Boosting Trees

Big Data y Machine Learning para Economía Aplicada

Ignacio Sarmiento-Barbieri

Universidad de los Andes

Agenda

1. Recap
   - Árboles
   - Bagging
     - Random Forests
2. Boosting
   - AdaBoost
   - Boosting Trees
   - XGBoost
3. Causal Trees
   - Causality Review: ATE, CATE, HTE
   - Empirical Example

Agenda

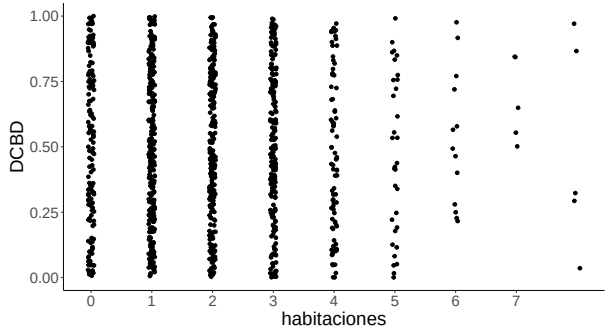
1. Recap
   - Árboles
   - Bagging
     - Random Forests
2. Boosting
   - AdaBoost
   - Boosting Trees
   - XGBoost
3. Causal Trees
   - Causality Review: ATE, CATE, HTE
   - Empirical Example

Agenda

Árboles

- Queremos predecir:

$$Precio = f(habitaciones, Distancia\,al\,CBD) \tag{1}$$



Motivación

- Podemos asumir f es lineal

$$f(\mathbf{x}_i; \beta) = \beta_0 + \beta_1 Habitaciones_i + \beta_2 DCBD_i + u_i \tag{2}$$

- o ajustar un modelo flexible e interpretable como son los arboles

$$f(x_i; \{R_j, \gamma_j\}_1^J) = T(x_i; \{R_j, \gamma_j\}_1^J) \tag{3}$$

Agenda

1. Recap
   - Árboles
   - Bagging
     - Random Forests
2. Boosting
   - AdaBoost
   - Boosting Trees
   - XGBoost
3. Causal Trees
   - Causality Review: ATE, CATE, HTE
   - Empirical Example

Bagging

- Problema con CART: pocos robustos.
- Podemos mejorar mucho el rendimiento mediante la agregación
- Idea: la varianza del promedio es menor que la de una sola predicción.

Bagging

- Bagging:
  - Obtenga repetidamente muestras aleatorias $(X_i^b, Y_i^b)_{i=1}^N$ de la muestra observada (bootstrap).
  - Para cada muestra, ajuste un árbol de regresión $\hat{f}^b(x)$
  - Promedie las muestras de bootstrap

$$\hat{f}_{bag} = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^b(x) \tag{4}$$

- Básicamente estamos suavizando las predicciones.

Random Forests

- Problema con el bagging: si hay un predictor fuerte, diferentes árboles son muy similares entre sí.
- Bosques (forests): reduce la correlación entre los árboles en el boostrap.
- Si hay p predictores, en cada partición use solo $m < p$ predictores, elegidos al azar.
- Bagging es forests con $m = p$ (usando todo los predictores en cada partición).
- m es un hiper-parámetro, $m = \sqrt{(p)}$ es un benchmark

Agenda

1. Recap
   - Árboles
   - Bagging
     - Random Forests
2. Boosting
   - AdaBoost
   - Boosting Trees
   - XGBoost
3. Causal Trees
   - Causality Review: ATE, CATE, HTE
   - Empirical Example

Boosting: Motivation

- Problema con CART: varianza alta.
- Podemos mejorar mucho el rendimiento mediante la agregación
- El boosting toma esta idea pero lo ”encara” de una manera diferente $\rightarrow$ viene de la computación
- Va a usar arboles “pequeños” y “aprende” sequencialmente

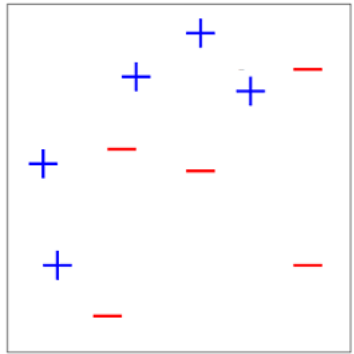
Boosting: Motivation

- Boosting es una de las ideas de aprendizaje más poderosas introducidas en los últimos veinte años.
- Originalmente fue diseñado para problemas de clasificación,
- Pero también puede extenderse a la regresión.

Boosting: Motivation

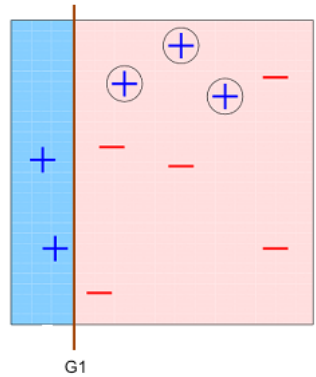
- Intuitivamente lo que hacen es "aprender" de los errores lentamente.
- Esto lo hace "ajustando" árboles pequeños.
- Esto permite mejorar lentamente aprendiendo $f(.)$ en áreas donde no funciona bien.
- OJO: a diferencia de *bagging*, la construcción de cada árbol depende en gran medida de los árboles anteriores

AdaBoost.M1



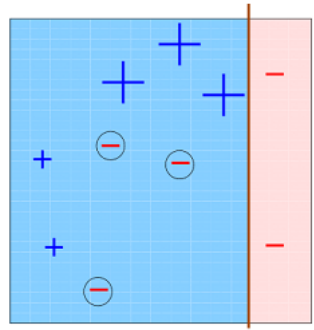
AdaBoost.M1

- Iteración 1



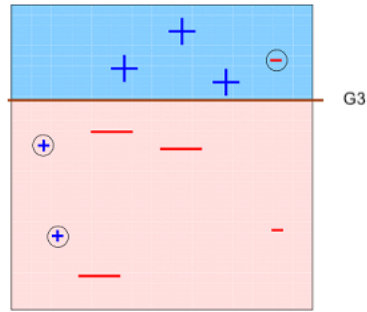
AdaBoost.M1

- Iteración 2



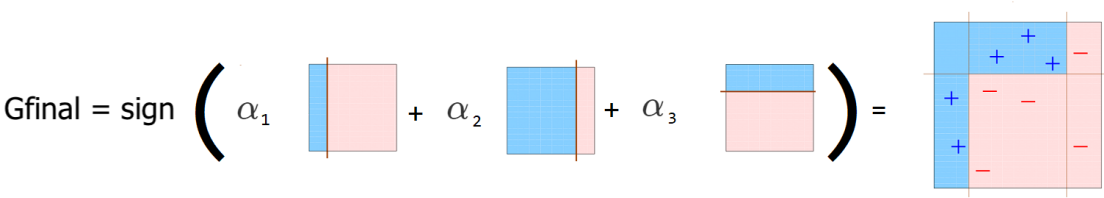
AdaBoost.M1

- Iteración 3



AdaBoost.M1

- Resultado Final



AdaBoost.M1

1. Comenzamos con ponderadores $w_i = 1/N$
2. Para m = 1 hasta M:
   1. Estimar $G_m(x)$ usando ponderadores w_i .
   2. Computar el error de predicción

$$err_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i} \tag{5}$$

   3. Obtener $\alpha_m = ln\left[\frac{(1-err_m)}{err_m}\right]$
   4. Actualizar los ponderadores : $w_i \leftarrow w_i c_i$

$$c_i = exp\left[\alpha_m I(yi \neq G_m(x_i))\right] \tag{6}$$

3. Resultado: $G(x) = sign[\sum_{m=1}^{M} \alpha_m G_m(x)]$

AdaBoost.M1

- $c_i = exp\left[\alpha_m I(y_i \neq G_m(x_i))\right]$
- Si fue correctamente predicho, $c_i = 1$.
- En caso contrario, $c_i = exp(\alpha_m) = \frac{(1-err_m)}{err_m} > 1$
- En cada paso el algoritmo da mas importancia relativa a las predicciones incorrectas.
- Paso final: promedio ponderado de estos pasos

$$G(x) = sign[\sum_{m=1}^{M} \alpha_m G_m(x)] \tag{7}$$

Boosting Trees

- El objetivo es

$$\hat{f} = \underset{f}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i; \Theta)) \right\} \tag{8}$$

- Boosting

$$f(x_i; \{R_j, \gamma_j\}_1^J, M) = \sum_{m=1}^{M} T(x_i; \Theta_m) \tag{9}$$

- el problema es

$$\hat{f} = \underset{f}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} L(y_i, \sum_{m=1}^{M} T(x_i; \Theta_m)) \right\} \tag{10}$$

Boosting Trees

- El problema

$$\hat{f} = \underset{f}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} L(y_i, \sum_{m=1}^{M} T(x_i; \Theta_m)) \right\} \tag{11}$$

- Se puede aproximar por

$$\hat{\Theta}_m = \underset{\Theta_m}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} L(y_i, f_{m-1} + T(x_i; \Theta_m)) \right\} \tag{12}$$

- Para una función de perdida cuadrática

$$\hat{\Theta}_m = \underset{\Theta_m}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} (y_i - f_{m-1} - T(x_i; \Theta_m))^2 \right\} \tag{13}$$

Boosting Trees: Algoritmo

1. Iniciamos fijando $\hat{f}(x) = 0$ y $r_i = y_i$ para todos los i del training set
2. Para $m = 1, 2, ..., M$
   1. Ajustamos un árbol $\hat{f}^m(x) = 0$ con J bifurcaciones
   2. Actualizamos $\hat{f}(x)$ con una versión "shrunken" del nuevo árbol

$$\hat{f}(x) \leftarrow \hat{f}(x) + \hat{f}^m(x) \tag{14}$$

   3. Actualizamos los residuales

$$r_i \leftarrow r_i - \hat{f}^m(x) \tag{15}$$

3. El modelo final es

$$\hat{f}_{boost} = \sum_{m=1}^{M} \hat{f}^m(x) \tag{16}$$

Boosting Trees: Hiper-parámetros

- Cuantas iteraciones (M) usar?
  - Cada iteración generalmente reduce el error de ajuste, de modo que para M lo suficientemente grande este error puede hacerse arbitrariamente pequeño (sesgo se va a cero).
  - Sin embargo, ajustar demasiado bien los datos de entrenamiento puede llevar a overfit (sobreajuste)
  - Por lo tanto, hay un número óptimo $M^\star$ que minimiza el error fuera de muestra
  - Una forma conveniente de encontrar $M^\star$ con validación cruzada

Boosting Trees: Hiper-parámetros

- Podemos utilizar "shrinkage"? $\Rightarrow$ Yes!
- Se escala la contribución de cada arbol por un factor $\lambda \in (0, 1)$

1. Iniciamos fijando $\hat{f}(x) = 0$ y $r_i = y_i$ para todos los i del training set
2. Para $m = 1, 2, ..., M$
   1. Ajustamos un árbol $\hat{f}^m(x) = 0$ con J bifurcaciones
   2. Actualizamos $\hat{f}(x)$

$$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^m(x) \tag{17}$$

   3. Actualizamos los residuales

$$r_i \leftarrow r_i - \lambda \hat{f}^m(x) \tag{18}$$

3. El modelo final es

$$\hat{f}_{boost} = \sum_{m=1}^{M} \lambda \hat{f}^m(x) \tag{19}$$

Boosting Trees: Iteraciones

- Los otros hiperparámetros a fijar son
  - λ la tasa a la que aprende, los valores típicos son 0.01 o 0.001
  - El tamaño del árbol.
    - $4 \leq J \leq 8$ funciona bien
    - $J = 2$ suele ser insuficiente
    - $J > 10$ rara vez se utiliza

Example

photo from https://www.dailydot.com/parsec/batman-1966-labels-tumblr-twitter-vine/

Gradient Boosting Trees

- For **arbitrary** loss functions this "simplified" problem

$$\hat{\Theta}_m = \underset{\Theta_m}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} L\left(y_i, f_{m-1} + T(x_i; \Theta_m)\right) \right\} \tag{20}$$

- is still a tricky problem
- We can solve it by using an implementation of gradient descent

$$f_m = f_{m-1} - \epsilon \sum_{i=1}^{n} \nabla_{f_{m-1}} L(y_i, f_{m-1}(x)) \tag{21}$$

- How do we implement this search?

Gradient Boosting Trees:Algorithm

1. Initialize $f_0(x) = \underset{\gamma}{\text{argmin}} \sum_{i=1}^{n} L(y_i, \gamma)$.
2. For m=1 to M:
   1. For $i = 1, 2, ..., N$ compute

$$r_{im} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f=f_{m-1}} \tag{22}$$

   2. Fit a regression tree to the targets r_{im} giving terminal regions $R_{jm}, j = 1, 2, ..., J_m$.
   3. For $j = 1, 2, ..., J_m$ compute

$$\gamma_{jm} = \underset{\gamma}{\text{argmin}} \sum_{x_i \in R_{jm}} L(y_i, f_{m-1} + \gamma) \tag{23}$$

   4. Update

$$f_m = f_{m-1} + \lambda \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm}) \tag{24}$$

3. El modelo final es

$$\hat{f}(x) = \hat{f}_M(x) \tag{25}$$

XGBoost: Motivation

- XGBoost is an implementation was specifically engineered for optimal performance and speed.
- Why talk about XGBoost?
  - Among the 29 challenge winning solutions published in Kaggle's blog during 2015, 17 solutions used XGBoost.
  - Among these solutions, eight solely used XGBoost to train the model, while most others combined XGBoost with neural nets in ensembles. (The second most popular method, deep neural nets, was used in 11 solutions)
  - The success of the system was also witnessed in 2015 Data Mining and Knowledge Discovery competition organized by ACM (KDD Cup) , where XGBoost was used by every winning team in the top-10.
  - Historically, XGBoost has performed quite well for structured, tabular data. But, if you are dealing with non-structured data such as images, neural networks are usually a better option (more on this later)

XGBoost is a Boosting Tree

- Which tree do we want at each step?
- Add the one that optimizes our objective.

$$\mathcal{L} = \sum_{i=1}^{N} L(y_i, \hat{y}_i) + \sum_{k=1}^{m} \Omega(f_k) \tag{26}$$

- $L(.)$ is a differentiable convex loss function that measures the difference between the prediction $\hat{y}_i$ and the target y_i.
- The second term $\Omega(f)$ penalizes the complexity of the model, where

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda ||\omega||_2 \tag{27}$$

XGBoost is a Boosting Tree

- For **arbitrary** loss functions the function above cannot be optimized using traditional optimization methods

$$\mathcal{L}^m = \sum_{i=1}^{N} (y_i - \hat{y}_i^{m-1} + f_m(x_i))^2 + \Omega(f_t) \tag{28}$$

- So in the general case, we take a second order Taylor expansion of the loss function:

$$\mathcal{L}^m = \sum_{i=1}^{N} \left[L(y_i, \hat{y}_i^{(m-1)}) + g_{im} f_m(x_i) + \frac{1}{2} h_i f_m^2(x_i) \right] + \Omega(f_t) \tag{29}$$

XGBoost is a Boosting Tree

- After we remove all the constants, the specific objective at step m becomes

$$\mathcal{L}^m = \sum_{i=1}^{N} \left[g_{im} f_m(x_i) + \frac{1}{2} h_i f_m^2(x_i) \right] + \Omega(f_t) \tag{30}$$

- This becomes our optimization goal for the new tree.
- One important advantage of this definition is that the value of the objective function only depends on g_{im} and h_{im}

XGBoost Model Complexity

- Let's refine the definition of the tree $f(x)$ as

$$f_t(x) = w_{q(x)}, \tag{31}$$
$$w \in R^T, \tag{32}$$
$$q : R^d \to \{1, 2, \cdots, T\}. \tag{33}$$

- Here w is the vector of predictions on leaves,
- q is a function assigning each data point to the corresponding leaf, and
- T is the number of leaves.

XGBoost Model Complexity

- With the above notation we can rewrite the objective funtion as

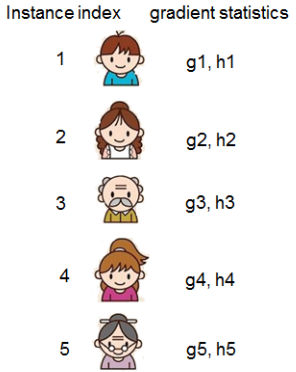
$$\mathcal{L}^{(t)} \approx \sum_{i=1}^{n} [g_i w_{q(x_i)} + \frac{1}{2} h_i w^2_{q(x_i)}] + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 \tag{34}$$

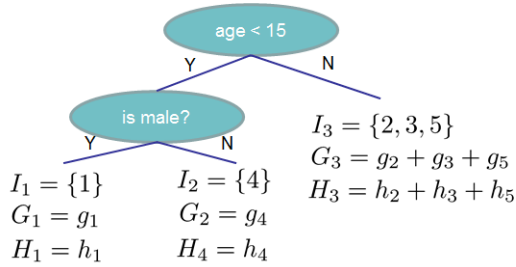
$$= \sum_{j=1}^{T} [(\sum_{i \in I_j} g_i) w_j + \frac{1}{2} (\sum_{i \in I_j} h_i + \lambda) w_j^2] + \gamma T \tag{35}$$

- Defining $G_j = \sum_{i \in I_j} g_i$ and $H_j = \sum_{i \in I_j} h_i$

$$\mathcal{L}^{(t)} = \sum_{j=1}^{T} [G_j w_j + \frac{1}{2} (H_j + \lambda) w_j^2] + \gamma T \tag{36}$$

XGBoost Model Complexity: Example





$$Obj = -\sum_j \frac{G_j^2}{H_j + \lambda} + 3\gamma$$

The smaller the score is, the better the structure is

source:https://xgboost.readthedocs.io/en/latest/tutorials/model.html

Learn the tree structure

- Now that we have a way to measure how good a tree is,
- Ideally we would enumerate all possible trees and pick the best one.
- In practice this is intractable, so we will try to optimize one level of the tree at a time.

Example

photo from https://www.dailydot.com/parsec/batman-1966-labels-tumblr-twitter-vine/

Agenda

1. Recap
   - Árboles
   - Bagging
     - Random Forests
2. Boosting
   - AdaBoost
   - Boosting Trees
   - XGBoost
3. Causal Trees
   - Causality Review: ATE, CATE, HTE
   - Empirical Example

Agenda

1. Recap
   - Árboles
   - Bagging
     - Random Forests
2. Boosting
   - AdaBoost
   - Boosting Trees
   - XGBoost
3. Causal Trees
   - Causality Review: ATE, CATE, HTE
   - Empirical Example

Treatment Effects Review

- We observe a sequence of triples $\{(W_i, Y_i, X_i)\}_i^N$, where
  - $W_i \in \{0, 1\}$: is a binary variable indicating whether the individual was treated (1) or not (0)
  - $Y_i^{obs} \in \mathbb{R}$: a real variable indicating the observed outcome for that individual
  - X_i: is a p-dimensional vector of observable pre-treatment characteristics
- Moreover, in the Neyman-Rubin potential-outcomes framework, we will denote by
  - $Y_i(1)$: the outcome unit i would attain if they received the treatment
  - $Y_i(0)$: the outcome unit i would attain if they were part of the control group

Treatment Effects Review

The individual treatment effect for subject i can then be written as

$$Y_i(1) - Y_i(0)$$

Unfortunately, in our data we can only observe one of these two potential outcomes.

Education (X_i)	Treated W_i	No Subsidy $Y_i(0)$	Subsidy $Y_i(1)$	Treatment effect $\tau_i = Y_i(1) - Y_i(0)$
High	1	?	$Y_1(1)$	?
High	0	$Y_2(0)$	?	?
Low	0	$Y_3(0)$	?	?
Low	1	?	$Y_4(1)$	?

Using the potential outcome notation above, the observed outcome can also be written as

$$Y_i = W_i Y_i(1) + (1 - W_i) Y_i(0)$$

Average Treatment Effects Review

- Computing the difference for each individual is impossible.
- But we can get the Average Treatment Effect (ATE):

$$\tau = E[Y_i(1) - Y_i(0)] \tag{37}$$

- Conditional Average Treatment Effect (CATE)

$$\tau(x) = E[Y_i(1) - Y_i(0)|X_i = x] \tag{38}$$

- Heterogeneous Treatment Effects: Same treatment may affect different individuals differently

Heterogeneous Treatment Effects Review

Concerns

- Issues:
  - Ad hoc searches for particularly responsive subgroups may mistake noise for a true treatment effect.
  - Concerns about ex-post "data-mining" or p-hacking
    - preregistered analysis plan can protect against claims of data mining
    - But may also prevent researchers from discovering unanticipated results and developing new hypotheses
- But how is researcher to predict all forms of heterogeneity in an environment with many covariates?
- Athey and Imbens to the rescue
  - Allow researcher to specify set of potential covariates
  - Data-driven search for heterogeneity in causal effects with valid standard errors

Causal Tree: Theory Details

- Work well in RCTs
- Issue: we do not observe the ground truth
- Honest estimation (Innovation):
  - One sample to choose partition
  - One sample to estimate leaf effects
- Why is the split critical?
- Fitting both on the training sample risks overfitting: Estimating many "heterogeneous effects" that are really just noise idiosyncratic to the sample.
- We want to search for true heterogeneity, not noise

Heterogeneous Treatment Effects Assumptions

- Before proceeding we need to make a couple of assumptions
- Assumption 1: Unconfoundedness

$$Y_i(1), Y_i(0) \perp W_i \mid X_i \tag{39}$$

  - The *unconfoundedness* assumption states that, once we condition on observable characteristics, the treatment assignment is independent to how each person would respond to the treatment.
  - i.e., the rule that determines whether or not a person is treated is determined completely by their observable characteristics.
  - This allows, for example, for experiments where people from different genders get treated with different probabilities,
  - **rules out** experiments where people self-select into treatment due to some characteristic that is not observed in our data.

Heterogeneous Treatment Effects

- Assumption 2: Overlap

$$\forall\, x \in \text{supp}\,(X), \qquad 0 < P\,(W = 1 \mid X = x) < 1 \tag{40}$$

  - The *overlap* assumption states that at every point of the covariate space we can always find treated and control individuals.
  - i.e., in order to estimate the treatment effect for a person with particular characteristics $X_i = x$, we need to ensure that we are able to observe treated and untreated people with those same characteristics so that we can compare their outcomes.

Agenda

1. Recap
   - Árboles
   - Bagging
     - Random Forests
2. Boosting
   - AdaBoost
   - Boosting Trees
   - XGBoost
3. Causal Trees
   - Causality Review: ATE, CATE, HTE
   - Empirical Example

Causal Trees: Empirical Example (Green and Kern)

photo from https://www.dailydot.com/parsec/batman-1966-labels-tumblr-twitter-vine/