

Métodos de ensambles

Outline

- Ensembles en general
- Bagging
- Random Forest
- Boosting

Introducción

- Ensamble: conjunto grande de modelos que se usan juntos como un “meta modelo”
- Idea base conocida: usar conocimiento de distintas fuentes al tomar decisiones

Ejemplos conocidos

- El comité de expertos:

- muchos elementos
- todos con alto conocimiento
- todos sobre el mismo tema
- votan

Ensamblajes planos:

- Fusión
- Bagging
- Boosting
- Random Forest

- Gabinete de asesores:

- expertos en diferentes áreas.
- alto conocimiento
- hay una cabeza que decide quién sabe del tema

Ensamblajes divisivos:

- Mixture of experts
- Stacking

Componentes base

Dos componentes base:

- Un método para seleccionar o construir los miembros
 - Ejemplo: misma o distinta área
- Un método para combinar las decisiones
 - Votación, promedio, función de disparo

Ensamblajes divisivos

- Dividir el problema en una serie de sub-problemas con mínima superposición
- Estrategia de “divide & conquer”
- Útiles para atacar problemas grandes
- Se necesita una función que decida que clasificador tiene que actuar

Ensamblés “planos”

- Muchos expertos, todos buenos
 - Necesito que sean lo mejor posible individualmente
 - De lo contrario, usualmente no sirven
 - Pero necesito que opinen distinto en algunos casos
 - Si todos opinan siempre igual... me quedo con uno solo!

Caso de éxito

Netflix Prize

HomeRulesLeaderboardUpdate

Leaderboard

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

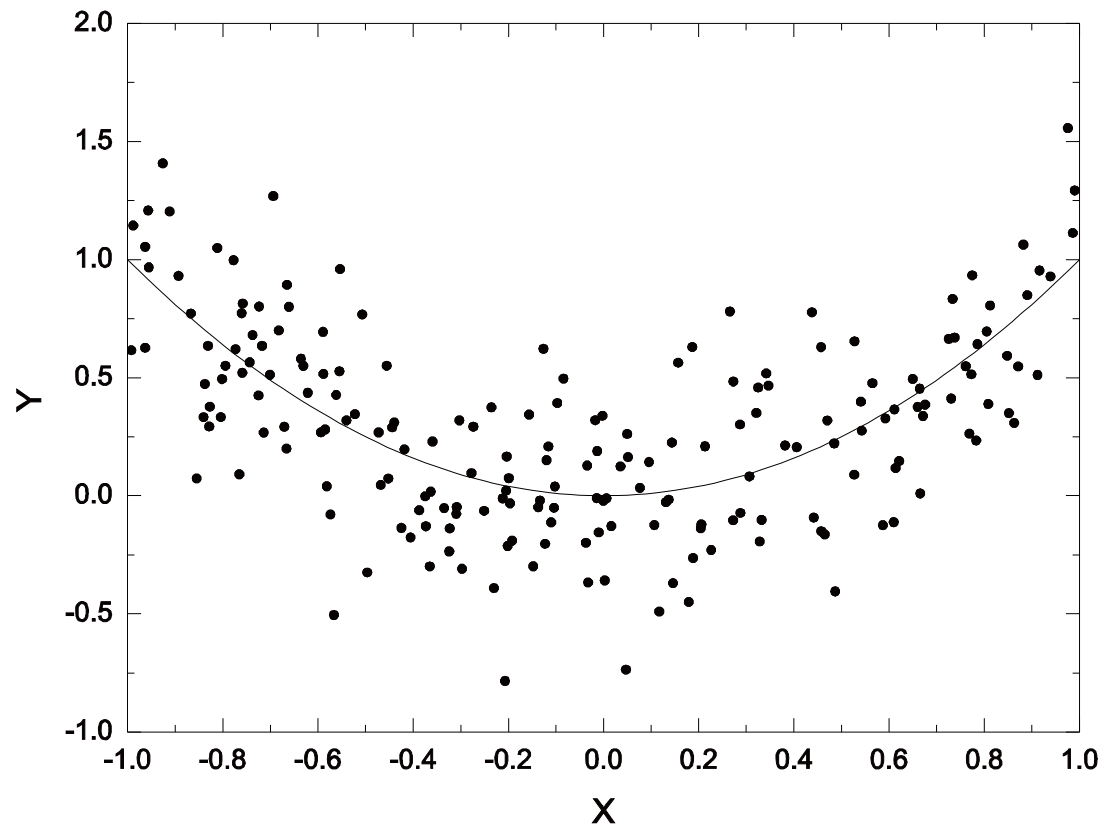
Display top leaders.

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

Formalización

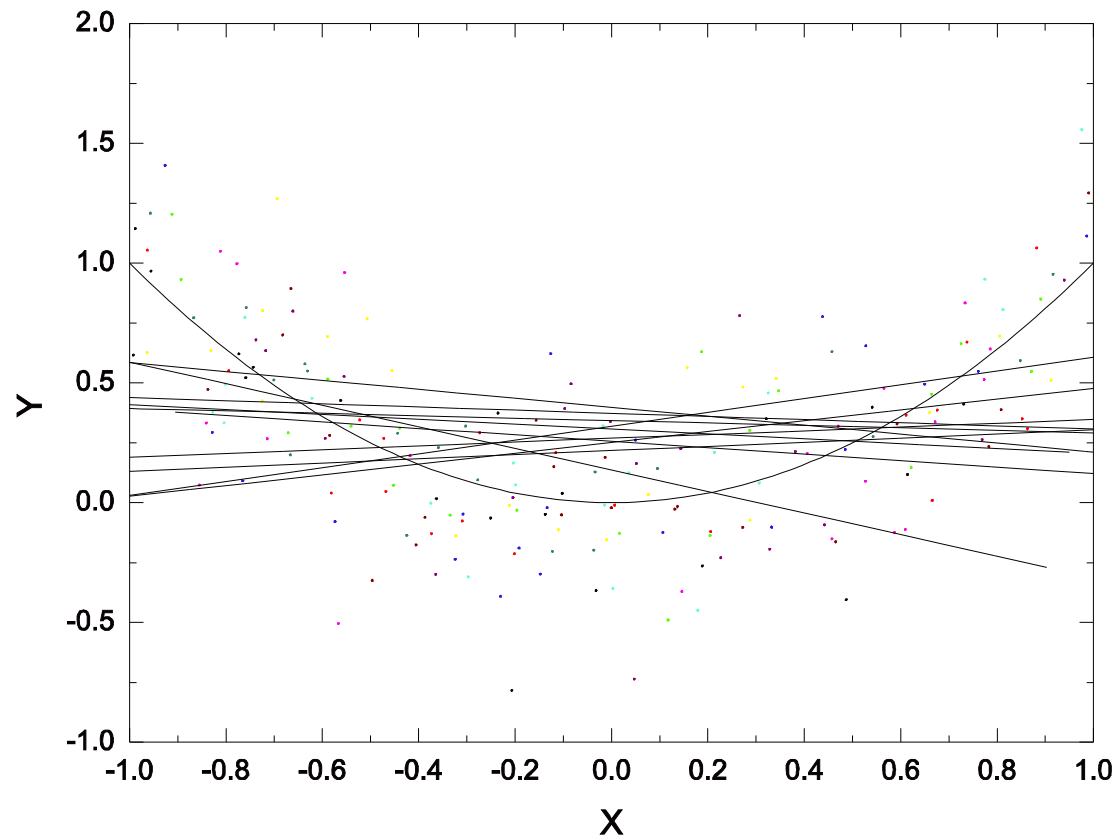
- Un camino posible es la vía estadística
- Se habla del dilema Sesgo-varianza
 - O Bias-variance
- Ejemplo

Ejemplo: Datos



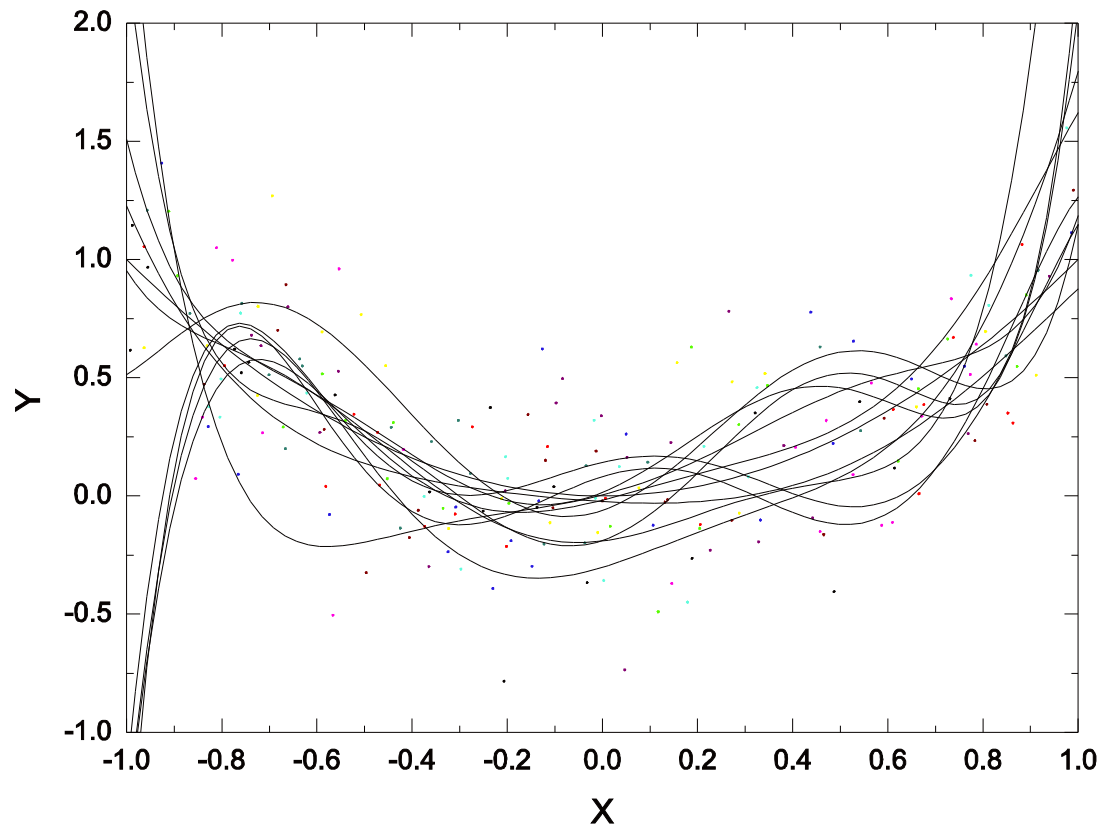
$$y(x) = x^2 + \varepsilon$$

Ejemplo: Sesgo



$$f_i(x) = a_{i1}x + a_{i0}$$

Ejemplo: Varianza



$$g_i(x) = \sum_{j=0}^6 b_{ij} x^j$$

Sesgo y Varianza

Que funciones utilizar?

- Funciones rígidas:

Buena estimación de los
parámetros óptimos –
poca flexibilidad

→ Error de sesgo

- Funciones flexibles:

Buen ajuste – mala
estimación de los
parámetros óptimos

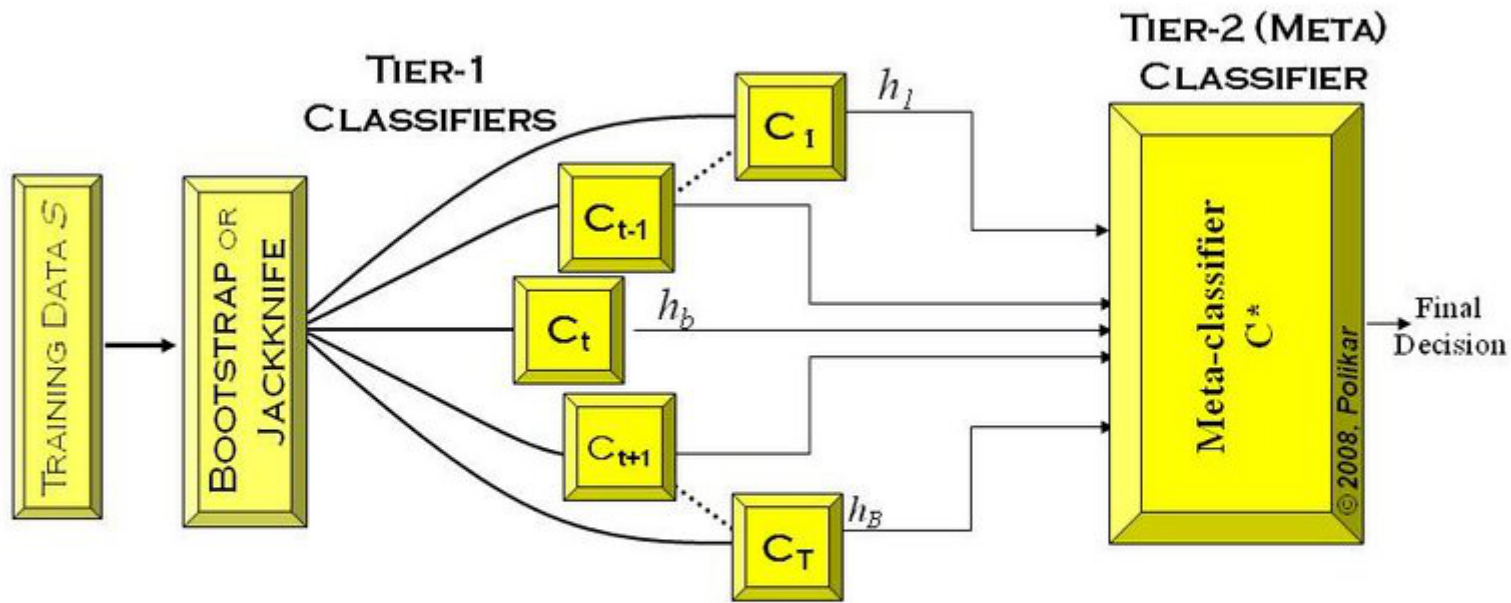
→ Error de varianza

El dilema

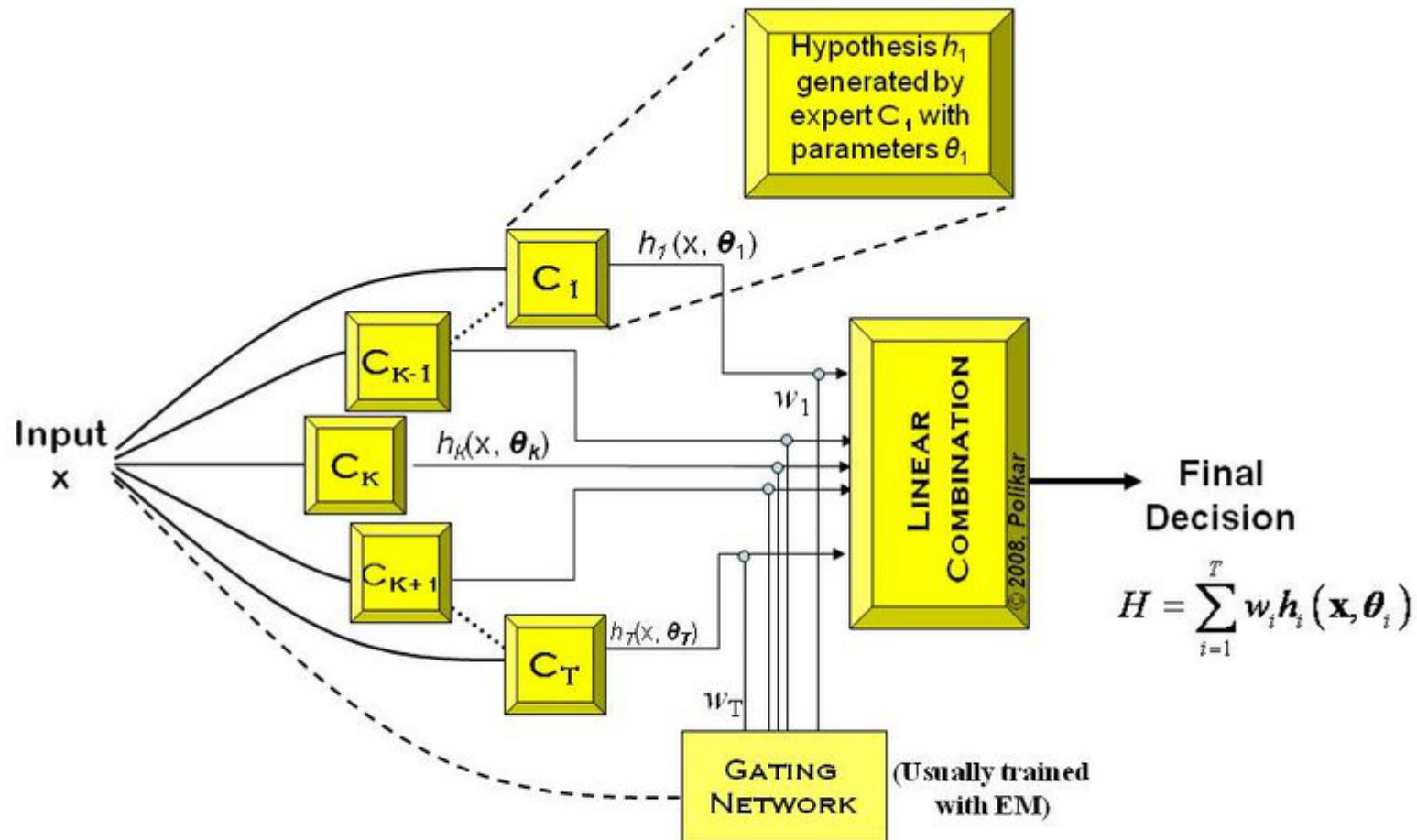
- Los predictores sin sesgo tienen alta varianza (y al revés)
- Hay dos formas de resolver el dilema:
 - Disminuir la varianza de los predictores sin sesgo
 - Construir muchos predictores y promediarlos: Bagging y Random Forest
 - Reducir el sesgo de los predictores estables
 - Construir una secuencia tal que la combinación tenga menos sesgo: Boosting

Divisivos

Stacking



Mixture of experts



Bagging



Disminuir la varianza

- Si tenemos realmente predictores sin sesgo, la media de estos en cada punto tiene que ser el valor real de la función a predecir.
- En ese caso, los errores que cometen los distintos predictores se cancelan.
- Eso es equivalente a pedir que los errores de los predictores estén decorrelacionados, que sean lo más diversos entre sí.

Disminuir... (2)

- Idea:
 - Construir un conjunto de predictores
 - Sin sesgo (lo más precisos posible)
 - Decorrelacionados (lo más diversos posible)
 - Promediarlos para conseguir un predictor con poco sesgo y poca varianza

Bagging

- Bootstrap **aggregating** (Leo Breiman, 96)
- Usar predictores buenos pero inestables
 - Árboles y ANN sobre todo
- Conseguir diversidad perturbando los datos
 - Como los predictores son inestables, pequeños cambios en los datos dan cambios importantes en los modelos

Datos perturbados

- La idea de Bagging es usar bootstraps
- Un bootstrap es una muestra al azar de la misma longitud, con repetición

```
>sample(1:10,rep=T)  
>3 5 4 3 1 7 9 9 2 1
```
- El bootstrap no introduce bias en ninguna estadística (propiedad importante)
- Cada predictor se entrena sobre un bootstrap del conjunto de entrenamiento

Regla de combinación

- Para regresión/ranking: promedio simple de las predicciones de cada modelo
- Para clasificación:
 - Cada modelo vota por una clase. La que tiene más votos es elegida.
 - Cada modelo estima la probabilidad de clase. Se promedian. La de mayor probabilidad se elige.

	Modelo 1	Modelo 2	Modelo 3	Votos	Probs.
Clase A	0,45	0,45	0,95	1	0,62
Clase B	0,55	0,55	0,05	2	0,38

Regla de combinación

- Para clustering:
 - Creamos una matriz que cuenta cuantas veces cada par de puntos termina en el mismo cluster, y clusterizamos esa matriz de similitud.

Algoritmo: Bagging (clasificación)

Entrada: Conjunto de datos $D = \{(x_i, y_i), i=1, m\}$, donde $x_i \in X$ e $y_i \in Y = \{1, \dots, k\}$
Algoritmo de aprendizaje *WeakLearn*

Inicializar: $w_i(1) = 1/m \quad \forall i=1, m$

Repetir $t = 1, 2, \dots, T$:

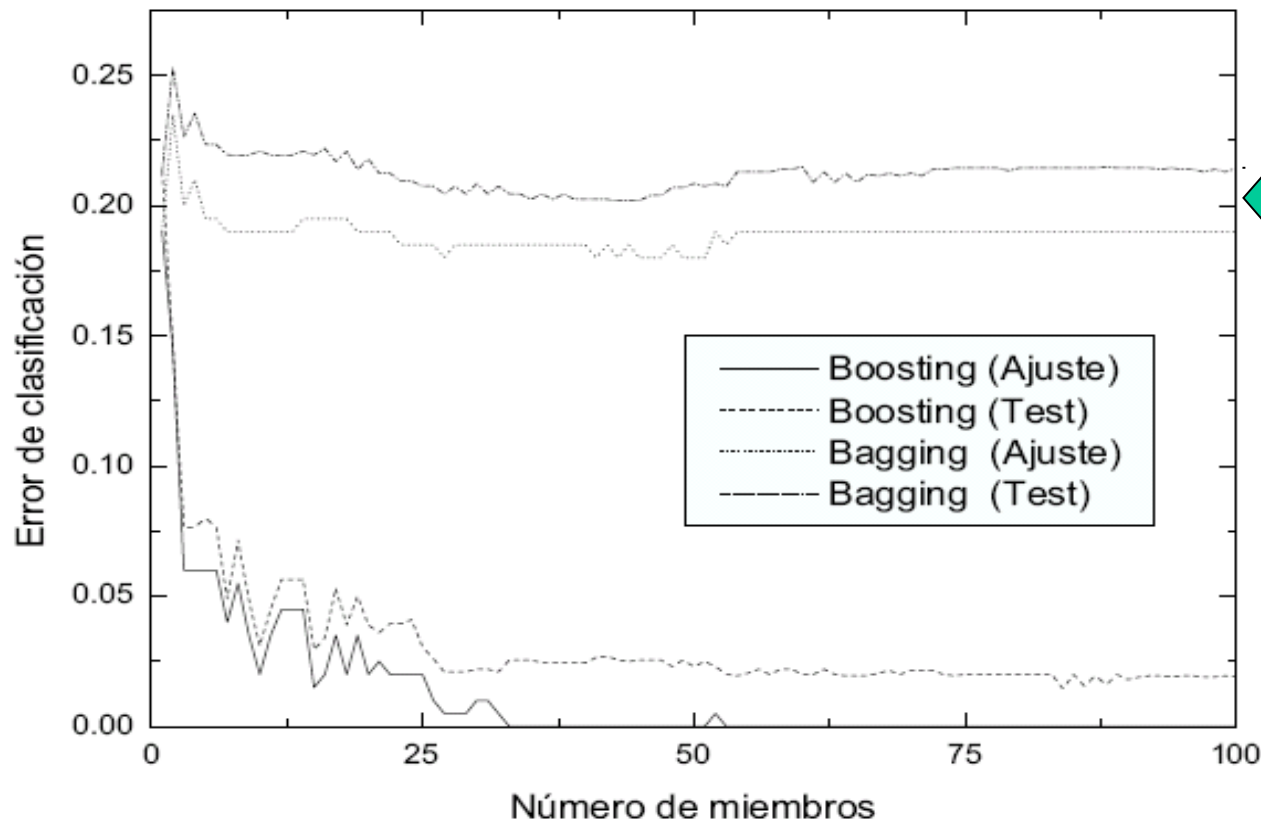
- tomar una muestra D_t a partir de D muestreando con probabilidad $w_i(t)$.
- utilizar *WeakLearn* en D_t para obtener un clasificador $f_t : X \rightarrow Y$

Salida: Hipótesis final:

$$\max_j \sum_{t=1}^T I[f_t(x) = j]$$
$$\max_j \sum_{t=1}^T P_t(x, j)$$

Cuántos predictores?

El único parámetro a fijar es el total de predictores T



Los errores de Train y test convergen suavemente.

Se usa un número grande de predictores directamente (100, 500)

Comparación de resultados

DATA SETS			
Data Set	# Samples	# Variables	# Classes
waveform	300	21	3
heart	1395	16	2
breast cancer	699	9	2
ionosphere	351	34	2
diabetes	768	8	2
glass	214	9	6
soybean	683	35	19

De: *Bagging Predictors*, Leo Breiman, **Machine Learning**, 24:123-140, 1996

Comparación de resultados

MISCLASSIFICATION RATES (%)			
Data Set	\bar{e}_S	\bar{e}_B	Decrease
waveform	29.1	19.3	34%
heart	4.9	2.8	43%
breast cancer	5.9	3.7	37%
ionosphere	11.2	7.9	29%
diabetes	25.3	23.9	6%
glass	30.4	23.6	22%
soybean	8.6	6.8	21%

De: *Bagging Predictors*, Leo Breiman, **Machine Learning**, 24:123-140, 1996
 e_S es modelo individual, e_B es Bagging

Comparación de resultados

LARGER DATA SETS

Data Set	#Training	#Variables	#Classes	#Test Set
letters	15,000	16	26	5000
satellite	4,435	36	6	2000
shuttle	43,500	9	7	14,500
DNA	2,000	60	3	1186

De: *Bagging Predictors*, Leo Breiman, **Machine Learning**, 24:123-140, 1996

Comparación de resultados

TEST SET MISCLASSIFICATION RATES (%)			
Data Set	e_S	e_B	Decrease
letters	12.6	6.4	49%
satellite	14.8	10.3	30%
shuttle	.062	.014	77%
DNA	6.2	5.0	19%

De: *Bagging Predictors*, Leo Breiman, **Machine Learning**, 24:123-140, 1996
 e_S es modelo individual, e_B es Bagging

Resultados: clasificador estable

MISCLASSIFICATION RATES FOR NEAREST NEIGHBOR

Data Set	\bar{e}_S	\bar{e}_B
waveform	26.1	26.1
heart	5.1	5.1
breast cancer	4.4	4.4
ionosphere	36.5	36.5
diabetes	29.3	29.3
glass	30.1	30.1

De: *Bagging Predictors*, Leo Breiman, **Machine Learning**, 24:123-140, 1996
 e_S es modelo individual, e_B es Bagging

Variantes

- Otras formas de generar diversidad:
 - Sub-sampling de los datos
 - Hasta el 50% (Half & Half Bagging)
 - Agregar ruido
 - Variar los modelos en vez de los datos
 - Cambiar la forma de ajustar los modelos
 - Cambiar la condición inicial (ANN)

Resultados: ANN

De: *Popular ensemble methods*, D.Opitz y R. Maclin,
Journal of Artificial Intelligence research, 11:169-198, 1999

Stan es modelo individual, Bag es Bagging, Simp es ensemble variando solo la condición inicial

Data Set	Stan	Simp	Bag
breast-cancer-w	3.4	3.5	3.4
credit-a	14.8	13.7	13.8
credit-g	27.9	24.7	24.2
diabetes	23.9	23.0	22.8
glass	38.6	35.2	33.1
heart-cleveland	18.6	17.4	17.0
hepatitis	20.1	19.5	17.8
house-votes-84	4.9	4.8	4.1
hypo	6.4	6.2	6.2
ionosphere	9.7	7.5	9.2
iris	4.3	3.9	4.0
kr-vs-kp	2.3	0.8	0.8
labor	6.1	3.2	4.2
letter	18.0	12.8	10.5
promoters-936	5.3	4.8	4.0
ribosome-bind	9.3	8.5	8.4
satellite	13.0	10.9	10.6
segmentation	6.6	5.3	5.4
sick	5.9	5.7	5.7
sonar	16.6	15.9	16.8
soybean	9.2	6.7	6.9
splice	4.7	4.0	3.9
vehicle	24.9	21.2	20.7

Variantes

- Otras formas de combinar:
 - Usar probabilidades en vez de votos
 - Usar pesos estadísticos en la combinación
 - Votos “pesados”: los mejores clasificadores tienen más fuerza en la decisión
 - Funciona igual o mejor que el uniforme

Resumen: Bagging

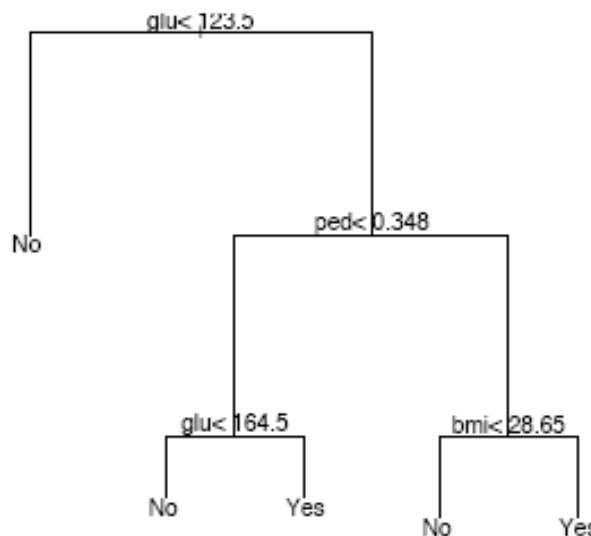
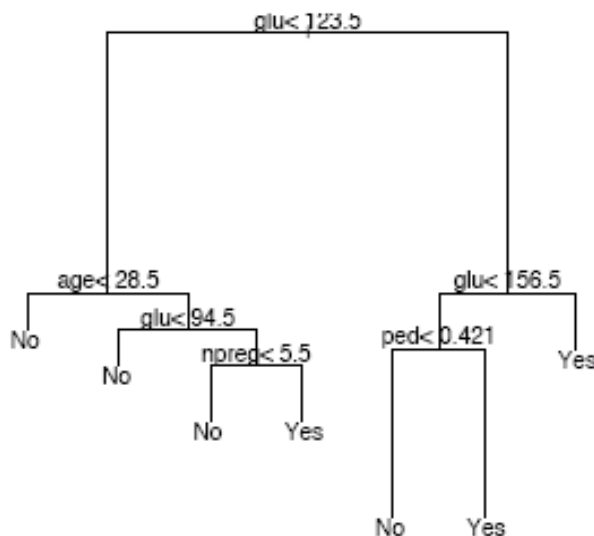
- Método simple y efectivo para crear ensembles
- Reduce la varianza al promediar predictores diversos
- Usa bootstraps y predictores inestables
- Se suele usar con árboles y redes
 - Con redes BP es mejor usar sólo la diversidad natural del entrenamiento

Random Forest

Generando más
diversidad

Problema con bagging + trees

- Usar bootstraps genera diversidad, pero los árboles siguen estando muy correlacionados
 - Las mismas variables tienden a ocupar los primeros cortes siempre.



Ejemplo:

Dos árboles generados con rpart a partir de bootstraps del dataset Pima.tr. La misma variable está en la raíz

Solución propuesta

- Agregar un poco de azar al crecimiento
- En cada nodo, seleccionar un grupo chico de variables al azar y evaluar sólo esas variables.
 - No agrega sesgo: A la larga todas las variables entran en juego
 - Agrega varianza: pero eso se soluciona fácil promediando modelos
 - Es efectivo para decorrelacionar los árboles

Random Forest: Algorithm

Algorithm 15.1 *Random Forest for Regression or Classification.*

1. For $b = 1$ to B :
 - (a) Draw a bootstrap sample \mathbf{Z}^* of size N from the training data.
 - (b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.
 - i. Select m variables at random from the p variables.
 - ii. Pick the best variable/split-point among the m .
 - iii. Split the node into two daughter nodes.
2. Output the ensemble of trees $\{T_b\}_1^B$.

To make a prediction at a new point x :

Regression: $\hat{f}_{\text{rf}}^B(x) = \frac{1}{B} \sum_{b=1}^B T_b(x)$.

Classification: Let $\hat{C}_b(x)$ be the class prediction of the b th random-forest tree. Then $\hat{C}_{\text{rf}}^B(x) = \text{majority vote } \{\hat{C}_b(x)\}_1^B$.

Observaciones

- Construye los árboles hasta separar todo. No hay podado. No hay criterio de parada.
- El valor de m (`mtry` en R) es importante. El default es \sqrt{p} que suele ser bueno.
 - Si uso $m=p$ recupero bagging
- El número de árboles no es importante, mientras sean muchos. 500, 1000, 2000.

Resultados

TABLE 1
Data set descriptions

Data set	Training Sample size	Test Sample size	Variables	Classes
Cancer	699	—	9	2
Ionosphere	351	—	34	2
Diabetes	768	—	8	2
Glass	214	—	9	6
Soybean	683	—	35	19
Letters	15,000	5000	16	26
Satellite	4,435	2000	36	6
Shuttle	43,500	14,500	9	7
DNA	2,000	1,186	60	3
Digit	7,291	2,007	256	10

Resultados

TABLE 2
Test set misclassification error (%)

Data set	Forest
Breast cancer	2.9
Ionosphere	5.5
Diabetes	24.2
Glass	22.0
Soybean	5.7
Letters	3.4
Satellite	8.6
Shuttle $\times 10^3$	7.0
DNA	3.9
Digit	6.2

TEST SET MISCLASSIFICATION ERROR (%)		
Data Set	e_S	e_B
letters	12.6	6.4
satellite	14.8	10.3
shuttle	.062	.014
DNA	6.2	5.0

from Breiman: "Statistical Modelling: the two cultures".

Resultados

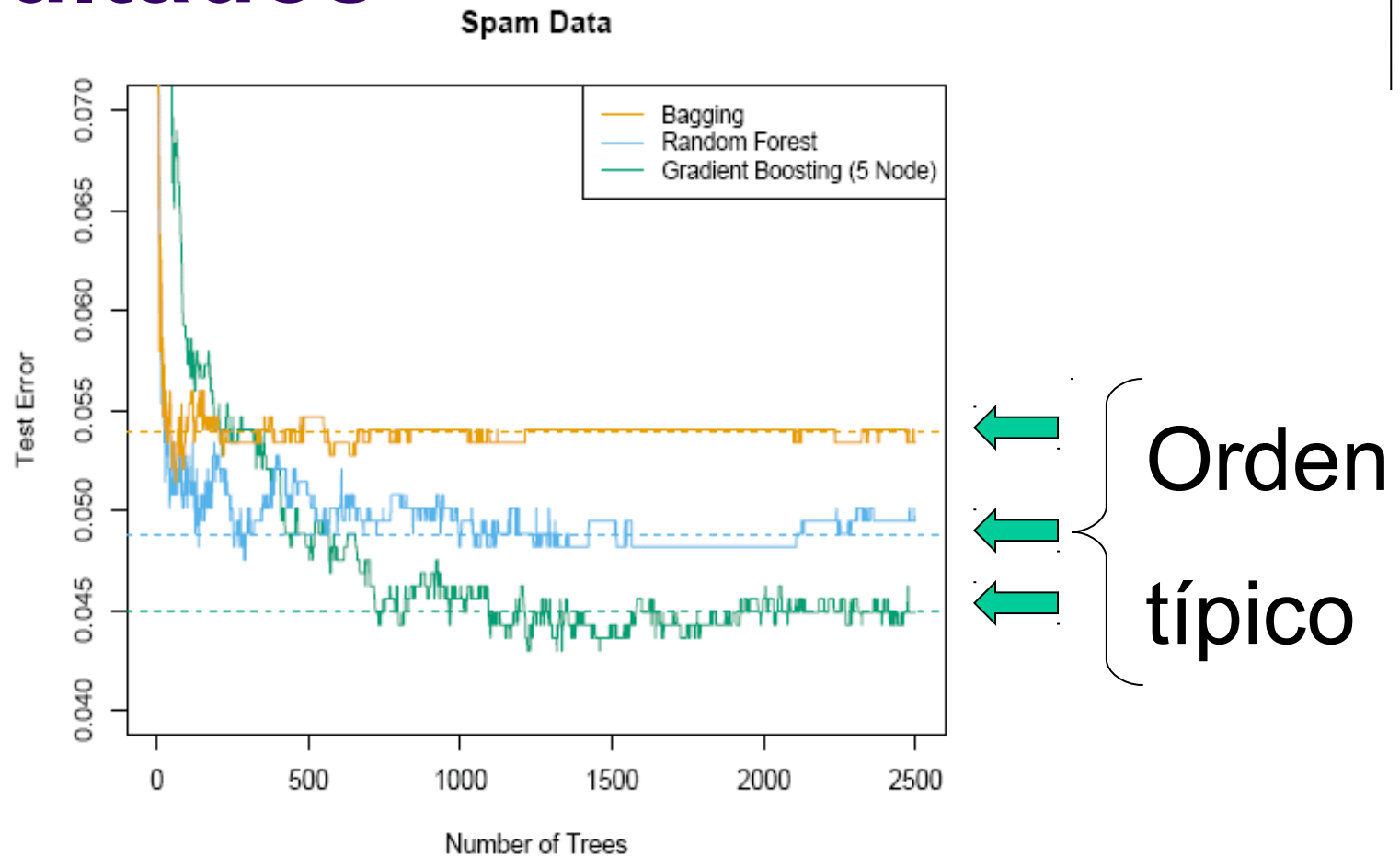


FIGURE 15.1. Bagging, random forest, and gradient boosting, applied to the spam data. For boosting, 5-node trees were used, and the number of trees were chosen by 10-fold cross-validation (2500 trees). Each “step” in the figure corresponds to a change in a single misclassification (in a test set of 1536).

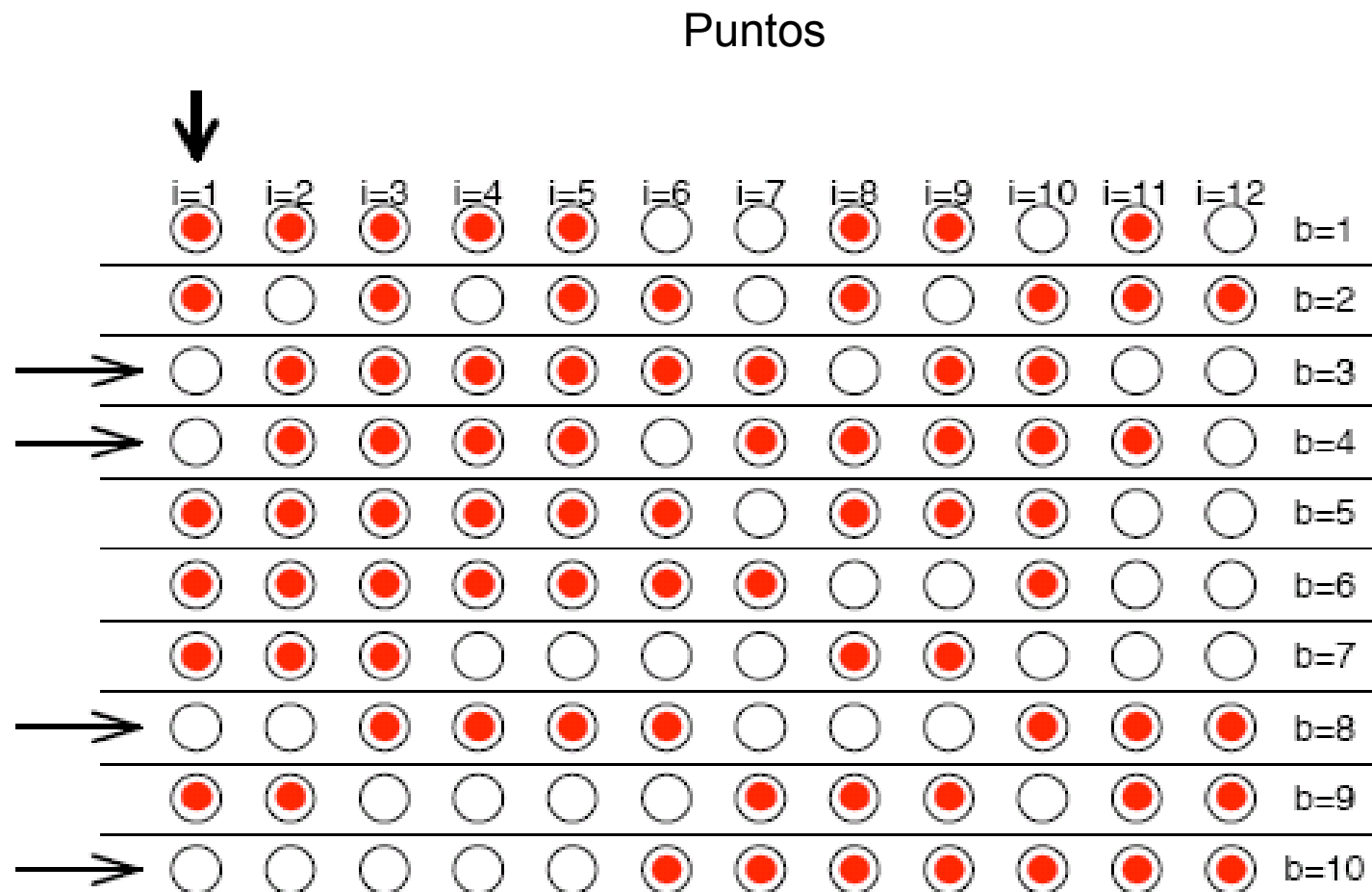
Sub-productos de RF

- Out-of-bag estimation
- Medida de importancia de las variables
- Gráfica de proximidad

Out-of-bag estimation

- Cuando se toma un bootstrap hay puntos que quedan fuera (se los llama OOB)
- Son muchos, un tercio del total en media
- Para cada predictor hay un conjunto OOB que no usó durante el entrenamiento
- Se pueden usar para estimar errores de predicción sin sesgo

Out-of-bag estimation



Predictores -
Bootstraps

Out-of-bag estimation

- Para cada punto del train set formo un sub-ensamble que tiene sólo los clasificadores que no entrenaron con ese punto.
- Hago predicciones sobre todo el conjunto de train y promedio. Se llaman predicciones OOB, errores OOB
- Son una buena estima del error de test
 - Tienen un sesgo “seguro”, estiman de más, porque están calculadas con un ensamble más chico

Ejemplo: OOB

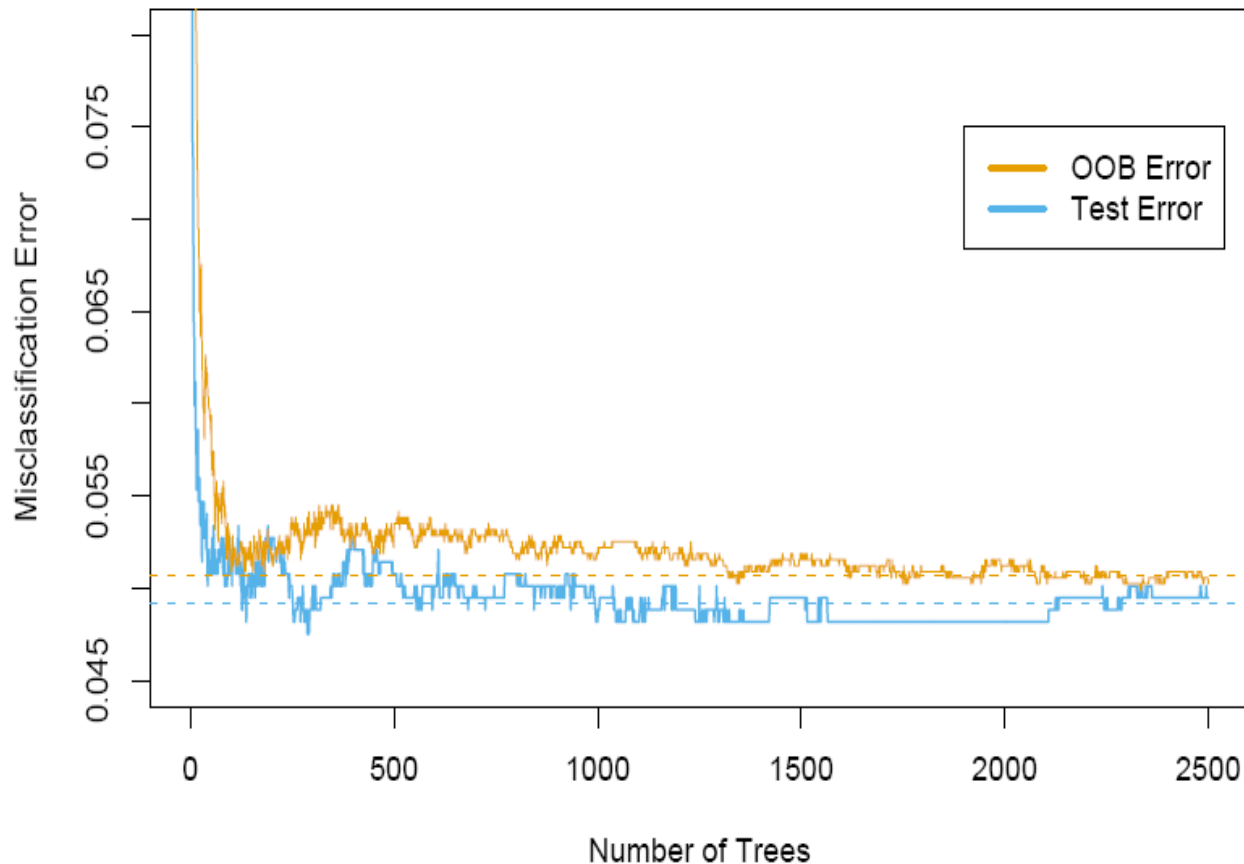


FIGURE 15.4. OOB error computed on the `spam` training data, compared to the test error computed on the test set.

Importancia de las variables

Dos criterios:

- Gini: Media de la reducción del Gini cuando se usó cada variable, promediada en el RF
- Randomization: Aumento del error OOB cuando cada variable se randomiza por separado
- Los dos se estiman fácilmente
- Suelen ser equivalentes (con excepciones)

Gráficas de proximidad

- Modo interno de proyectar datos en bajas dimensiones (como PCA o MDS)
- Calcula una matriz de distancias entre los datos basada en cuantas veces los puntos terminan en un mismo nodo de un árbol
 - Puntos cercanos deberían terminar juntos más seguido
- Hace un MDS de esas distancias

Ejemplo

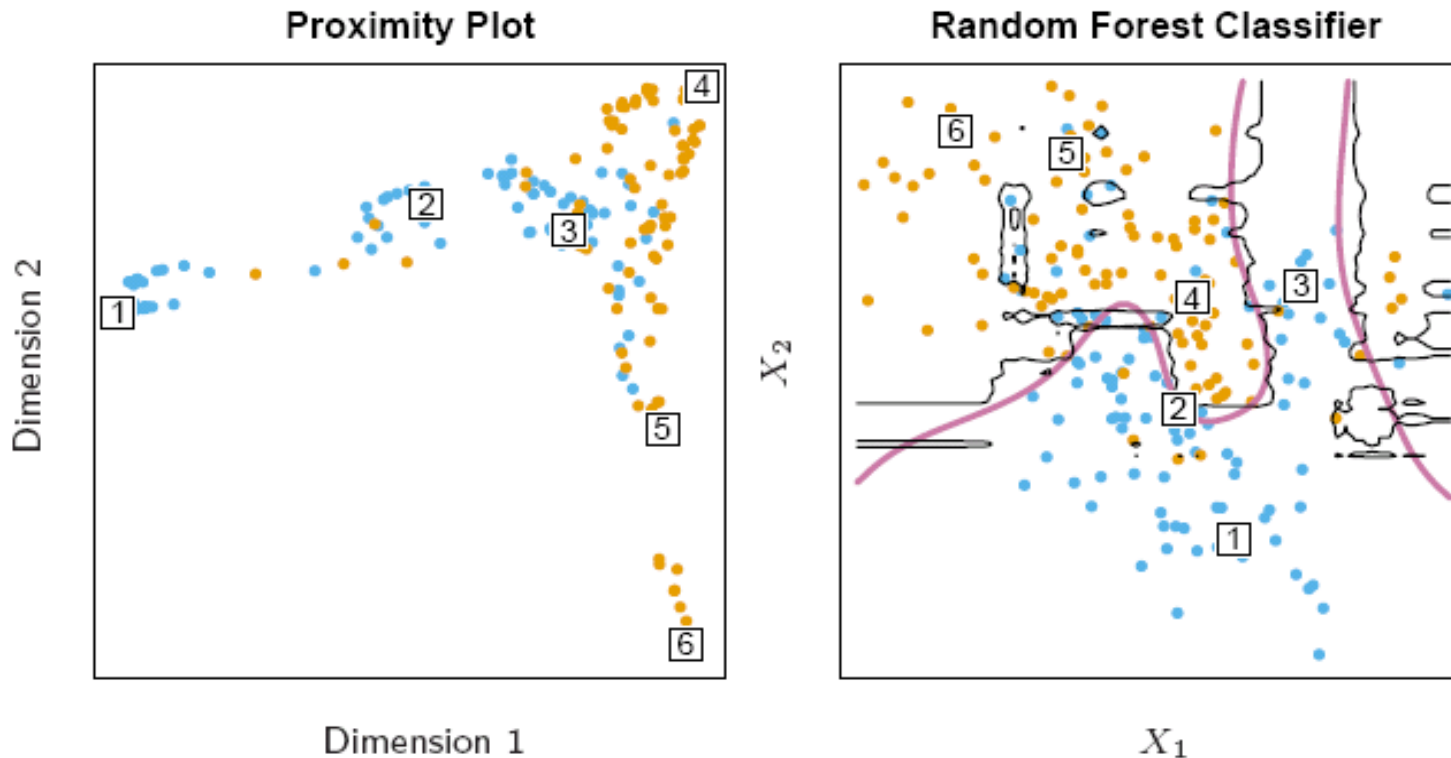


FIGURE 15.6. (Left): Proximity plot for a random forest classifier grown to the mixture data. (Right): Decision boundary and training data for random forest on mixture data. Six points have been identified in each plot.

Resumen: RF

- Mejora de bagging sólo para árboles
- Mejores predicciones que Bagging.
- Muy usado. Casi automático.
- Resultados comparables a los mejores métodos actuales.
- Subproductos útiles, sobre todo la estima OOB y la importancia de variables.

Boosting



El dilema sesgo-varianza

- Los predictores sin sesgo tienen alta varianza (y al revés)
- Hay dos formas de resolver el dilema:
 - Disminuir la varianza de los predictores sin sesgo
 - Construir muchos predictores y promediarlos: Bagging y Random Forest
 - Reducir el sesgo de los predictores estables
 - Construir una secuencia tal que la combinación tenga menos sesgo: Boosting

Introducción a boosting

- Es relativamente fácil construir un clasificador con un error apenas menor al 50% en cualquier problema de dos clases balanceado. (weaklearner, rules-of-thumb)
- Es mucho más difícil conseguir uno que dé un error arbitrariamente chico en cualquier problema (stronglearner)
- Problema central en teoría del aprendizaje: se puede crear un stronglearner combinando weaklearners?

Idea central

- Construir una secuencia de clasificadores débiles, donde cada uno aprenda un poco de lo que le falta a la secuencia previa
- Como los clasificadores débiles pueden mejorar un poco en cualquier problema, la secuencia converge a error cero

Cómo se implementa?

- Es un método de ensemble, hay que definir 2 cosas:
 - Como se construye cada clasificador, con que datos
 - Como se combinan los clasificadores



Conjuntos de entrenamiento

- Se trabaja con bootstraps, como en bagging.
- Pero la muestra no se toma con pesos estadísticos uniformes.
- Se busca poner énfasis en aprender los puntos que fueron mal clasificados previamente por el ensamble
- Para eso, se le aumenta el peso estadístico a los errores, y se le baja a los aciertos

Combinación

- Se hace una suma de votos (como Bagging)
- Los pesos en la suma no son uniformes
 - Se da más peso a los clasificadores que son mejores

Algoritmo Adaboost (1)

- given training set $(x_1, y_1), \dots, (x_m, y_m)$
- $y_i \in \{-1, +1\}$ correct label of instance $x_i \in X$
- for $t = 1, \dots, T$:
 - construct distribution D_t on $\{1, \dots, m\}$ 
 - find weak classifier (“rule of thumb”)
 $h_t : X \rightarrow \{-1, +1\}$
with small error ϵ_t on D_t :
 $\epsilon_t = \Pr_{D_t}[h_t(x_i) \neq y_i]$
- output final classifier H_{final} 

Algoritmo Adaboost (2)

- constructing D_t :

- $D_1(i) = 1/m$
- given D_t and h_t :

$$\begin{aligned} D_{t+1}(i) &= \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } y_i = h_t(x_i) \\ e^{\alpha_t} & \text{if } y_i \neq h_t(x_i) \end{cases} \\ &= \frac{D_t(i)}{Z_t} \exp(-\alpha_t y_i h_t(x_i)) \end{aligned}$$

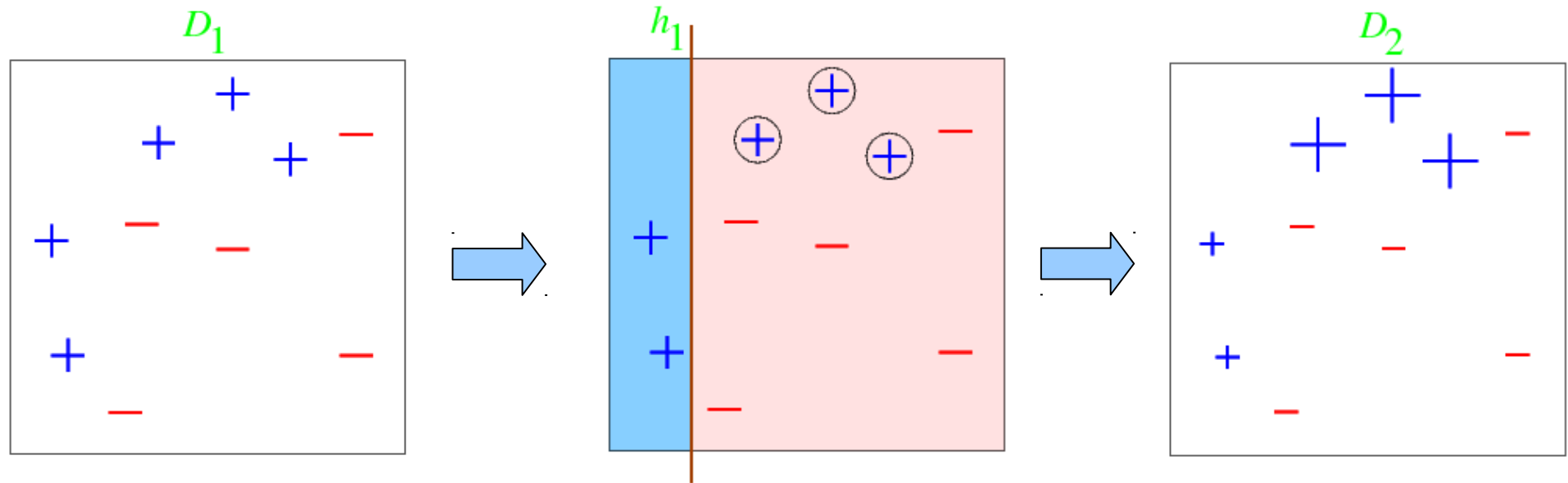
where $Z_t =$ normalization constant

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right) > 0$$

- final classifier:

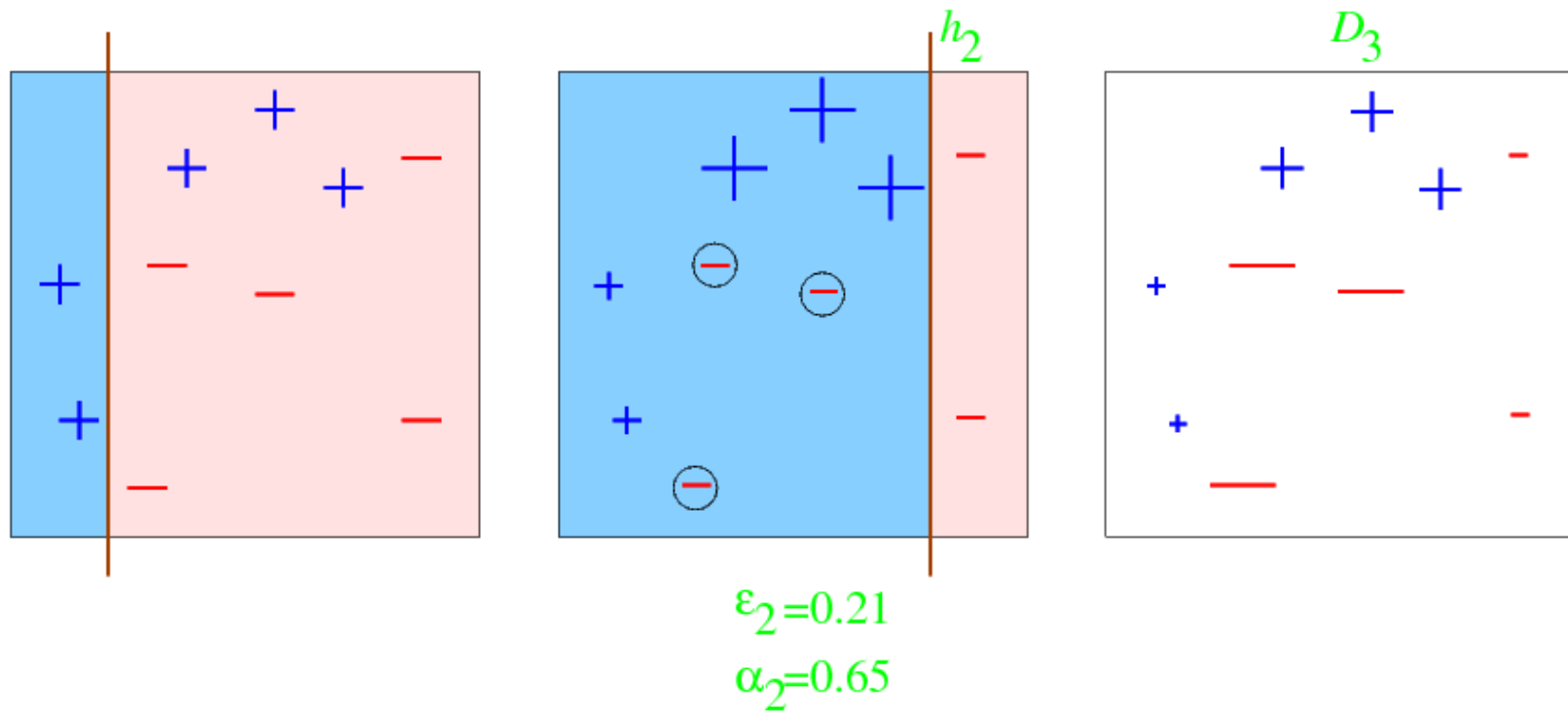
- $H_{\text{final}}(x) = \text{sign} \left(\sum_t \alpha_t h_t(x) \right)$

Ejemplo

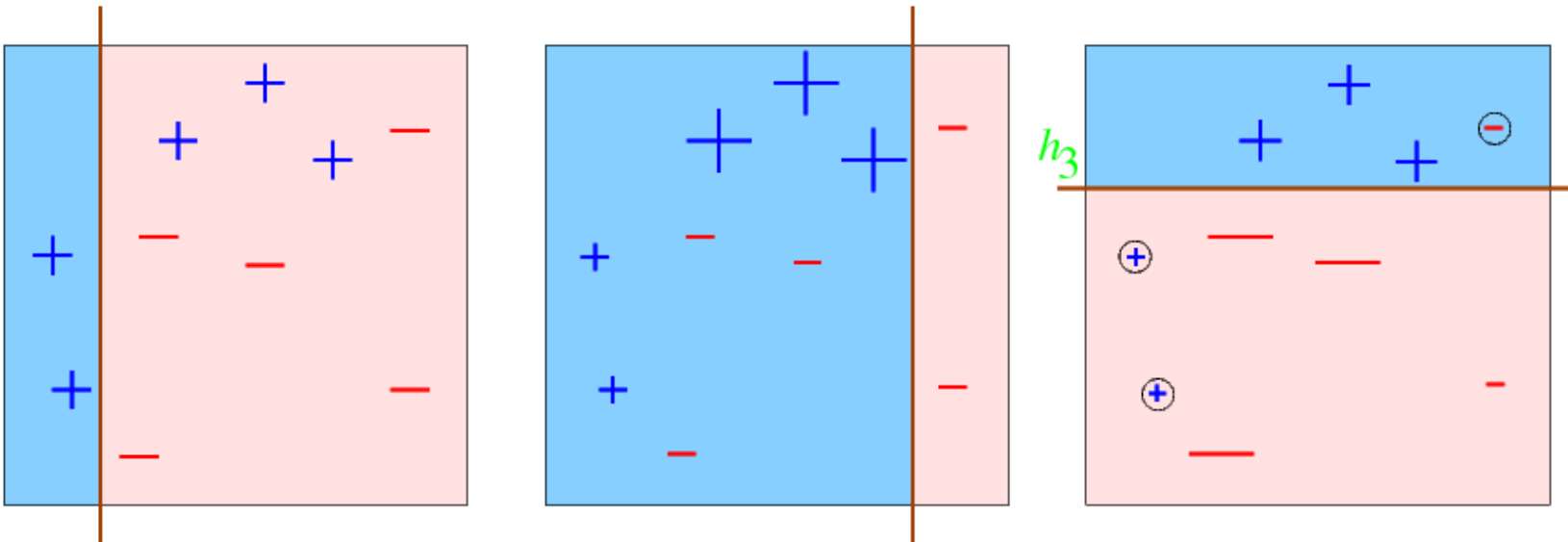


Weak learner: partición
paralela a un eje

Ejemplo (2)



Ejemplo (3)

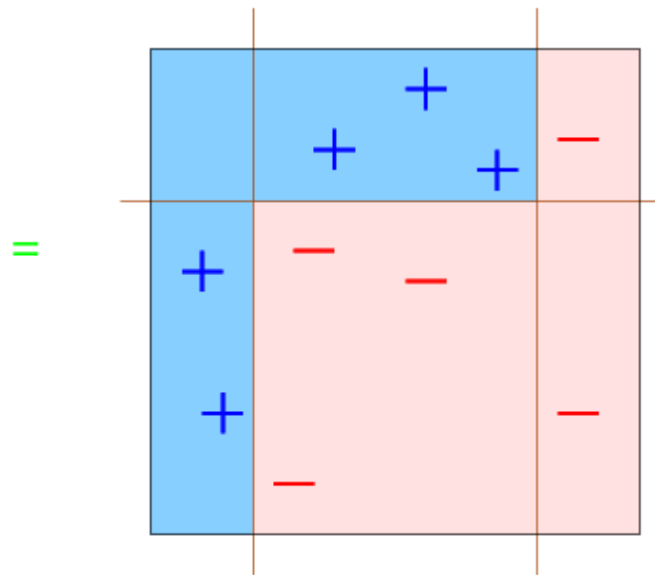


$$\epsilon_3 = 0.14$$

$$\alpha_3 = 0.92$$

Ejemplo (y 4)

$$H_{\text{final}} = \text{sign} \left(0.42 \begin{array}{|c|} \hline \text{blue} \\ \hline \text{red} \end{array} + 0.65 \begin{array}{|c|} \hline \text{blue} \\ \hline \text{red} \end{array} + 0.92 \begin{array}{|c|} \hline \text{blue} \\ \hline \text{red} \end{array} \right)$$

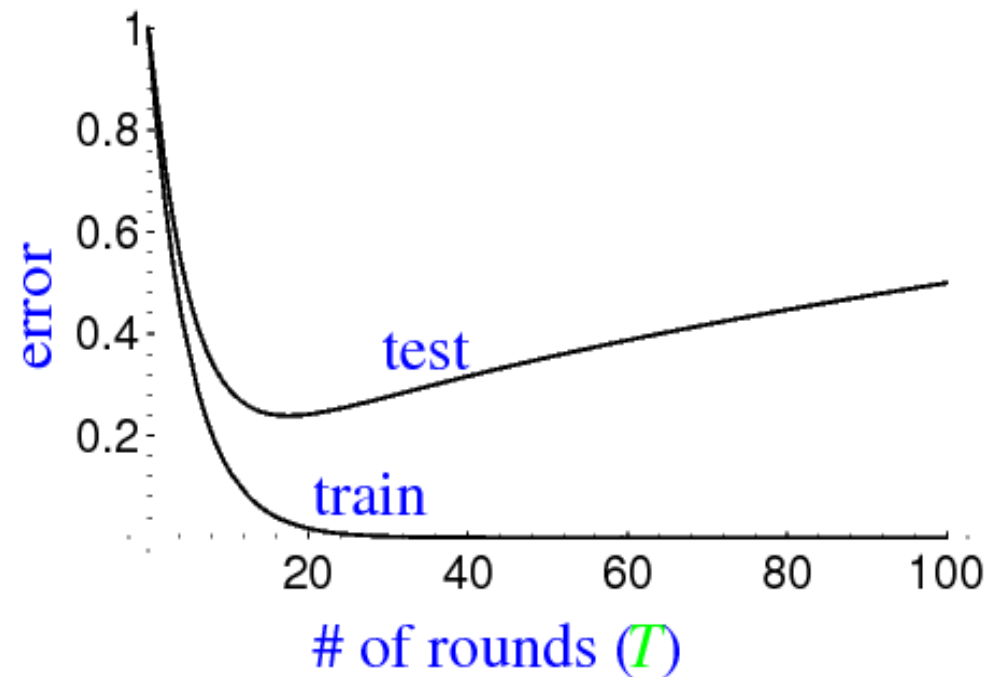


Error de entrenamiento

- Se puede probar que el error sobre los datos de ajuste converge a cero
 - Si el weaklearn de verdad puede siempre hacer algo mejor que el azar

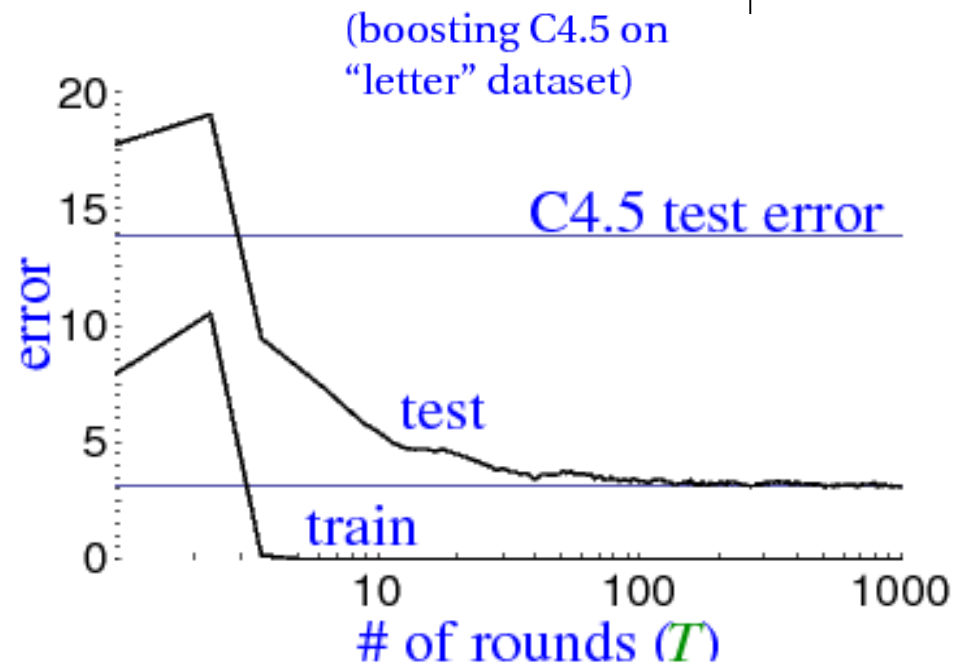
Error de test

- Se espera sobreajuste
- El error de train tiende a cero
- El clasificador se vuelve más complejo a cada paso



Error de test real

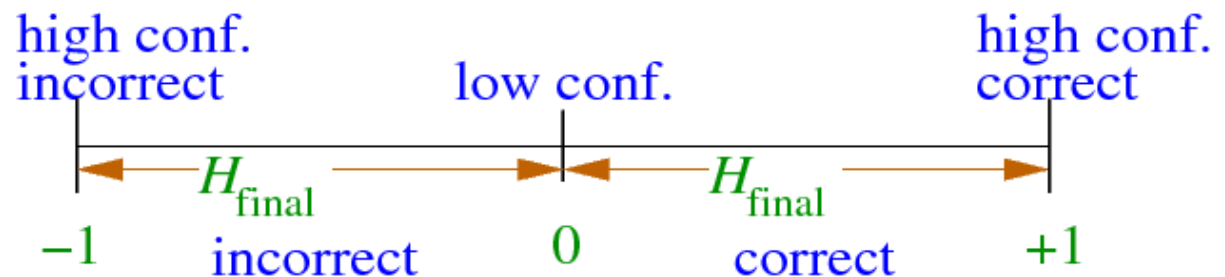
- Pero en realidad no sobreajusta
- Peor, el error en test sigue bajando con el error de train en 0.
- Se necesita una explicación!



	# rounds		
	5	100	1000
train error	0.0	0.0	0.0
test error	8.4	3.3	3.1

Margen

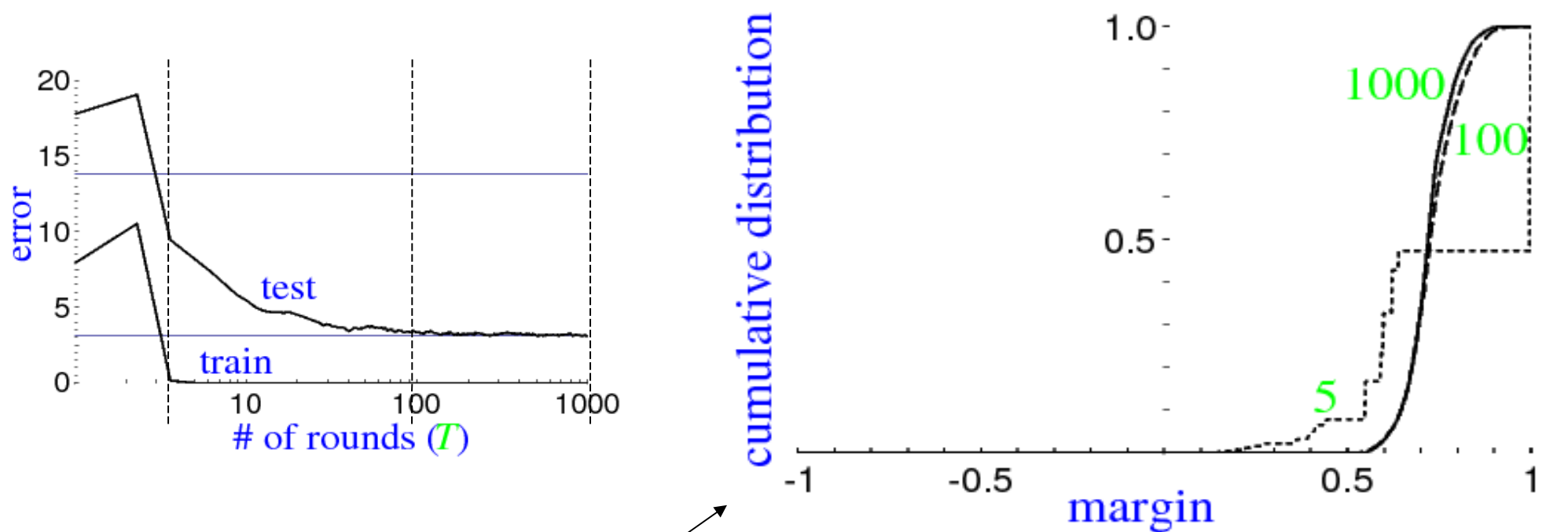
- El error de test solo mide la clase
- La respuesta de adaboost es mas compleja, es la suma pesada de votos por una clase y la otra
- Hay que tener en cuenta la “fuerza” de la respuesta, la calidad
- Margen: (fracción pesada de votos correctos) – (fracción incorrecta)



Margen y error

- Teorema 1: Mayor margen es equivalente a una menor cota superior en el error de test
- Teorema 2: Boosting incrementa en cada iteración el margen de los datos
- Conclusión: El error de test baja porque boosting aumenta el margen

Distribución del margen



Siempre sobre el conjunto de entrenamiento

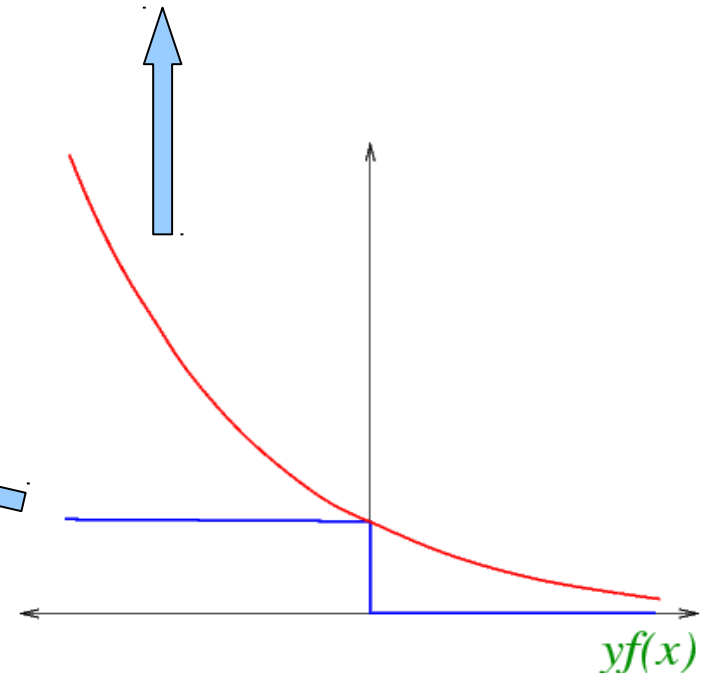
	# rounds		
	5	100	1000
train error	0.0	0.0	0.0
test error	8.4	3.3	3.1
% margins ≤ 0.5	7.7	0.0	0.0
minimum margin	0.14	0.52	0.55

Otras derivaciones

- Costo exponencial
 - Adaboost minimiza a cada paso una función de costo exponencial

$$\prod_t Z_t = \frac{1}{m} \sum_i \exp(-y_i f(x_i))$$

El costo exponencial es
una cota superior del
error de clasificación



Otras derivaciones (2)

- Teoría de juegos
 - Boosting es un juego entre el booster y el weeklearn. La estrategia óptima da el algoritmo Adaboost
- Y varias más.

Extensiones

- Originalmente solo clasificación binaria
- Fue extendido eficientemente para:
 - Regresión
 - Clasif. Multiclase
 - Ranking

Resultados

Boosting sobre árboles

De: *Popular ensemble methods*, D.Opitz y R. Maclin, **Journal of Artificial Intelligence research**, 11:169-198, 1999

Stan es modelo individual, Bag es Bagging, Simp es ensemble variando solo la condición inicial, Ada es adaboost

Data Set	C4.5			
	Stan	Bag	Arc	Ada
breast-cancer-w	5.0	3.7	3.5	3.5
credit-a	14.9	13.4	14.0	13.7
credit-g	29.6	25.2	25.9	26.7
diabetes	27.8	24.4	26.0	25.7
glass	31.3	25.8	25.5	23.3
heart-cleveland	24.3	19.5	21.5	20.8
hepatitis	21.2	17.3	16.9	17.2
house-votes-84	3.6	3.6	5.0	4.8
hypo	0.5	0.4	0.4	0.4
ionosphere	8.1	6.4	6.0	6.1
iris	5.2	4.9	5.1	5.6
kr-vs-kp	0.6	0.6	0.3	0.4
labor	16.5	13.7	13.0	11.6
letter	14.0	7.0	4.1	3.9
promoters-936	12.8	10.6	6.8	6.4
ribosome-bind	11.2	10.2	9.3	9.6
satellite	13.8	9.9	8.6	8.4
segmentation	3.7	3.0	1.7	1.5
sick	1.3	1.2	1.1	1.0
sonar	29.7	25.3	21.5	21.7
soybean	8.0	7.9	7.2	6.7
splice	5.9	5.4	5.1	5.3
vehicle	29.4	27.1	22.5	22.9

Resultados

Boosting sobre ANN

De: *Popular ensemble methods*, D.Opitz y R. Maclin, **Journal of Artificial Intelligence research**, 11:169-198, 1999

Stan es modelo individual, Bag es Bagging, Simp es ensemble variando solo la condición inicial, Ada es adaboost

Data Set	Neural Network				
	Stan	Simp	Bag	Boosting	
	Arc	Ada			
breast-cancer-w	3.4	3.5	3.4	3.8	4.0
credit-a	14.8	13.7	13.8	15.8	15.7
credit-g	27.9	24.7	24.2	25.2	25.3
diabetes	23.9	23.0	22.8	24.4	23.3
glass	38.6	35.2	33.1	32.0	31.1
heart-cleveland	18.6	17.4	17.0	20.7	21.1
hepatitis	20.1	19.5	17.8	19.0	19.7
house-votes-84	4.9	4.8	4.1	5.1	5.3
hypo	6.4	6.2	6.2	6.2	6.2
ionosphere	9.7	7.5	9.2	7.6	8.3
iris	4.3	3.9	4.0	3.7	3.9
kr-vs-kp	2.3	0.8	0.8	0.4	0.3
labor	6.1	3.2	4.2	3.2	3.2
letter	18.0	12.8	10.5	5.7	4.6
promoters-936	5.3	4.8	4.0	4.5	4.6
ribosome-bind	9.3	8.5	8.4	8.1	8.2
satellite	13.0	10.9	10.6	9.9	10.0
segmentation	6.6	5.3	5.4	3.5	3.3
sick	5.9	5.7	5.7	4.7	4.5
sonar	16.6	15.9	16.8	12.9	13.0
soybean	9.2	6.7	6.9	6.7	6.3
splice	4.7	4.0	3.9	4.0	4.2
vehicle	24.9	21.2	20.7	19.1	19.7

Resumen - Pros

- Eficiente, simple y fácil de programar
- Sólo un parámetro ajustable (T , fácil)
- Flexible, funciona con casi cualquier learner
- Efectivo, tiene garantías de funcionamiento (siempre que el weaklearn cumpla)
- Extensión a muchos problemas en ML

Resumen - Contraste

- La performance depende no sólo de los datos, también del Weaklearn
- Falla si:
 - Weak son demasiado fuertes (overfitting)
 - Weak demasiado débiles
- Tiene problemas con el ruido, en particular con el ruido uniforme