

# Conjunto de clasificadores

# Guión

- ¿Qué es un conjunto de clasificadores? ¿Cómo construirlo?
- Bagging, Boosting, Random forests, class-switching, Gradient Boosting
- 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%

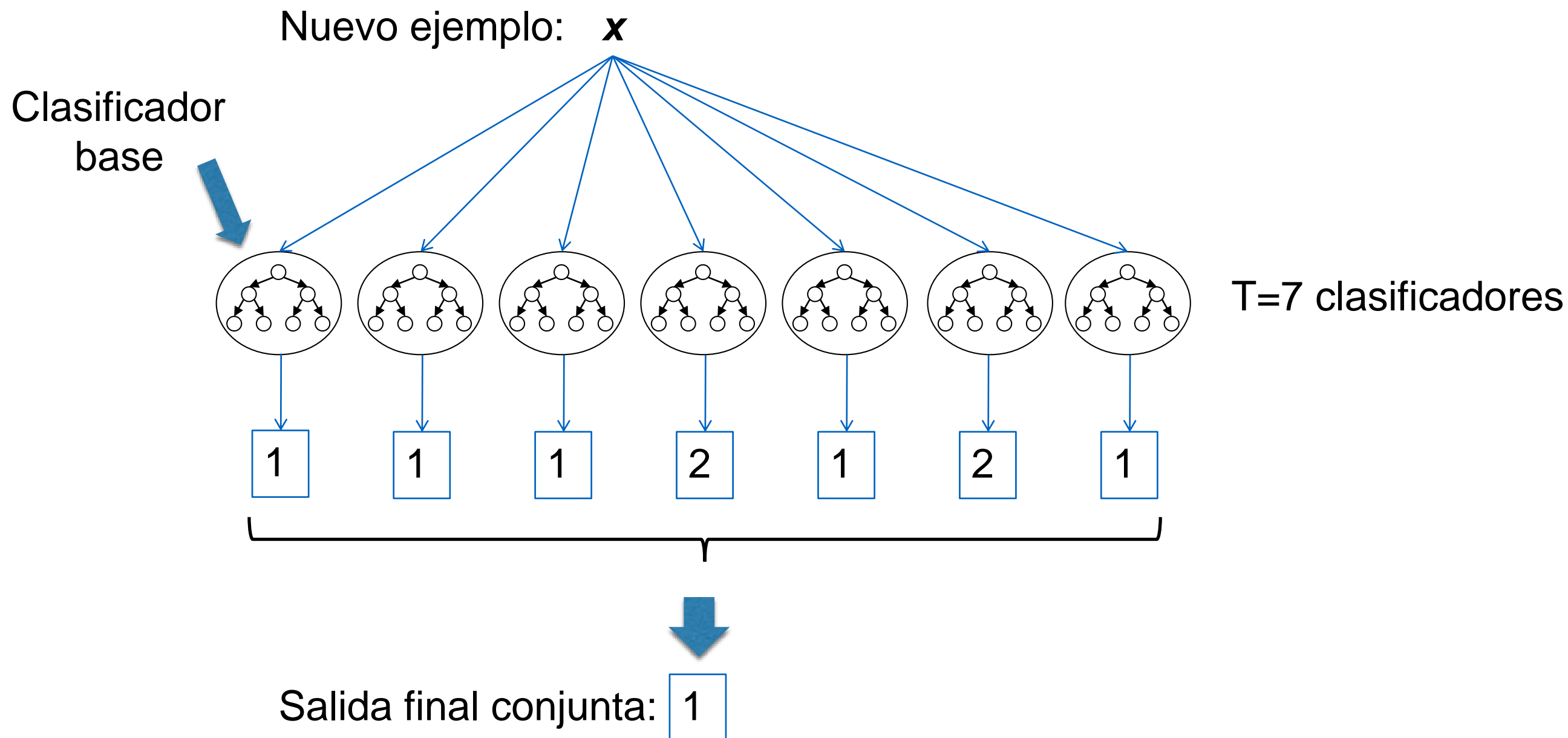
La probabilidad del jurado de acertar tiende a 100% al aumentar el numero de miembros del jurado



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 entrenan 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)

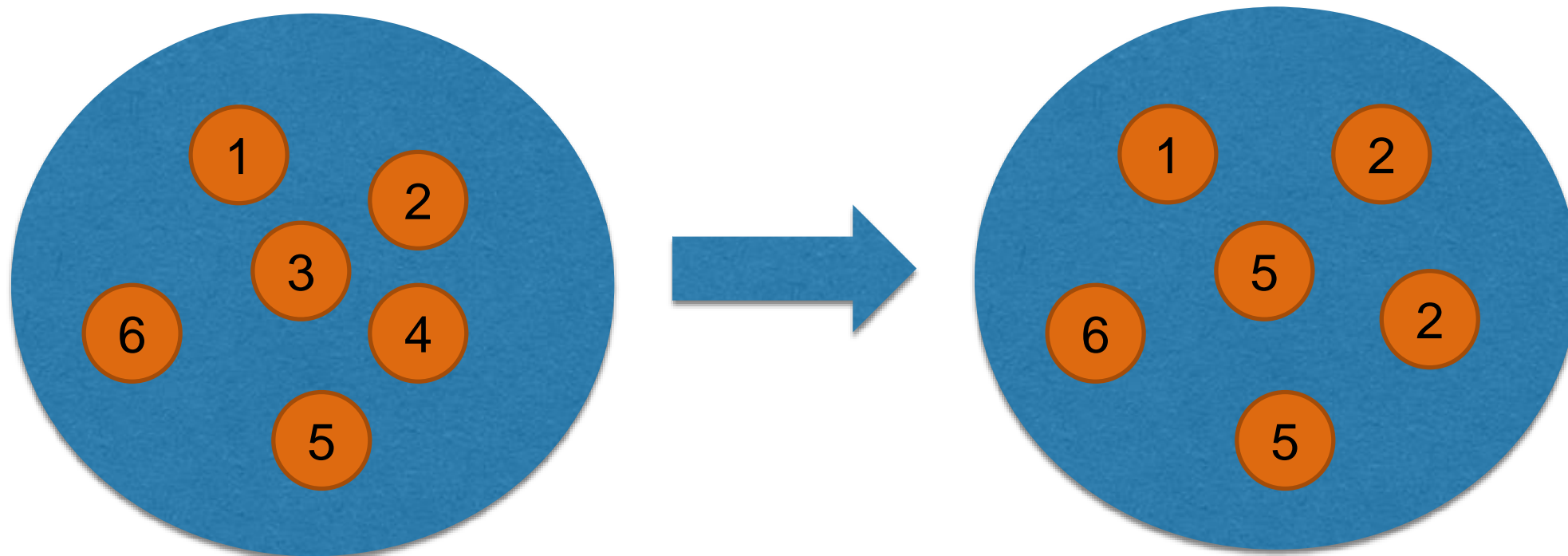
# ¿Cómo construirlos?

- 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*

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



6 extracciones con repetición

- Variantes:
  - Extraer  $M$  elementos con  $M \neq N$
  - Realizar extracciones sin reemplazamiento

# Bagging

## Input:

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

Ensemble size  $T$

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

2.      $\text{sample} = \text{BootstrapSample}(D)$

3.      $h_t = \text{TrainClassifier}(\text{sample})$

*Bootstrap*

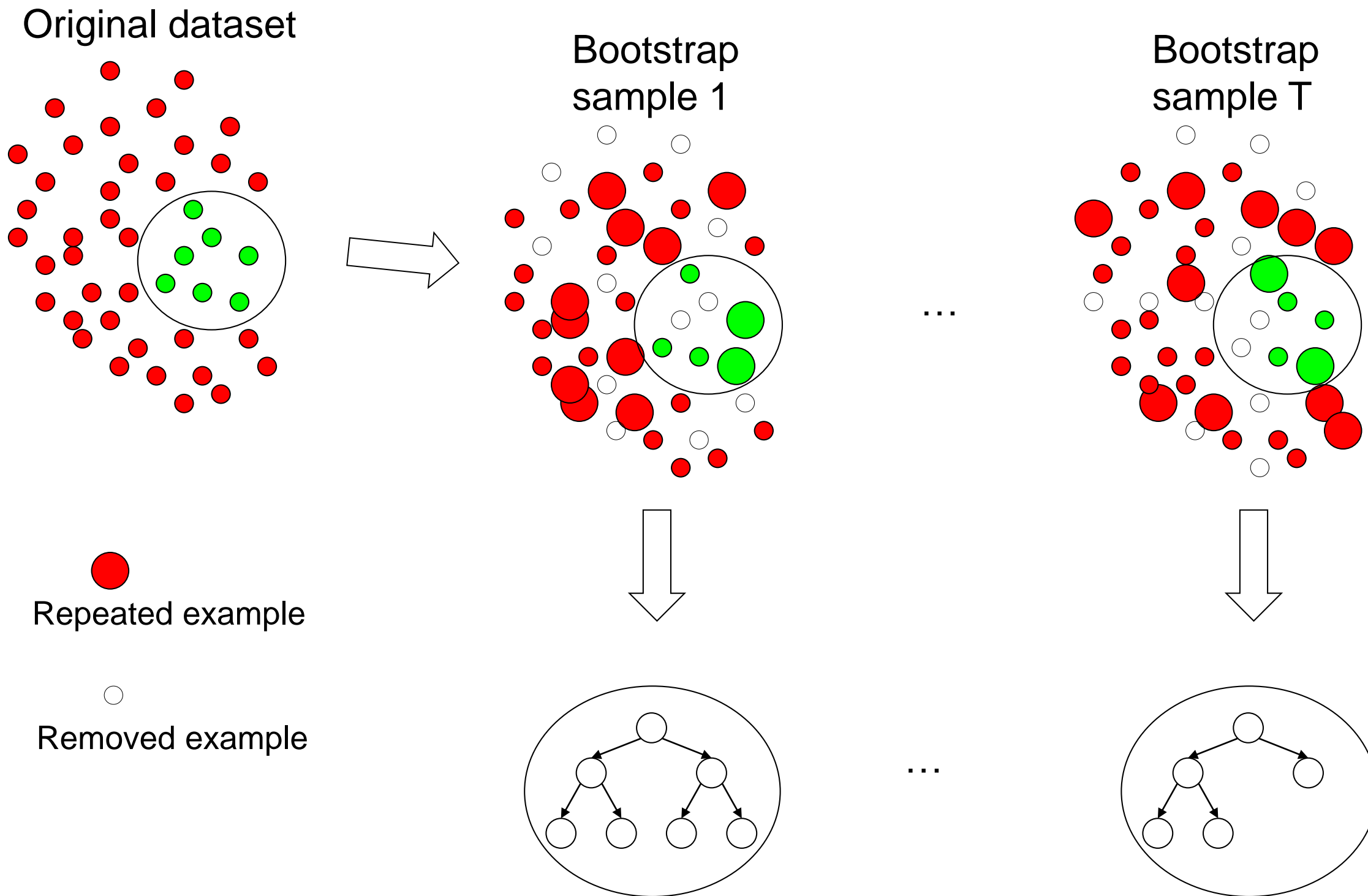
+

## Output:

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

*Agregación  
(por voto)*

# Bagging



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

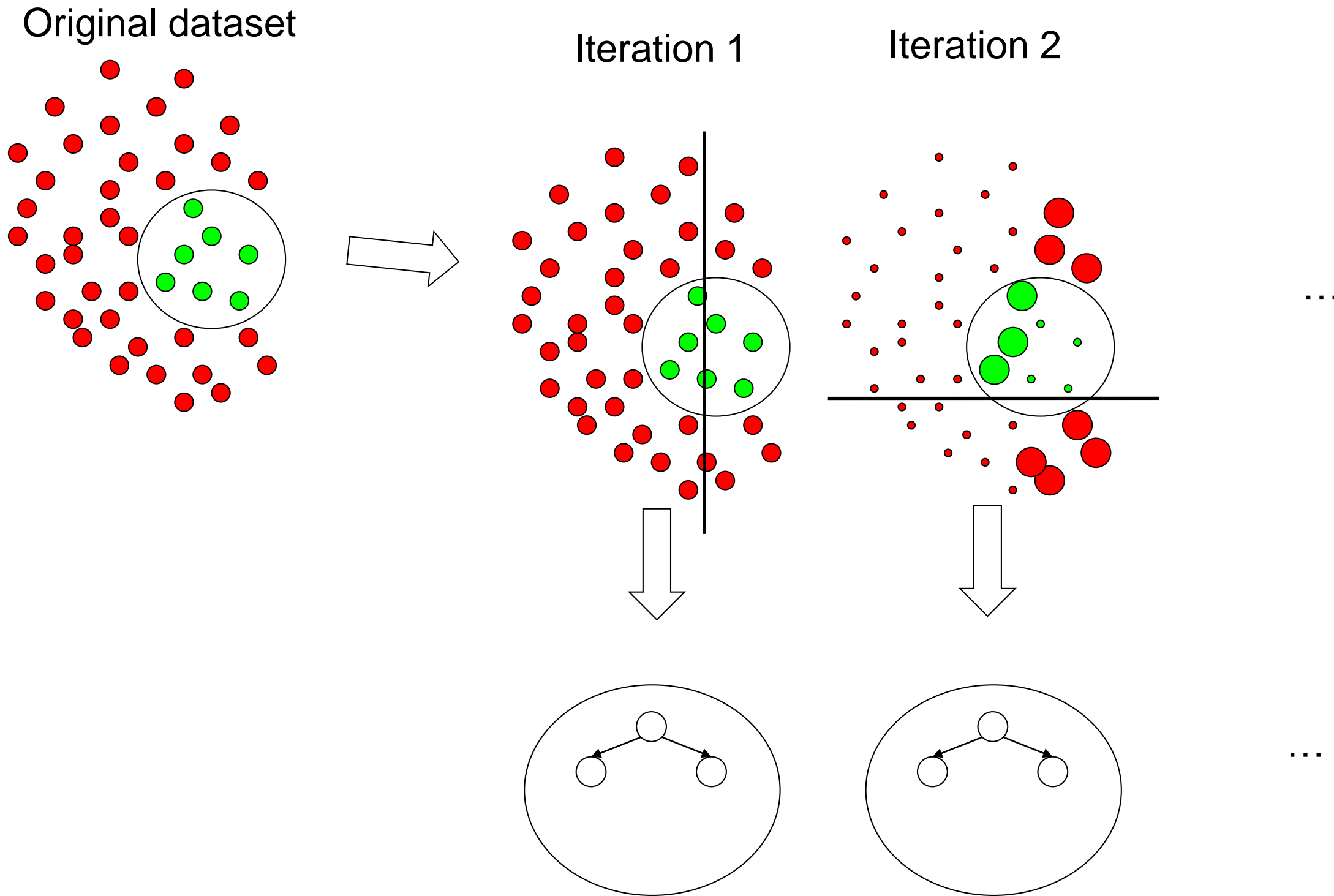
Tamaño del conjunto  $T$

1.  $w_i = 1/N$  para  $i=1 \dots N$
2. **for**  $t=1$  **to**  $T$ :
3.      $h_t = \text{BuildClassifier}(D, \mathbf{w})$
4.      $e_t = \text{WeightedError}(D, \mathbf{w})$
5.     **if**  $e_t == 0$  **or**  $e_t \geq 0.5$  **break**
6.     Modificar pesos de los ejemplos a

$$* \ w_i = \frac{w_i}{2e_t} \quad \forall i \mid h_t(X_i) \neq y_i$$

$$* \ w_i = \frac{w_i}{2(1-e_t)} \quad \forall i \mid h_t(X_i) = y_i$$

# Boosting



# Consideraciones sobre boosting

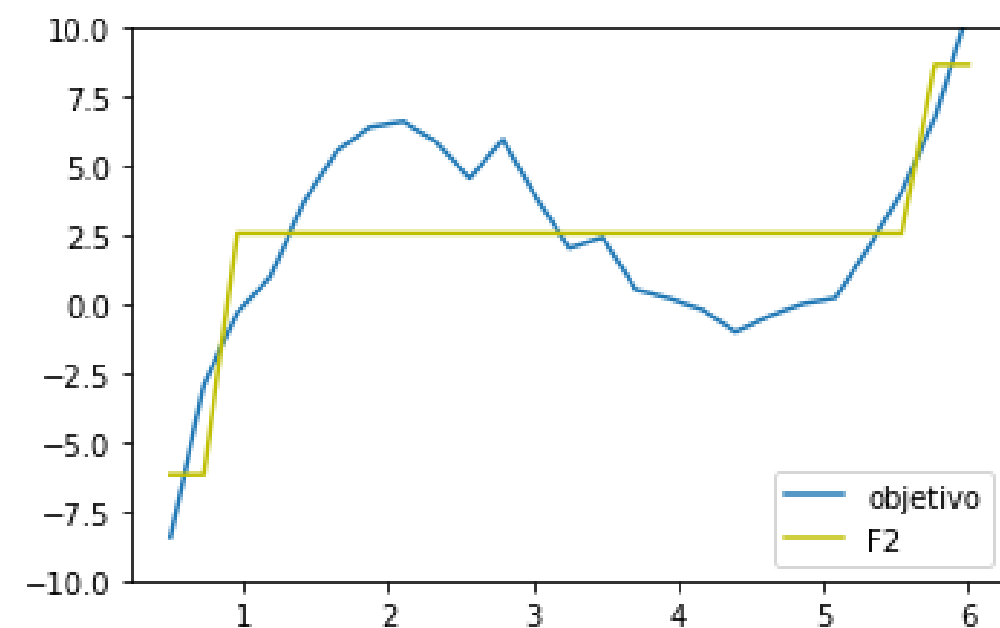
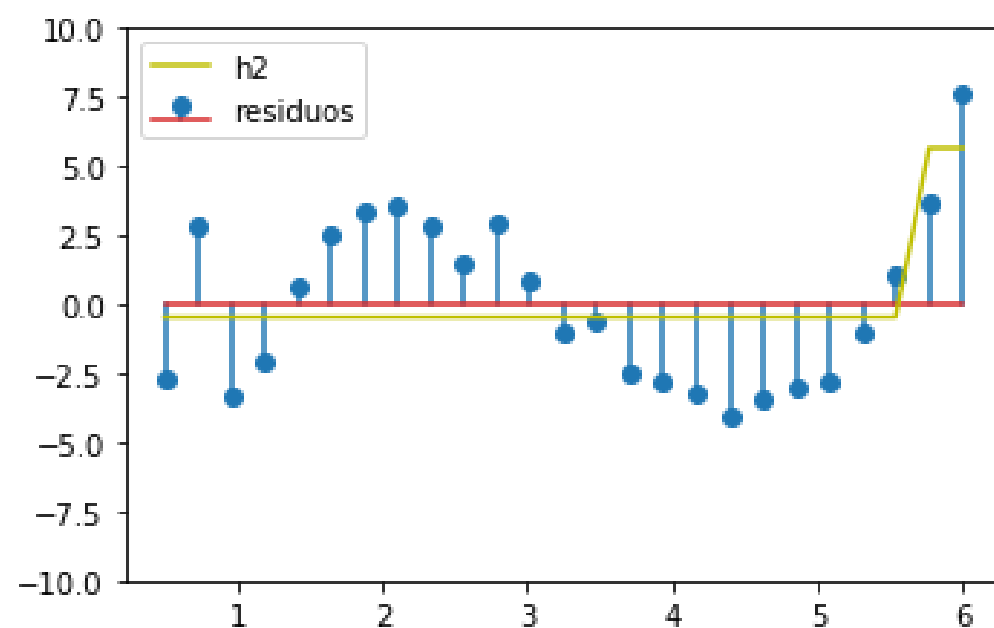
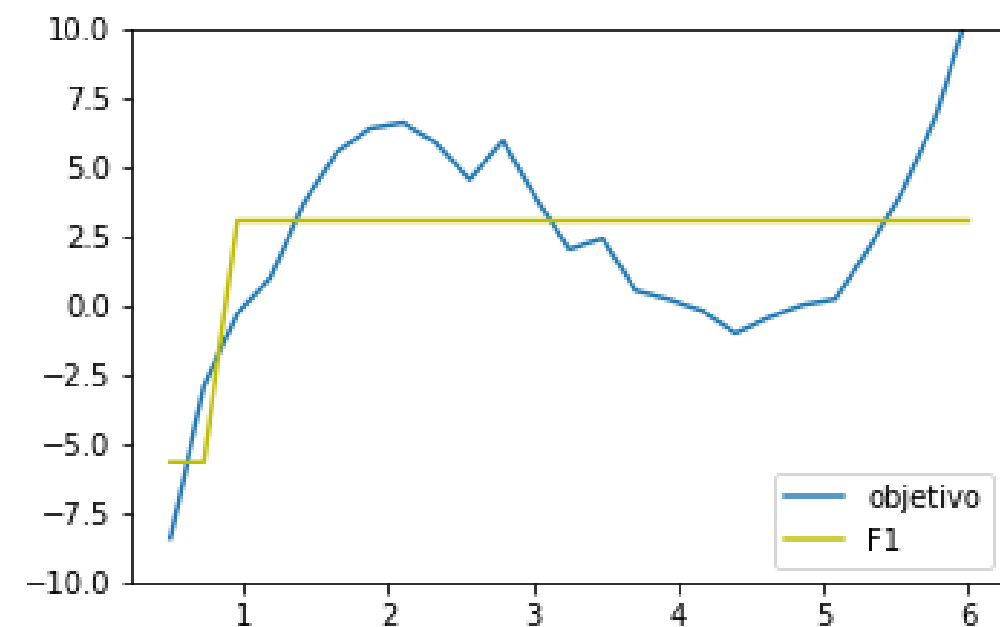
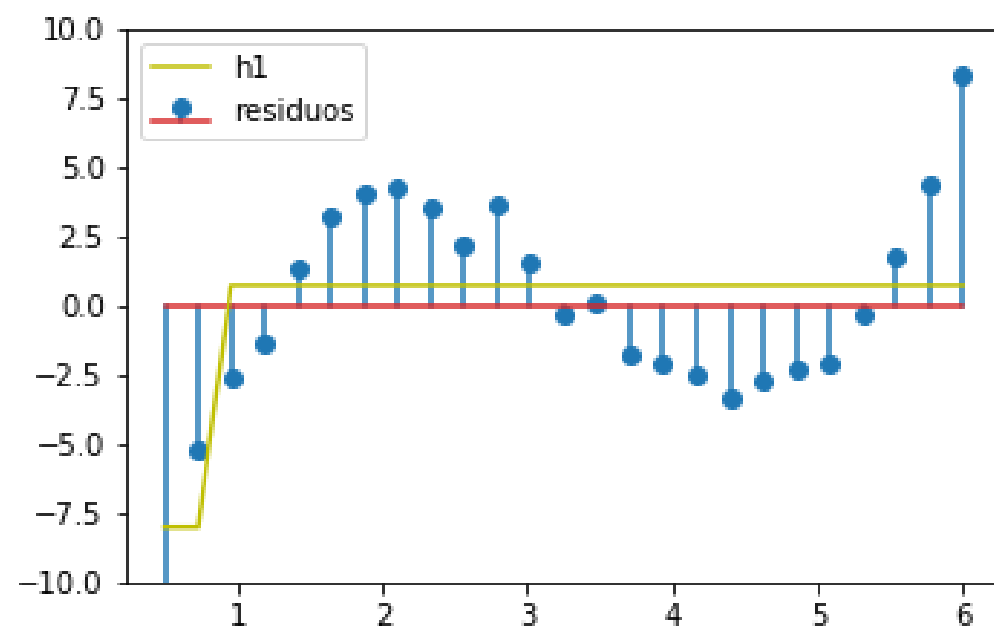
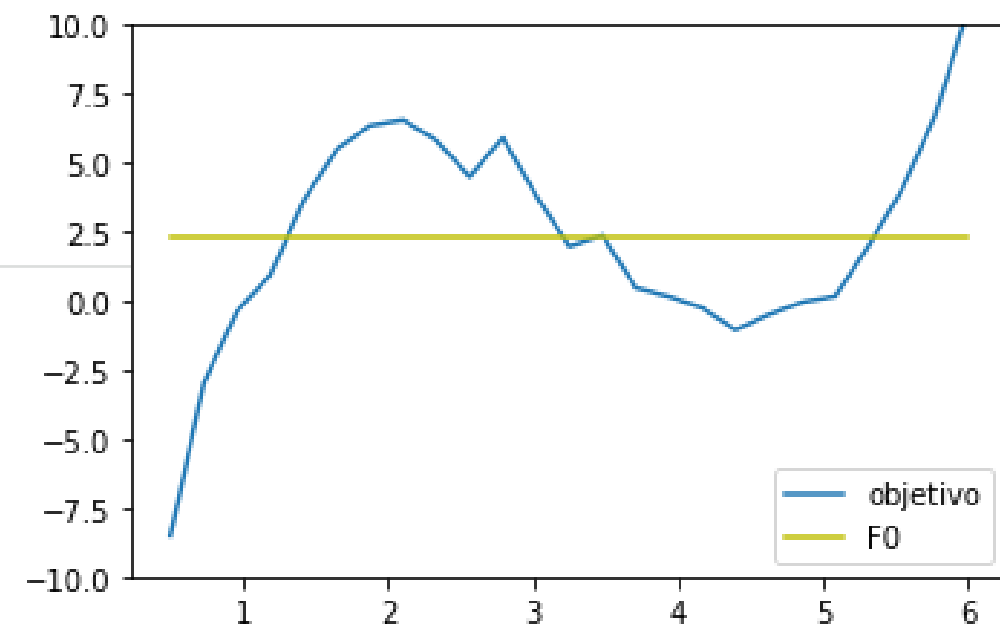
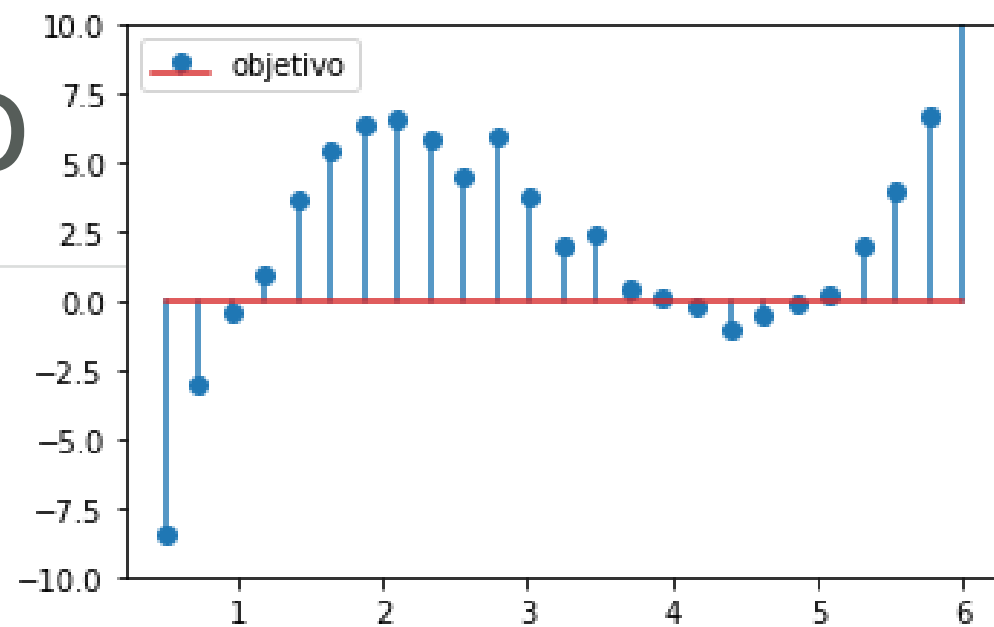
- Obtiene muy buenos resultados en general
- No es robusto frente a ruido en las etiquetas de clase
- Puede incrementar el error del clasificador base
- No se puede implementar fácilmente en paralelo

# GradientBoosting

---

**En otra presentación**

# Ejemplo GB



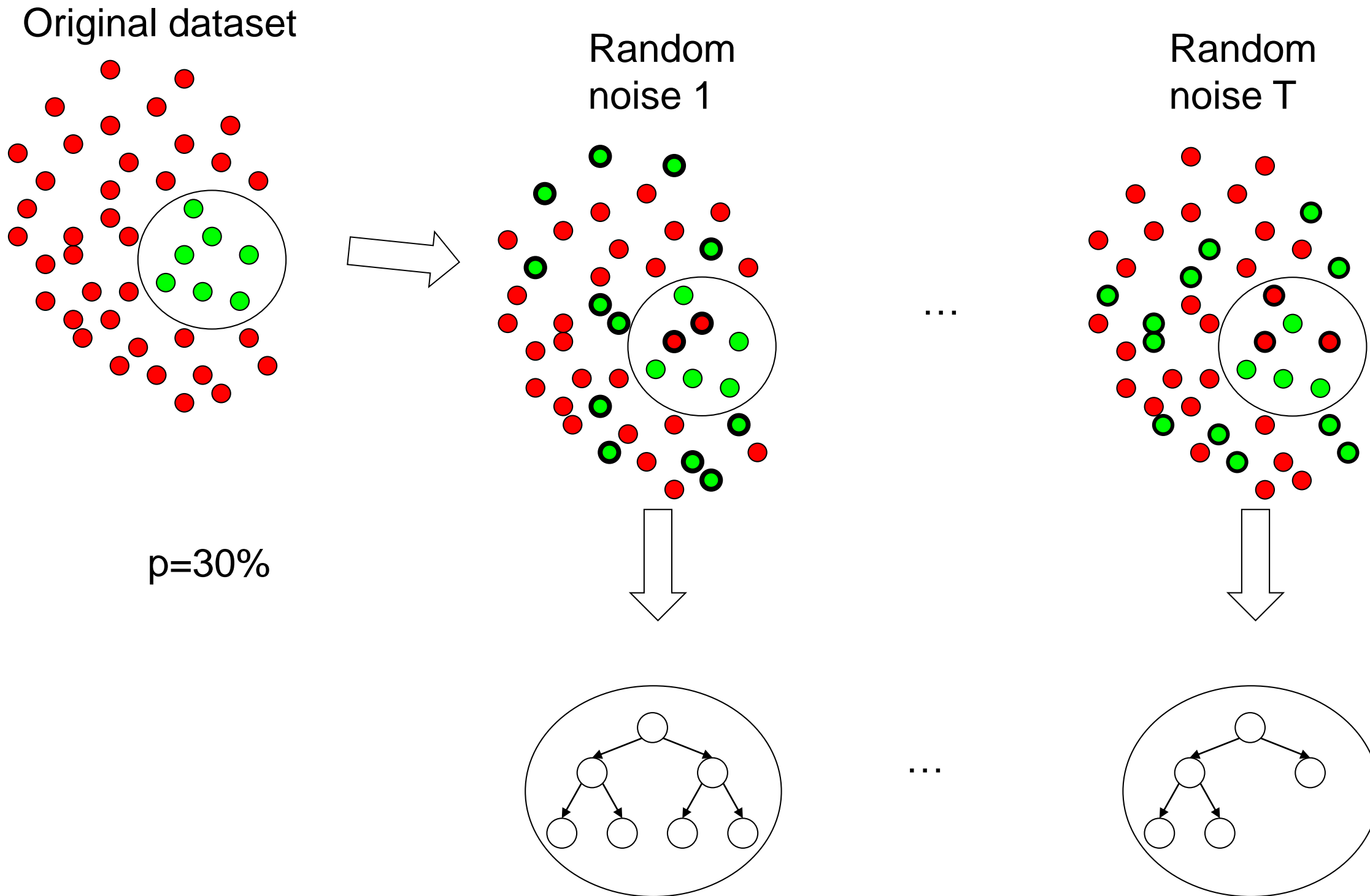
# Consideraciones sobre GradientBoosting

- 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

# 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  $p$ .

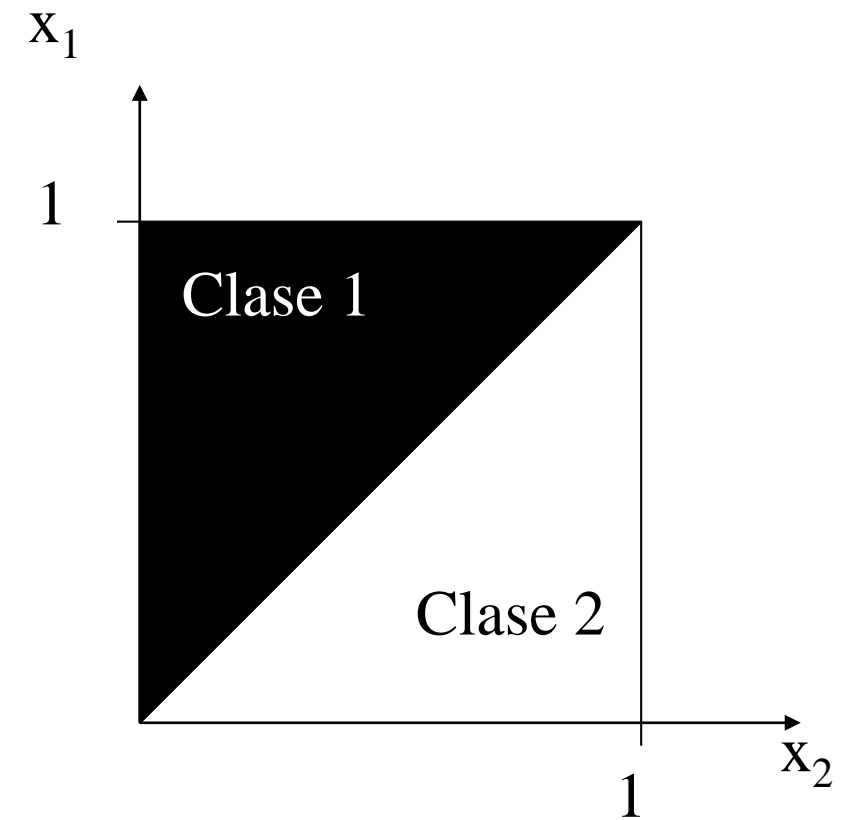
# Class switching



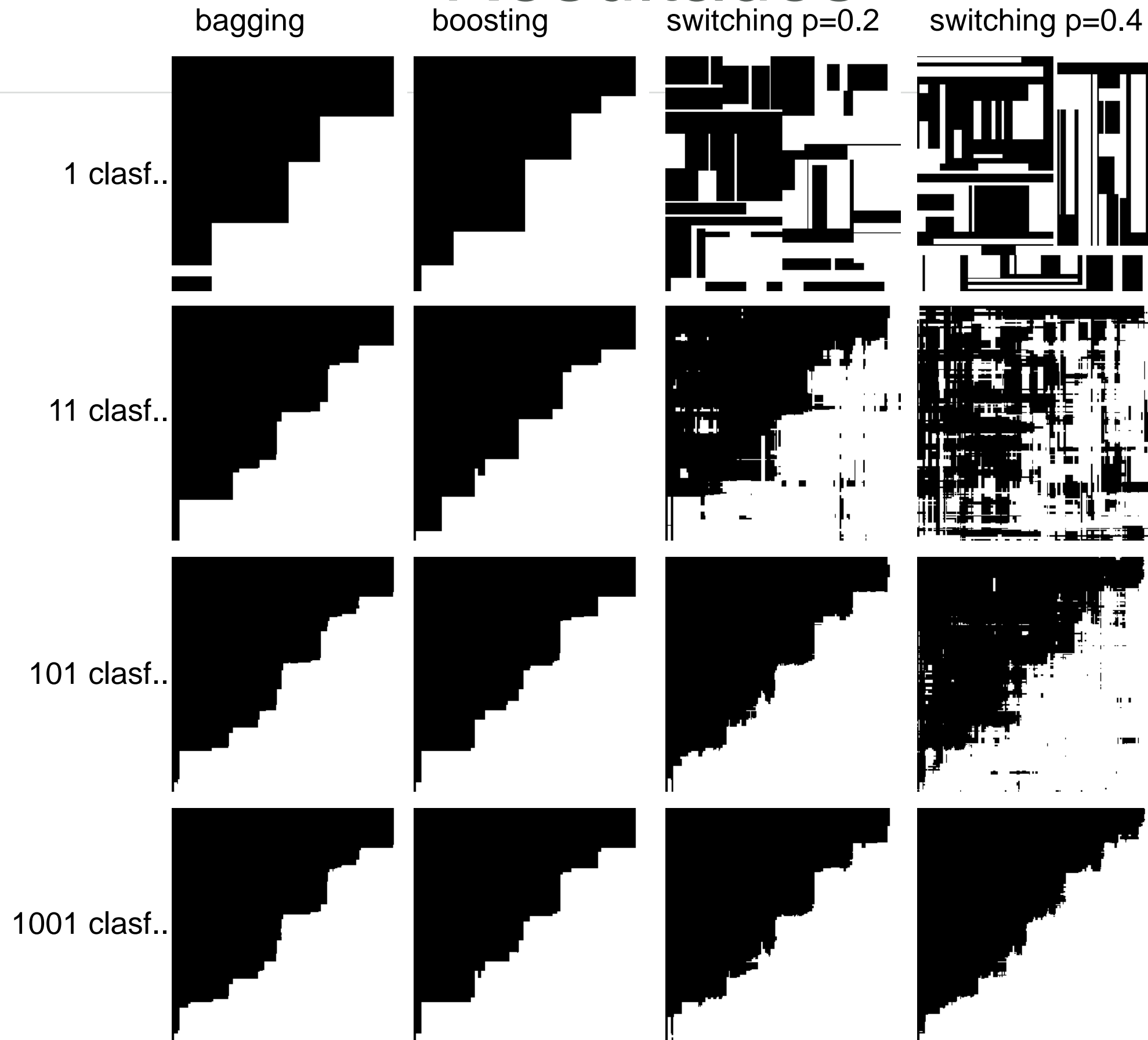


# Ejemplo

- 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>

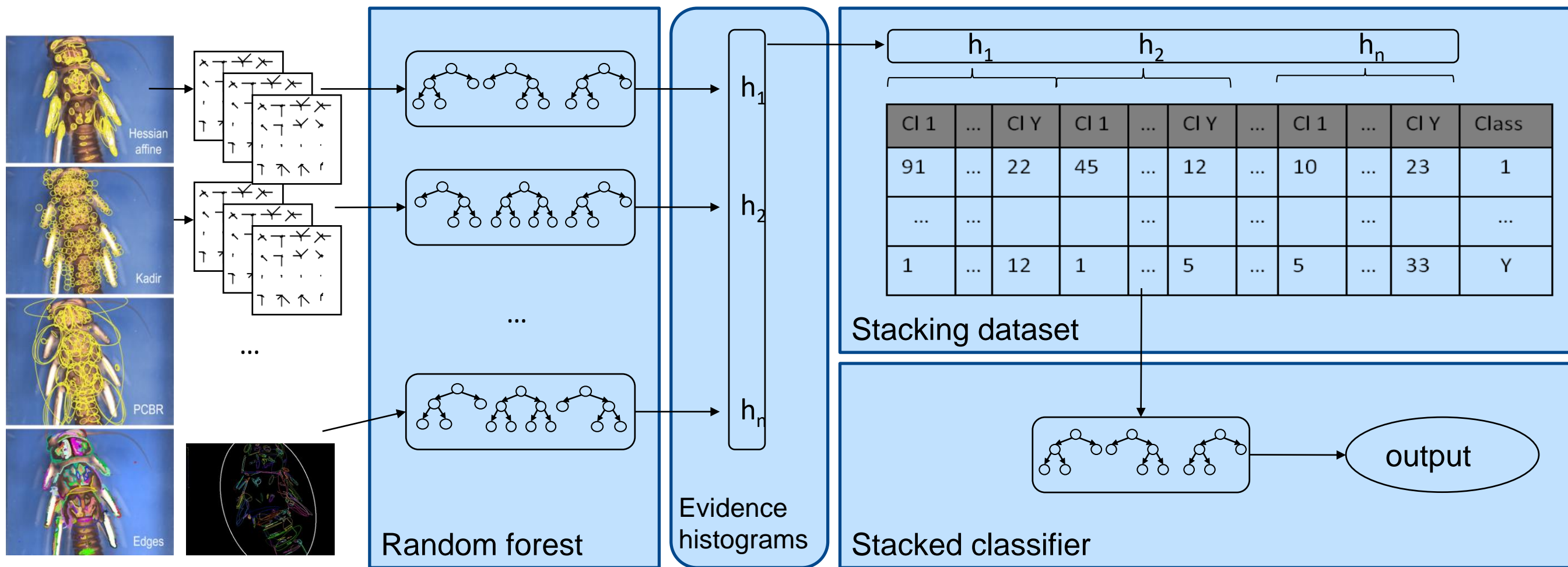
# Combinar la salida

- 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



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

# Poda de conjuntos

1.- El orden aleatorio dado por *bagging*

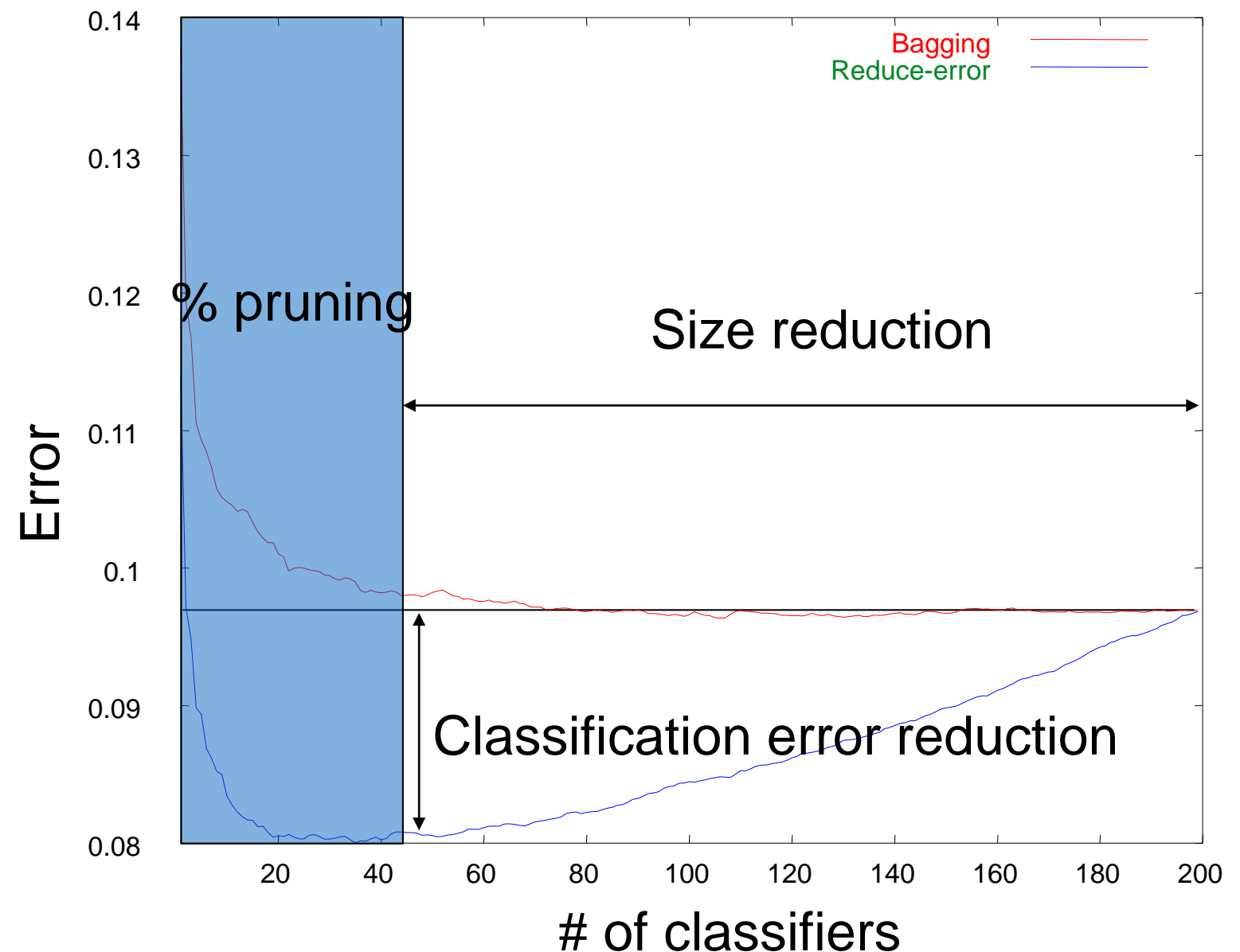
$h_1, h_2, h_3, \dots, h_T$

2.- Nueva ordenación

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

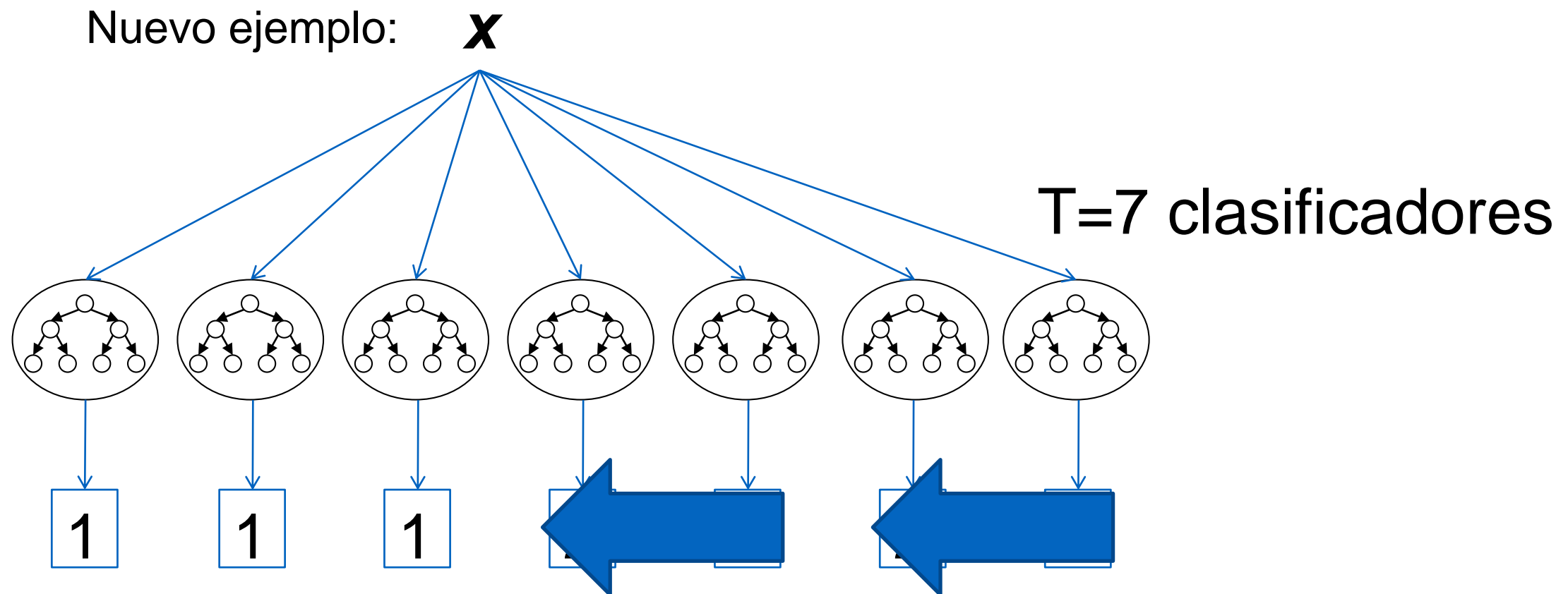
3.- Poda

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





# Poda dinámica de conjuntos



Votos acumulados:  $\mathbf{t} \rightarrow \begin{matrix} \boxed{0} & \boxed{0} \\ t_1 & t_2 \end{matrix} \rightarrow \text{Clase final } \boxed{1}$

¿Necesitamos preguntar a todos los clasificadores?

NO

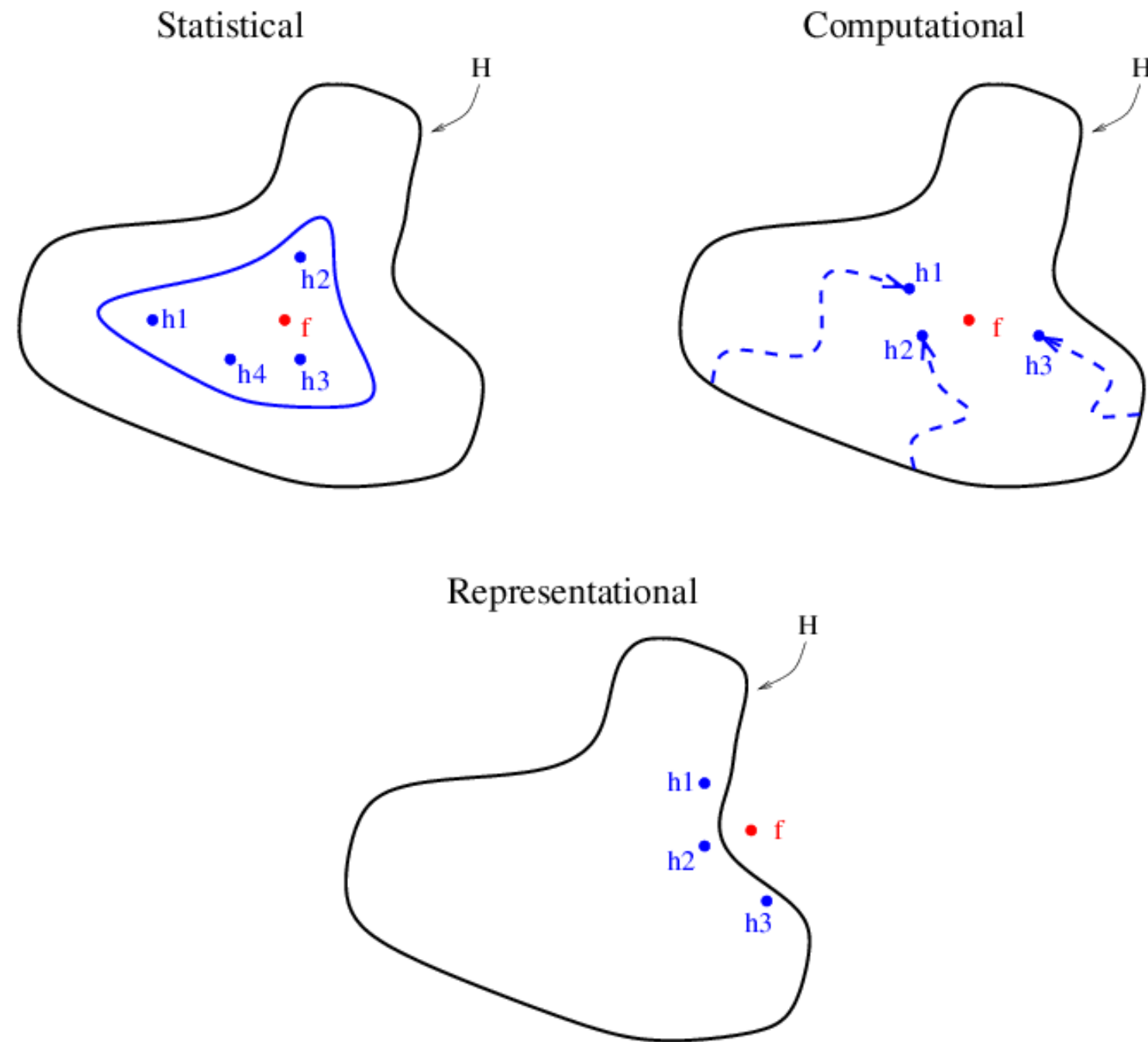
# Importancia de atributos

- 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 deteriore 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?

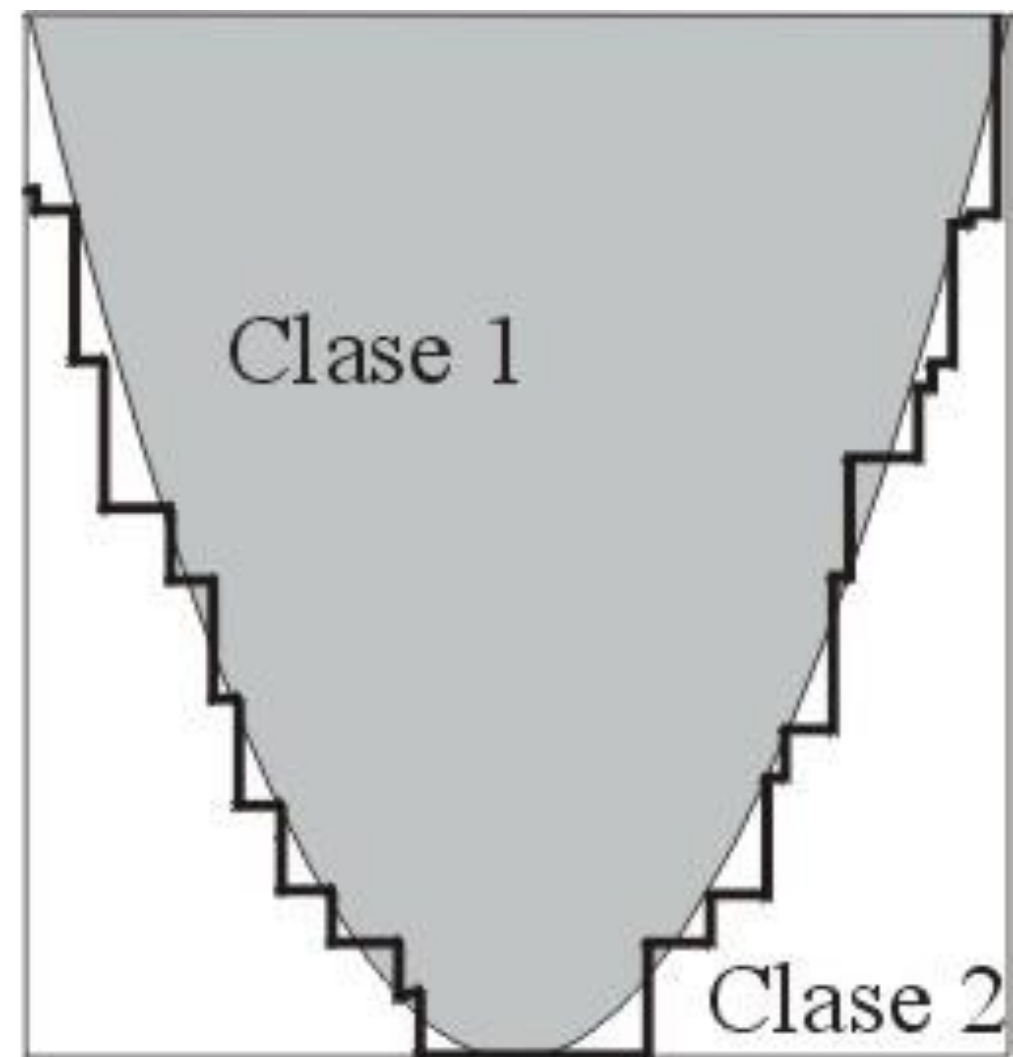
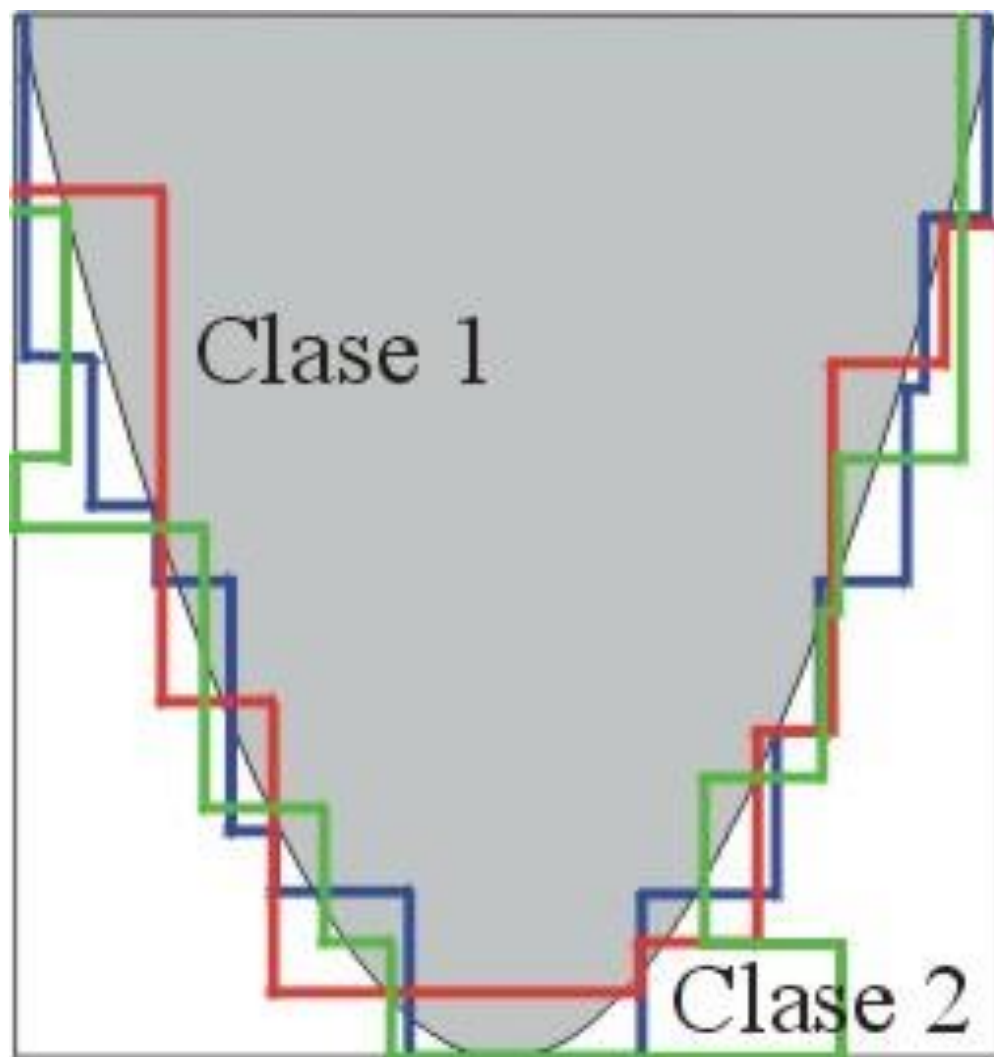


Thomas Dietterich

**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  leaders.

Combina cientos de modelos de tres equipos

Variante de stacking

Rank	Team Name	Best Test Score	% Improvement	Best Submit Time
Grand Prize - RMSE = 0.8567 - Winning Team: BellKor's Pragmatic Chaos				
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 I</a>	0.8591	9.81	2009-07-10 00:32:20
6	<a href="#">PragmaticTheory</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

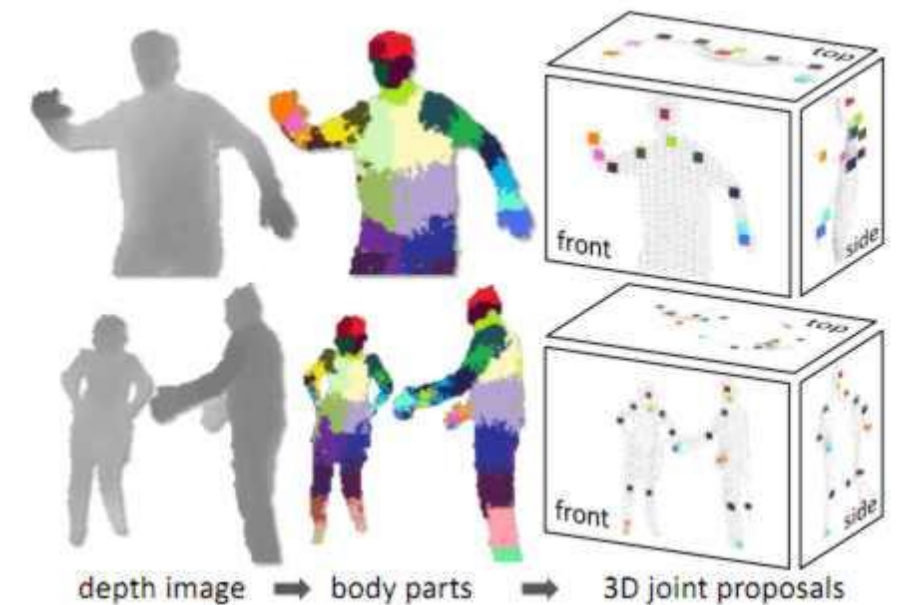
# Historia de éxito 2: KDD cup

- 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
  - 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 mitigar 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.