

Boosting

Una Introducción

Fernando Arias-Rodríguez

Banco Central de Bolivia

30 de agosto de 2024



- ① Introducción
- ② AdaBoost
- ③ L_2 Boosting
- ④ Gradient Boosting
- ⑤ Gradient Boosting Decision Tree
- ⑥ Inferencia

- 1 Introducción
- 2 AdaBoost
- 3 L_2 Boosting
- 4 Gradient Boosting
- 5 Gradient Boosting Decision Tree
- 6 Inferencia

- Surge de la hipótesis de propulsar un modelo de aprendizaje probable y aproximadamente correcto.
- Suponga que un algoritmo de aprendizaje produce un clasificador a partir de muestras aleatorias provenientes de un proceso generador de datos **desconocido**.
- El trabajo del algoritmo es clasificar nuevas muestras derivadas del proceso generador de datos como ejemplos positivos o negativos del fenómeno de estudio.
- Un algoritmo **débil** produce clasificadores que se desempeñan ligeramente mejor que una adivinación aleatoria.
- Un algoritmo **fuerte** produce clasificadores con una alta precisión, dada una cantidad suficiente de muestras.

- En 1997, Freund & Schapire proponen el algoritmo *Adaptive Boost* (AdaBoost) como un ejemplo de *Boosting* aplicado a clasificación.
- AdaBoost funciona muy bien en la práctica y ha dado lugar al desarrollo de otros algoritmos.
- Si bien este curso es enfocado a regresión, aquí se hace una pequeña excepción y se explicará la lógica de *Boosting* con AdaBoost.

- Un problema típico de clasificación es el siguiente: sea

$$\pi(x) = Pr(y = 1|x)$$

con y igual a 1 con probabilidad $\pi(x)$ y -1 con probabilidad $1 - \pi(x)$.

- El objetivo es predecir el valor desconocido para y , a partir de un conjunto de información conocido, x .

Algunas definiciones previas:

- Sea y una variable binaria, la cual toma los valores $\{-1, 1\}$, la cual se desea predecir.
- Sea $f_m(x)$ el algoritmo débil (un clasificador débil) para predecir y y que se estima con un conjunto amplio de variables x , en una iteración m de un total de M .
- Sea err_m el error del algoritmo débil, $f_m(x)$.
- n es el número de observaciones; ω_i denota el peso i y el símbolo $1_{(\cdot)}$ denota una función indicadora que es igual a 1 si la condición dentro del paréntesis se satisface y cero en otro caso.
- El símbolo $sign(z)$ es 1 si $z > 0$ y -1 si $z < 0$. Por ende, $sign(z) = 1_{(z>0)} - 1_{(z<0)}$.

Tres elementos a tener en cuenta

- 1 Este algoritmo se modifica para que, además de arrojar el resultado del agregado de clasificadores, se obtenga la probabilidad de predicción, $\hat{\pi}(x)$.
- 2 El único hiperparámetro (parámetro especificado por el usuario) es el número de iteraciones, M . En la práctica, se puede usar validación cruzada o criterios de información como AIC.
- 3 En la mayoría de las aplicaciones, la elección de M se encuentra incorporada en la metodología, por lo que el usuario no debería preocuparse por su determinación.

1 Introducción

2 AdaBoost

Descripción del algoritmo

Ejemplo de cómo funciona AdaBoost

3 L_2 Boosting

4 Gradient Boosting

5 Gradient Boosting Decision Tree

6 Inferencia

- El algoritmo débil más usado es el árbol de clasificación, donde su versión más simple (llamado *the stump*), toma la forma funcional:

$$f(x_j, a) = \begin{cases} 1 & x_j > a \\ -1 & x_j < a \end{cases}$$

donde a es el parámetro que minimiza la tasa de error

$$\min_a \sum_{i=1}^n \omega_i 1(y_i \neq f(x_{ji}, a))$$

- Ng (2014) utiliza este acercamiento para predecir los ciclos de negocios en EE.UU.
- Problema: clasificar si en los 12 meses de 2001, la economía se encontraba en expansión o recesión.
- Variables: rezagos de 3 meses para: índice de vacantes (*help-wanted index*, HWI), nuevas órdenes de producción (NAPM) y la diferencia entre la tasa de interés de bonos a 10 años y la tasa de interés de la FED (10yr-FF spread (SPREAD)).
- Variable resultado: una variable dicótoma que toma el valor de 1 si hay recesión en el mes y -1 si hay una expansión, según la *National Bureau of Economic Research* (NBER).

- La información se encuentra en las columnas 2 a 5 del Cuadro 1.
- Se usa un *stump* como el algoritmo débil (f).
- Este *stump* usa un umbral escogido de manera óptima, para dividir la información en dos particiones (como ya se vio en las secciones anteriores de *Regression Trees*).
- El algoritmo comienza con asignar un peso igual a cada observación, es decir $\omega_i^{(1)} = \frac{1}{n}$ con $n = 12$ (ver paso 1).
- Se genera la primera bifurcación del árbol a partir de la primera variable, HWI (se minimiza el error con HWI y se clasifica 1 si $HWI < 0,044$).

Fecha	Datos rezagados 3 meses				NBER	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$	$f_5(x)$
	HWI	NAPM	SPREAD	HWI		NAPM	HWI	SPREAD	NAPM	
	-0,066	48,550	0,244	< -0,044		< 49,834	< -0,100	> -0,622	< 47,062	
2001.1	0,014	51,1	-0,77	-1	-1	-1	-1	-1	-1	
2001.2	-0,091	50,3	-0,79	-1	1	-1	-1	-1	-1	
2001.3	0,082	52,8	-1,16	-1	-1	-1	-1	-1	-1	
2001.4	-0,129	49,8	-0,82	1	1	1	1	1	1	
2001.5	-0,131	50,2	-0,39	1	1	-1	1	1	1	
2001.6	-0,111	47,7	-0,42	1	1	1	1	1	1	
2001.7	-0,056	47,2	0,34	1	1	1	1	1	1	
2001.8	-0,103	45,4	1,18	1	1	1	1	1	1	
2001.9	-0,093	47,1	1,31	1	1	1	1	1	1	
2001.1	-0,004	46,8	1,47	1	-1	1	-1	1	1	
2001.11	-0,174	46,7	1,32	1	1	1	1	1	1	
2001.12	-0,007	47,5	1,66	-1	-1	1	-1	1	-1	
c					0,804	1,098	0,71	0,783	0,575	
Error rate					0,167	0,1	0,138	0,155	0	

Cuadro 1: Tomado de Ng (2014)

- El valor de dicho umbral, que minimiza el error, es igual a $-0,044$. El mismo procedimiento se repite con NAPM y con SPREAD. Luego se comparan los tres errores y se escoge la partición según la variable que tenga el menor error (ver paso 2a1).
- En la primera iteración, resulta ser HWI la variable con menor error. Así, el primer modelo le asigna el valor de 1 a $N\hat{B}ER$ si $HWI_i < -0,044$.
- El resultado de esta primera iteración se encuentra en la columna 6 del Cuadro 1.

Fecha	Datos rezagados 3 meses				NBER	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$	$f_5(x)$
	HWI	NAPM	SPREAD			HWI	NAPM	HWI	SPREAD	NAPM
	-0,066	48,550	0,244			< -0,044	< 49,834	< -0,100	> -0,622	< 47,062
2001.1	0,014	51,1	-0,77	-1		-1	-1	-1	-1	-1
2001.2	-0,091	50,3	-0,79	-1		1	-1	-1	-1	-1
2001.3	0,082	52,8	-1,16	-1		-1	-1	-1	-1	-1
2001.4	-0,129	49,8	-0,82	1		1	1	1	1	1
2001.5	-0,131	50,2	-0,39	1		1	-1	1	1	1
2001.6	-0,111	47,7	-0,42	1		1	1	1	1	1
2001.7	-0,056	47,2	0,34	1		1	1	1	1	1
2001.8	-0,103	45,4	1,18	1		1	1	1	1	1
2001.9	-0,093	47,1	1,31	1		1	1	1	1	1
2001.10	-0,004	46,8	1,47	1		-1	1	-1	1	1
2001.11	-0,174	46,7	1,32	1		1	1	1	1	1
2001.12	-0,007	47,5	1,66	-1		-1	1	-1	1	-1
c						0,804	1,098	0,71	0,783	0,575
Error rate						0,167	0,1	0,138	0,155	0

Cuadro 1: Tomado de Ng (2014)

- Comparado con los valores observados de la serie observada de la NBER (columna 5, Cuadro 1), los meses 2 y 10 están incorrectamente clasificados (números en rojo, Cuadro 1).
- Lo anterior genera una tasa de incorrecta clasificación de $\frac{2}{12} = 0,167$. Esto es igual a err_1 como se describe en paso 2aII.
- Aplicando la fórmula del paso 2c, $c_1 = \log\left(\frac{1-err_1}{err_1}\right) = 1,607$, o 0,804 para cada observación.
- Con este resultado se recalculan los pesos (*weights*) $\omega_i^{(2)}$ se actualizan, como se propone en el paso 2d.

- Para cada uno de los dos meses mal clasificados,
 $\omega_k = e^{1,607} * \frac{1}{12} = 0,416, k = 2, 10.$
- Para los restantes 10 meses, $\omega_i = \frac{1}{12} = 0,0834,$
 $i = 1, 3, 4, \dots, 9, 11, 12.$
- Reponderando, para que la suma de pesos sea 1, $\omega_k = 0,25,$
 $k = 2, 10$ y $\omega_i = 0,05, i = 1, 3, 4, \dots, 9, 11, 12.$
- Con estos nuevos pesos se estima un nuevo modelo
(clasificador, como en paso 2a1). En esta segunda iteración, la
bifurcación con NAPM resulta ser la de menor error ponderado
(1 si $NAPM < 49.834$). Sus resultados se resumen en la
columna 7 del Cuadro 1.

Fecha	Datos rezagados 3 meses				NBER	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$	$f_5(x)$
	HWI	NAPM	SPREAD			HWI	NAPM	HWI	SPREAD	NAPM
	-0,066	48,550	0,244			< -0,044	< 49,834	< -0,100	> -0,622	< 47,062
2001.1	0,014	51,1	-0,77	-1	-1	-1	-1	-1	-1	-1
2001.2	-0,091	50,3	-0,79	-1	1	-1	-1	-1	-1	-1
2001.3	0,082	52,8	-1,16	-1	-1	-1	-1	-1	-1	-1
2001.4	-0,129	49,8	-0,82	1	1	1	1	1	1	1
2001.5	-0,131	50,2	-0,39	1	1	-1	1	1	1	1
2001.6	-0,111	47,7	-0,42	1	1	1	1	1	1	1
2001.7	-0,056	47,2	0,34	1	1	1	1	1	1	1
2001.8	-0,103	45,4	1,18	1	1	1	1	1	1	1
2001.9	-0,093	47,1	1,31	1	1	1	1	1	1	1
2001.10	-0,004	46,8	1,47	1	-1	1	-1	1	1	1
2001.11	-0,174	46,7	1,32	1	1	1	1	1	1	1
2001.12	-0,007	47,5	1,66	-1	-1	1	-1	1	1	-1
c						0,804	1,098	0,71	0,783	0,575
Error rate						0,167	0,1	0,138	0,155	0

Cuadro 1: Tomado de Ng (2014)

- Comparado con la columna 5, se ve que los meses 5 y 12 se encuentran mal clasificados (Cuadro 1, observaciones en azul).
- Nótese que, en este caso, la tasa de error baja a 0,100 ($\omega_i = 0,05$ para las observaciones 5 y 12).
- Usando la fórmula del paso 2d de nuevo, los pesos de la nueva iteración, $\omega_t^{(3)}$, son iguales a 0,25 para los meses 5 y 12, 0,138 para los meses 2 y 10 y 0,027 para los meses remanentes.
- Siguiendo la fórmula del paso 3, es posible ir calculando el signo de la estimación, el cual resulta de operar:

$$F_2(x) = 0,804 \times 1_{(HWI < -0,044)} + 1,098 \times 1_{(NAPM < 49,834)}$$

- Este mismo procedimiento se sigue hasta que la tasa de error sea lo más pequeña posible.
- En este ejemplo, luego de 5 iteraciones, y utilizando dos *stumps* adicionales con $\text{SPREAD} > -0,622$ y $\text{NAPM} < 47,062$, se llega a que el error es cero y la variable NBER se encuentra correctamente clasificada (ver columna 10, Cuadro 1).
- Nótese que el algoritmo (clasificador) fuerte es un ensamble de 5 algoritmos (clasificadores) débiles, definido por el $\text{sign}(F_5(x))$, con

$$\begin{aligned} F_5(x) = & 0,804 \times 1_{(HWI < -0,044)} + 1,098 \times 1_{(NAPM < 49,834)} \\ & + 0,710 \times 1_{(HWI < -0,100)} + 0,783 \times 1_{(SPREAD > -0,622)} \\ & + 0,575 \times 1_{(NAPM < 47,062)} \end{aligned}$$

Fecha	Datos rezagados 3 meses			NBER	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$	$f_5(x)$
	HWI	NAPM	SPREAD		HWI	NAPM	HWI	SPREAD	NAPM
	-0,066	48,550	0,244		< -0,044	< 49,834	< -0,100	> -0,622	< 47,062
2001.1	0,014	51,1	-0,77	-1	-1	-1	-1	-1	-1
2001.2	-0,091	50,3	-0,79	-1	1	-1	-1	-1	-1
2001.3	0,082	52,8	-1,16	-1	-1	-1	-1	-1	-1
2001.4	-0,129	49,8	-0,82	1	1	1	1	1	1
2001.5	-0,131	50,2	-0,39	1	1	-1	1	1	1
2001.6	-0,111	47,7	-0,42	1	1	1	1	1	1
2001.7	-0,056	47,2	0,34	1	1	1	1	1	1
2001.8	-0,103	45,4	1,18	1	1	1	1	1	1
2001.9	-0,093	47,1	1,31	1	1	1	1	1	1
2001.1	-0,004	46,8	1,47	1	-1	1	-1	1	1
2001.11	-0,174	46,7	1,32	1	1	1	1	1	1
2001.12	-0,007	47,5	1,66	-1	-1	1	-1	1	-1
c					0,804	1,098	0,71	0,783	0,575
Error rate					0,167	0,1	0,138	0,155	0

Cuadro 1: Tomado de Ng (2014)

- Nótese que AdaBoost se puede usar una variable más de una vez.
- Los pesos son ajustados en cada iteración para que se enfoquen más en las observaciones mal clasificadas.
- La decisión final se basa en un ensamble de modelos.
- Ninguna variable individual puede lograr una clasificación correcta, la cual es la premisa de una regla de decisión basada en ensambles de modelos.

- 1 Introducción
- 2 AdaBoost
- 3 L_2 Boosting
- 4 Gradient Boosting
- 5 Gradient Boosting Decision Tree
- 6 Inferencia

- Bühlmann (2003, 2006) propone un algoritmo basado en las premisas del modelo de regresión lineal, con el cual se puede aplicar el principio de *Boosting*.
- Este acercamiento es muy útil, especialmente cuando se tiene un número grande de variables. Inclusive, funciona cuando hay más variables que observaciones.
- Se parte de una regresión lineal simple, $y = x\beta + u$, donde y es la variable dependiente, x es la variable independiente y $u \sim N(0, 1)$.
- La idea básica de L_2 Boosting es usar una variable explicativa al tiempo. En otras palabras, usa iterativamente la técnica de mínimos cuadrados ordinarios para explicar los residuales de regresiones hechas en iteraciones anteriores.

El algoritmo funciona así:

- 1 Se parte de una muestra de entrenamiento.
- 2 Para $m = 1$ hasta M :
 - a Para $j = 1$ hasta k (para cada variable):
 - i Estimar la regresión $y_i = \beta_{m,0,j} + \beta_{m,ji}x_{ji} + u_i$ usando mínimos cuadrados, y_i y x_{ji} .
 - ii Computar $err_{mj} = 1 - R_{mj}^2$, donde R_{mj}^2 es el coeficiente de determinación de la regresión en paso 2ai.
 - b Hallar $\hat{j}_m = \min_j err_{mj}$.
 - c Se definen los nuevos y_i como y_i del paso previo menos \hat{y}_i , es decir, $y_i - \hat{\beta}_{m,0,\hat{j}_m} - \hat{\beta}_{m,\hat{j}_m}x_{\hat{j}_m,i}$, $i = 1, \dots, n$.
- 3 Calcular el modelo de regresión final

$$F_M(x) = \sum_{m=1}^M \hat{\beta}_{m,0,\hat{j}_m} + \hat{\beta}_{m,\hat{j}_m}x_{\hat{j}_m}.$$

- El parámetro M debe ser calibrado previamente. Para hacerlo, suele usarse validación cruzada o algún criterio de información, como Akaike por ejemplo.
- Según Bühlmann (2003), $\lim_{M \rightarrow \infty} MSE = \sigma^2$, ya que el sesgo al cuadrado decae exponencialmente y la varianza se incrementa exponencialmente despacio cuando M crece.
- L_2 Boosting es computacionalmente simple y exitoso si el algoritmo débil es suficientemente débil. Si este es muy fuerte, habrá sobreajuste.
- L_2 Boosting no solo funciona con regresiones lineales. Puede usarse *splines* o *Regression Trees* como los algoritmos débiles.

- 1 Introducción
- 2 AdaBoost
- 3 L_2 Boosting
- 4 Gradient Boosting**
- 5 Gradient Boosting Decision Tree
- 6 Inferencia

- $f_m(x)$ se define como:

$$f_m(x) = E_y \left[- \frac{\partial L(y, F(x))}{\partial F(x)} \Big| x \right]_{F(x)=F_{m-1}(x)}$$

y marca la dirección en la que $R(F)$ decrece más rápido.

- c_m se puede calcular, dado $f_m(x)$, así:

$$c_m = \min_{c_m} E_{y,x} L(y, F_{m-1}(x) + c_m f_m(x))$$

- Finalmente, la función estimada se actualiza de la siguiente manera:

$$F_m(x) = F_{m-1}(x) + c_m f_m(x)$$

El algoritmo de Gradient Boosting es el siguiente:

- 1 Se calcula $F_0(x) = \min_{\text{constante}} \sum_{i=1}^n L(y_i, \text{constante})$.
- 2 Desde $m = 1$ hasta M :
 - a Calcular pseudo-residuales $r_i^m = - \left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)} \right]_{F(x)=F_{m-1}(x)}$
 $i = 1, \dots, n$.
 - b Calcular $f_m(x) = \min_{f_m(x)} \sum_{i=1}^N (r_i^m - f_m(x_i))^2$.
 - c Calcular $c_m = \min_c \sum_{i=1}^N L(y_i, F_{m-1}(x_i) + c_m f_m(x_i))$
 - d Calcular $F_m(x) = F_{m-1}(x) + c_m f_m(x)$.
- 3 Calcular $F_M(x) = \sum_{m=1}^M c_m f_m(x)$.

- En términos de este acercamiento, el gradiente negativo también se conoce como "pseudo residuales", r_m y se calculan en cada iteración.
- Nótese que en el paso 2a del algoritmo, cualquier criterio que estime la media condicional del fenómeno de estudio puede ser usado.
- La elección más popular para hacer las veces el algoritmo es, en nuestro caso, *Regression Trees*.
- Esto da lugar al método conocido como *Gradient Boosting Decision Tree*.

- 1 Introducción
- 2 AdaBoost
- 3 L_2 Boosting
- 4 Gradient Boosting
- 5 Gradient Boosting Decision Tree**
- 6 Inferencia

- Usado de manera masiva para implementar modelos No-lineales.
- Combina la teoría de árboles de decisión y de *gradient boosting*, las cuales ya hemos revisado.
- En pocas palabras, *Boosting Tree* es un método de agregación de *Regression Trees* similar a *Random Forest*, pero con una diferencia marcada: aquí, **cada nuevo árbol crece a partir del error del árbol que creció en la iteración anterior.**

El algoritmo consta de los pasos:

- ① Estimar los primeros residuales,
$$r_i^0 = -2(y_i - \bar{y}) = -2(y_i - f_1(x_i)).$$
- ② Desde $m = 1$ hasta M :
 - Ⓐ A partir de nuevas muestras (r_i^m, x_i) , $i = 1, \dots, n$ se estima un *Regression Tree*, $h_m(x)$.
 - Ⓑ Sea $f_{m+1}(x) = f_m(x) + \lambda_m h_m(x)$, entonces
$$\lambda_m = \min_{\lambda} L(y, f_m(x) + \lambda h_m(x)).$$
 - Ⓒ Actualizar $f_{m+1}(x)$ vía $f_{m+1} = f_m(x) + \lambda_m h_m(x)$.
- ③ Calcular el *Gradient Boosting Decision Tree*,
$$F_M(x) = \sum_{m=1}^M \lambda_m f_m(x).$$

- Para implementar *Gradient Boosting Decision Tree*, se necesita escoger dos hiperpárametros:
 - 1 N , el número de nodos terminales en los árboles.
 - 2 M , el número de iteraciones en el procedimiento de *boosting*.
- Hastie *et al.* (2009) comentan que típicamente $4 < N < 8$ funciona bien para *boosting*. los resultados son insensibles a la elección de N en este rango.
- $N = 2$ es insuficiente para muchas aplicaciones y $N > 10$ es poco probable que sea requerido.
- Para definir M se deben aplicar dos técnicas del reino de la regularización.

Las dos técnicas de regularización para este caso son:

- 1 *Early Stopping*: consiste en controlar el número de iteraciones en la metodología. Específicamente, se trata M como un hiperparámetro, definiéndolo con validación cruzada.
- 2 *Shrinkage Method*: se trata de añadir un parámetro de contracción durante el proceso de entrenamiento. Esto consiste en contraer el *step size*, añadiendo el parámetro de contracción v :

$$f_{m+1}(x) = f_m(x) + v\lambda_m h_m(x)$$

La intención es ralentizar el proceso de aprendizaje de la metodología, para hacerlo más preciso en cada iteración. La consecuencia de tener $v < 1$ es el tener que aumentar M para minimizar el error.

- 1 Introducción
- 2 AdaBoost
- 3 L_2 Boosting
- 4 Gradient Boosting
- 5 Gradient Boosting Decision Tree
- 6 Inferencia**

- $$I_j^2 = \frac{1}{M} \sum_{m=1}^M I_j^2(m)$$

$$I_j^2(m) = \sum_{t=1}^{T_m-1} e_t^2 I(v(t)_m = j)$$

A set of small navigation icons typically found in Beamer presentations, including symbols for back, forward, search, and other slide controls.

- A diferencia de *Random Forest*, donde se puede medir la importancia de una variable a partir de *Out-of-Bag errors* (OOB), en este acercamiento no hay muestra para hacer dicho cálculo.
- Así, solo puede usarse I_j^2 .
- En la práctica, OOB e I_j^2 muestran resultados similares y I_j^2 funciona muy bien cuando M es muy grande.