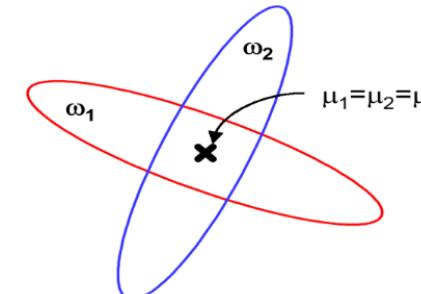
- LDA produce como mucho **C - 1** características proyectadas
  - Si el error de clasificación estimado es demasiado alto, necesitaremos más características, con lo que deberemos utilizar otro método que proporcione esas características adicionales.
- LDA es un **método paramétrico** ya que asume implícitamente distribuciones **unimodales gaussianas**.
  - Si las distribuciones distan de ser gaussianas, las proyecciones LDA no serán capaces de preservar ninguna estructura compleja en los datos, lo que puede ser necesario para la clasificación.



$$S_B = 0$$

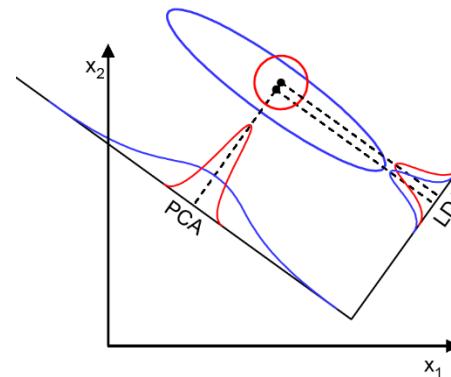


$$S_B = 0$$



## Limitaciones de LDA

- LDA falla cuando la información discriminatoria no está en la media sino en la varianza de los datos:



- Precisa que  $S_W$  sea no singular  $\rightarrow (S_W^{-1} S_B) W^* = \lambda W^*$ 
  - No es aplicable a datos altamente dimensionados donde el número de patrones es menor que el número de características.
  - Como discriminante será lineal

## Variantes de LDA (1)

### □ LDA no paramétrico, “NPLDA” (Fukunaga)

- Este método no necesita la suposición de gaussianidad en las distribuciones. Para ello, calcula la matriz de dispersión inter-clase  $\mathbf{S}_B$  usando información local y la regla de  $K$  vecinos más próximos.
- Como resultado de esto:
  - La matriz  $\mathbf{S}_B$  tiene orden máximo, permitiéndonos extraer más de  $C-1$  características.
  - Las proyecciones son capaces de preservar la estructura de los datos de una manera más precisa.

## Variantes de LDA (2)

- **LDA ortonormal (Okada y Tomita)**
  - Se computan proyecciones que maximizan  $J(w)$  y a la vez son ortonormales entre sí.
  - Se combina lo obtenido con Fisher con el proceso de ortonormalización de Gram-Schmidt
  - Es capaz de encontrar más de  $C-1$  características.

## Variantes de LDA (3)

- **LDA generalizado (Lowe)**
  - Se generaliza lo desarrollado con Fisher incluyendo funciones de costo similares a las usadas al calcular el Riesgo de Bayes.
  - El efecto es una proyección LDA cuya estructura está sesgada por la función de coste.
  - Las clases con costos  $C_{ij}$  mayores se separarán más en el espacio de proyecciones.

## Variantes de LDA (4)

- Perceptrones multicapa (Webb y Lowe)
  - Estos autores demostraron que las capas ocultas de perceptrones multi-capa (MLP) efectúan un análisis discriminante no lineal maximizando  $\text{Tr} [ S_B S_T^{-1} ]$ , donde las matrices de dispersión se miden a la salida de la última capa oculta. [Nota:  $S_T = S_W + S_B$ ].

# Feature Extraction (1)

## (Extracción de características)

# Reducción de la Dimensionalidad

Dos estrategias:

**Selección de características:** se selecciona un subconjunto de los atributos originales.

- **Algoritmos de Filtrado:** la calidad de los atributos se mide usando alguna medida estadística general (Chi Cuadrado, Correlación, Información Mutua, etc.)
- **Algoritmos de Wrapping:** la calidad de un conjunto de atributos se evalúa calculando la calidad promedio de un modelo entrenado con esos atributos
- **Algoritmos embebidos:** la selección es parte del algoritmo de aprendizaje

**Extracción / construcción de características:** la idea es construir nuevas características que condensen / resuman la información relevante de los atributos originales.  
Por ejemplo PCA, LDA

# Reducción de la dimensionalidad con transformaciones de los atributos (1)

- Idea: encontrar una transformación  $y=f(x)$  que conserve la información acerca del problema, minimizando el número de componentes
- En general, la función óptima  $y=f(x)$  será no lineal
- Sin embargo, no hay una forma de generar sistemáticamente transformaciones no lineales:
  - La selección de un subconjunto particular de transformaciones depende del problema
  - Por esta razón, la limitación a transformaciones lineales ha sido ampliamente aceptada,  $y = W^T x$   
**→ y es una proyección lineal de x**

# Reducción de la dimensionalidad con transformaciones de los atributos (2)

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} \xrightarrow{\text{Transformación lineal}} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_M \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & \cdots \\ w_{21} & w_{22} & \cdots \\ \vdots & \vdots & \ddots \\ w_{M1} & w_{M2} & \cdots \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}$$

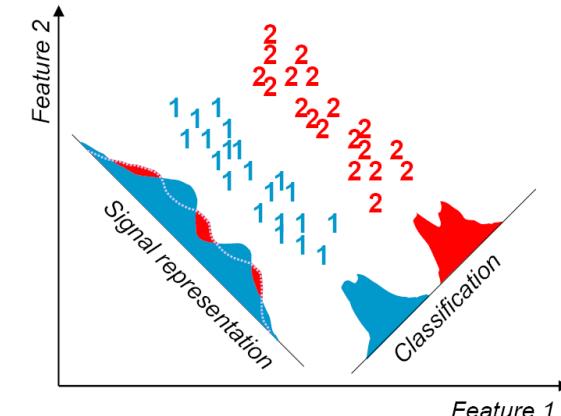
*M-dimensional*  
*M < N*

*N-dimensional*

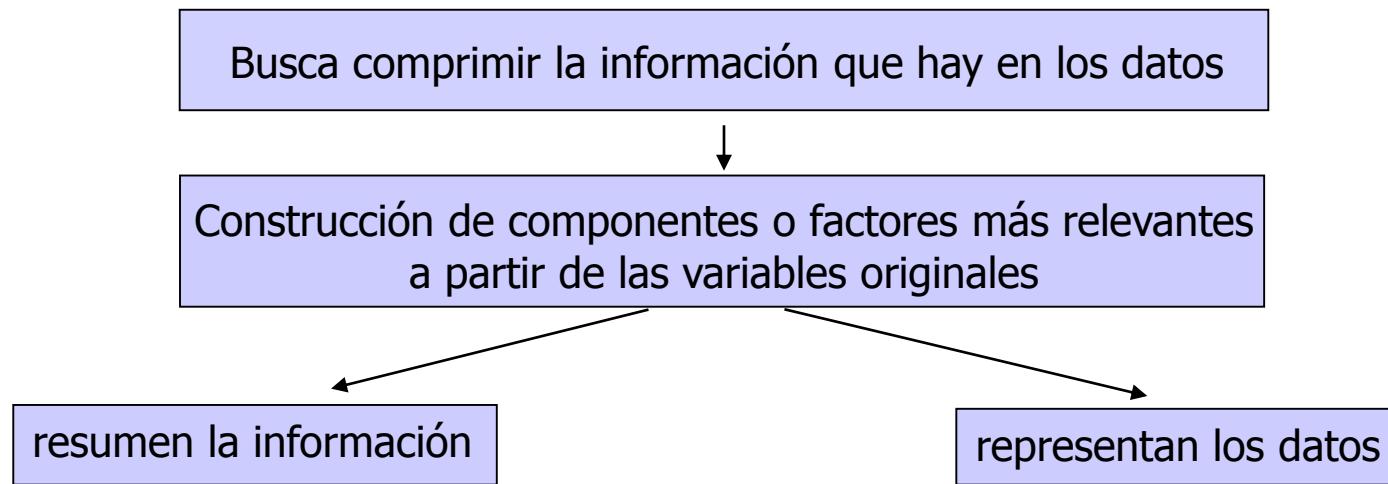
- Por el momento nos centraremos en transformaciones lineales

## Representación de la señal versus clasificación (PCA vs. LDA)

- La selección de la transformación extractora de características,  $y=f(x)$ , está guiada por una función objetivo que buscamos maximizar (o minimizar)
- Dependiendo del criterio usado por la función objetivo, las técnicas de extracción de características se dividen en dos categorías:
  - **Clasificación:** El objetivo de la transformación extractora de características es resaltar en un espacio de menos dimensiones la información discriminante de clases
  - **Representación de la señal:** El objetivo de la transformación extractora de características es representar los vectores de atributos de manera precisa en un espacio de menos dimensiones
- Hay dos técnicas principales en la extracción lineal de características:
  - Análisis Discriminante Lineal (LDA), que utiliza el criterio de clasificación
  - Análisis de Componentes Principales (PCA), que usa el criterio de representación de la señal



## PCA: Definición y objetivo



Simplificar la estructura de los datos transformando las variables originales en otras llamadas componentes principales a través de combinaciones lineales de las mismas:

$$y = W^T x$$

## Definición formal de PCA (1)

- El objetivo de **PCA** es realizar una **reducción de la dimensionalidad** preservando lo máximo posible la información contenida en los datos originales **sin tener en cuenta la clase**.
- PCA busca reducir la dimensionalidad proyectando los datos originales en M ejes  $\{ \mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_M \}$  ( $M < N$ , y cada uno de los vectores  $\mathbf{w}$  tiene longitud 1). Los datos comprimidos  $\mathbf{y}$  tienen M dimensiones, el componente i es  $\mathbf{y}_i = \mathbf{w}_i^T \mathbf{x}$

## Definición formal de PCA (2)

- Supongamos sin pérdida de generalidad que la media de los datos  $\mathbf{x}$  es  $\mathbf{0}$ . En el caso de que nos interese comprimir los datos a una sola dimensión ( $M=1$ ) la proyección de cada punto  $\mathbf{x}$  en el eje  $\mathbf{w}_1$  es:

$$\tilde{\mathbf{x}} = (\mathbf{w}_1^T \cdot \mathbf{x})$$

- ¿Cuál es el eje  $\mathbf{w}_1$  que maximiza la información que se mantiene de los datos originales?  $\Rightarrow$  el que minimiza el promedio del error  $\|\tilde{\mathbf{x}} - \mathbf{x}\|$  calculado a lo largo de la base de datos de entrenamiento
- Se puede demostrar que el  $\mathbf{w}_1$  óptimo es el autovector de  $\Sigma$  (matriz de covarianza de los datos originales) con mayor autovalor, y normalizado para que tenga longitud 1.

## Definición formal de PCA (3)

- ¿Qué es un autovector de  $\Sigma$ ? Es un vector  $\mathbf{u}$  que cumple:

$$\Sigma \cdot \mathbf{u} = \lambda \cdot \mathbf{u}$$

- El número  $\lambda$  es el “autovalor” correspondiente al autovector  $\mathbf{u}$

Propiedades:

1. El autovalor  $\lambda_i$  es exactamente igual a la varianza de la componente  $y_i$ . Su valor no puede ser negativo.
2. Si se asume que la nube de puntos es Gaussiana, la nube tiene una forma “elipsoidal”. Los autovectores de  $\Sigma$  representan los ejes de simetría de esta elipsoide
3. Si  $\mathbf{w}_i$  y  $\mathbf{w}_j$  son dos autovectores que tienen diferentes autovalores, entonces son perpendiculares

## Definición formal de PCA (4)

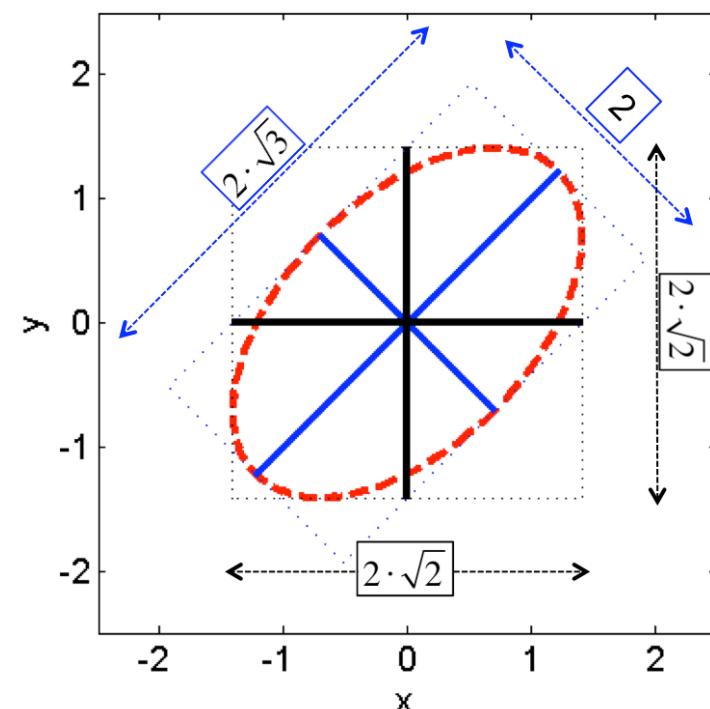
Ejemplo:  $\Sigma = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$

Varianza de atributo 1 = varianza de atributo 2 = 2

- $\begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix} \cdot \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 3 \\ 3 \end{bmatrix} = 3 \cdot \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix}$
- $\begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix} \cdot \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix} = 1 \cdot \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix}$
- $\Rightarrow \mathbf{w}_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix}$  autovector con autovalor 3
- $\Rightarrow \mathbf{w}_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix}$  autovector con autovalor 1

Varianza en eje principal 1 = 3

Varianza en eje principal 2 = 1



## Definición formal de PCA (5)

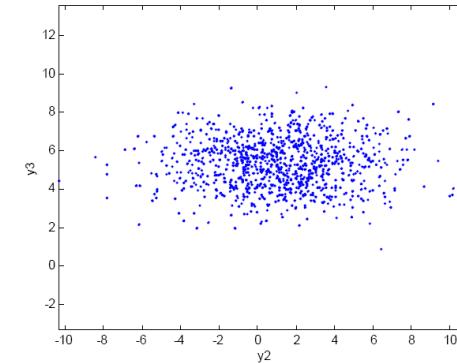
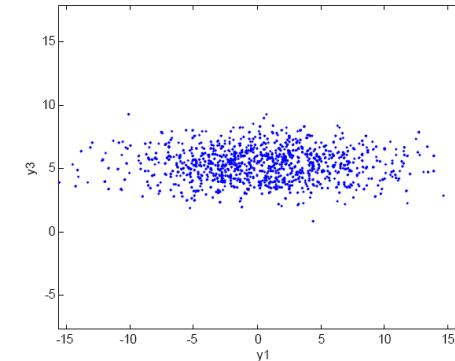
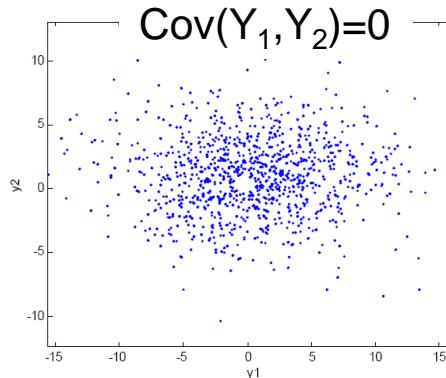
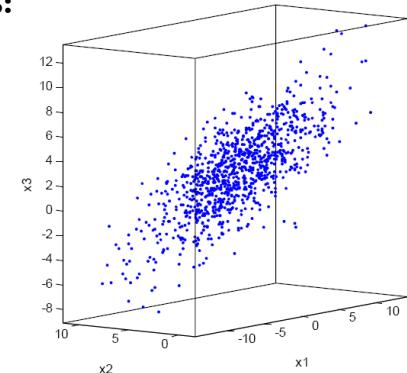
- Supongamos que queremos comprimir los datos a M dimensiones. ¿Cuáles son los ejes  $w_i$  que maximizan en conjunto la información que se mantiene de los datos originales?  $\Rightarrow$  los que minimizan el promedio del error  $\|\tilde{x} - x\|$  calculado a lo largo de la base de datos de entrenamiento
- Se puede demostrar que el conjunto  $\{ w_1, w_2, \dots, w_M \}$  óptimo es el formado por los M autovectores de  $\Sigma$  (matriz de covarianza de los datos originales) con autovalores más grandes, y normalizados para que tengan longitud 1

## PCA: Ejemplo 1

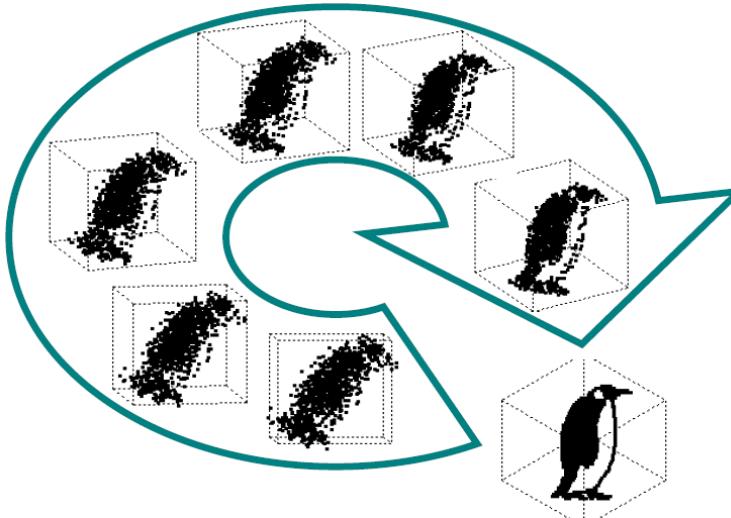
- Ejemplo: distribución gaussiana en tres dimensiones con los siguientes parámetros:

$$\mu = [0 \ 5 \ 2]^T \quad \Sigma = \begin{bmatrix} 25 & -1 & 7 \\ -1 & 4 & -4 \\ 7 & -4 & 10 \end{bmatrix}$$

- Ahora mostramos los tres pares de proyecciones en los componentes principales
  - La primera proyección tiene la mayor varianza, seguida por la segunda
  - Las proyecciones PCA “decorrelacionan” los ejes.  $\text{Cov}(Y_i, Y_k) = 0$

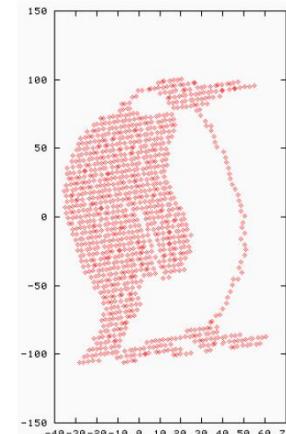


## PCA: Ejemplo 2



- PCA nos puede ayudar en encontrar la estructura implícita en nuestros datos. Selecciona una rotación tal que casi toda la variabilidad de los datos es representada en las primeras componentes principales.
  - En nuestro ejemplo no parece de mucha ayuda.
  - Sin embargo, cuando tenemos docenas de dimensiones, PCA es muy potente

- Ahora tenemos una nube de datos en 3 dimensiones
- Inicialmente, excepto por un alargamiento en la nube de puntos, no hay estructura aparente
- Elegir una rotación apropiada nos permite descubrir la estructura que hay por debajo (podemos pensar en esta rotación como el “caminar” en 3 dimensiones, buscando el mejor punto de vista).



## PCA: Ejemplo 3

- Tenemos los siguientes datos en dos dimensiones:

$$X = (x_1, x_2) = \{(1,2), (3,3), (3,5), (5,4), (5,6), (6,5), (8,7), (9,8)\}$$

- La estimación del promedio y de la matriz de covarianza es:

$$\mu = \begin{bmatrix} 5 \\ 5 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 7.14 & 4.86 \\ 4.86 & 4.00 \end{bmatrix}$$

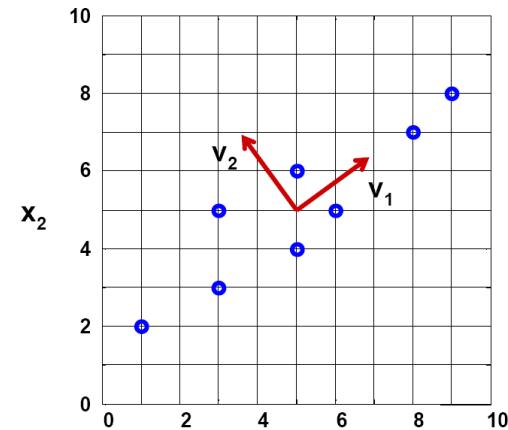
Los autovectores son:

$$\Sigma \omega = \lambda \omega \Rightarrow |\Sigma - \lambda I| = 0$$

$$\mathbf{w}_1 = \begin{bmatrix} 0.81 \\ 0.59 \end{bmatrix} \text{ con autovalor } 10.68$$

$$\mathbf{w}_2 = \begin{bmatrix} -0.59 \\ 0.81 \end{bmatrix} \text{ con autovalor } 0.47$$

- La primera componente explica el  $10.68/(10.68+0.47) \cdot 100 = 95.8\%$  de los datos originales



## Comentarios sobre PCA (1)

- Ya que PCA elige los autovectores de la matriz de covarianza  $\Sigma_x$ , es capaz de encontrar los ejes independientes de los datos cuando éstos están distribuidos gaussianamente
  - Para distribuciones no Gaussianas (multimodales, por ejemplo), PCA simplemente *decorrelaciona* los ejes (las nuevas variables tienen correlación 0 entre ellas).
- La principal limitación de PCA es que no tiene en cuenta la separabilidad de las clases ya que no tiene en cuenta las clases de los vectores  $x$  → **Método no supervisado**
  - PCA simplemente realiza una rotación de coordenadas que alinea los ejes transformados con las direcciones de máxima varianza.
  - **No hay garantía alguna de que los ejes de máxima varianza contengan una buena información para la clasificación**

## Comentarios sobre PCA (2)

### □ Comentarios históricos:

- PCA es la técnica más antigua de análisis multivariable
- Se conoce también como “transformada de Karhunen-Loève” en otros campos como la Teoría de la Comunicación y la Física.

## Uso práctico de PCA (1)

Matemáticamente: Partiendo de N variables iniciales

Construcción de una matriz a partir de los datos de partida

→ Matriz de Covarianza si los datos no están estandarizados  
Matriz de Correlaciones si los datos están estandarizados

Cálculo de los autovectores y autovalores de la matriz

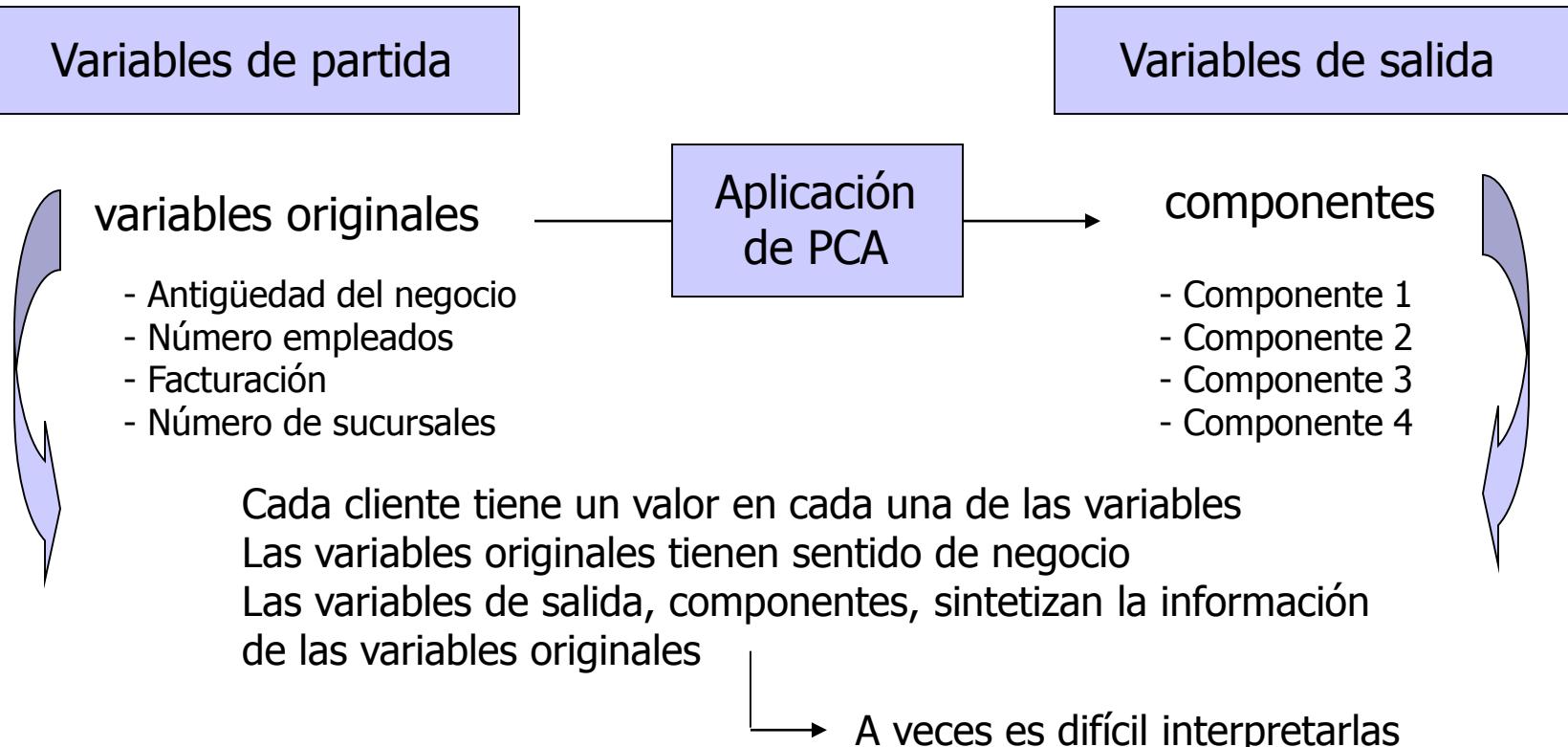
→  $\lambda_i$  y  $(w_{i1}, w_{i2}, \dots, w_{iN})^T$  para  $i = 1, \dots, N$

Cálculo de los nuevos atributos ("componentes")

→ Las componentes sintetizan la información de las variables

$$\begin{cases} y_1 = w_{11} \cdot x_1 + \dots + w_{1N} \cdot x_N \\ \vdots \\ y_M = w_{M1} \cdot x_1 + \dots + w_{MN} \cdot x_N \end{cases}$$

## Uso práctico de PCA (2)



## Uso práctico de PCA (3)

### Tipo de las variables

Por las propiedades y características del modelo **sólo está permitido el uso de variables numéricas**

Si se dispone de variables categóricas que describan directamente el problema:



Para que intervengan en el análisis de componentes principales, es necesario convertirlas en numéricas



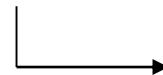
Transformar la variable creando variables dicotómicas

## Uso práctico de PCA (4)

Escala de medida → PCA no es invariante a la escala!!!

Las escalas en la que estén medidas las variables influyen en esta técnica

Si no se quiere dar importancia a una variable por la escala en la que viene medida



Normalización (o “estandarización”) de los datos

Correlaciones

PCA es una técnica que tiene sentido aplicarse en el caso de existir **correlaciones altas** entre las variables (indicio de que existe información redundante).

⇒ Como consecuencia, pocos factores explicarán gran parte de la variabilidad total.

## Uso práctico de PCA (5)

- Sea  $\mathbf{X}$  la matriz de datos de entrenamiento (cada columna un patrón, cada fila un atributo), y consideremos que los atributos han sido previamente estandarizados.

- En este caso la matriz de covarianza equivale a la de correlación, y es igual a

$$\boldsymbol{\Sigma} = \frac{1}{N_{\text{tr}} - 1} \cdot \mathbf{X} \cdot \mathbf{X}^T$$

- En algunas situaciones el número de dimensiones  $N$  de los datos de partida es mucho mayor que el número de patrones de entrenamiento  $N_{\text{tr}}$ . Por ejemplo:

- En bases de datos de genomas
- En bases de datos de imágenes
- En bases de datos de audios

- Esto hace que el tamaño de la matriz  $\boldsymbol{\Sigma}$  sea muy grande, y por tanto el cálculo de los autovectores sea muy costoso. Por otra parte, muchos de los autovalores serán nulos (una condición necesaria para que no salga ningún autovalor nulo es que  $N_{\text{tr}} > N$ )

## Uso práctico de PCA (6)

### Características de los componentes o factores

La correlación entre factores diferentes es 0,  $\text{Cov}(y_i, y_j)=0$

Los primeros tienen más relevancia (más información) que los últimos

Los factores sintetizan la información de las variables originales

### Si los atributos antiguos estaban estandarizados:

Los nuevos atributos (“componentes” o “factores”) tienen media 0

La varianza de cada factor es exactamente igual al autovalor asociado

El Análisis de Componentes Principales estudia las relaciones que las variables tienen entre sí, descubriendo grupos de variables muy correlacionadas entre sí

## Uso práctico de PCA (7)

### Elección de componentes

Si nos quedamos con N componentes no reducimos dimensionalidad

#### Criterios para la ayuda a la elección de factores o componentes

- 1.- Seleccionar los componentes necesarios para sumar al menos el 80% del valor total de la varianza
- 2.- En el caso de utilizar la matriz de correlación, seleccionar todos los factores con autovalor  $\geq 1$ .  
En caso de utilizar la matriz de covarianza, seleccionar aquellas componentes cuyos autovalores son mayor o igual al promedio de todos los autovalores  $\frac{1}{N} \sum_{i=1}^N \lambda_i$
- 3.- Representar en una gráfica los valores propios y seleccionar el número de componentes en función de un cambio brusco

## Ejemplo: Calificaciones de 15 alumnos en distintas asignaturas

Se dispone de las calificaciones de 15 alumnos en 8 materias: lengua, matemáticas, física, inglés, filosofía, historia, química y gimnasia.

Componente	Varianza explicada (autovalor / suma de los 8 autovalores)	Varianza acumulada
1	0.464	0.464
2	0.358	0.821 (0.464+0.358)
3	0.119	0.941 (0.464+0.358+0.119)
4	0.027	0.968 (0.464+ ... + 0.027)
...	...	...
8	0.002	1

## Ejemplo: Calificaciones de 15 alumnos en distintas asignaturas

Construcción de 8 componentes como mucho porque hay 8 variables independientes

Con 8 componentes se consigue explicar toda la información

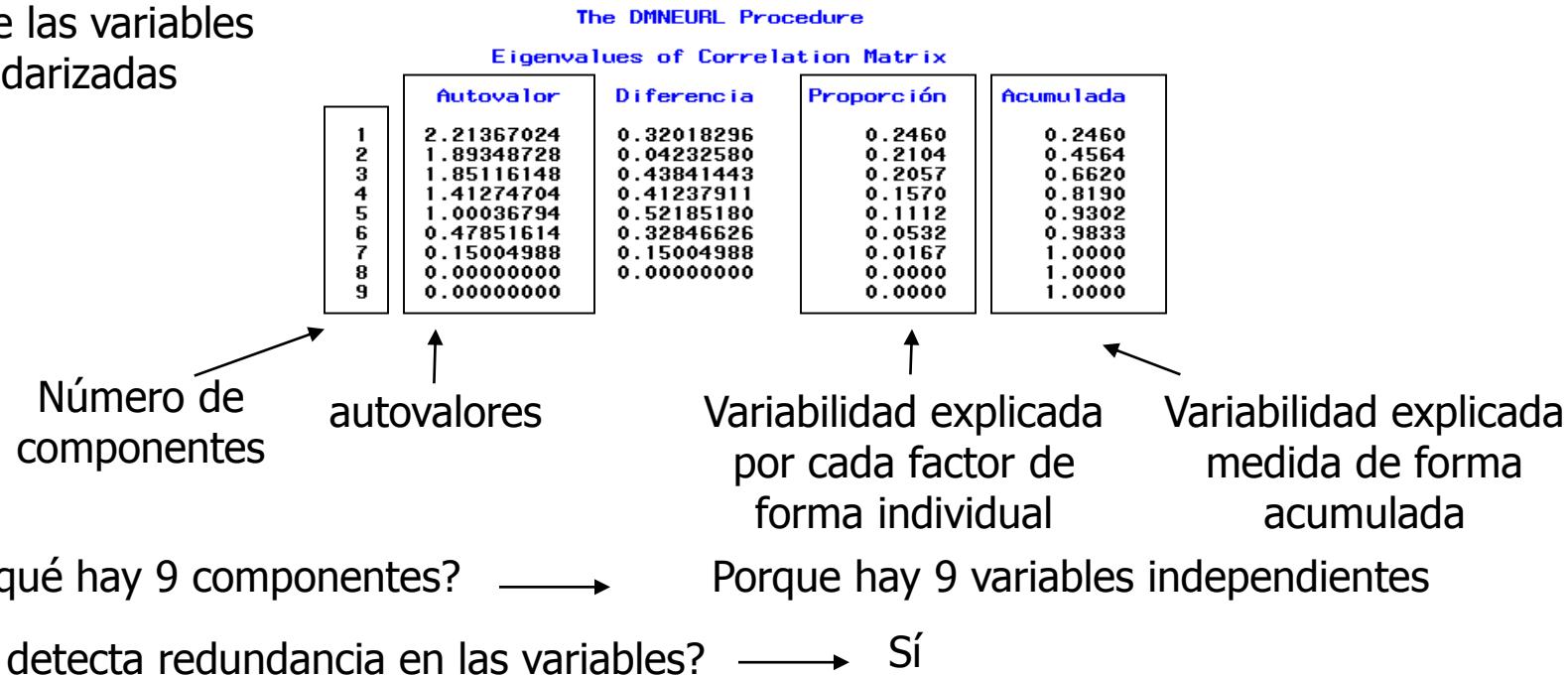
Cada componente aporta menos información que el anterior

El primer componente explica casi el 50% de la información

Con 3 componentes se consigue explicar el 94% de la información

## Ejemplo: eliminación de variables redundantes en problema de impago en PYMES

PCA sobre las variables estandarizadas

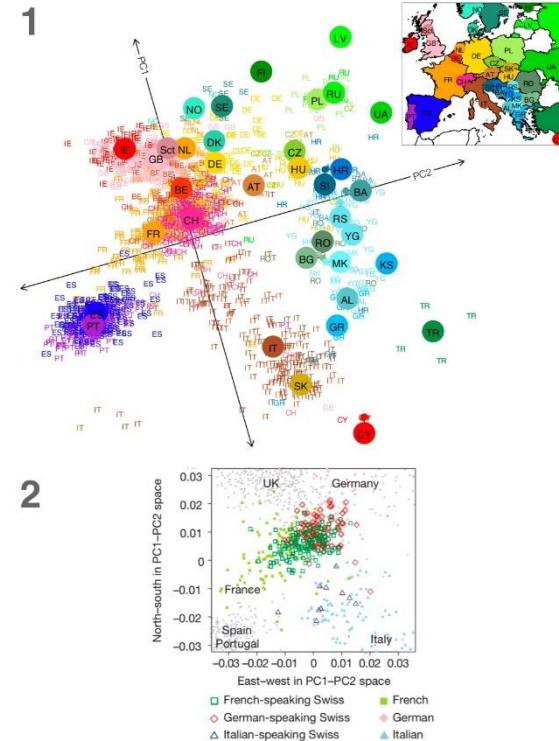


## Ejemplo: estudio de datos genéticos

En el artículo “Genes mirror geography within Europe”, Novembre et al. 2008, Nature vol. 456, pp. 98-10, Novembre y colaboradores toman los datos genéticos de 1387 personas europeas y realizan PCA, mostrando la proyección en los dos primeros componentes principales.

Cada punto: una persona. Cada círculo: medianas en cada país.

Resultado: hay una correspondencia muy fuerte entre el mapa genético y el mapa geográfico.



courtesy: John Novembre, UCLA

## Aplicaciones de PCA (1)

1. Reducir el número de variables del problema mediante extracción de características:  $y = W^T x$

Nos olvidamos de las antiguas variables y trabajamos en adelante con las variables sintéticas obtenidas de los componentes principales.

Por ejemplo, entrenamos un árbol de decisión con ellas.

2. Reducir el número de variables del problema mediante selección de variables:

Usamos PCA para que nos “diga” qué variables antiguas podemos “tachar” (no aportan información relevante o son redundantes).

Aquellas características o variables cuyos coeficientes sean elevados para los primeros componentes principales, serán las que aporten la mayor capacidad discriminatoria

A partir de ese momento, trabajamos sólo con las variables antiguas relevantes.

## Aplicaciones de PCA (2)

### 3. Detección de “outliers”:

Usamos PCA para detectar qué ejemplos son atípicos. Diagramas de dispersión en los factores.

### 4. Descubrimiento de “clusters”:

Usamos PCA para detectar grupos diferentes de ejemplos.

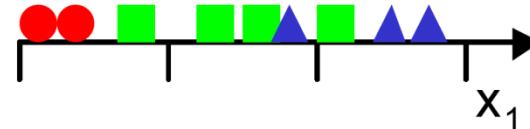
# Feature Selection

# The curse of dimensionality (I)

Consider for example a classification problem with **3 classes** and a training dataset with **9 patterns**. A simple strategy could be:

- Divide the features space in uniform cells
- Calculate the percent of examples of each class in each cell
- Given a new pattern to classify, check in which cell is. The class predicted for the new pattern is the most frequent class in the cell

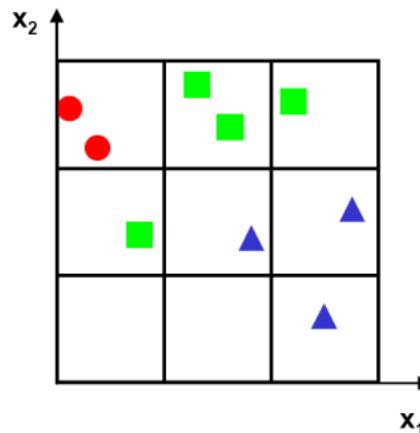
Example: one attribute (feature), we decide to use only 4 cells.



What happens if we decide to take into account an additional feature?

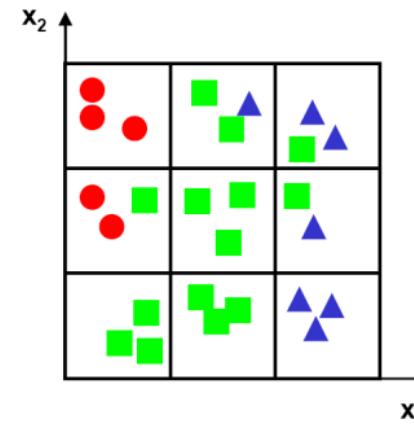
# The curse of dimensionality (II)

We decide to include an additional feature. If the granularity of the new feature is also 3, we have  $3^2=9$  cells:



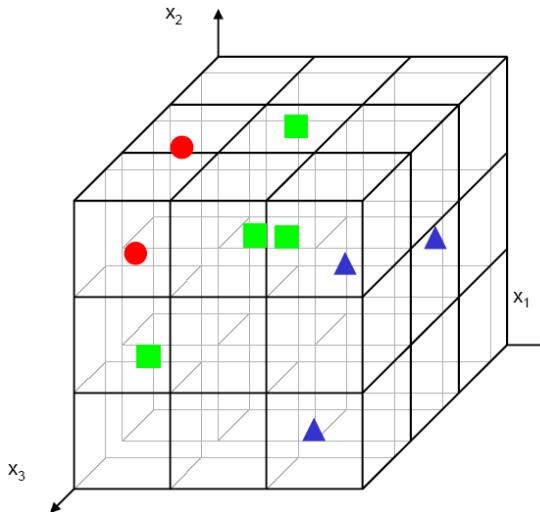
Low number of examples per cell, and even **cells with no training patterns**

If we want to measure robustly the class probabilities in each cell, we need much more patterns in our database:



# The curse of dimensionality (III)

With 3 features the problem is even worst,  
with many cells without any example:



We need much more examples in order to measure the classes frequencies in each cell

**How many more examples do we need?**

# The curse of dimensionality (IV)

Let us call:

- **d**: the number of attributes
- **Nc**: the number of patterns we want per cell
- **g**: granularity (number of segments per attribute).

We will now consider a simple situation where g is the same for all attributes.

How many examples **N** do we need (at least) if we want **Nc** patterns per cell?

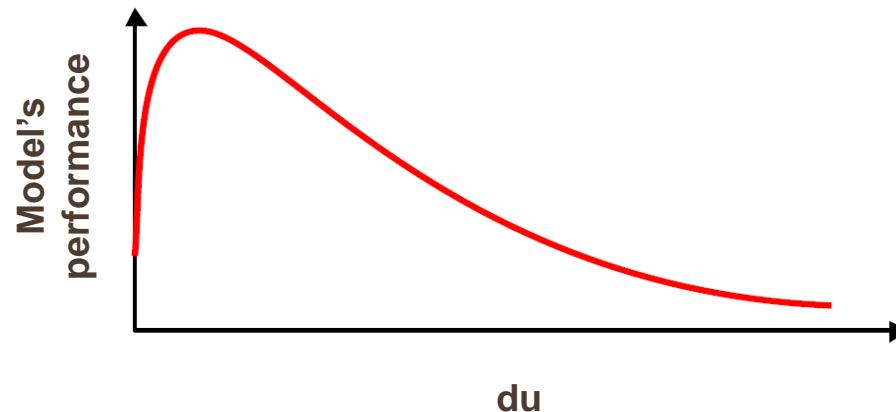
$$N = Nc \times (\# \text{ cells}) = Nc \cdot (g^d)$$

How large is it?

If  $Nc = 10$ ,  $g = 5$  and  $d$  (# attributes) = 20, then  $N = 1,000,000,000,000,000$  !!!!

# The curse of dimensionality (V)

In general, in real problems the total number of patterns is fixed or constrained. If  $du$  is the number of attributes used by our model, then:



Which means that a minimum number of attributes is needed for a good model, **but** too many attributes is bad for model's performance: overfitting

# Idea:

Use in your model only a set of attributes, the ones with more information about the problem: **Dimensionality reduction**

This will:

- Reduce computation time of the model (learning step).
- Avoid overfitting.
- Filter the attributes that do not carry substantial information about the problem (noise attributes, etc.). This will increase the model's performance.

# Dimensionality Reduction

Two strategies:

**Feature selection:** the idea is to select a subset of the original attributes.

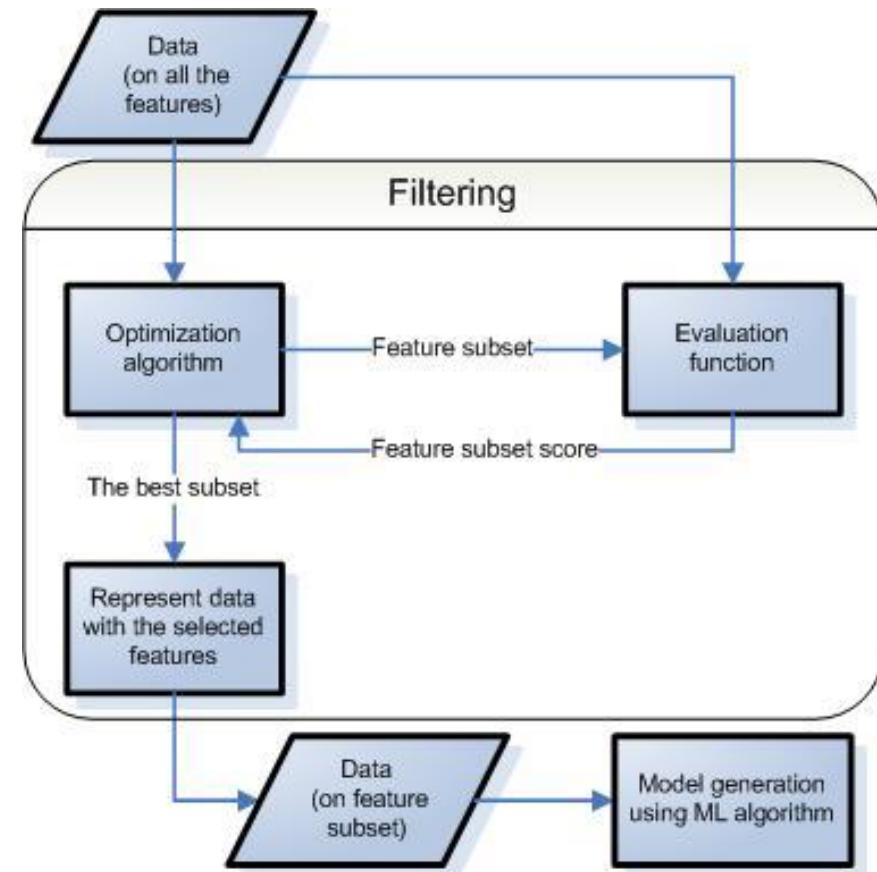
- **Filtering algorithms:** the quality of the attributes is measured using some general statistical measure. For example Chi Square, Correlation, Mutual Information
- **Wrapping algorithms:** the quality of a set of attributes is measured as the average performance of a model trained using those attributes
- **Embedded algorithms:** the selection is included as part of the learning algorithm

**Feature extraction / construction:** the idea is to construct new features that condense / summarize the relevant information of the original ones. For example PCA, LDA

**Feature selection:** the idea is to select a subset of the original attributes

**Filtering algorithms:** the quality of the attributes is measured using some general statistical measure. For example Significance Tests, Correlation, Mutual Information

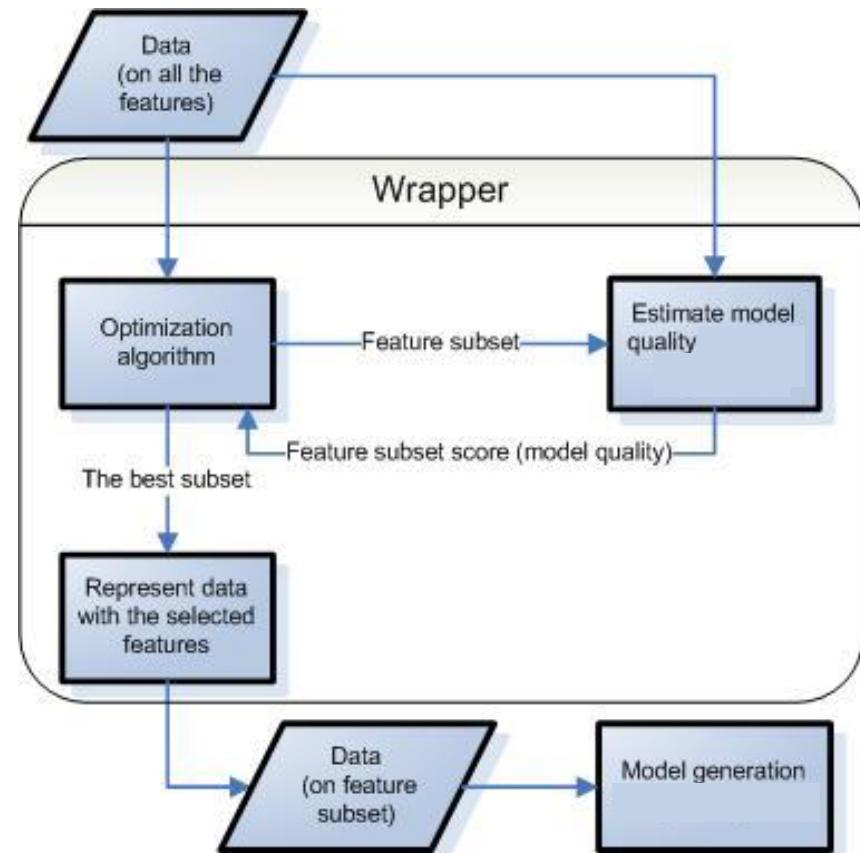
The evaluation does not depend on a model



**Feature selection:** the idea is to select a subset of the original attributes

**Wrapping algorithms:** the quality of a set of attributes is measured as the average performance of a model trained using those attributes

The evaluation **does depend** on a model



# Simple Filtering: Pearson's correlation coefficient

Pearson's correlation coefficient is a measure of the strength of **linear** association between two variables:

$$r_{x,y} = \frac{\sum_i [(x_i - \bar{x}) \cdot (y_i - \bar{y})]}{\sqrt{\sum_i (x_i - \bar{x})^2} \cdot \sqrt{\sum_i (y_i - \bar{y})^2}} \quad \text{where} \quad \bar{x} = \frac{\sum_i x_i}{N} \quad \bar{y} = \frac{\sum_i y_i}{N}$$

**r is always in the [-1,1] interval.**

+1: perfectly correlated. -1: perfectly anticorrelated. 0: linearly independent

**Idea:** compute Pearson's correlation coefficient between each attribute and the target variable, and select the attributes with greatest correlation coefficients:

**In other words:** rank the attributes by their correlation coefficients and select the top ones

# Simple Filtering: Pearson's correlation

**Note 1:**  $r$  is independent on the scale: good!

**Note 2:** The square of  $r$ ,  $r^2$ , should be used ("coefficient of determination")

**Note 3:** We are implicitly treating the attributes as independent

**Note 3:** attributes and target must be continuous

**Note 4:** assumes linear relationships between each attribute and the target variable.

# Simple Filtering: Student's t-test

**Student's t-test** tell us if two sets of observations of a single variable follow the same distribution or not. Assumption: the observations follow a Gaussian distribution

**“Null Hypothesis”**: the two sets of observations follow the same distribution

**“Alternative Hypothesis”**: the two sets of observations follow distributions with different means

**How can we use this in simple filtering? If we want to check if we should use attribute x or not:**

Set 1 of observations: set of observed values of x (with repetitions) when the class = 1

Set 2 of observations: set of observed values of x (with repetitions) when the class = 2

**If Student's t-test concludes the Alternative Hypothesis, use attribute x**

# How does Student's t-test work?

1. **Choose a significance level  $\alpha$**  (probability that the observation differences are due to chance). For example, 0.05
2. **Compute the “degrees of freedom”:**

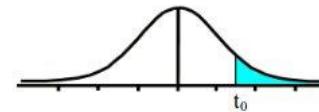
$$v = N_1 + N_2 - 2$$

$N_1$ : number of cases where class = 1  
 $N_2$ : number of cases where class = 2

3. **Go to Student's Table and check what is  $t_{critical}$  for those  $\alpha$  and  $v$**

# How does Student's t-test work?

Tabla t-Student



Grados de libertad	0.25	0.1	0.05	0.025	0.01	0.005
1	1.0000	3.0777	6.3137	12.7062	31.8210	63.6559
2	0.8165	1.8856	2.9200	4.3027	6.9645	9.9250
3	0.7649	1.6377	2.3534	3.1824	4.5407	5.8408
4	0.7407	1.5332	2.1318	2.7765	3.7469	4.6041
5	0.7267	1.4759	2.0150	2.5706	3.3649	4.0321
6	0.7176	1.4398	1.9432	2.4469	3.1427	3.7074
7	0.7111	1.4149	1.8946	2.3646	2.9979	3.4995
8	0.7064	1.3968	1.8595	2.3060	2.8965	3.3554
9	0.7027	1.3830	1.8331	2.2622	2.8214	3.2498
10	0.6998	1.3722	1.8125	2.2281	2.7638	3.1693
11	0.6974	1.3634	1.7959	2.2010	2.7181	3.1058
12	0.6955	1.3562	1.7823	2.1788	2.6810	3.0545
13	0.6938	1.3502	1.7709	2.1604	2.6503	3.0123
14	0.6924	1.3450	1.7613	2.1448	2.6245	2.9768
15	0.6912	1.3406	1.7531	2.1315	2.6025	2.9467
16	0.6901	1.3368	1.7459	2.1199	2.5835	2.9208
17	0.6892	1.3334	1.7396	2.1098	2.5669	2.8982
18	0.6884	1.3304	1.7341	2.1009	2.5524	2.8784
19	0.6876	1.3277	1.7291	2.0930	2.5395	2.8609
20	0.6870	1.3253	1.7247	2.0860	2.5280	2.8453
21	0.6864	1.3232	1.7207	2.0796	2.5176	2.8314
22	0.6858	1.3212	1.7171	2.0739	2.5083	2.8188
23	0.6853	1.3195	1.7139	2.0687	2.4999	2.8073
24	0.6848	1.3178	1.7109	2.0639	2.4922	2.7970
25	0.6844	1.3163	1.7081	2.0595	2.4851	2.7874

3. Go to Student's Table, which tell us what is the value for  $t_{critical}$  for a given  $\alpha$  and a given  $v$

**Example:** if  $\alpha = 0.05$  and  $v=16$ ,  $t_{critical}$  is 1.7459

# How does Student's t-test work?

## 4. Compute the “t-Student”:

$$t = \frac{\mu_1 - \mu_2}{s \sqrt{\frac{1}{N_1} + \frac{1}{N_2}}}$$

$$s = \sqrt{\frac{(N_1 - 1)s_1^2 + (N_2 - 1)s_2^2}{(N_1 - 1) + (N_2 - 1)}}$$

$\mu_1$  = Mean of x when class = 1

$\mu_2$  = Mean of x when class = 2

$s_1$ : standard deviation of x when class = 1

$s_2$ : standard deviation of x when class = 2

$$\left\{ \begin{array}{l} s_1 = \sqrt{\frac{\sum_{i=1}^N (x_i^{\Omega_1} - \mu_1)^2}{N_1 - 1}} \\ s_2 = \sqrt{\frac{\sum_{i=1}^N (x_i^{\Omega_2} - \mu_2)^2}{N_2 - 1}} \end{array} \right.$$

## 5. If $\text{abs}(t) > t_{\text{critical}}$ , reject the Null Hypothesis:

In other words, we conclude that the two sets of observations are significantly different -> the attribute gives significant information about the class

# How does Student's t-test work?

## Example

Two attributes (age, height), two classes

class	1	1	1	1	2	2	2	2	2
age	32	28	36	34	26	30	24	26	22
height	180	170	160	175	182	168	170	180	174

1. We choose  $\alpha=0.05$

$$2. v = N_1 (4) + N_2 (6) - 2 = 8$$

3. t-Student's table gives a value of  $t_{critical} = 1.8595$  for those  $\alpha$  and  $v$

# How does Student's t-test work?

class	1	1	1	1	2	2	2	2	2	2
age	32	28	36	34	26	30	24	26	22	25
height	180	170	160	175	182	168	170	180	174	172

## 4. Attribute “age”:

$$\begin{array}{ll}\mu_1 = 32.5, & \mu_2 = 25.5 \\ s_1 = 3.4157, & s_2 = 2.6646\end{array}$$

With this we can calculate  $t$ .  $t = 3.6530$ , which is greater than  $t_{critical}$

-> “age” has significant information about the class

# How does Student's t-test work?

class	1	1	1	1	2	2	2	2	2	2
age	32	28	36	34	26	30	24	26	22	25
height	180	170	160	175	182	168	170	180	174	172

## 4. Attribute “height”:

$$\begin{array}{ll}\mu_1 = 171.25, & \mu_2 = 174.33 \\ s_1 = 8.5391, & s_2 = 5.5737\end{array}$$

With this we can calculate t. t = 0.6985, which is smaller than t\_critical  
-> “height” **does not have significant information about the class**

# Simple Filtering: Student's t-test

## Summary

Assumptions: Attributes are continuous and follow Gaussian distributions (no realistic in many situations). Target: class with 2 possible values (“2 classes”)

Attributes can be ranked by “t”. Greater t: more informative about the class.

The version we showed assumes  $s_1$  and  $s_2$  are identical. There is a improvement, “Welch-Satterthwaite approximation” that does not assume this.

# Simple Filtering: Mutual Information

The mutual information MI between two variables with a finite number of values (discretized / segmented / categorical / nominal / boolean variables) is defined as:

$$\text{MI}[x, y] \equiv \sum_{i=1}^{Nx} \sum_{j=1}^{Ny} p(x_i, y_j) \cdot \log \frac{p(x_i, y_j)}{p(x_i) \cdot p(y_j)}$$

Terms in the summation where some  $p$  is null are ignored.

# Simple Filtering: Mutual Information

$$\text{MI}[x, y] \equiv \sum_{i=1}^{N_x} \sum_{j=1}^{N_y} p(x_i, y_j) \cdot \log \frac{p(x_i, y_j)}{p(x_i) \cdot p(y_j)}$$

## Properties:

- MI  $\geq 0$ , being 0 only if x and y are **statistically independent**
- Given variable x, y maximizes MI if  $y=f(x)$  with x invertible  
-> MI is a measure of the **level of independence** between x and y
- MI units depend on the logarithm base:  
Base=2: "bits". Base=e ("natural base"): "nats"  
Different bases give **identical** attribute rankings

# Simple Filtering: Mutual Information

## Practical use:

- In a classification problem, compute the MI between each attribute and the class (now there can be any number of classes). Rank the attributes according to their MI with the class and select the top ones.
- Note: continuous attributes must be discretized / segmented

# Simple Filtering: Mutual Information

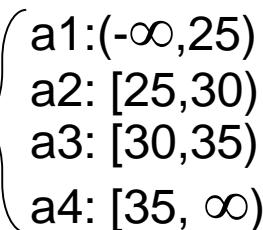
Example:

Two attributes (age, height), two classes

class	1	1	1	1	2	2	2	2	2
age	32	28	36	34	26	30	24	26	22
height	180	170	160	175	182	168	170	180	174

First step: recode the continuous attributes / attributes with many values

class	1	1	1	1	2	2	2	2	2
age'	a3	a2	a4	a3	a2	a3	a1	a2	a1
height'	h3	h2	h1	h2	h3	h1	h2	h3	h2

age'   
a1: (-∞, 25)  
a2: [25, 30)  
a3: [30, 35)  
a4: [35, ∞)

# Simple Filtering: Mutual Information

**Second step**

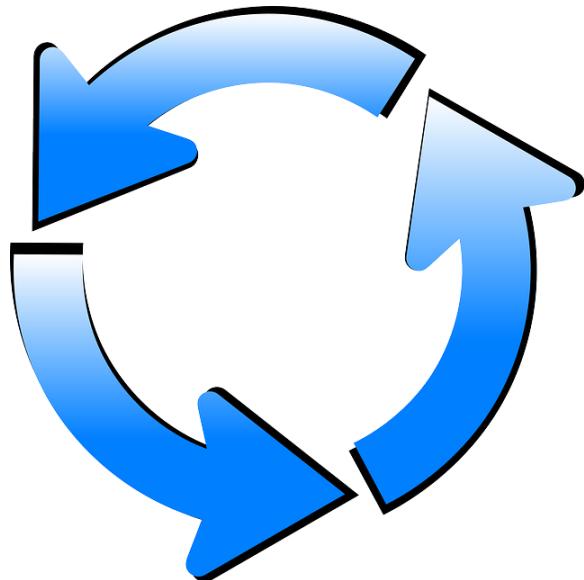
class	1	1	1	1	2	2	2	2	2	2
age'	a3	a2	a4	a3	a2	a3	a1	a2	a1	a1
height'	h3	h2	h1	h2	h3	h1	h2	h3	h2	h2

$$\begin{aligned}
 \text{MI[age', class]} &= p(a1,1) \cdot \log \frac{p(a1,1)}{p(a1) \cdot p(1)} + p(a1,2) \cdot \log \frac{p(a1,2)}{p(a1) \cdot p(2)} + \\
 &\quad p(a2,1) \cdot \log \frac{p(a2,1)}{p(a2) \cdot p(1)} + p(a2,2) \cdot \log \frac{p(a2,2)}{p(a2) \cdot p(2)} + \\
 &\quad p(a3,1) \cdot \log \frac{p(a3,1)}{p(a3) \cdot p(1)} + p(a3,2) \cdot \log \frac{p(a3,2)}{p(a3) \cdot p(2)} + \\
 &\quad p(a4,1) \cdot \log \frac{p(a4,1)}{p(a4) \cdot p(1)} + p(a4,2) \cdot \log \frac{p(a4,2)}{p(a4) \cdot p(2)} = \\
 &= 0 + \frac{3}{10} \cdot \log \frac{3/10}{3/10 \cdot 6/10} + \\
 &\quad \frac{1}{10} \cdot \log \frac{1/10}{3/10 \cdot 4/10} + \frac{2}{10} \cdot \log \frac{2/10}{3/10 \cdot 6/10} + \\
 &\quad \frac{2}{10} \cdot \log \frac{2/10}{3/10 \cdot 4/10} + \frac{1}{10} \cdot \log \frac{1/10}{3/10 \cdot 6/10} + \\
 &\quad \frac{1}{10} \cdot \log \frac{1/10}{1/10 \cdot 4/10} + 0 = 0.3827 \text{ nats}
 \end{aligned}$$

# Simple Filtering: Chi Square

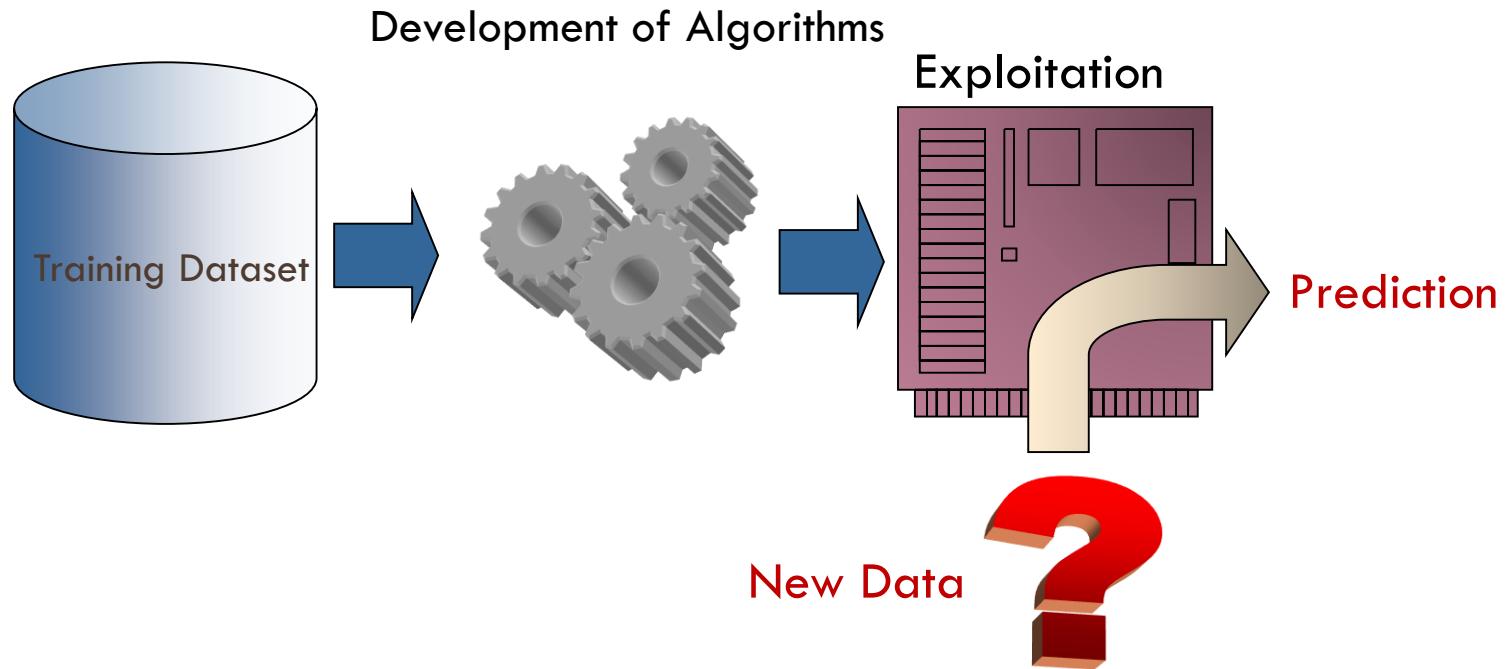
The Chi Square statistic measures the degree of dependence between two **categorical** variables

If one of the two variables is continuous, it should be discretized / categorized.



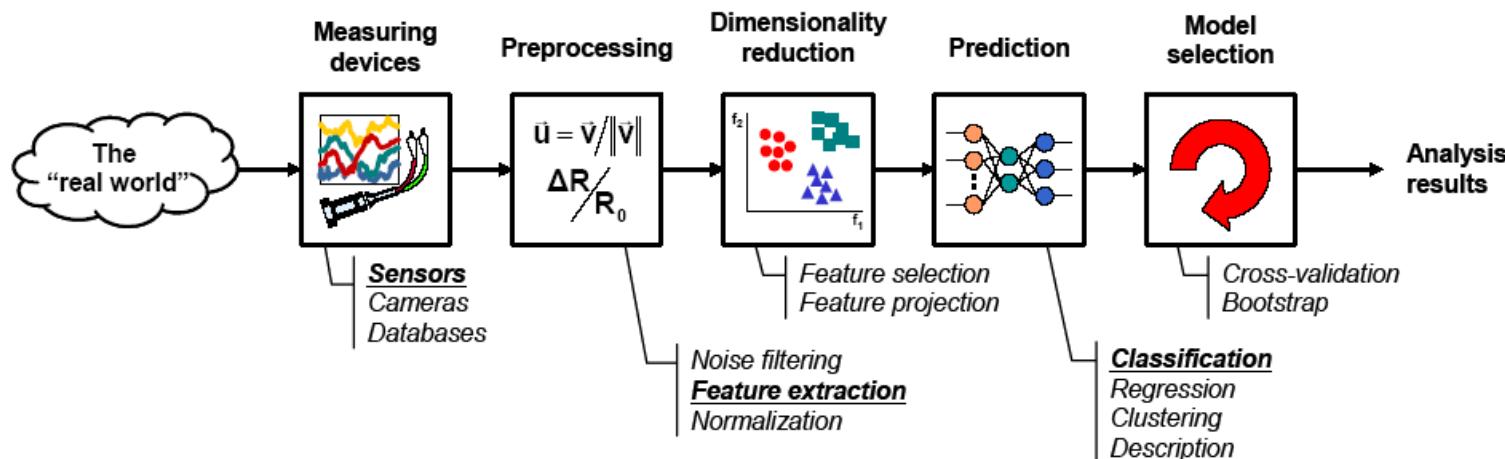
# Introducción de Conceptos

# Design Cycle

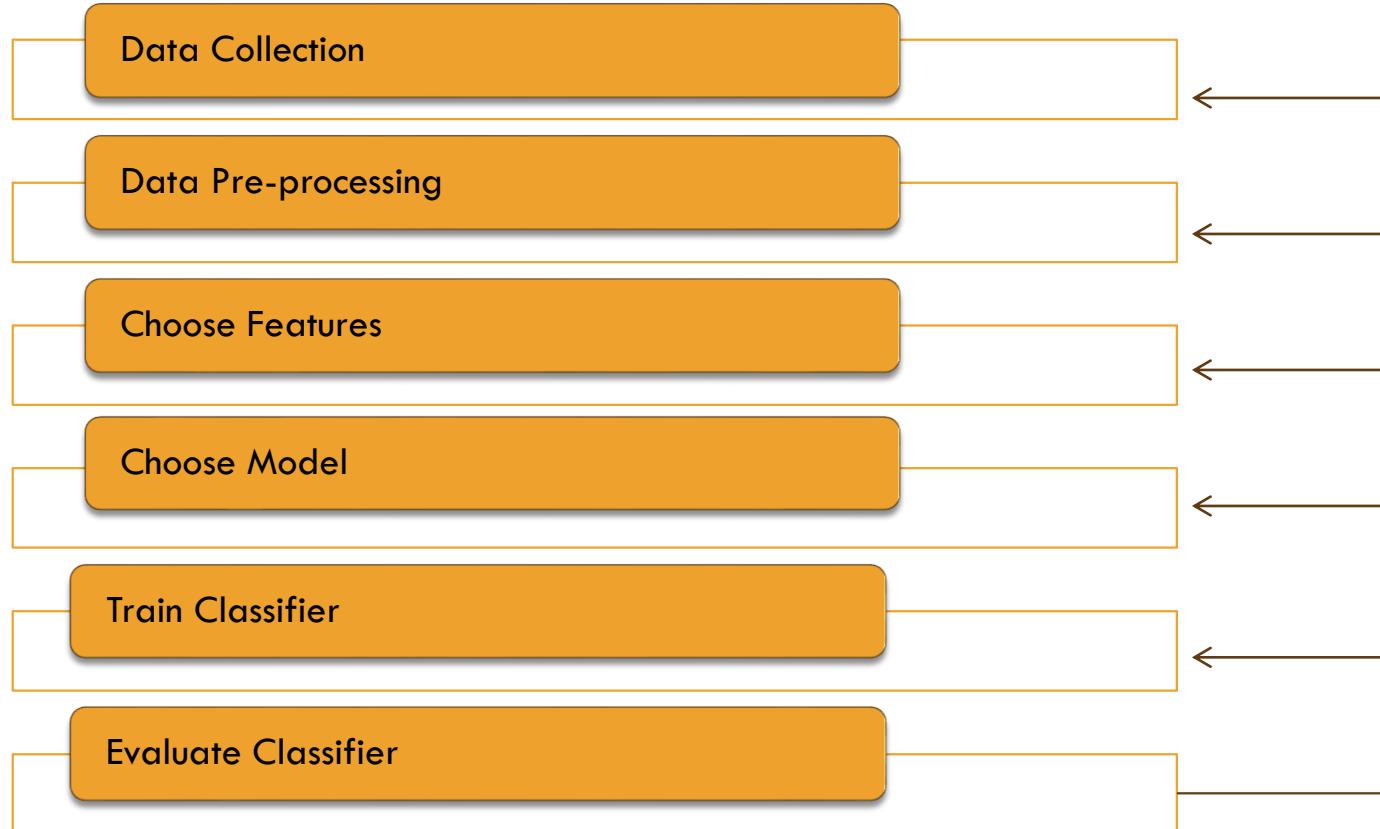


# Design Cycle

- A pattern classification system contains:
  - Acquisition Sensor → Raw database
  - Preprocessing Mechanisms
  - Mechanism of dimensionality reduction
  - Learning Algorithms
  - Mechanisms for validation

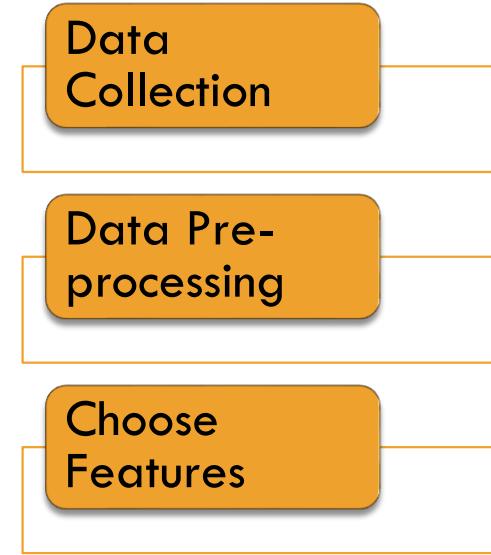


# Design Cycle



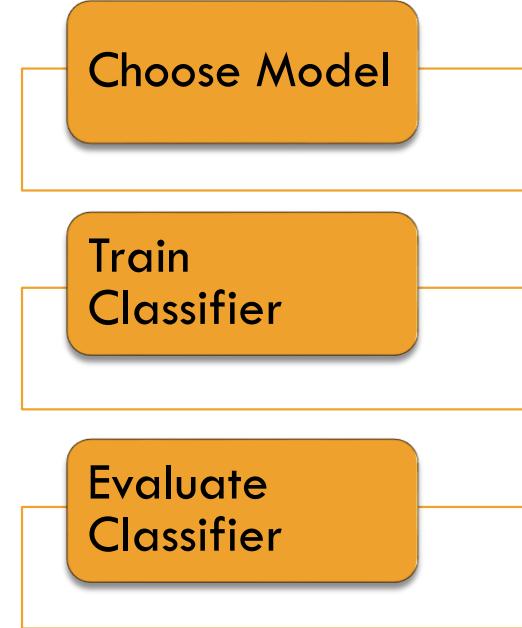
# Design Cycle

- **Data Collection**
  - Probably the most time-consuming part
  - How much data?
    - Sufficient large number of instances
    - Representative
- **Data Preprocessing**
  - Data Cleaning: Missing Values, Outliers, Signal/Noise Rate
  - Data Integration
  - Data Transformation
  - Data Reduction
- **Choose Features**
  - It is critical for success in a Pattern Recognition problem
  - It requires a basic understanding of the a priori problem
  - Ideally:
    - Simple to extract
    - Invariant to irrelevant transformation
    - Insensitive to noise

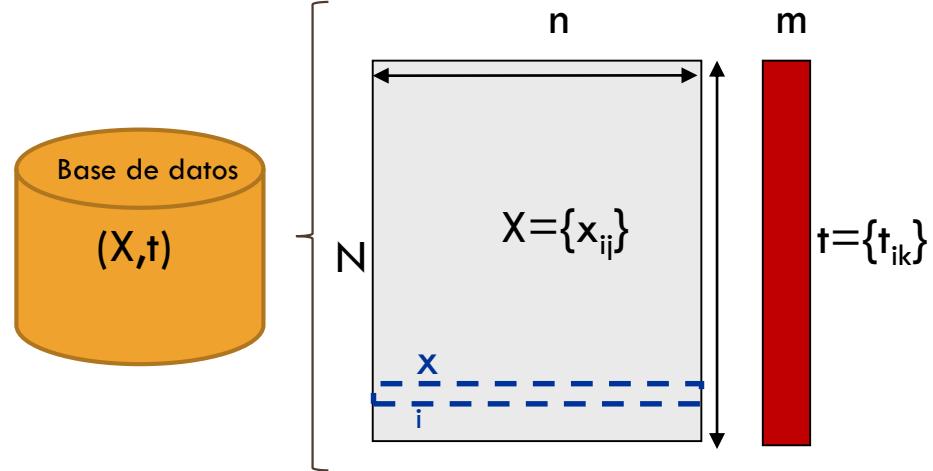
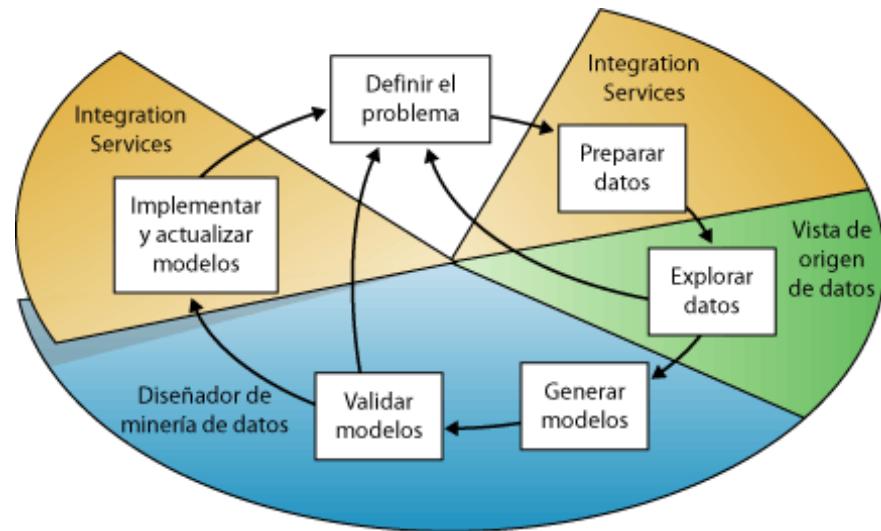


# Design Cycle

- Choose the model
  - Different types of models
  - Different parameters to play
- Training
  - Use data to obtain a good classifier
    - Identify best model
    - Determine appropriate models
- Evaluation
  - How good is the trained model?
  - Measure error rate → Performance
  - Overfitting versus generalization
  - May suggest switching:
    - From one set of feature to another set or adding new features
    - From one model to another one.



# Design Cycle



Representación de la Base de datos

Base de datos:  $X$

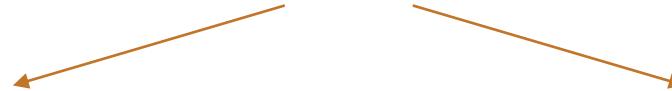
$N$  filas = patrones, casos, ejemplos, instancias, ...

$n$  columnas = atributos, variables, covariables, características, ...

# Steps to develop a prediction model: training, validation and test sets

- Dividir el conjunto de datos iniciales en subconjuntos de forma aleatoria, manteniendo la representación de partida.

Conjunto de entrenamiento: conjunto de datos con el que



Se construyen distintos modelos para resolver el problema

Se seleccionan uno o varios modelos finales

# Steps to develop a prediction model: training, validation and test sets

Conjunto de validación: conjunto de datos con el que

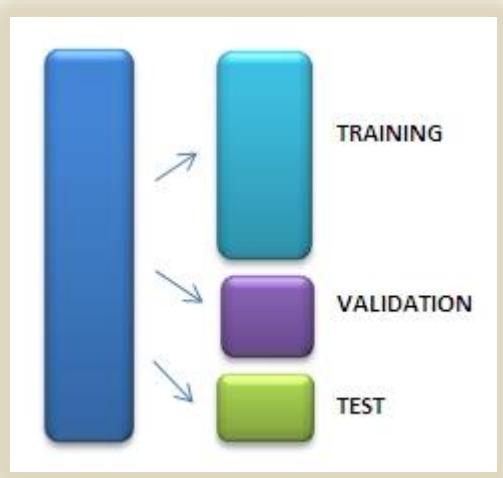
Se validan los modelos  
del punto anterior.  
Selección de los  
parámetros.

Se determina el modelo definitivo según:  
1.- Modelo con mejor ajuste a los datos  
2.- Modelo más “cercano” al problema  
de predicción

Conjunto de prueba/test: conjunto de datos con el que

Se realizan pruebas de funcionamiento del modelo.  
Estos datos no han jugado ningún papel en la selección del  
modelo

# Steps to develop a prediction model: training, validation and test sets



Posibles mecanismos de selección de conjuntos →

Validación cruzada (Cross-validation):

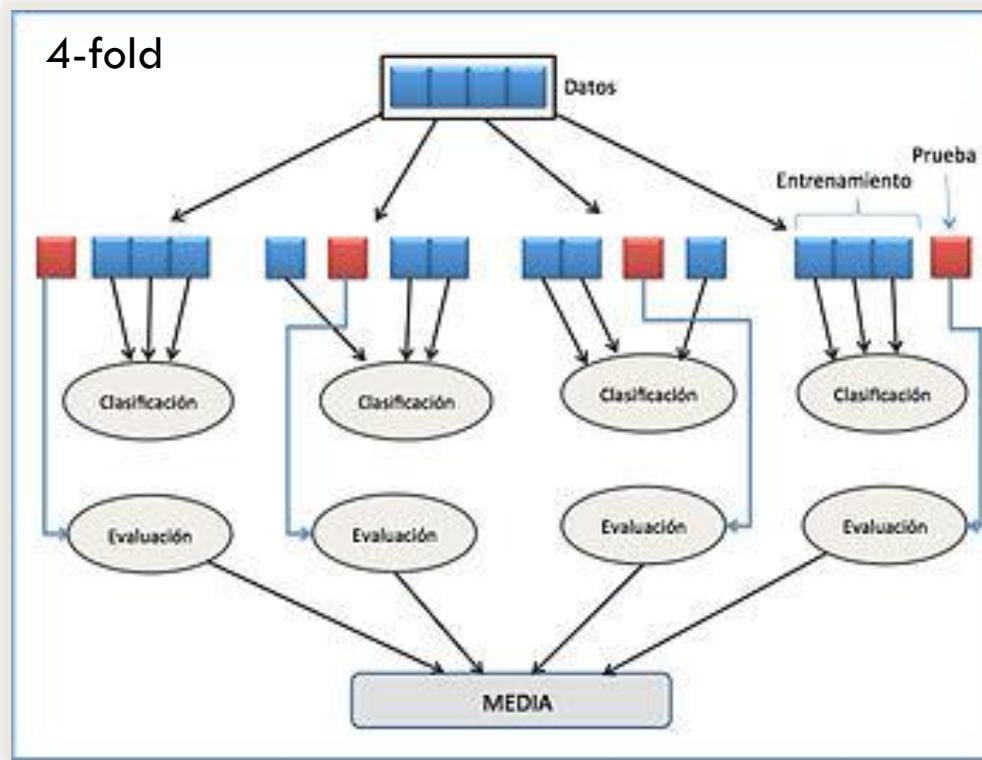
a) Muestra suficiente:

- Con conjunto de validación:  
50% Training + 25% Validation + 25% Test
- Sin validación  
60% Training + 40% Test

b) Muestras reducidas:

- K-fold ( $K=10$ )
- leave-one-out = n-fold
- Bootstrap = muestreo con reemplazamiento o sin reemplazo

# Steps to develop a prediction model: training, validation and test sets



# Steps to develop a prediction model: training models

## Modelos lineales:

- Análisis Discriminantes Lineales / Regresión Lineal

## Redes neuronales:

- Perceptrón Multicapa, PCM (“Multilayer Perceptron”, MLP)

## Modelos basados en núcleos:

- Máquinas de vectores de soporte (“Support Vector Machine”, SVM)

## Modelos probabilísticos:

- Redes Bayesianas

## Árboles de decisión:

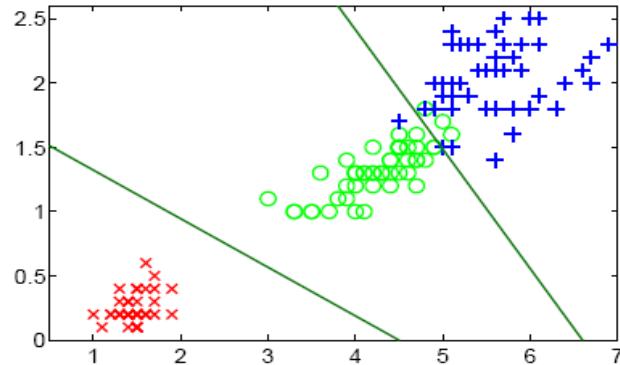
- ID3, C4.5

## Métodos de “clustering”:

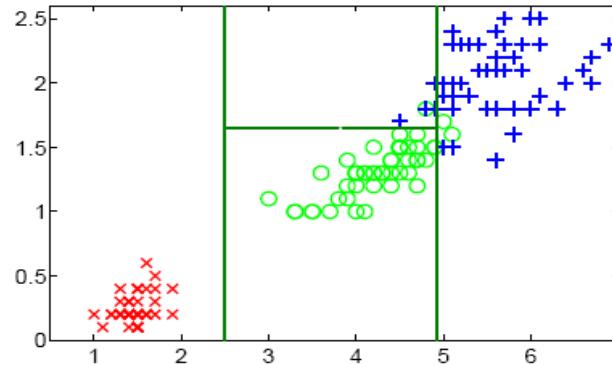
- “K-means”, Métodos Espectrales

# Steps to develop a prediction model: select models

Discriminante Lineal



Árboles de decisión

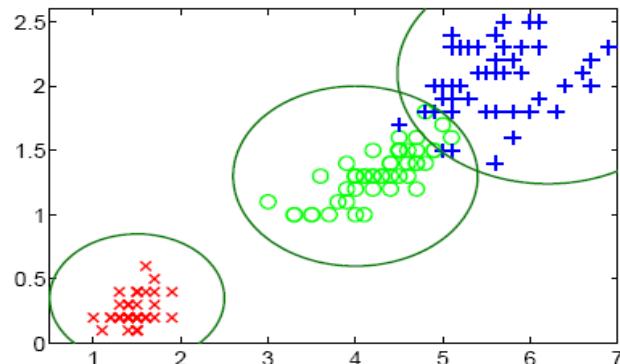


Iris Data Set  
(Fisher, 1936)

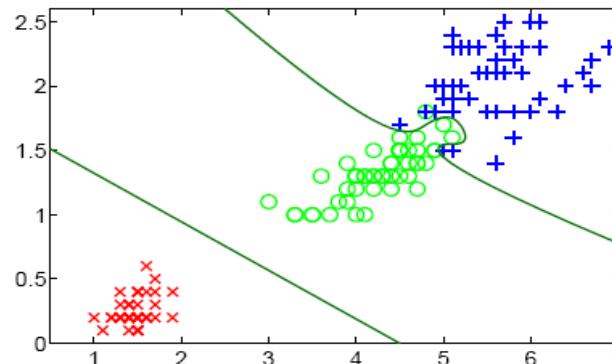
*setosa*

*versicolor*

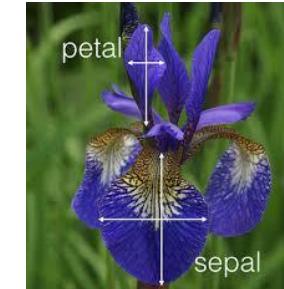
*virginica*



Mezclas de Gausianas

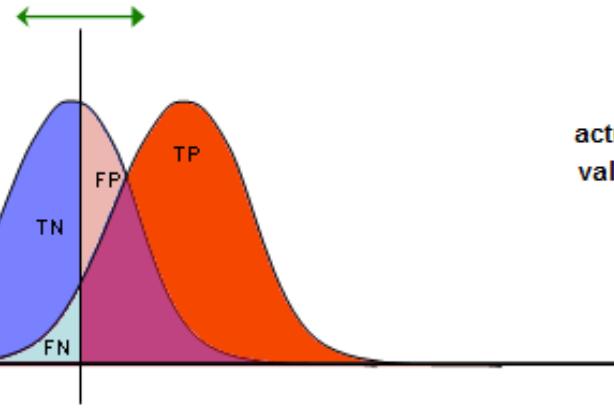


Kernel method (SVM)



# Steps to develop a prediction model: select models

- Clasification with two classes



		prediction outcome		total
actual value	<i>p'</i>	True Positive	False Negative	P'
	<i>n'</i>	False Positive	True Negative	
total		P	N	

$$Sens = \frac{TP}{TP + FN}$$

$$Spec = \frac{TN}{TN + FP}$$

$$acc = \frac{TN + TP}{P + N}$$

Sensitivity=Recall=TPR=Hit Rate

Specificity>Selectivity=FNR

$$Precision = \frac{TP}{TP + FP}$$

# Steps to develop a prediction model: select models

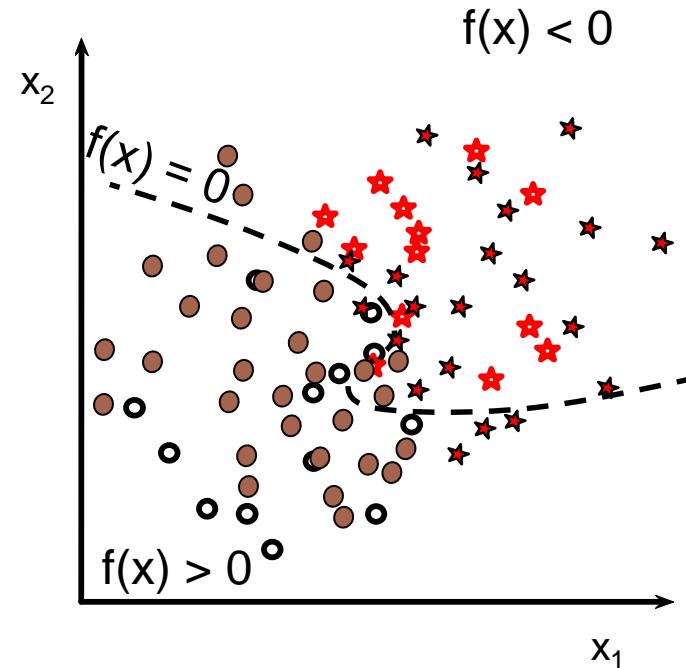
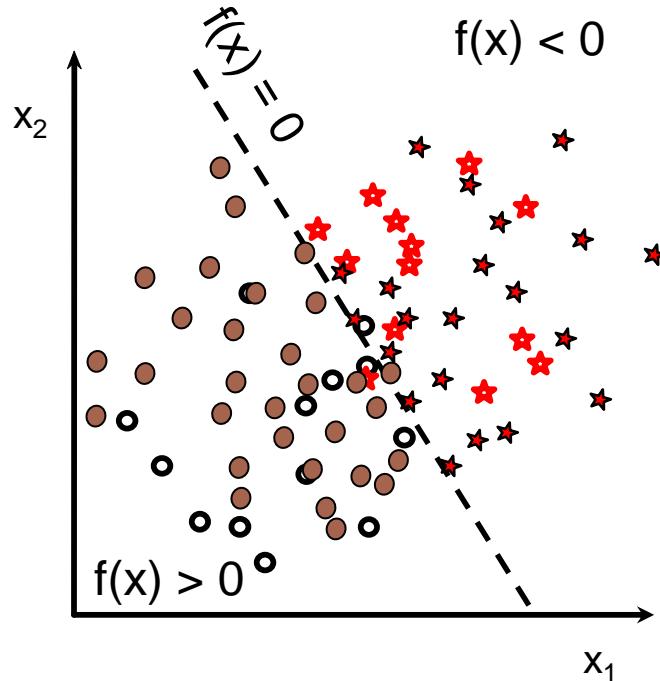
## Ejemplo: (+) Estar enfermo, (-) Estar sano

- La sensibilidad nos indica la capacidad de nuestro estimador para dar como casos positivos:
  - los casos realmente enfermos;
  - proporción de enfermos correctamente identificados.

Es decir, la sensibilidad caracteriza la capacidad de la prueba para detectar la enfermedad en sujetos enfermos.
- La especificidad nos indica la capacidad de nuestro estimador para dar como casos negativos los casos realmente sanos:
  - proporción de sanos correctamente identificados.

Es decir, la especificidad caracteriza la capacidad de la prueba para detectar la ausencia de la enfermedad en sujetos sanos.

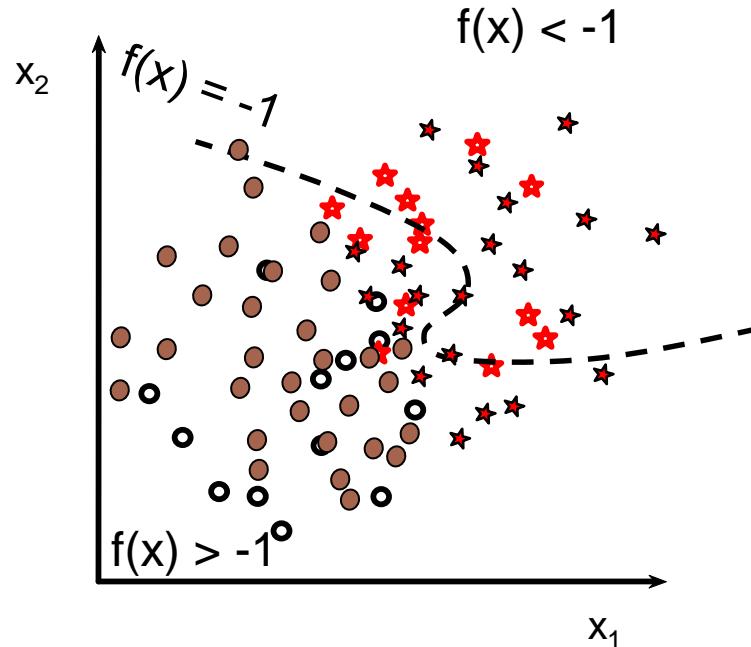
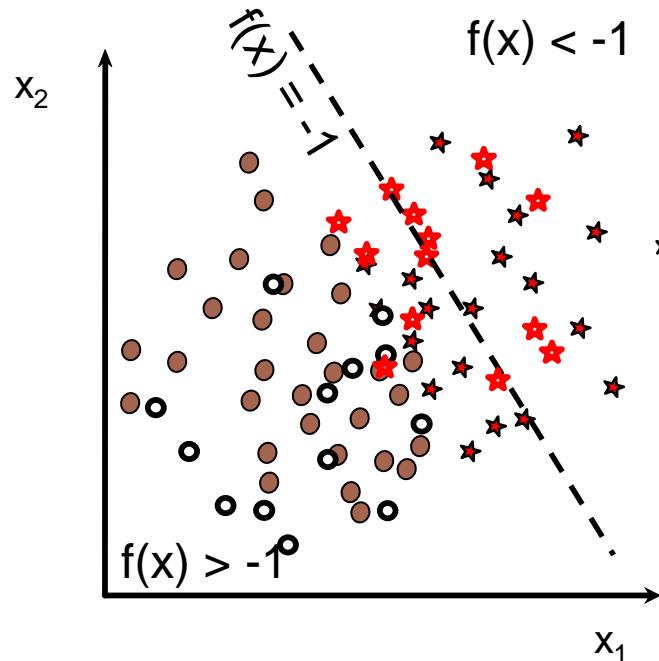
# Steps to develop a prediction model: select models



# Steps to develop a prediction model: select models

## ➤ Coste

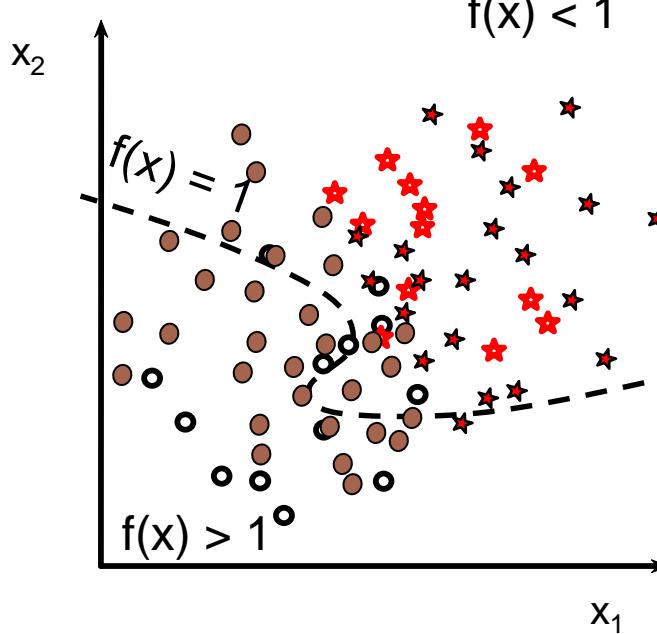
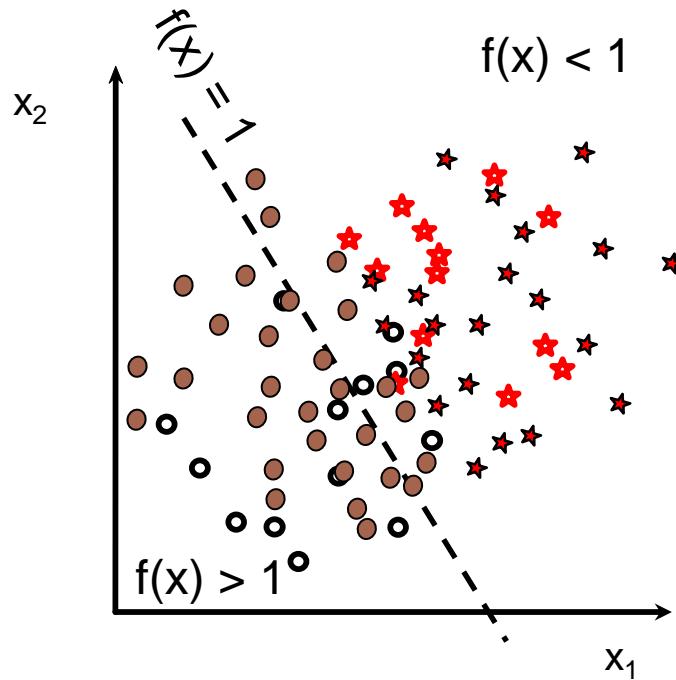
Variación del umbral → Elección de la frontera de decisión



# Steps to develop a prediction model: select models

## ➤ Coste

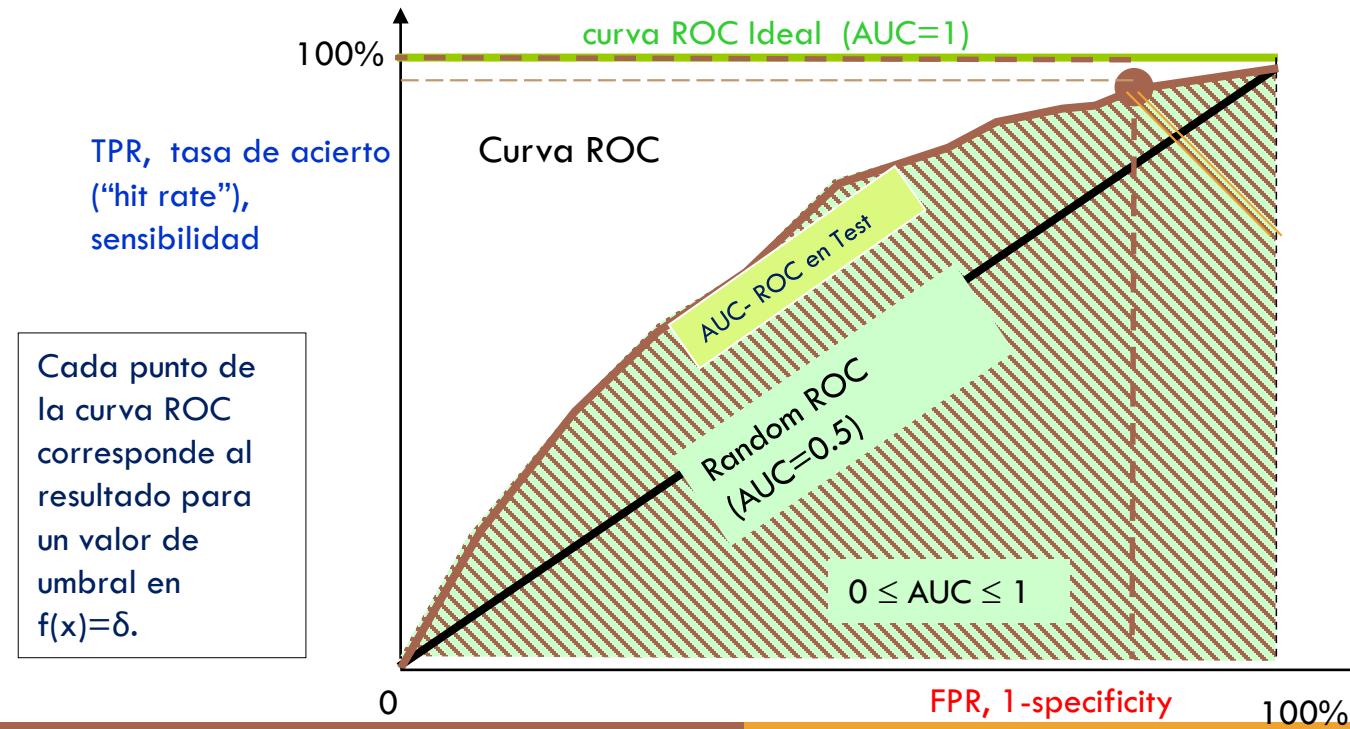
Variación del umbral  $\rightarrow$  Elección de la frontera de decisión



# Steps to develop a prediction model: select models

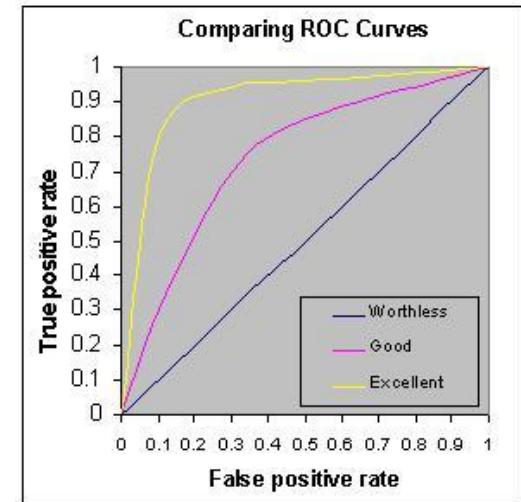
## ➤ Curva ROC y AUC-ROC

Requisito: un estimador de una variable con un parámetro ajustable



# Steps to develop a prediction model: select models

- Puntos de la curva ROC son **distintos niveles de decisión o valores de corte** que permiten una clasificación dicotómica de los valores de la prueba según sean superiores o inferiores al valor elegido.
- La diferencia esencial con el caso más simple es que ahora no tenemos un único par de valores de sensibilidad y especificidad que definan la exactitud de la prueba, sino **un conjunto de pares** correspondientes a cada uno de los distintos niveles de decisión.
- El área bajo la curva (AUC-ROC) es una medida de la calidad del clasificador
  - 0.90-1.00 = excelente
  - 0.80-0.90 = buena
  - 0.70-0.80 = justa
  - 0.60-0.70 = pobre
  - 0.50-0.60 = mala



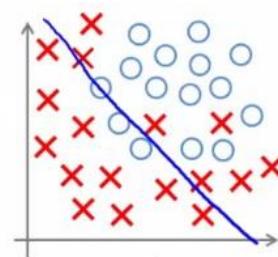
# Generalization

## ➤ Over-fitting

- Typically occurs when the ratio complexity of the model training set size is too high:

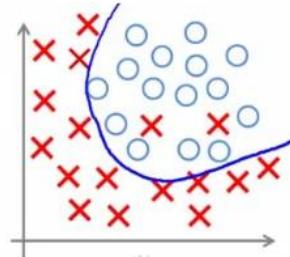
Occam's Razor:  
simpler theories are  
easier to understand

$$\frac{\text{Complexity of the model}}{\text{training set size } (N)}$$

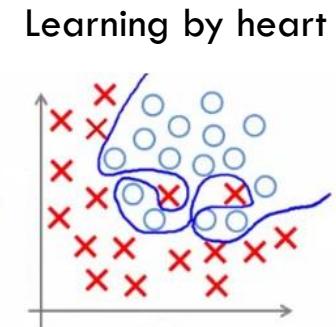


Under-fitting

(too simple to  
explain the  
variance)



Appropriate-fitting



Over-fitting

(forcefitting -- too  
good to be true)

Learning by heart

# Generalization

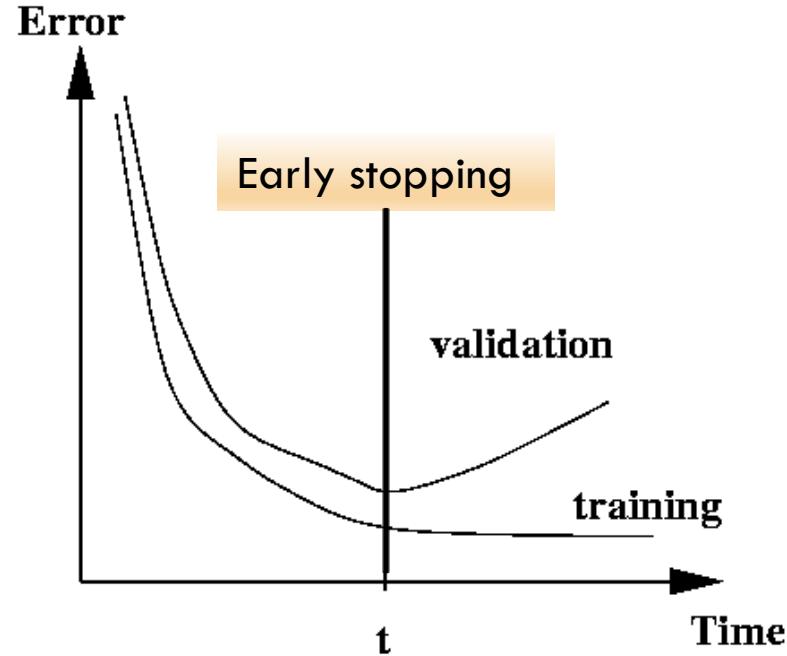
- Preventing over-fitting
  - Early stopping
  - Weight-Decay
    - Adding a penalty to the error function

$$\tilde{E} = E + \lambda \Omega$$

$$\Omega = \frac{1}{2} \sum_i w_i^2$$

- Training with random noise

$$\tilde{x} = x + \epsilon$$



# Kernel Methods

# Introduction (I)

- ▶ Parametric models for classification/regression:
  - ▶ The objective is to learn a set of adaptive parameters  $\mathbf{w}$  which determine the mapping from the input  $\mathbf{x}$  to the target  $y$ :

$$\hat{y} = f(\mathbf{x}, \mathbf{w})$$

- ▶ Need of a training phase
- ▶ One example is linear regression:

$$\hat{y} = \mathbf{w}^T \phi(\mathbf{x}) = \sum_{j=0}^{M-1} w_j \phi_j(\mathbf{x})$$

- ▶ Non-parametric models:
  - ▶ A subset of the input data points are used during classification
  - ▶ No training is needed, but methods are slow at making predictions
  - ▶ Examples: Parzen windows, Nearest Neighbors, etc.

## Introduction (II)

- ▶ Some linear parametric models can be reformulated using a *dual representation*
- ▶ The predictions on the test data are based only on a linear combination of a *kernel* function which is evaluated on a subset of the training data
- ▶ Somehow we manage to express the parametric model as a non-parametric one
- ▶ One of such models is regularized linear regression

# Regularized linear regression

- ▶ The attribute vectors are  $\mathbf{x}_i$ ,  $i = 1, \dots, N$
- ▶ The targets are  $y_i$ ,  $i = 1, \dots, N$
- ▶ The model (estimation of the target) is:

$$\hat{y}_i = \mathbf{w}^T \phi(\mathbf{x}_i) = \sum_{j=0}^{M-1} w_j \phi_j(\mathbf{x}_i)$$

- ▶ The base functions  $\phi(\mathbf{x}_i) = (\phi_0(\mathbf{x}_i), \phi_1(\mathbf{x}_i), \dots, \phi_{M-1}(\mathbf{x}_i))^T$  represent the attributes
- ▶ We assume  $\phi_0(\mathbf{x}_i) = 1$
- ▶ We minimize the following error function:

$$J(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^N \{y_i - \mathbf{w}^T \phi(\mathbf{x}_i)\}^2 + \frac{\lambda}{2} \mathbf{w}^T \mathbf{w}$$

# Minimization of $J(\mathbf{w})$

- We want to minimize:

$$\begin{aligned} J(\mathbf{w}) &= \frac{1}{2} \sum_{i=1}^N \{y_i - \mathbf{w}^T \phi(\mathbf{x}_i)\}^2 + \frac{\lambda}{2} \mathbf{w}^T \mathbf{w} \\ &= \frac{1}{2} \sum_{i=1}^N \left\{y_i - \sum_{j=0}^{M-1} w_j \phi_j(\mathbf{x}_i)\right\}^2 + \frac{\lambda}{2} \sum_{j=0}^{M-1} w_j^2 \end{aligned}$$

- With respect to the parameters  $w_k$ :

$$\frac{\partial J(\mathbf{w})}{\partial w_k} = 0$$

# Minimization of $J(\mathbf{w})$

► Operating:

$$\begin{aligned}\frac{\partial J(\mathbf{w})}{\partial w_k} &= - \sum_{i=1}^N y_i \phi_k(\mathbf{x}_i) + \sum_{i=1}^N \sum_{j=0}^{M-1} w_j \phi_j(\mathbf{x}_i) \phi_k(\mathbf{x}_i) + \lambda w_k \\ &= - \sum_{i=1}^N y_i \Phi_{ik} + \sum_{i=1}^N \sum_{j=0}^{M-1} w_j \Phi_{ij} \Phi_{ik} + \lambda w_k \\ &= - \sum_{i=1}^N \Phi_{ki}^T y_i + \sum_{j=0}^{M-1} \sum_{i=1}^N \Phi_{ki}^T \Phi_{ij} w_j + \lambda w_k \\ &= - (\Phi^T \mathbf{y})_k + \sum_{j=0}^{M-1} (\Phi^T \Phi)_{kj} w_j + \lambda w_k \\ &= - (\Phi^T \mathbf{y})_k + (\Phi^T \Phi \mathbf{w})_k + \lambda w_k = 0\end{aligned}$$

## Minimization of $J(\mathbf{w})$

- Finally:

$$-\Phi^T \mathbf{y} + \Phi^T \Phi \mathbf{w} + \lambda \mathbf{w} = \mathbf{0}$$

$$\mathbf{w} = (\mathbf{A} + \lambda I)^{-1} \Phi^T \mathbf{y}$$

- Where  $\Phi$  is the *design matrix*,  $\Phi_{ij} = \phi_j(\mathbf{x}_i)$ :

$$\Phi = \begin{pmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \dots & \phi_{M-1}(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \dots & \phi_{M-1}(\mathbf{x}_2) \\ \dots & \dots & \dots & \dots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \dots & \phi_{M-1}(\mathbf{x}_N) \end{pmatrix}$$

- And  $\mathbf{A} = \Phi^T \Phi$  is a  $M \times M$  matrix

## An example

- ▶ Four pairs  $(x; y)$ :  $\{(1; 0,8), (4; 4,1), (6; 6,2), (9; 8,5)\}$
- ▶ The vector of attributes is  $\phi(x) = (1, x)^T$
- ▶ And the design matrix is:

$$\Phi = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \\ 1 & x_4 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 4 \\ 1 & 6 \\ 1 & 9 \end{pmatrix}$$

- ▶ Let us assume  $\lambda = 1$
- ▶ Full solution in the IPython Notebook

# Reformulation of regularized linear regression

- Let's do it a different way:

$$\frac{\partial J(\mathbf{w})}{\partial w_k} = - \sum_{i=1}^N \Phi_{ki}^T y_i + \sum_{j=0}^{M-1} \sum_{i=1}^N \Phi_{ki}^T \Phi_{ij} w_j + \lambda w_k = 0$$

- So:

$$\begin{aligned} w_k &= -\frac{1}{\lambda} \sum_{i=1}^N \left\{ \sum_{j=0}^{M-1} \Phi_{ki}^T \Phi_{ij} w_j - \Phi_{ki}^T y_i \right\} \\ &= -\frac{1}{\lambda} \sum_{i=1}^N \Phi_{ki}^T \left\{ \sum_{j=0}^{M-1} \Phi_{ij} w_j - y_i \right\} \\ &= -\frac{1}{\lambda} \sum_{i=1}^N \Phi_{ki}^T \left\{ \sum_{j=0}^{M-1} w_j \phi_j(\mathbf{x}_i) - y_i \right\} \end{aligned}$$

# Reformulation of regularized linear regression

- This leads to:

$$\begin{aligned} w_k &= -\frac{1}{\lambda} \sum_{i=1}^N \Phi_{ki}^T \left\{ \sum_{j=0}^{M-1} w_j \phi_j(\mathbf{x}_i) - y_i \right\} \\ &= -\frac{1}{\lambda} \sum_{i=1}^N \Phi_{ki}^T \{\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_i) - y_i\} \\ &= \sum_{i=1}^N \Phi_{ki}^T a_i \end{aligned}$$

- Where:

$$a_i = -\frac{1}{\lambda} \{\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_i) - y_i\}$$

# Reformulation of regularized linear regression

- ▶ In matrix form:

$$\mathbf{w} = \Phi^T \mathbf{a}$$

- ▶ Now we can substitute  $w_k$  into  $J(\mathbf{w})$  to obtain an expression that depends only on  $\mathbf{a}$ :

$$J(\mathbf{a}) = \frac{1}{2} \sum_{i=1}^N \left\{ y_i - \sum_{j=0}^{M-1} \sum_{k=1}^N \Phi_{jk}^T a_k \Phi_{ij} \right\}^2 + \frac{\lambda}{2} \sum_{j=0}^{M-1} \left( \sum_{k=1}^N \Phi_{jk}^T a_k \right)^2$$

- ▶ The *dual problem* consists of minimizing  $J(\mathbf{a})$  with respect to  $a_k$ :

$$\frac{\partial J(\mathbf{a})}{\partial a_k} = 0$$

# Reformulation of regularized linear regression

- ▶ After some algebra we obtain:

$$\mathbf{a} = (\mathbf{K} + \lambda I)^{-1} \mathbf{y}$$

- ▶ Where  $\mathbf{K} = \Phi \Phi^T$  is a  $N \times N$  matrix that satisfies that:

$$\mathbf{K}_{ij} = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) \equiv k(\mathbf{x}_i, \mathbf{x}_j)$$

- ▶ The function  $k(\mathbf{x}_i, \mathbf{x}_j)$  is known as a *kernel* function

# Conclusion

- ▶ In the primal formulation:

$$\hat{y} = \mathbf{w}^T \phi(\mathbf{x}) = \sum_{j=0}^{M-1} w_j \phi_j(\mathbf{x})$$

To obtain  $\mathbf{w}$  we have to invert the matrix  $\mathbf{A} + \lambda I$ , which is  $M \times M$

- ▶ In the dual formulation:

$$\hat{y} = \mathbf{a}^T \Phi \phi(\mathbf{x}) = \sum_{i=1}^N a_i k(\mathbf{x}_i, \mathbf{x})$$

To obtain  $\mathbf{a}$  we have to invert the matrix  $\mathbf{K} + \lambda I$ , which is  $N \times N$

- ▶ This technique is generalizable to other problems

# Machine Learning in the context of Big Data

Luis Fernando Lago Fernández

*EPS - UAM*

[www.catedrauamibm.com](http://www.catedrauamibm.com)



# Outline

- ▶ Introduction to ML in the context of Big Data
- ▶ The ML design cycle

# PART 1: MACHINE LEARNING IN THE CONTEXT OF BIG DATA

# Overview

- ▶ **Big Data = Data + Analytics**
- ▶ What do we mean by “Big”?
  - ▶ Volume
  - ▶ Velocity
  - ▶ Variety
  - ▶ But also, veracity and value
- ▶ **Predictive Analytics** uses data to make future predictions with the help of **Machine Learning** algorithms

# Is Big Data just data?

**Without analytics Big Data is just data**

## Data Analysis and Data Analytics

- ▶ **Data analysis** is the process of examining data to find facts, relationships, patterns, insights and/or trends. The overall goal is to support better decision making.
- ▶ **Data analytics** is a broader term than encompasses data analysis. It includes the management of the complete data lifecycle: collecting, cleansing, organizing, storing, analyzing and governing

(From *Big Data Fundamentals: Concepts, Drivers and Techniques*, Prentice Hall, 2016)

# Data Analytics

- ▶ Descriptive analytics
- ▶ Diagnostic analytics
- ▶ Predictive analytics
- ▶ Prescriptive analytics

# Descriptive Analytics

Answer questions about events that have already occurred

- ▶ What was the sales volume over the past 6 months?
- ▶ How many new customers in the last year?

Determine the cause of something that happened in the past

- ▶ Why were sales in June less than sales in May?
- ▶ Why the demand for product A has decreased in the last 2 months?

# Predictive Analytics

Determine the outcome of an event that might occur in the future.  
Models generate future predictions based on past events

- ▶ If a customer has purchased products A and B, what are the chances that he will also purchase product C?
- ▶ What is the probability of my customers going to another company for the same service?

# Prescriptive Analytics

Built upon the results of predictive analytics by prescribing actions that should be taken.

- ▶ Which products should I offer to customers that have purchased products A and B?
- ▶ Which actions should I take to keep my customers?

# Predictive Analytics and Machine Learning

In this course we will focus on Predictive Analytics and Machine Learning

- ▶ How to build models to extract useful information from data
- ▶ How to learn from data in order to make predictions

# Machine Learning

“Field of study that gives computers the ability to learn without being explicitly programmed.”

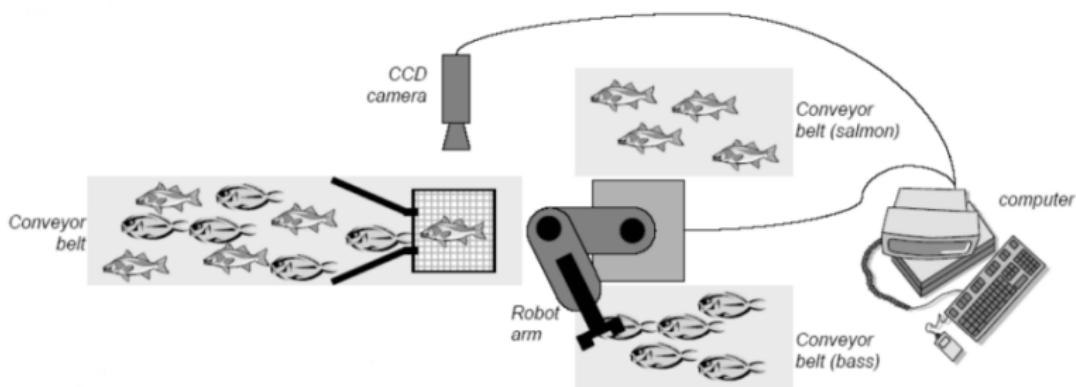
(Attributed to A. Samuel, 1959)

“Machine learning explores the study and construction of algorithms that can learn from and make predictions on data.”

(From Wikipedia: [https://en.wikipedia.org/wiki/Machine\\_learning](https://en.wikipedia.org/wiki/Machine_learning))

# Machine Learning - An example

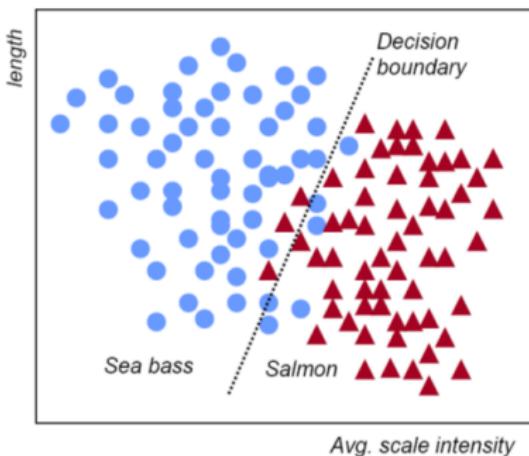
- ▶ A fishing company wants to automate the process of separation of fish (salmon vs sea bass), using images recorded by a CCD camera



(From Duda, Hart and Stork, *Pattern Classification*, 2001)

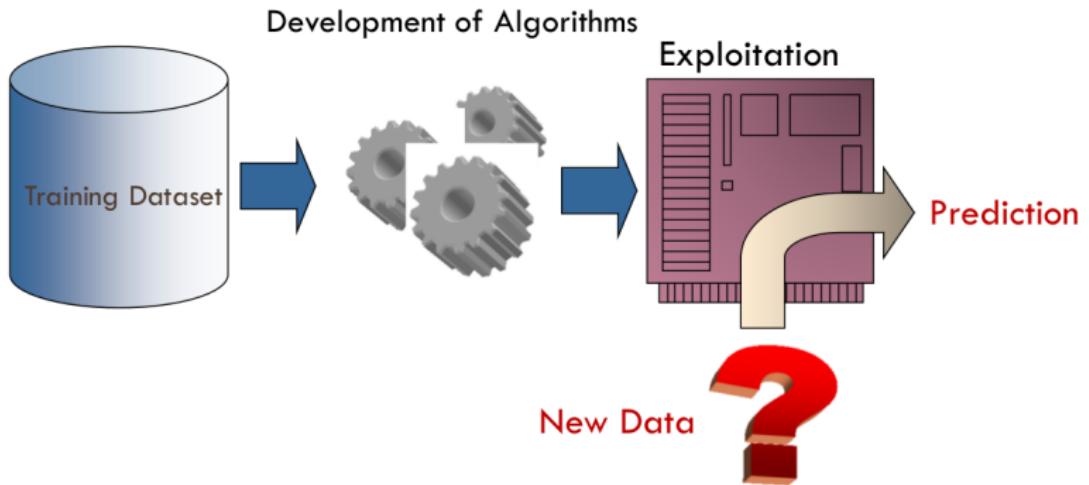
# Machine Learning - An example

- ▶ Combining fish length and scale intensity they build the following linear classifier, which achieves a 95,7 % classification accuracy on the *training* data
- ▶ What do you expect to happen on new, unlabeled data?
- ▶ Will this model make good predictions?



(From Duda, Hart and Stork, *Pattern Classification*, 2001)

# The Machine Learning design cycle



# Machine Learning types

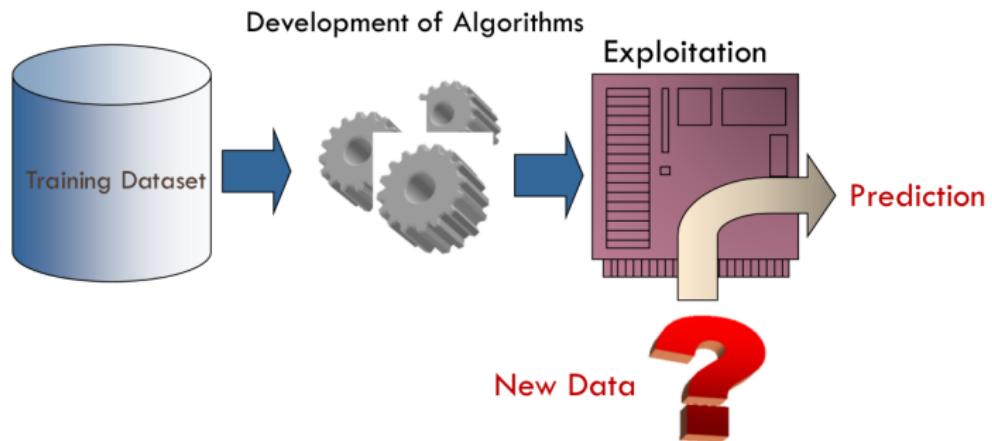
- ▶ Supervised learning
- ▶ Unsupervised learning
- ▶ (Semi-supervised learning)
- ▶ Reinforcement learning

## PART 2: THE ML DESIGN CYCLE

# Outline

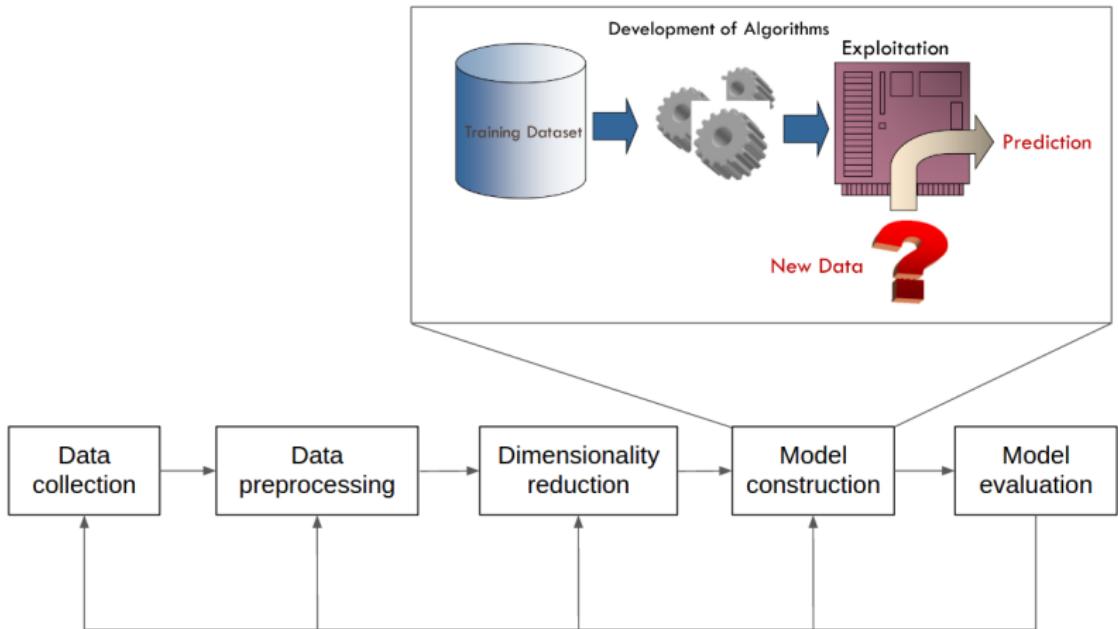
- ▶ Overview of the ML design cycle
- ▶ Pattern classification
- ▶ Model evaluation and selection
- ▶ Data preparation and audit
- ▶ Attribute selection and dimensionality reduction

# The Machine Learning design cycle

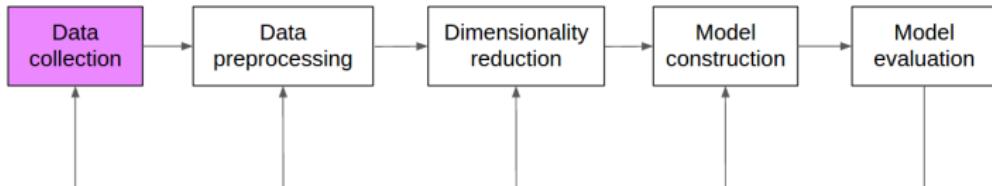


- ▶ Building the model is just one of the steps

# The Machine Learning design cycle



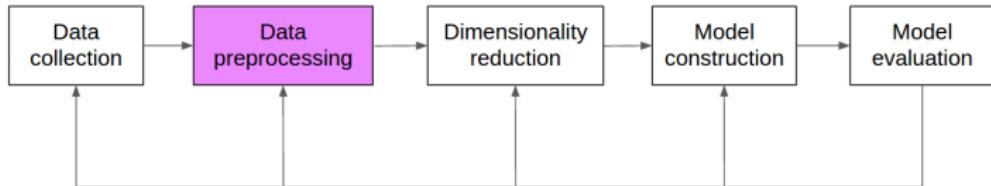
# Data collection



## Data Collection

- ▶ Probably the most time-consuming part
- ▶ How much data?
  - ▶ Sufficiently large number of instances
  - ▶ Representative

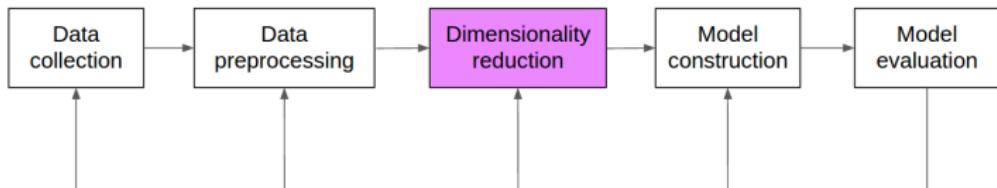
# Data preprocessing



## Data Preprocessing

- ▶ Correct inconsistencies in data
- ▶ Data Cleansing: missing values, outliers, noise, etc.
- ▶ Data Transformation: normalization, smoothing, segmentation, etc.

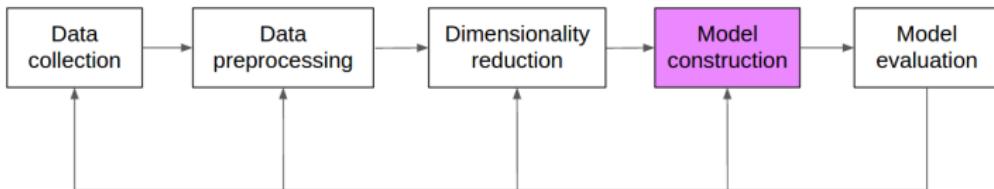
# Feature selection and dimensionality reduction



## Feature selection

- ▶ Critical in any Pattern Recognition problem
- ▶ Requires a basic understanding of the problem
- ▶ Ideal features:
  - ▶ Simple to extract
  - ▶ Invariant to irrelevant transformation
  - ▶ Insensitive to noise

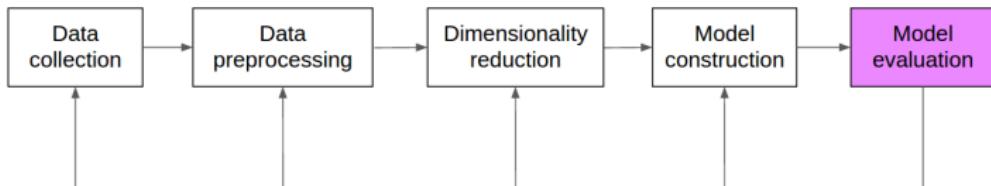
# Model construction (pattern classification)



## Model construction

- ▶ Select the model: linear discriminant, neural net, decision tree, etc.
- ▶ Use the data to train the classifier

# Model evaluation

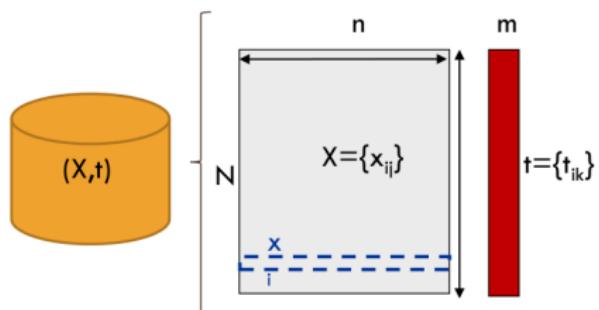


## Evaluation

- ▶ How good is the trained model?
- ▶ Measure error rate to obtain model performance
- ▶ Compare the performances of different models
- ▶ Overfitting versus generalization

# Pattern classification

- The problem data is the set of patterns:  
 $\{(\mathbf{x}_1, t_1), (\mathbf{x}_2, t_2), \dots, (\mathbf{x}_n, t_n)\}$ 
  - $n$  is the number of training patterns
  - $\mathbf{x}_i$  is the attribute vector for pattern  $i$
  - $t_i$  is the class label for pattern  $i$



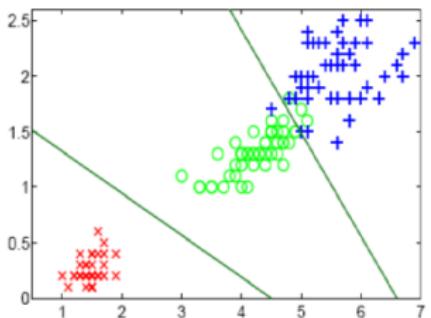
- The goal is to predict the class of each pattern

# Pattern classification

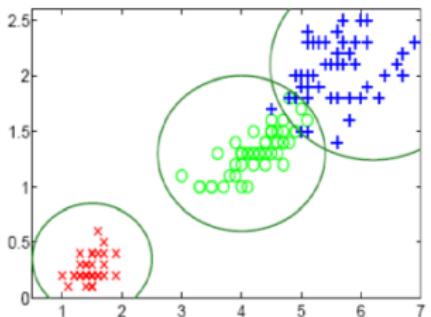
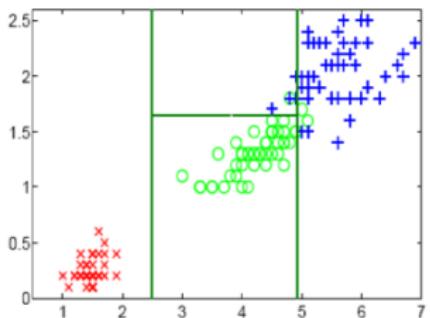
- ▶ A classifier is a function  $f(\mathbf{x}, \Theta)$  that assigns each pattern  $\mathbf{x}_i$  an estimation of its class  $y_i = f(\mathbf{x}_i, \Theta)$
- ▶ Training the classifier means tuning the parameters  $\Theta$  in order to minimize an error function that measures the discrepancy between the real classes  $t_i$  and the predictions  $y_i$ .
- ▶ Different function families define different types of classifiers: linear discriminant analysis, neural networks, decision trees, Bayesian methods, support vector machines, etc.

# Pattern classification

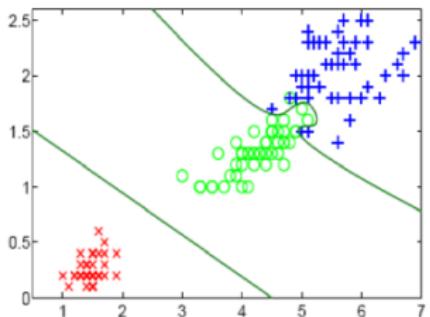
Linear discriminant



Decision tree

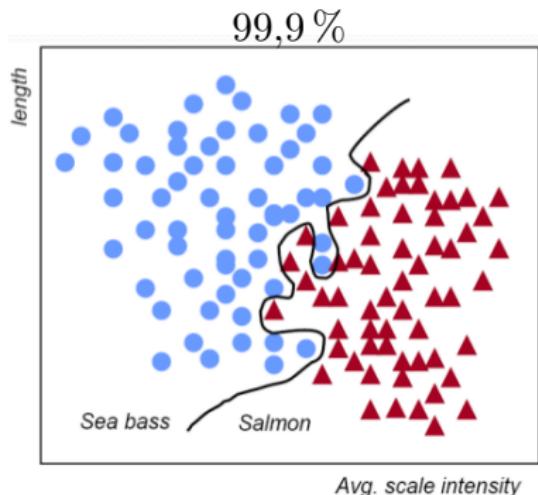
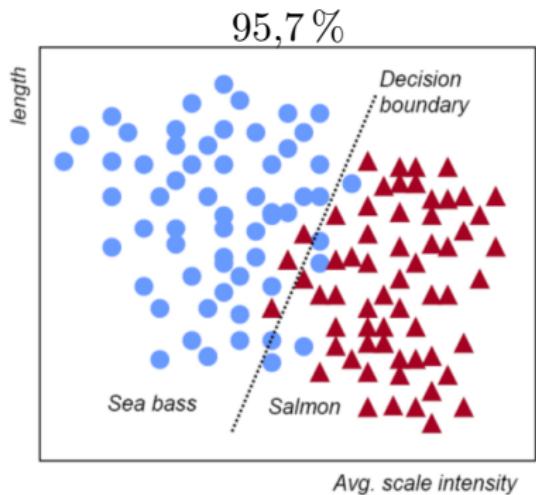


Gaussian mixture



Support vector machine

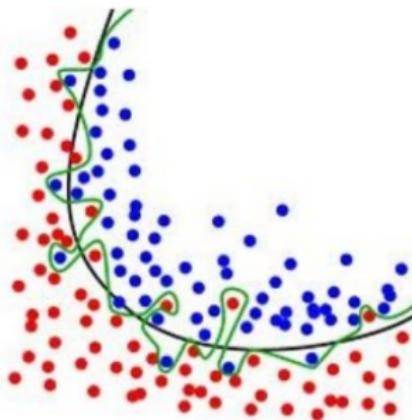
# Model evaluation and selection



(From Duda, Hart and Stork, *Pattern Classification*, 2001)

- ▶ Which model is better?

## Overfitting vs generalization



- ▶ Measuring the classification accuracy on the training data is not the best way to evaluate the model
- ▶ It is better to use a different data set not used during the training phase

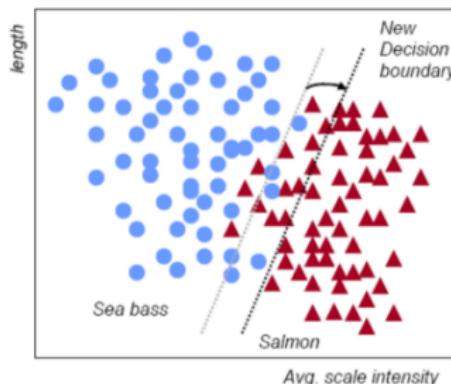
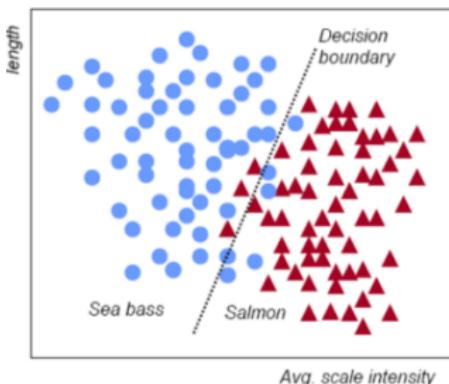
# Training, validation, test



- ▶ Training set: used to train the models
- ▶ Validation set: used to validate the models and to select the final classifier
- ▶ Test set: used to test the model performance

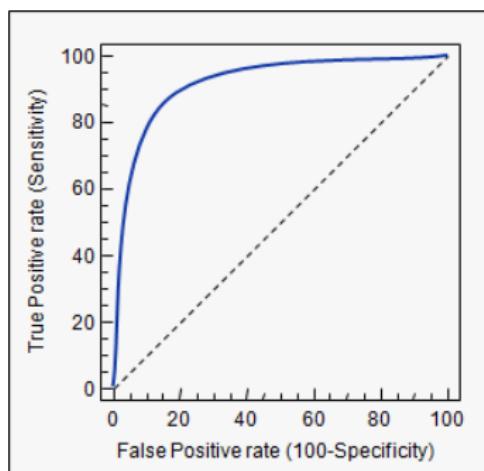
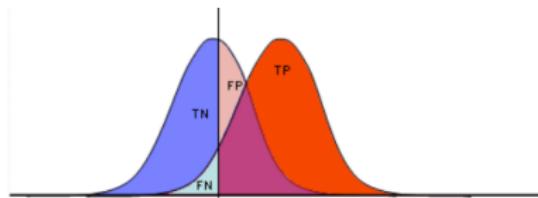
## Cost versus classification rate

- ▶ Usually misclassified patterns from different classes imply different costs
- ▶ In those cases we might want to adjust the decision boundary in order to minimize the cost associated to misclassifications



(From Duda, Hart and Stork, *Pattern Classification*, 2001)

# ROC analysis



- ▶ True positive rate:

$$TPR = \frac{TP}{TP + FN}$$

- ▶ False positive rate:

$$FPR = \frac{FP}{FP + TN}$$

- ▶ The area under the curve (AUC) is a good measure of model performance

# Data preparation and audit

- ▶ Process that transforms the raw input data into the set of patterns used to train the models
- ▶ Initial data may have any form
- ▶ Final data used to train the models:
  - ▶ Attribute matrix (one row per pattern, one column per attribute), usually numeric data
  - ▶ Vector of targets, usually symbolic data
- ▶ Note that
  - ▶ Training data must be measured in the same conditions as evaluation data
  - ▶ Any bias in the training data should be avoided

# Data preprocessing

- ▶ Correct inconsistencies in data that may negatively affect the training phase
- ▶ Data preprocessing has a strong impact in the model performance
- ▶ Data preprocessing includes
  - ▶ Data cleansing: filter data that is not useful, correct errors in the database
  - ▶ Data transformation: change coding into a more appropriate format that facilitates learning

# Data cleansing and transformation

- ▶ Cleansing:
  - ▶ Fill in missing values
  - ▶ Filter noise
  - ▶ Remove inconsistencies
  - ▶ Remove correlations
  - ▶ Remove outliers
- ▶ Transformation:
  - ▶ Normalization and/or standardization
  - ▶ Smoothing
  - ▶ Data aggregation
  - ▶ Segmentation
  - ▶ Compression of temporal series

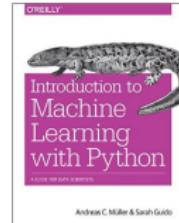
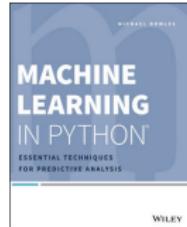
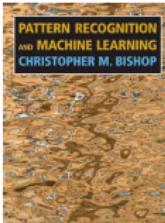
- ▶ Avoid the *curse of dimensionality*
- ▶ Select the most informative attributes, those which convey the most information about the target
- ▶ Avoid false predictors: attributes that are strongly correlated with the target, but that are not available in a realistic prediction scenario
- ▶ Reduce dimensionality by making projections in the attribute space
  - ▶ Principal Component Analysis
  - ▶ Linear Discriminant Analysis

# Summary

- ▶ Machine Learning is in the core of Big Data
- ▶ Models/classifiers are trained using labeled data, and used to make predictions on unlabeled data
- ▶ Prevent overfitting: models should be evaluated using data that was not used for training
- ▶ Building the model is just a single step in the whole process (data collection, preprocessing, dimensionality reduction, etc.)

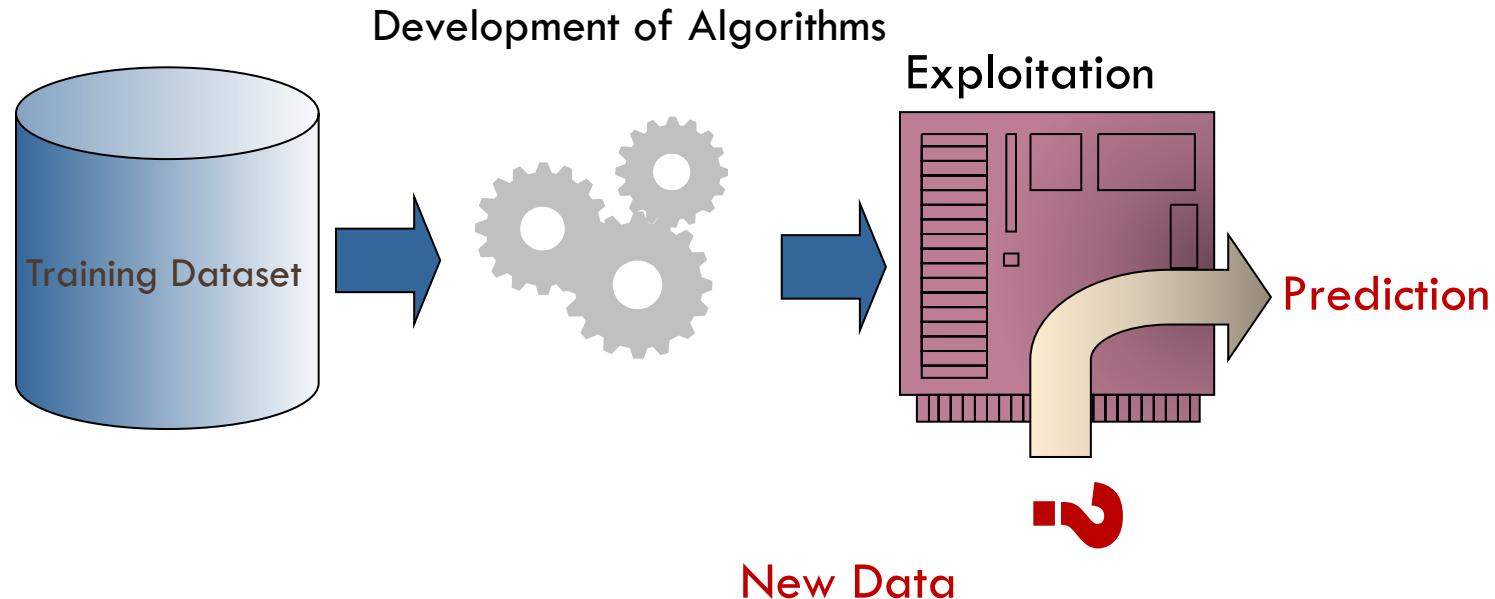
# Bibliography

- ▶ *Pattern Classification*. R.O. Duda, P.E. Hart, D.G. Stork. Wiley, 2001.
- ▶ *Pattern Recognition and Machine Learning*. C. Bishop. Springer, 2006.
- ▶ *Machine Learning in Python: Essential Techniques for Predictive Analysis*. M. Bowles. Wiley, 2015.
- ▶ *Introduction to Machine Learning with Python. A Guide for Data Scientists*. A.C. Mueller, S. Guido. O'Reilly, 2016.



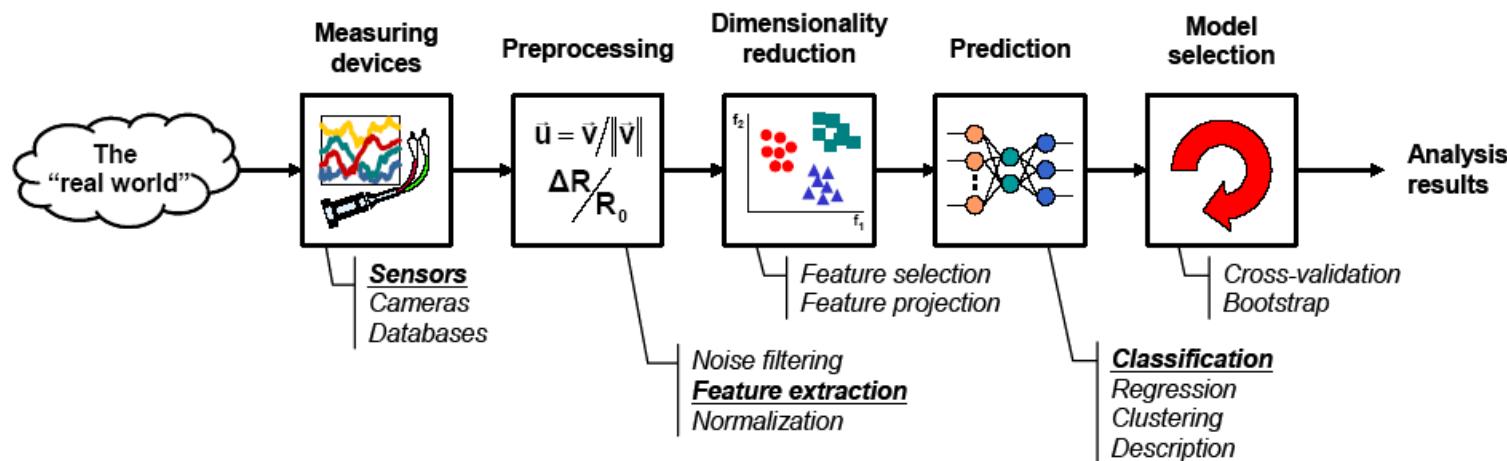
# Preprocesamiento de Datos

# Design Cycle



# Design Cycle

- A pattern classification system contains:
  - Acquisition Sensor → Raw database
  - Preprocessing Mechanisms
  - Mechanism of dimensionality reduction
  - Learning Algorithms
  - Mechanisms for validation



# Design Cycle

Data Collection

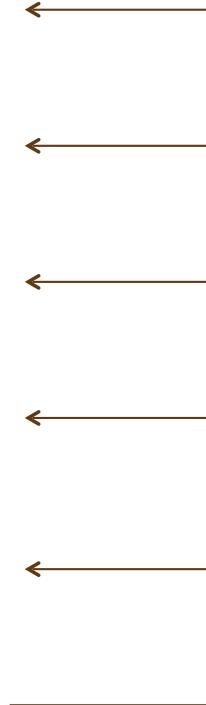
Data Pre-processing

Choose Features

Choose Model

Train Classifier

Evaluate Classifier



# Introduction

- Objective: To construct / use algorithms that learn from data
- Data is cheap and abundant:
  - data warehouses
  - data marts
- Knowledge is expensive and scarce.



Volumen

Velocity

Variety

# Terminology: Attributes and patterns

**Attribute (or variable or characteristic or descriptor)**, it is any distinctive aspect, quality or characteristic.

- nominal (i.e, color: white, red, yellow, green, blue, ...),
- numeric (i.e., height, measured in meters).

## Patterns (or Cases or Instances)

Collection (possibly ordered and structured) of descriptors (features) that represent an object.

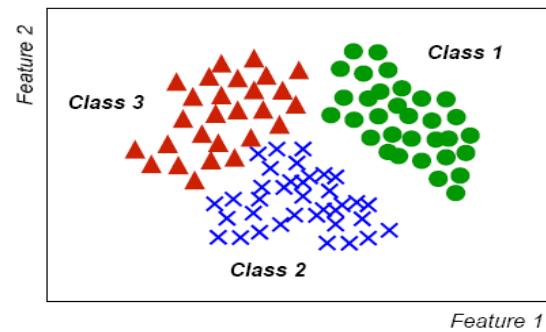
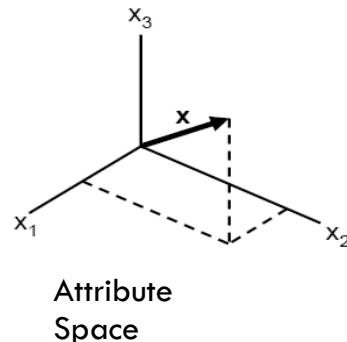
Important: patterns that describe objects of the same class have similar characteristics.

# Terminology: Attributes and patterns

- Each pattern is represented by a set of attributes → a column vector of d dimensions called attribute vector
- d-dimensional space defined by this vector is the attribute space
- The patterns are represented as points in attribute space

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}$$

Vector of attribute

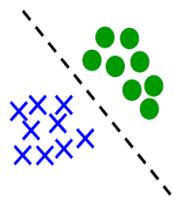


# Terminology: Attributes and patterns

What is a "good" vector of attributes?

The quality of a vector of attributes is related to its ability to discriminate examples from different classes:

- The attributes of instances of the same class should have similar values
- The attributes of instances from different classes should have different values



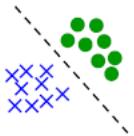
*"Good" features*



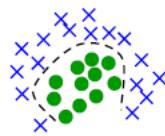
*"Bad" features*

# Terminology: Attributes and patterns

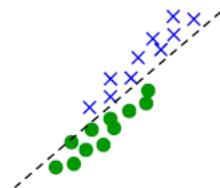
Additional properties related to attributes :



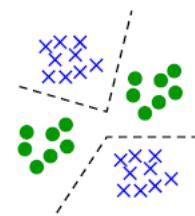
Lineal Separability



Non lineal Separability



Highly  
Correlated attributes



Multi-modal

# Data Preprocessing

- How can data be prepared for the next text mining /machine learning process?
  - **Data cleaning**, is the process oriented to eliminate data with noise or incorrect.
  - **Data integration**, tries to integrate different data sources in a coherent and homogeneous warehouse such as a data warehouse or a data cube.
  - **Data transformation**, or transformation of the data as, for example, a normalization.
  - **Data reduction**, it is aimed at reducing size of the data by aggregation and / or elimination of redundant features.

# Data cleaning

- Real-world data is often presented in an incomplete, inconsistent and noisy way. We will learn to clean up the data by:
  - eliminating missing data,
  - softening the effect of noise,
  - eliminating data out of range and
  - correcting inconsistencies.
- The representation and quality of data is first and foremost before running any analysis.
- Data preparation and filtering steps can be considerable time-consuming.

# Data Preprocessing

- Knowledge Discovery during the training phase is more difficult when:
  - there is much irrelevant and redundant information
  - noisy
  - unreliable data
- Data gathering methods are often loosely controlled
  - Resulting in out-of-range values (e.g., Income: -100),
  - Impossible data combinations (e.g., Gender: Male, Pregnant: Yes),
  - Missing values
- The product of data pre-processing is the final dataset.

# “Garbage In – Garbage Out”

## MODEL CALCULATIONS

”Garbage In-garbage Out” Paradigm



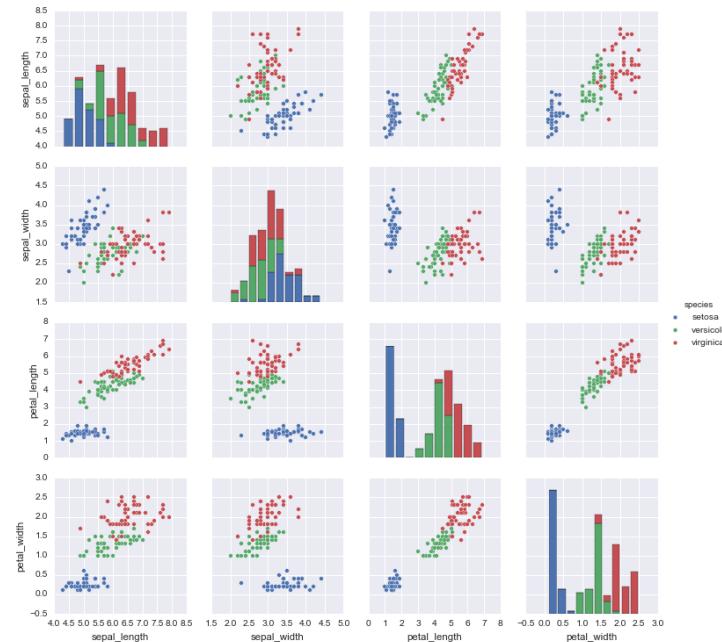
**Garbage** is an unwanted or undesired material or substance

# Data Cleaning or Data Cleansing



**Data auditing:** The data is audited with the use of statistical and database methods to detect anomalies and contradictions.

- Incomplete
  - lacking attribute values
  - certain attributes of interest
  - containing only aggregate data
- Noisy
  - Containing errors
- Outliers, values which deviate from the expected
- Inconsistent



# Data Cleaning – Incomplete Data

- Attributes of interest may not always be available
- Data may not be included simply because it was not considered important at the time of entry
- Relevant data may not be recorded due to a misunderstanding
- Incomplete Data due to equipment malfunctions



# Data Cleaning – Incomplete Data (II)

- What to do with missing values?
  - Ignore the tuple (pattern, instance, case, ...)
    - This is usually done when the class label is missing (Classification Problem)
    - In general, it's not very effective, unless the tuple contains several attributes with missing values.
  - Fill in the missing value manually
  - Use a global constant to fill in the missing value
  - Use the attribute mean to fill in the missing value
  - Use the attribute mean for all samples belonging to the same class as the given tuple
  - Use the most probable value to fill in the missing value. Inference-based tools: regression, bayesian formalism or decision tree induction, KNN Imputation

BIAS the data:  
filled-in value may not  
be correct

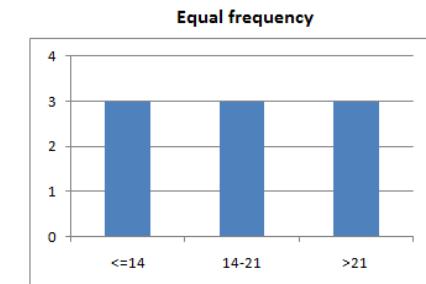
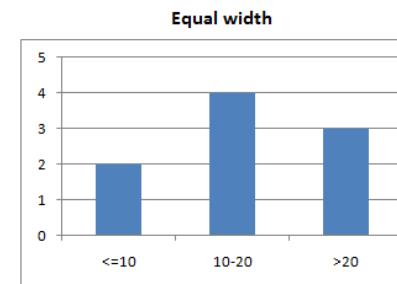
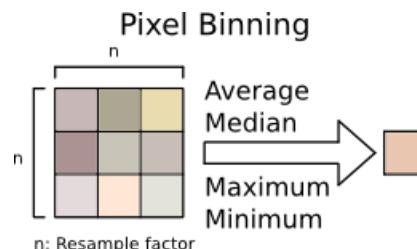
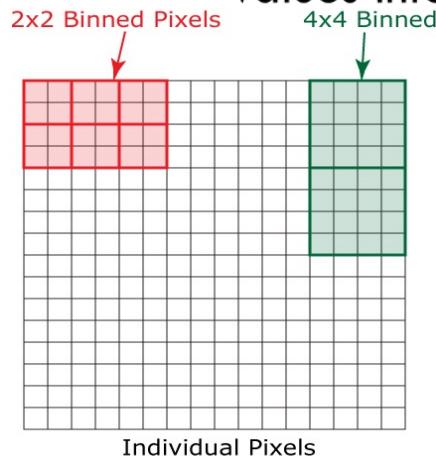
# Data Cleaning – Noisy

- Noise is a random error or variance in a measured variable

$$\hat{X} = X + \varepsilon$$

- Smoothing noisy data:

- Binning methods are a way to group a number of more or less continuous values into a smaller number of "bins"



Data={0, 4, 12, 16, 16, 18, 24, 26, 28}

# Data Cleaning – Outlier Detection

## Outliers

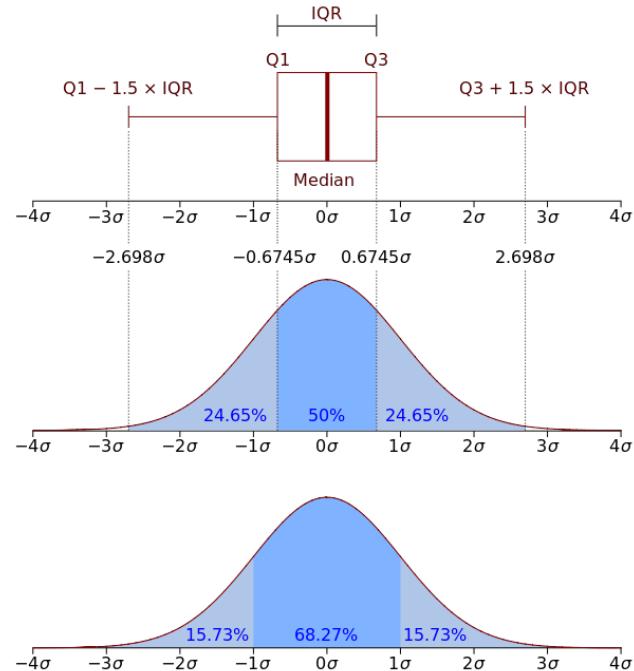
- **CASE I:** the outlying value should be deleted from the analysis (or corrected if possible)

An outlier may indicate bad data

- the data may have been coded incorrectly
- an experiment may not have been run correctly

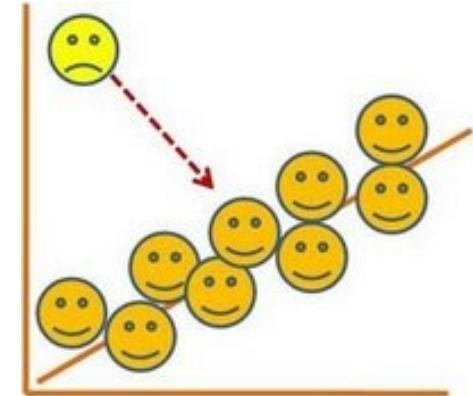
- **CASE II:**

- outliers may be due to random variation
- may indicate something scientifically interesting



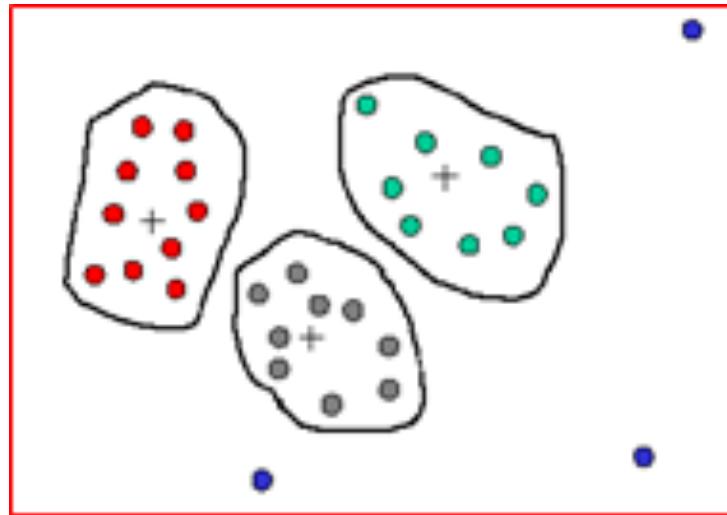
# Data Cleaning – Outlier Detection

- Outliers should be investigated carefully.
- Often they contain valuable information about the process under investigation or the data gathering and recording process.
- Before considering the possible elimination of these points from the data, one should try to understand why they appeared and whether it is likely similar values will continue to appear.
- Of course, outliers **are often bad data points.**



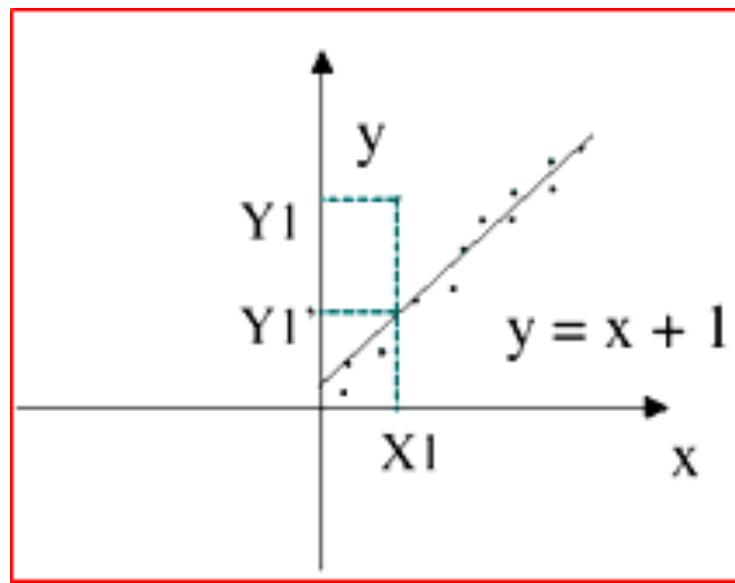
# Data Cleaning – Outlier Detection (II)

- Clustering: Similar values are organized in groups or clusters. Values falling outside of the cluster maybe considered as outliers and may be candidate for elimination



# Data Cleaning – Outlier Detection (III)

- Regression: Fit Data to a Function. The new values given by the function are used instead of the original values.

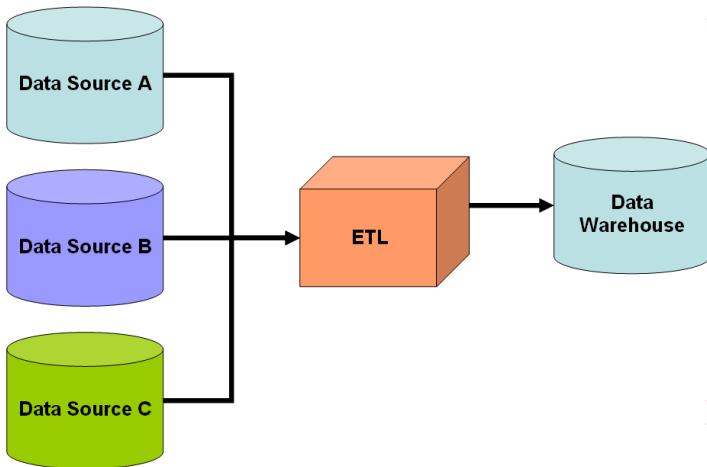


# Data Cleaning – Inconsistencies

- Correct inconsistent data: use domain knowledge or expert decision.
- Inconsistent: containing discrepancies in codes or names, e.g.
  - Age = “42”, Date Birth = “03/07/2010”
  - Was rating “1, 2 or 3”, now rating “A, B or C”
- Inconsistent data are handled by:
  - Manual Corrections: tedious and expensive
  - Develop routines to detect inconsistencies

# Data Integration

- Data integration involves combining data residing in different sources and providing users with a unified view of these data.



- Detecting and resolving data value conflicts:
  - Different scales, metric (feet or meters).
  - Entity identification problem, some attributes representing a given concept may have different names in different databases.  
Patients: “Bill”, “William”, “B.”
- Removing duplicates and redundant data

# Data Transformation

- Normalization, where the attribute data are scaled so as to fall within a small specified range, such as [-1.0, 1.0], or [0, 1.0]
  - Min-Max normalization, linear transformation of the original data based on the minimum and maximum values of an attribute

$$x' = nuevo_{min_A} + (nuevo_{max_A} - nuevo_{min_A}) \frac{x - min_A}{max_A - min_A}$$

- Z-Score normalization, adjusts the data from the initial distribution to a normal  
Also called “standarization”

$$x' = \frac{x - \mu_A}{\sigma_A}$$

# Data Transformation

- Normalization by decimal scale, normalizes by moving the decimal points of the values of attribute A

$$x' = \frac{x}{10^j}$$

where  $j$  is the smallest integer such that  $\max(|x'|) < 1$

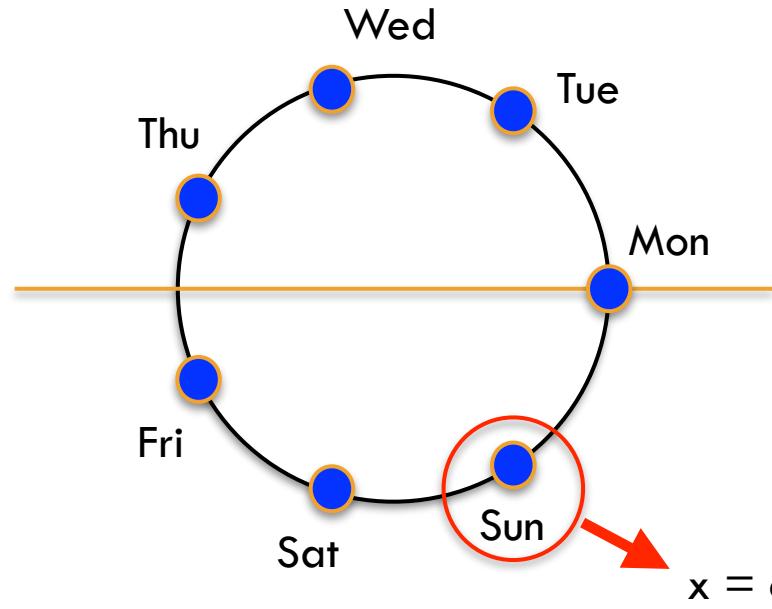
# Data Transformation

- Dummy Coding: categorical predictor variables cannot be entered directly into a regression model and be meaningfully interpreted
- Dichotomous variables from categorical variables are created → **dummy coding.**

	<b>Dept</b>	<b>FamilyS</b>	<b>Biology</b>
Family Studies	1	1	0
Biology	2	0	1
Business	3	0	0

# Data Transformation

- Cyclic categories
  - Example: Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday
  - Option 1: **dummy coding** (7 dummy variables, one can be removed)
  - Option 2: cyclic coding:



N categories -> N angles

$$\text{angle } i = i \cdot (360^\circ/N)$$

$$\text{Mon: angle} = 0^\circ$$

$$\text{Wed: angle} = 2 \cdot (360^\circ/7)$$

**2 variables x, y**

$$x(i) = \cos(\text{angle } i), y(i) = \sin(\text{angle } i)$$

$$x = \cos(6 \cdot 360^\circ / 7), y = \sin(6 \cdot 360^\circ / 7)$$

# Data Transformation

- Cyclic categories
  - Example: Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday
    - Option 1: **dummy coding** (7 dummy variables, one can be removed)
    - Option 2: cyclic coding:

**Warning: usually numerical libraries such as bumpy work with angles measured in radians, not degrees**

Thus we have to change degrees to radians:

$$\text{angle in radians} = \text{angle in degrees} \cdot \pi / 180$$

# Data Transformation: Synthetic Variables

- New attributes, for what and how?
  - Add / replace attributes makes it easier for algorithms to analyze the dataset
  - If we can combine attributes through some interesting expression, we get that the algorithm does not have to discover that expression
    - We can create an attribute area from height and width
    - Knowledge from experts is required

$$NPI = [0.2 \times S] + N + G, \text{ where}$$

S =Size, N=#nodes (0 =1, 1-3 = 2, >3 = 3), G=Grade (1, 2 o 3)

# False Predictor

- **Be careful not to include False Predictors. Find and eliminate them.**

A false predictor is a variable that is **strongly correlated** with the output class, but that is not available in a realistic prediction scenario.

- An expert in a modeling domain can spot when false predictor is buried among the input variables, because the model will be performing better than could be expected given the uncertain nature of the task.
- If the results are too good to be true, you probably have found false predictors.

# Regression and Classification Basics

José R. Dorronsoro

Dpto. de Ingeniería Informática, Escuela Politécnica Superior  
Universidad Autónoma de Madrid  
28049 Madrid, Spain

# Dónde estamos?

- ▶ Introducción
- ▶ Preprocesado de datos
- ▶ Reducción de la dimensionalidad
- ▶ **Regresión Lineal y Logística**
- ▶ Redes Neuronales
- ▶ Máquinas de vectores soporte
- ▶ Conjuntos de clasificadores y árboles de decisión
- ▶ Deep Learning
- ▶ Clustering
- ▶ ...

# Modeling Basics

## 1 Machine Learning Modeling Basics

- 2 Basic Regression
- 3 Bias, Variance and Cross Validation
- 4 Data and Model Analysis
- 5 Basic Classification
- 6 Logistic Regression
- 7 Practical Classification
- 8 Nearest Neighbor Classification

# What Is Machine Learning (ML)?

- ▶ Lofty definition: make machines learn!!!
  - ▶ Have to make “machines” and “learn” more precise
- ▶ The machines of ML: mathematical input–output processes that lend themselves to some form of (numerical) parameterization
- ▶ The learning process: adjust the machine’s parameters until a goal is reached
- ▶ New thing: “goal”?
  - ▶ At first sight, get something done
  - ▶ Ultimately, to minimize some error measure
- ▶ Summing things up: a ML process tries to find a concrete mathematical/algorithmic **input–output parameterized transformation** that **minimizes an error measure** by iteratively **adjusting the transformation’s parameters**

# Where Lies ML?

- ▶ In the middle of a possibly long process chain
- ▶ Before ML starts we must
  - ▶ Go from **raw to organized** data: accesing, gathering, cleaning, formatting, ...
  - ▶ Go from **organized to** (potentially) **informative** data: extracting basic and derived features
- ▶ After ML finishes and we have a model, we must perform
  - ▶ Outcome **evaluation**: how good/actionable the model is
  - ▶ Outcome **exploitation**: collect, organize, act
  - ▶ **Individual model maintenance**: monitor performance, tune hyper-parameters
  - ▶ **Modeling life cycle maintenance**: discard old models, introduce new ones and **communicate** our work/results
- ▶ ML is in the middle of the global process chain

# Supervised/Unsupervised Models

- ▶ Model types: **supervised, unsupervised**
- ▶ Supervised models:
  - ▶ Targets  $y^p$  are known and the model tries to predict or estimate them
  - ▶ These known targets guide, or **supervise**, model building
  - ▶ Main emphasis here
- ▶ Unsupervised models:
  - ▶ There are no predetermined or supervising outputs
  - ▶ But nevertheless the model is supposed to learn relations or find structure in the data
  - ▶ Sometimes as a first step towards a supervised model

# Regression and Classification

- ▶ Problems (usually) to be solved by models: regression, classification
- ▶ Patterns come in pairs  $(x, y)$ 
  - ▶  $x$ : inputs, predictors, features, independent variables
  - ▶  $y$ : target, response, dependent variable; numerical in regression, class labels in classification
- ▶ **Regression:** the desired output  $y$  is regressed into the inputs  $x$  to derive a model  $\hat{y} = f(x)$ 
  - ▶ We want  $y \simeq \hat{y}$  so having  $y - \hat{y}$  "small" is the natural goal
- ▶ **Classification:** inputs are derived from several classes  $C_1, \dots, C_K$ , to which labels  $\ell_k$  are assigned
  - ▶ The model now assigns a label  $\ell(x)$  to an input  $x$
  - ▶ If  $x$  is derived from  $C_k$  we want to have  $\ell(x) = \ell_k$
  - ▶ Here having  $\ell(x) - \ell_k$  "small" may not make sense

# The Boston Housing Problem

- ▶ This is a first “toy” regression problem
- ▶ We want to estimate the median of house values over an area from some information about it which we believe relevant
- ▶ Features  $x$ : several real estate-related variables of Boston areas
  - ▶ CRIM: per capita crime rate by town
  - ▶ RM: average number of rooms per dwelling
  - ▶ NOX: nitric oxides concentration (parts per 10 million)
  - ▶ AGE: proportion of owner-occupied units built prior to 1940
  - ▶ LSTAT: % lower status of the population
  - ▶ ...
- ▶ Target  $y$ : MEDV, median value of owner-occupied homes in \$1,000's

# How to Build Regression Models

- ▶ In general we have a sample  $S = \{x^p, y^p\}, 1 \leq p \leq N$ , with  $x^p \in \mathbf{R}^d$  the **features** and  $y^p$  the **targets**
- ▶ We want to build a model  $\hat{y} = f(x)$  so that  $\hat{y}^p = f(x^p) \simeq y^p$ ; i.e., we want to **regress**  $y$  to the  $x$
- ▶ The concrete  $f$  is chosen within a certain family  $\mathcal{F}$ 
  - ▶ Examples here: linear regression, multilayer perceptrons (MLPs), SVMs
  - ▶ And also: Random Forests (RF), Gradient Boosting (GB), Nearest Neighbor (NN)
- ▶ Natural option to ensure  $f(x^p) \simeq y^p$ : choose  $f$  to minimize the sample mean square error (MSE)

$$\widehat{e}(f) = \widehat{e}_S(f) = \frac{1}{2N} \sum_{p=1}^N (y^p - f(x^p))^2$$

- ▶ Thus, the model we select is  $\widehat{f} = \widehat{f}_S = \arg \min_{f \in \mathcal{F}} \widehat{e}_S(f)$

# Model Parameterization

- ▶ Usually individual models are selected through (ideally optimal) **parameter sets**
  - ▶ The parameters (weights)  $W \in R^M$  select a concrete  $f$  in  $\mathcal{F}$
- ▶ **Parametric** models have a fixed functional form  $f(x) = f(x; W)$
- ▶ Simplest example: linear regression, where  $M = d + 1$  and  $W = (w_0, w)$

$$f(x; w_0, w) = w_0 + \sum_{j=1}^d w_j x_j = w_0 + w \cdot x$$

- ▶ **Semi-parametric** models also use weights but without a predefined functional form; MLPs but also RF or GBR
- ▶ **Non parametric** models do not use weights nor follow any broad functional form; Nearest Neighbor models

# Model Estimation as Error Minimization

- For a parametric or semiparametric  $f(x; W)$  we can write  
 $\hat{e}_S(f) = \hat{e}_S(W)$
- The problem to solve becomes

$$\hat{W}^* = \hat{W}_S^* = \arg \min_W \hat{e}_S(f(\cdot; W)), \text{ i.e., } \hat{e}_S(\hat{W}^*) \leq \hat{e}_S(W) \forall W$$

- In linear regression

$$\hat{e}(w_0, w) = \frac{1}{2N} \sum_{p=1}^N (y^p - w_0 - w \cdot x^p)^2$$

which ends up in a simple **quadratic form**

- The regression problem reduces to **minimize**  $\hat{e}_S(W)$ 
  - Something in principle well understood in mathematical optimization

# Regression Basics

1 Machine Learning Modeling Basics

2 Basic Regression

3 Bias, Variance and Cross Validation

4 Data and Model Analysis

5 Basic Classification

6 Logistic Regression

7 Practical Classification

8 Nearest Neighbor Classification

# Regression Assumptions

- ▶ **Key assumption:**  $x$  and  $y$  are related as  $y = \phi(x) + n$  where
  - ▶  $\phi(x)$  is the **true** underlying function
  - ▶  $n$  is **additive noise** with 0 mean and finite variance  $\sigma_N^2$
- ▶ Our sample is just a particular instance of a deeper **sample generation process**
- ▶ Thus  $x, n$  are produced by **random variables**  $X, N$ 
  - ▶ And so is  $y$ , given by  $Y = \phi(X) + N$
- ▶ Moreover,  $X$  and  $N$  are **independent distributions**
- ▶ These assumptions are basic in what follows
- ▶ We should check our final models verify them

# Linear Models

- ▶ Assuming  $x \in \mathbf{R}^d$ , the basic linear model is

$$f(x) = w_0 + \sum_1^d w_i x_i = w_0 + w \cdot x$$

- ▶  $w_0$  complicates notation and, to drop it, we center  $x$  and  $y$  so that  $E[x_i] = E[y] = 0$ ; then  $w_0 = 0$
- ▶ Then we are left with the simpler homogeneous model  
 $f(x) = w \cdot x$
- ▶ In practice we will always **normalize**  $x$ , for instance to have 0 mean and 1 standard deviation (std) on each feature
  - ▶ But not  $y$  if we may help it
- ▶ But: how do we find  $w$ ?

# 1-dimensional Linear Regression (LR)

- ▶ Assume that features  $X$  and target  $Y$  are **centered**, i.e., have 0 means
- ▶ For 1-dimensional patterns  $x$  the LR model then becomes

$$f(x) = w x$$

- ▶ And the error is then the function  $e(w)$

$$\begin{aligned}\widehat{e}(w) &= \frac{1}{2N} \sum_{p=1}^N (w x^p - y^p)^2 = \frac{1}{2N} \sum_p (w^2 (x^p)^2 - 2x^p y^p w + (y^p)^2) \\ &= w^2 \left( \frac{1}{2N} \sum_p (x^p)^2 \right) - w \left( \frac{1}{N} \sum_p x^p y^p \right) + \frac{1}{2N} \sum_p (y^p)^2\end{aligned}$$

- ▶ The problem has obviously a minimum  $w^*$
- ▶ To find it we just solve  $\widehat{e}'(w) = 0$

## Solving $\widehat{e}'(w) = 0$

- To compute  $\widehat{e}'(w)$  we have

$$\widehat{e}'(w) = w \left( \frac{1}{N} \sum_p (x^p)^2 \right) - \left( \frac{1}{N} \sum_p x^p y^p \right)$$

- The optimal  $w^*$  solves  $\widehat{e}'(w) = 0$  and is given by

$$w^* = \frac{\frac{1}{N} \sum_p x^p y^p}{\frac{1}{N} \sum_p (x^p)^2} = \frac{\frac{1}{N} X \cdot Y}{\frac{1}{N} X \cdot X} = \frac{\frac{1}{N} X \cdot Y}{\text{var}(x)} = \frac{1}{\text{var}(x)} \text{covar}(x, y)$$

where  $X$  and  $Y$  denote the  $N$  dimensional vectors  $(x^1, \dots, x^N)^t$ ,  $(y^1, \dots, y^N)^t$

# General Linear Regression

- ▶ Assume again that  $X$  and  $Y$  are centered
- ▶ The LR model becomes now  $f(x) = \sum_1^d w_i x_i = w \cdot x$
- ▶ If  $Y$  is the  $N \times 1$  **target** vector and we organize the sample  $S$  in a  $N \times d$  **data matrix**  $X$ , the sample mse is given by

$$\begin{aligned}\hat{e}(w) &= \frac{1}{2N} \sum_p (w \cdot x^p - y^p)^2 = \frac{1}{2N} (Xw - Y)^t (Xw - Y) \\ &= \frac{1}{2N} (w^t X^t X w - 2w^t X^t Y + Y^t Y)\end{aligned}$$

- ▶ Now we have to solve  $\nabla \hat{e}(w) = 0$ , i.e.,  $\frac{\partial \hat{e}}{\partial w_i}(w) = 0$
- ▶ It is easy to see that

$$\nabla \hat{e}(w) = \frac{1}{N} X^t X w - \frac{1}{N} X^t Y = \widehat{R} w - \widehat{b}$$

# Solving the Linear Equations

- Thus, the optimal  $\hat{w}^*$  must solve the **normal equation**  
 $\hat{R}\hat{w} - \hat{b} = 0$ , where we recall that

$$\hat{R} = \frac{1}{N}X^tX, \quad \hat{b} = \frac{1}{N}X^tY$$

- Over the original, non-centered data matrix we have

$$\hat{R} = \frac{1}{N}(X - \bar{X})^t(X - \bar{X});$$

i.e.,  $\hat{R}$  is the **sample covariance matrix**

- If  $\hat{R}$  is **invertible**, we just solve the linear system  $\hat{R}\hat{w} - \hat{b} = 0$
- And obtain the sample-dependent optimal  $\hat{w}^*$  as

$$\hat{w}^* = \hat{R}^{-1}\hat{b} = (X^tX)^{-1}X^tY$$

# Finding Optimal Models

- ▶ Computing the covariance matrix has a  $O(N \times d^2)$  cost and invert it has a  $O(d^3)$  cost
  - ▶ For big data problems it may not possible to solve analytically the normal equation  $\nabla \hat{e}(w) = 0$
- ▶ The simplest numerical alternative is **gradient descent**:
  - ▶ Starting from some random  $w^0$  we iteratively compute

$$w^{k+1} = w^k - \rho_k \nabla \hat{e}(w^k) = w^k - \frac{\rho}{n_B} \left( \hat{X}_B^t \hat{X}_B w^k - \hat{X}_B^t Y \right)$$

- over a **mini-batch**  $B$  with  $n_B$  samples
- ▶ Component wise:  $w_i^{k+1} = w_i^k - \rho_k \frac{\partial \hat{e}}{\partial w_i}(w^k)$
- ▶  $\rho_k$  is the **learning rate**
- ▶ If  $w^k \rightarrow w^*$ , then  $\nabla \hat{e}(w^*) = 0$ 
  - ▶ Since our problems have obviously minima, this should be enough

# Measuring Model Fit

- ▶ First option: **Root Square Error**  $RSE = \sqrt{\frac{1}{N} \sum (y^p - \hat{y}^p)^2}$
- ▶ OK, but how good is this? We must always have a **base model** to benchmark our results
- ▶ Simplest “model”: a constant  $w_0$ , which yields the mean  $\bar{y} = \frac{1}{N} \sum_1^N y^p$ , with square error

$$\frac{1}{N} \sum (y^p - \bar{y})^2 = \text{Var}(y)$$

- ▶ We can compare our model against this base by computing

$$\frac{\sum (y^p - \hat{y}^p)^2}{\sum (y^p - \bar{y})^2} = \frac{RSE^2}{\text{Var}(y)}$$

- ▶ The widely used  $R^2$  coefficient is simply  $R^2 = 1 - \frac{RSE^2}{\text{Var}(y)}$

# Regularization

- Our regression solution  $\hat{w}^* = (X^t X)^{-1} X^t Y$  won't work if  $X^t X$  is not invertible
  - For instance, when some features are correlated
- We could fix this working instead with  $X^t X + \alpha I$  for some  $\alpha > 0$
- To make this practical, one can show that  
 $\hat{w}^* = (X^t X + \alpha I)^{-1} X^t Y$  minimizes

$$e_R(w) = \frac{1}{2N} \sum_p (y^p - w \cdot x_p^p)^2 + \frac{\alpha}{2} \|w\|^2,$$

- This is the **Ridge Regression** problem
  - Our first example of **regularization**, a key technique in Machine Learning
  - **All ML models must be regularized in some way**
- Important issue: how to find the right choice for  $\alpha$ ?

# Takeaways on Linear Regression

1. We introduced **supervised** models
2. We have reviewed the essentials of the **linear regression model** (always the first thing to try)
3. We have considered model estimation as a problem on **error minimization**
4. We have seen how to build linear models **analytically and numerically**
5. We have defined how to **measure model fit**
6. We have introduced **regularization**

# Bias, Variance and Cross Validation

1 Machine Learning Modeling Basics

2 Basic Regression

3 Bias, Variance and Cross Validation

4 Data and Model Analysis

5 Basic Classification

6 Logistic Regression

7 Practical Classification

8 Nearest Neighbor Classification

# Sample Dependence

- ▶ Important: **everything is sample dependent** for if we change  $S$  we get a different model

- ▶ We thus write  $\hat{f}_S(x)$

- ▶ Therefore, for different samples  $S, S'$  we want our models to verify

$$\hat{f}_S(x) \simeq \hat{f}_{S'}(x)$$

- ▶ That is, we want our models to have small **variance** with respect to sample changes

- ▶ Intuitively this can be achieved using simple models with few parameters

- ▶ But we also want that  $\hat{f}_S \simeq \phi(x)$

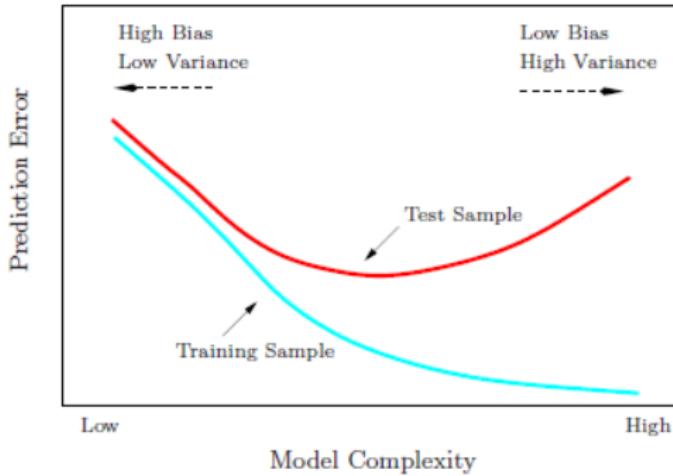
- ▶ That is, we want our models to have a small **bias**, i.e., get as close as possible to the “true” model  $\phi$

- ▶ Intuitively this can be achieved using highly flexible models with many parameters

- ▶ But obviously both goals are contradictory to a large extent

# The Bias–Variance Tradeoff

- There is thus a **tradeoff** between bias (low for complex models) and variance (low for simple models)



Taken from *Hastie et al.*, p. 38

# Evaluating Expected Performance

- ▶ Over a single  $S$ , we apply **Cross Validation** (CV), where we
  - ▶ Randomly split the sample  $S$  in  $M$  subsets  $S_1, \dots, S_M$
  - ▶ Work with  $M$  **folds**: pairs  $(S_m, S_m^c)$ , with

$$S_m^c = S - S_m = \cup_{i \neq m} S_i$$

- ▶ Build  $M$  different models **using the  $S_m^c$  as training subsets**
- ▶ Compute their errors  $e_m$  on the folds' **validation subsets  $S_m$**
- ▶ Use these **errors' average** as a first estimate of the true model performance
- ▶ CV can and **must be used** in any model building procedure
- ▶ We will also use CV to find an **optimal value** for the hyper-parameter  $\alpha$  in Ridge Regression

# Grid Hyper-parameter Selection

- ▶ Consider for Ridge regression a hyper-parameter range  $[0, A]$ 
  - ▶  $\alpha = 0$ : no penalty and, thus, small bias but possibly high variance
  - ▶  $\alpha = A$ : large penalty and, thus, small variance but high bias
- ▶ Select an  $L + 1$  point **grid**  $[\alpha_0, \dots, \alpha_L]$ 
  - ▶ For instance a uniform one  $\alpha_\ell = \ell \frac{A}{L}$ ,  $\ell = 0, 1, \dots, L$
- ▶ Build  $M$  **folds**: pairs  $(S_m, S_m^c)$  and for each  $\alpha_\ell$ 
  - ▶ Train  $M$  Ridge models on the  $S_m^c$  using the hyper-parameter  $\alpha_\ell$
  - ▶ Average their  $M$  validation errors  $e_m$  on the  $S_m$  to get the CV error  $e(\alpha_\ell)$  for  $\alpha_\ell$
- ▶ Finally choose the (hopefully) optimal hyper-parameter  $\alpha^*$  as

$$\alpha^* = \arg \min_{0 \leq \ell \leq L} e(\alpha_\ell)$$

- ▶  $\alpha^*$  gives the model with the **best expected generalization among all possible  $\alpha$  choices**

# Takeaways on Bias, Variance and CV

1. We have stressed that **any model estimation is sample-dependent** and that this has to be controlled
2. We have introduced the **bias** and **variance** as the two key components of any model error
3. We have discussed **bias-variance trade-off**
4. We have introduced **Cross Validation** here as a tool to estimate a **model's generalization performance**
5. We have also introduced **Cross Validation** as a tool to estimate a **model's hyper-parameters**

# Data and Model Analysis

- 1 Machine Learning Modeling Basics
- 2 Basic Regression
- 3 Bias, Variance and Cross Validation
- 4 Data and Model Analysis**
- 5 Basic Classification
- 6 Logistic Regression
- 7 Practical Classification
- 8 Nearest Neighbor Classification

# And So What?

- ▶ Key question: what are models for?
  - ▶ First answer: to be used to derive new predictions
  - ▶ Better answer: to extract knowledge and to make inference on the underlying problem
- ▶ In this light, LR models are simple, perhaps not too powerful, but certainly useful
  - ▶ They are the first tool to apply in (almost) any problem analysis
- ▶ Some questions are easier to answer for them:
  - ▶ Which variables do influence the target and which do not?
  - ▶ What are the strongest predictive variables?
  - ▶ Are there related/redundant variables?
  - ▶ Is the relationship actually linear?

# Issues with LR

- ▶ Before building any model we must perform a prior **data analysis** to keep under control important issues:
  - ▶ **Collinearity**: predictor variables that are redundant
  - ▶ **Outliers**: points  $(x^p, y^p)$  with a “normal” pattern  $x$  but an unlikely target value  $y^p$ , or viceversa
- ▶ And after a model is built we must check if **its results agree with its assumptions**
  - ▶ **Linearity** of the response–predictor relationships: if not, the LR will be poor
  - ▶ **No correlation of error terms**, i.e. our basic model assumption does hold
  - ▶ **Homoscedasticity**, i.e., residuals are the same across all features and target

# Detecting and Handling Data Issues

- ▶ Before **any** model is built we **must** try detect possible data inconsistencies and/or redundancies
- ▶ Feature collinearity: look at least at the correlation matrix
- ▶ Analyze feature–target scatterplots; if possible, look also at the two–predictor scatterplots (though there are  $d(d - 1)/2$  of them)
- ▶ Outliers: will cause  $(x^p, y^p)$  to be far from the line fit or the residual to be out of range
  - ▶ Can detect them with box plots
- ▶ We consider all this over the Boston Housing dataset and notebook

# Housing: First Conclusions on the Data

- ▶ Collinearity: some predictor variables may be redundant
  - ▶ AGE–DIS: proportion of units built prior to 1940 and weighted distances to five employment centres
  - ▶ RAD–TAX: accessibility to radial highways and full-value property-tax rate
  - ▶ NOX–INDUS
- ▶ Outliers: points  $(x^p, y^p)$  with a normal pattern  $x$  but an unlikely target value  $y^p$ 
  - ▶ ???
- ▶ Something happens with the high price houses
  - ▶ It seems that the price is capped at \$ 50K
  - ▶ Better remove them from our first model

# Detecting and Handling Model Issues

- ▶ After the model is built we check whether it supports the basic LR assumptions
- ▶ Linearity: a residual plot should not have any structure
- ▶ Uncorrelated error terms: residuals do not change rather smoothly
- ▶ Error histograms should be symmetric and sharp at 0
- ▶ Homoscedasticity: residuals are the same across the target
- ▶ **Always address these possible problems:** if not, we may be fooling ourselves with an untenable model
- ▶ Let's build LR models over the Boston Housing data

# Housing: First Conclusions on the Linear Model

- ▶ Linearity of the response-predictor relationships: not bad
  - ▶ If perfect fit,  $y$  and  $\hat{y}$  in diagonal; here in near diagonal
- ▶ Correlation of residuals: there seems to appear for large targets
  - ▶ Perhaps we should think about two separate models?
- ▶ Homoscedasticity, i.e., constant variance of residuals
  - ▶ Again, perhaps for small targets but clearly not for large targets
- ▶ Build perhaps a second model?

# Takeaways on Data and Model Analysis

1. Before any model building we must **analyze and understand our data**
2. We must also understand the **assumptions** our model implies on the data
  - ▶ If they aren't true the model won't be very good
3. This must be checked **after the model is built**
4. LR models are simple but their assumptions are of interest to any other model
5. LR are the first models to build, to better understand the problem and its data and to have a **benchmark**
6. And
  - ▶ Always **tune the hyperparameters** for our models
  - ▶ Always try out **many different models**
  - ▶ Always explore several **feature representations** for our data

# Basic Classification

- 1 Machine Learning Modeling Basics
- 2 Basic Regression
- 3 Bias, Variance and Cross Validation
- 4 Data and Model Analysis
- 5 Basic Classification**
- 6 Logistic Regression
- 7 Practical Classification
- 8 Nearest Neighbor Classification

# Regression vs Classification

- ▶ Recall that in regression we have numerical continuous targets  $y$  and want our predictions  $\hat{y}$  to be as close to  $y$  as possible
- ▶ But in classification we have a finite number of labelled targets for which “selection by closeness” doesn’t make sense
- ▶ Natural alternative: select the **most probable label given the pattern** we have just received
  - ▶ The concrete labels used for targets do not matter anymore
  - ▶ Model learning should thus be “target” agnostic
  - ▶ And good probability estimates should be quite useful
- ▶ Let’s analyze this in an example

# A First Problem: Pima Indian Diabetes

- ▶ We want to diagnose whether a person may have diabetes from some clinical measures
- ▶ Features  $x$ : clinical measures
  - ▶ numPregnant
  - ▶ bloodPress
  - ▶ massIndex
  - ▶ age ...
- ▶ Target  $y$ : 0 (no diabetes), 1 (diabetes)
- ▶ Clear goal but perhaps too radical
- ▶ Better: try to estimate **the probability  $P(1|x)$  of having diabetes depending on the features  $x$  we measure**

# Classification Setup

- ▶ We have random patterns  $\omega$  from  $M$  classes,  $C_1, \dots, C_M$
- ▶ Over each pattern we “measure”  $d$  features  $x = x(\omega) \in \mathbb{R}^d$ 
  - ▶  $x$  inherits the randomness in  $\omega$  and becomes a **random variable**
- ▶ A  $\omega$  has a **prior probability**  $\pi_m$  of belonging to  $C_m$
- ▶ Inside each class  $C_m$  there is a **conditional class density**  $f(x|m)$  that “controls” the appearance of a given  $x$
- ▶ The  $\pi_m$  and  $f(x|m)$  determine the **posterior probability**  $P(m|x)$  that  $x$  comes from class  $C_m$
- ▶ **Intuition:** we should assign  $x$  to the class with the **largest**  $P(m|x)$ , that is, work with the classifier

$$\delta(x) = \arg \max_m P(m|x)$$

# The Obviously Optimal Classifier

- ▶ It can be shown that  $P(m|x) = \frac{\pi_m f(x|m)}{f(x)}$
- ▶ Thus, we should decide according to a **classifier** function  $\delta_B$

$$\begin{aligned}\delta_B(x) &= \arg \max_m P(m|x) = \arg \max_m \frac{\pi_m f(x|m)}{f(x)} \\ &= \arg \max_m \pi_m f(x|m)\end{aligned}$$

- ▶ With some extra work we can show that this **Bayes Classifier**  $\delta_B$  defines an optimal solution (in some precise sense) of the classification problem
- ▶ But ... this doesn't look too practical for we do not know either  $\pi_m$  or (much harder)  $f(x|m)$
- ▶ We will focus on estimating **directly** the label generating distribution  $P(m|x)$

# Logistic Regression (LR)

- 1 Machine Learning Modeling Basics
- 2 Basic Regression
- 3 Bias, Variance and Cross Validation
- 4 Data and Model Analysis
- 5 Basic Classification
- 6 Logistic Regression
- 7 Practical Classification
- 8 Nearest Neighbor Classification

# Logistic Regression (LR)

- We assume

$$P(1|x) = P(1|x; w_0, w) = \frac{1}{1 + e^{-(w_0 + w \cdot x)}}$$

- Then  $0 \leq P(1|x) \leq 1$  for any  $x$
- We then have

$$P(0|x) = 1 - P(1|x) = \frac{e^{-(w_0 + w \cdot x)}}{1 + e^{-(w_0 + w \cdot x)}} = \frac{1}{1 + e^{w_0 + w \cdot x}}$$

- Notice that if  $w_0 + w \cdot x = 0$ ,  $P(1|x) = P(0|x) = 0.5$
- The ratio  $\frac{P(1|x)}{P(0|x)} = e^{w_0 + w \cdot x}$  is called the **odds** of  $x$  and its log the **log odds** or **logit**
- Thus, the basic assumption in LR is that the **logit is a linear function**  $w_0 + w \cdot x$  of  $x$
- We have the model  $f(x; w)$ ; we need a **loss** function  $L(w_0, w)$  to minimize for which we use the sample's **likelihood**

## Estimating $w_0^*, w^*$

- ▶ Assume a single sample  $x, y$  and two possible model coefficients  $w_0, w$  and  $w'_0, w'$
- ▶ Denoting by  $p = P(y|x; w_0, w)$  and  $p' = P(y|x; w'_0, w')$ , it is clear that we should prefer  $w_0, w$  if  $p > p'$  and  $w'_0, w'$  if not
  - ▶ In other words, we prefer the coefficients that give a **higher posterior probability** to the sample  $(x, y)$
- ▶ For an independent sample  $S = \{(x^p, y^p)\}$ , its joint probability under a posterior model  $p = P(y|x; w_0, w)$  is

$$P(Y|X; w_0, w) = \prod_{p=1}^N P(y^p|x^p; w_0, w)$$

- ▶ And, again, given two possible model coefficients  $w_0, w$  and  $w'_0, w'$ , we should prefer  $w_0, w$  iff

$$P(Y|X; w_0, w) > P(Y|X; w'_0, w')$$

## Sample's Likelihood

- ▶ Therefore, we can estimate the optimal  $w_0^*, w^*$  as

$$w_0^*, w^* = \arg \max_{w_0, w} P(Y|X; w_0, w)$$

- ▶ By the independence assumption we have

$$\begin{aligned} P(Y|X; w_0, w) &= \left\{ \prod_{y^p=1} P(1|x^p) \right\} \left\{ \prod_{y^p=0} P(0|x^p) \right\} \\ &= \prod_{p=1}^N P(1|x^p)^{y^p} P(0|x^p)^{1-y^p} \end{aligned}$$

with the last equality follows from

- ▶ If  $y^p = 1$ ,  $P(1|x^p) = P(1|x^p)^{y^p} = P(1|x^p)^{y^p} P(0|x^p)^{1-y^p}$ , and
- ▶ If  $y^p = 0$ ,  $P(0|x^p) = P(0|x^p)^{1-y^p} = P(1|x^p)^{y^p} P(0|x^p)^{1-y^p}$

# Max Log-Likelihood Estimation

- The log-likelihood of  $w_0, w$  given  $S$  is then

$$\begin{aligned}\ell(w_0, w; S) &= \log P(Y|X; w_0, w) \\ &= \sum_p \{y^p \log p(1|x^p) + (1 - y^p) \log p(0|x^p)\} \\ &= \sum_p y^p \log \frac{p(1|x^p)}{p(0|x^p)} + \sum_p \log p(0|x^p) \\ &= \sum_p y^p (w_0 + w \cdot x^p) - \sum_p \log(1 + e^{w_0 + w \cdot x^p})\end{aligned}$$

- We can thus estimate the optimal  $\hat{w}_0^*, \hat{w}^*$  as

$$\hat{w}_0^*, \hat{w}^* = \arg \min_{w_0, w} -\ell(w_0, w; S)$$

- Extra bonus:  $-\ell$  is a convex differentiable function of  $(w_0, w)$  and, thus, it is enough to solve  $\nabla \ell(w_0, w) = 0$

# Newton–Raphson Solution

- ▶ However,  $\nabla \ell(W) = \nabla \ell(w_0, w) = 0$  doesn't admit a closed form solution but only an iterative, numerical one
- ▶ We solve it the **Newton–Raphson** iterative method (equivalent here to Newton's method for minimization)
- ▶ Starting from a random  $W^0 = (w_0^0, w^0)$ , Newton's iterations are

$$W^{k+1} = W^k + (\mathcal{H}_\ell(W^k))^{-1} \nabla \ell(W^k)$$

- ▶  $\mathcal{H}_\ell(W^k)$  denotes the Hessian of  $\ell$  at  $W^k$  (which may or may not be invertible)
  - ▶ Everything is fine if the  $W^k$  are close enough to the optimum  $W^*$  but far away things may get tricky
- ▶ Just as before, we can add a regularization term  $\frac{\alpha}{2} \|w\|^2$
- ▶ The iterations in Logistic Regression are again typical of many of the model building methods used in Machine Learning

# Learning in ML

- The general approach to **learning** is usually the following:
  - A **model**  $f(x; W)$  is chosen
  - Given a sample  $S = \{(x^1, y^1), \dots, (x^N, y^N)\}$  and a loss function  $\ell(y, \hat{y})$ , we define a **sample dependent loss function**

$$L(W) = L(W|S) = \sum \ell(y^p, \hat{y}^p = f(x^p; W))$$

- $L(W)$  is often minimized from some  $W^0$  by **iterations**

$$W^{k+1} = W^k - \rho_k G(W^k, S)$$

with  $\rho_k$  a **learning rate** and  $G$  some vectorial function

- When  $G(W) = \nabla L(W)$  we have **gradient descent**
- When  $G(W) = H(W)^{-1} \nabla L(W)$  we obtain **Newton's method**
- In **batch learning** the entire sample  $S$  is used at each iteration
- **On-line or minibatch learning:** we use either a single patterns  $(x^p, y^p)$  or small subsample
- Several such procedures will appear here in the coming weeks

# Practical Classification

- 1 Machine Learning Modeling Basics
- 2 Basic Regression
- 3 Bias, Variance and Cross Validation
- 4 Data and Model Analysis
- 5 Basic Classification
- 6 Logistic Regression
- 7 Practical Classification**
- 8 Nearest Neighbor Classification

# True/False Positives/Negatives

- ▶ Consider a two class problem with labels  $y = 0, 1$
- ▶ We will call patterns with label 1 **positive** and those with label 0 **negative**
  - ▶ Usually the positive patterns are the interesting ones: sick people, defaulted loans, . . .
- ▶ Let  $\hat{y} = \hat{y}(x)$  the label predicted at  $x$ ; we say that  $x$  is a
  - ▶ **True Positive (TP)** if  $y = \hat{y} = 1$
  - ▶ **True Negative (TN)** if  $y = \hat{y} = 0$
  - ▶ **False Positive (FP)** if  $y = 0$  but  $\hat{y} = 1$
  - ▶ **False Negative (FN)** if  $y = 1$  but  $\hat{y} = 0$
- ▶ The standard way of presenting these data is through the **confusion matrix**

# The Confusion Matrix

- ▶ Standard layout

	P' (Predicted)	N' (Predicted)
P (Actual)	True Positive	False Negative
N (Actual)	False Positive	True Negative

- ▶ Other layouts:
  - ▶ Positives(with label 1) at bottom, as done in `confusion_matrix` of `sklearn`
  - ▶ Predicted values in rows, real values in columns

# Classifier Metrics

- ▶ The classifier **accuracy** is  $acc = \frac{TP+TN}{N}$
- ▶  $acc$  is the first thing to measure but it may not be too significant: if the number  $N_0$  of negatives is  $\gg N_1$ , the number of positives
  - ▶ The classifier  $\delta(x) = 0$  will have a high accuracy  $N_0/N \simeq 1$
  - ▶ But it will also be useless!!
- ▶ First variant: Precision, Recall
  - ▶ **Recall:**  $TP/(TP + FN)$ , i.e., the fraction of positives detected
  - ▶ **Precision:**  $TP/(TP + FP)$ , i.e., the fraction of true alarms issued
- ▶ Recall measures how many positive cases we recover, i.e., how **effective** is our method
- ▶ Precision measures the effort we need for that, i.e., its **efficiency**
- ▶ Ideal classifier: high recall, high precision (i.e., effective and efficient!!)

# What's New from Regression?

- ▶ Some things change from regression, some don't
- ▶ We should check feature correlations, if only to remove too similar features
- ▶ Important: **positive and negative-class feature histograms**
  - ▶ Scatter plots  $(x_i, y)$  are usually less informative
- ▶ The **bias-variance trade-off** is subtler in classification
- ▶ Accuracy, recall, precision are the usual model quality measures
- ▶ We use CV with **stratified folds** to estimate generalization performance
- ▶ We also use CV for hyperparameter estimation, as regularization will also be needed
  - ▶ In LR we should minimize  $-\ell(w_0, w; S) + \frac{\alpha}{2} \|w\|^2$

# How to Handle Posterior Probabilities

- If possible, we want **posterior probabilities** as model outputs better than labels
- Most models give them as pairs

$$(\hat{P}(0|x), \hat{P}(1|x)) = (\hat{P}(0|x), 1 - \hat{P}(0|x))$$

- In principle we would decide 1 if  $\hat{P}(1|x) > 0.5$  and viceversa, but this may be too crude
- It may be advisable to set a **decision threshold**  $\kappa$  and decide 1 if  $\hat{P}(1|x) > \kappa$  and 0 if  $\hat{P}(1|x) < \kappa$
- For **imbalanced** problems where  $\pi_0 \gg \pi_1$  (usually the interesting ones) we would have  $\hat{P}(1|x) \simeq 0$  for most  $x$ 
  - In this case we may choose a  $\kappa < 0.5$  and **suggest** 1 if  $\hat{P}(1|x) > \kappa$

# Takeaways on Basic Classification

1. We have introduced the classification problem as one of computing **posterior probabilities**
2. We have defined the **optimal Bayes classifier**
3. We have introduced **Logistic Regression** to estimate posterior probabilities, and the numerical minimization of its (minus) log-likelihood
4. We have introduced **accuracy, recall, precision** as first classification metrics
5. We have reviewed some practical issues in classification

# Nearest Neighbor Classification

- 1 Machine Learning Modeling Basics
- 2 Basic Regression
- 3 Bias, Variance and Cross Validation
- 4 Data and Model Analysis
- 5 Basic Classification
- 6 Logistic Regression
- 7 Practical Classification
- 8 Nearest Neighbor Classification

# Approximating the Bayes Classifier

- ▶ Starting with the approximation  $P(m|x) \simeq P(m|B_r(x))$ , Bayes formula gives

$$P(m|x) \simeq P(m|B_r(x)) = \frac{P(C_m \cap B_r(x))}{P(B_r(x))} = \frac{\pi_m P(B_r(x)|m)}{P(B_r(x))}$$

- ▶ Assume a sample with  $N$  patterns of which
  - ▶  $N_m$  are in  $C_m$
  - ▶  $k$  sample patterns are in  $B(x, r)$
  - ▶  $k_m$  samples in class  $C_m$  are in  $B(x, r)$
- ▶ We then have

$$\pi_m \simeq \frac{N_m}{N}, \quad P(B_r(x)) \simeq \frac{k}{N}, \quad P(B_r(x)|m) = \frac{k_m}{N_m}$$

# The $k$ -NN Classifier

- We can thus approximate  $P(m|x)$  as

$$P(m|x) \simeq \frac{k_m}{N_m} \frac{N_m}{N} \frac{1}{\frac{k}{N}} = \frac{k_m}{k}$$

- And the optimal  $\delta_B$  as

$$\delta_B(x) \simeq \arg \max_m P(m|B_r(x)) = \arg \max_m \frac{k_m}{k}$$

- We have thus arrived to the  **$k$ -Nearest Neighbor** classifier

$$\delta_k^{NN}(x) = \arg \max_m \frac{k_m}{k} = \arg \max_m k_m$$

- Thus  $\delta_{kNN}(x)$  assigns  $x$  to the class that has more patterns in  $N_k(x)$

## By the Way: $k$ -NN Regression

- ▶  $k$ -NN Classification assumes that a pattern should belong to the class with the closest patterns
- ▶  $k$ -NN Regression also relies on a similar assumption: **Predictors that are close should give predictions that are also close**
- ▶ In  $k$ -NN Regression we fix a number  $k$  of neighbors to be considered and for an input  $x$  set

$$\hat{y} = Y_{kNN}(x) = \frac{1}{k} \sum_{x^p \in N_k(x)} y^p$$

where  $N_k(x)$  denotes again the  $k$  sample points closest to  $x$

- ▶ **Weighted variants:** for instance,  
$$Y_k^w(x) = \frac{1}{C_k(x)} \sum_{x^p \in N_k(x)} \frac{1}{\|x^p - x\|} y^p$$
  - ▶  $C_k(x) = \sum_{x^p \in N_k(x)} \frac{1}{\|x^p - x\|}$  is a normalizing constant

## Some $k$ -NN Issues

- ▶ **Q1: How do we choose  $k$ ?** Using CV, of course, and balancing again the bias–variance tradeoff
  - ▶ Small variance with large  $k$ : if  $k = N$ ,  $k$ -NN regression returns the mean
  - ▶ Small bias with small  $k$ : if  $k = 1$  a very close point should give a very close prediction
  - ▶ But also large variance: the nearest point to  $x$  in another sample may have a quite different target or belong to another class
- ▶ **Q2: Is  $k$ -NN meaningful?** Yes but only if predictors that are close give predictions that are also close, and **provided that there are enough of them close by**
- ▶ This will not be easy because of the **curse of dimensionality**:
  - ▶ Even for low dimensions and large samples, **the sample space is essentially empty**
  - ▶ And for most problems, **there never will be close enough points**
  - ▶ Thus, to get  $k$  observations we may go too far away from  $x$  and the  $k$ -NN predictions will not be meaningful