Tree-Based Models

Big Data y Machine Learning para Economía Aplicada

Ignacio Sarmiento-Barbieri

Universidad de los Andes

Agenda

1. Recap

2. Tree-Based Models
   - Motivation
   - Árboles
   - Bagging
     - Random Forests
   - Boosting
     - AdaBoost
     - Boosting Trees
     - XGBoost

Agenda

1. Recap

2. Tree-Based Models
   - Motivation
   - Árboles
   - Bagging
     - Random Forests
   - Boosting
     - AdaBoost
     - Boosting Trees
     - XGBoost

Recap

- Queremos predecir y en funcion de observables $(\mathbf{x}_i)$

$$y = f(\mathbf{x}_i) + u \tag{1}$$

- donde la estimación de f implica la que minimize el riesgo empírico (prediga mejor fuera de muestra):

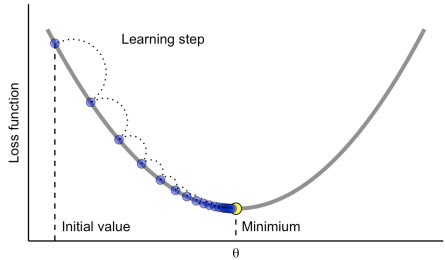
$$\hat{f} = \underset{f}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i; \Theta)) \right\} \tag{2}$$

Recap

- Por ejemplo si L es la perdida cuadrática y $f(x_i) = \mathbf{x}_i\beta$ el problema se resume a encontrar los β.

$$\hat{\beta} = \underset{\tilde{\beta}}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} (y_i - \mathbf{x}_i\beta)^2 \right\} \tag{3}$$

- para estimar los β usamos en ML gradiente descendiente



Source: Boehmke, B., & Greenwell, B. (2019)

Agenda

1. Recap

2. Tree-Based Models
   - Motivation
   - Árboles
   - Bagging
     - Random Forests
   - Boosting
     - AdaBoost
     - Boosting Trees
     - XGBoost

Agenda

1. Recap
2. Tree-Based Models
   - Motivation
   - Árboles
   - Bagging
     - Random Forests
   - Boosting
     - AdaBoost
     - Boosting Trees
     - XGBoost

Motivación

- Inspirados por Leo Breiman:

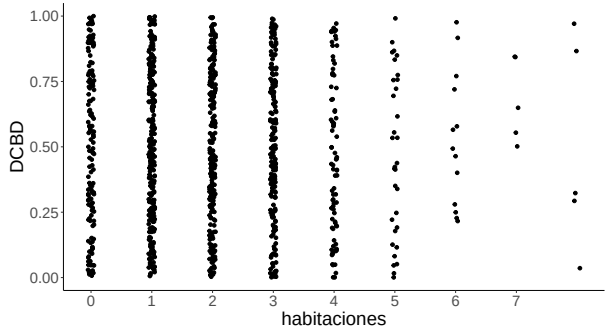
  "There are two cultures in the use of statistical modeling to reach conclusions from data. One assumes that the data are generated by a given stochastic data model. The other uses algorithmic models and treats the data mechanism as unknown." Breiman [2001b], p199.

  "The statistical community has been committed to the almost exclusive use of data models. This commitment has led to irrelevant theory, questionable conclusions, and has kept statisticians from working on a large range of interesting current problems. Algorithmic modeling, both in theory and practice, has developed rapidly in fields outside statistics. It can be used both on large complex data sets and as a more accurate and informative alternative to data modeling on smaller data sets. If our goal as a field is to use data to solve problems, then we need to move away from exclusive dependence on data models and adopt a more diverse set of tools." Breiman [2001b], p199.

Motivación

- Queremos predecir:

$$Precio = f(habitaciones, ubicacion, ...) \tag{4}$$



Agenda

Árboles

- Podemos asumir f es lineal

$$f(\mathbf{x}_i) = \beta_0 + \beta_1 Habitaciones_i + \beta_2 DCBD_i + u_i \tag{5}$$

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

$$f(x_i; \{R_j, \gamma_j\}_1^J) = T(x_i; \{R_j, \gamma_j\}_1^J) \tag{6}$$

- donde

$$T(x_i; \{R_j, \gamma_j\}_1^J) = \sum_{j=1}^{J} \gamma_j I(x \in R_j) \tag{7}$$

Árboles

- El problema de optimización

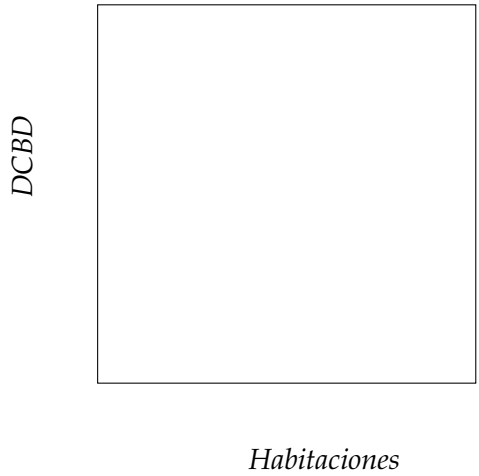
$$\hat{f} = \underset{f}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i; \Theta)) \right\} \tag{8}$$

- es ahora

$$\{\hat{R}_j, \hat{\gamma}_j\}_1^J = \underset{\{R_j, \gamma_j\}_1^J}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} L(y_i, T(\mathbf{x}_i; \{R_j, \gamma_j\}_1^J)) \right\} \tag{9}$$

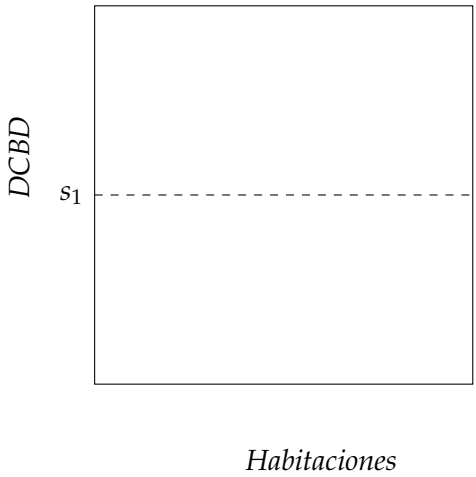
Árboles

"Recursive binary splitting"



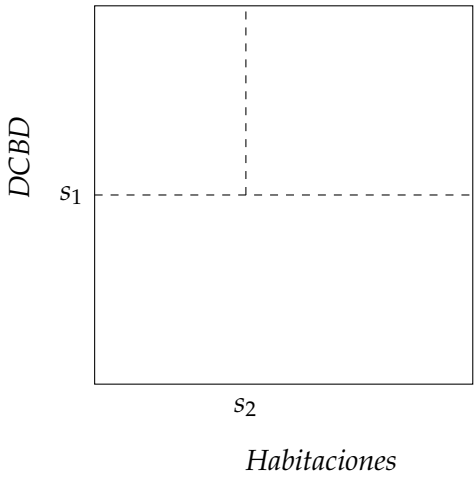
Árboles

"Recursive binary splitting"

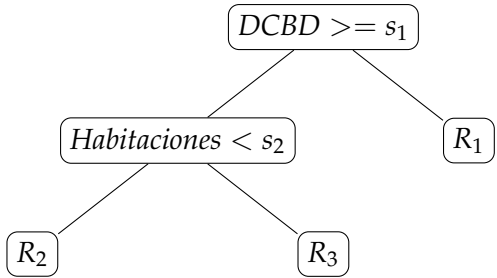


Árboles

"Recursive binary splitting"



Árboles



Ejemplo

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

Agenda

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

$$\hat{f}_{bag} = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^b(x) \tag{10}$$

- 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

Ejemplo

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

Agenda

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

- El AdaBoost.M1 es una implementación para clasificación

AdaBoost.M1

- El AdaBoost.M1 es una implementación para clasificación
- Como viene de la computación tiene un lenguaje ligeramente diferente.
- Vocabulario:
  - $y \in -1, 1$, X vector de predictores.
  - $\hat{y} = G(X)$ (clasificador)
  - $err = \frac{1}{N} \sum_i^N I(y_i \neq G(x_i))$

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

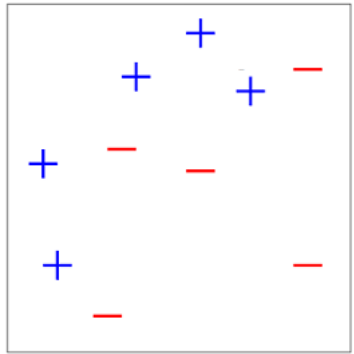
$$err_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i} \tag{11}$$

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

$$c_i = exp\left[\alpha_m I(yi \neq G_m(x_i))\right] \tag{12}$$

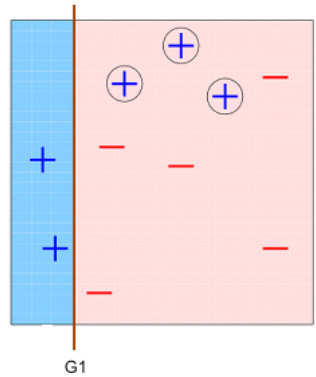
3. Resultado: $G(x) = sign[\sum_{m=1}^{M} \alpha_m G_m(x)]$

AdaBoost.M1



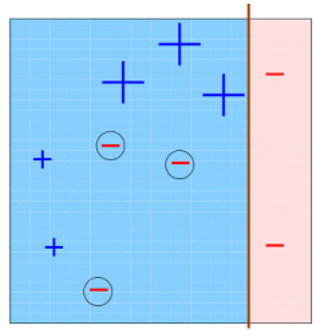
AdaBoost.M1

- Iteración 1



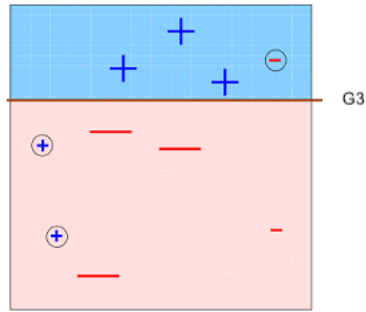
AdaBoost.M1

- Iteración 2



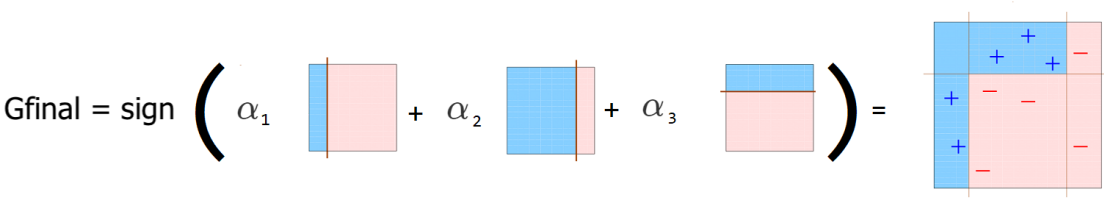
AdaBoost.M1

- 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

$$err_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i} \tag{13}$$

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

$$c_i = exp\left[\alpha_m I(yi \neq G_m(x_i))\right] \tag{14}$$

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

$$G(x) = sign[\sum_{m=1}^{M} \alpha_m G_m(x)] \tag{15}$$

Boosting Trees

- El objetivo es

$$\hat{f} = \underset{f}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i; \Theta)) \right\} \tag{16}$$

- Boosting

$$f(x_i; \{R_j, \gamma_j\}_1^J, M) = \sum_{m=1}^{M} T(x_i; \Theta_m) \tag{17}$$

- el problema es

$$\hat{f} = \underset{f}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} L(y_i, \sum_{m=1}^{M} T(x_i; \Theta_m)) \right\} \tag{18}$$

Boosting Trees

- El problema

$$\hat{f} = \underset{f}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} L(y_i, \sum_{m=1}^{M} T(x_i; \Theta_m)) \right\} \tag{19}$$

- Se puede aproximar por

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

- Para una función de perdida cuadrática

$$\hat{\Theta}_m = \underset{\Theta_m}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} (y_i - f_{m-1} - T(x_i; \Theta_m))^2 \right\} \tag{21}$$

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

$$\hat{f}(x) \leftarrow \hat{f}(x) + \hat{f}^m(x) \tag{22}$$

   3. Actualizamos los residuales

$$r_i \leftarrow r_i - \hat{f}^m(x) \tag{23}$$

3. El modelo final es

$$\hat{f}_{boost} = \sum_{m=1}^{M} \hat{f}^m(x) \tag{24}$$

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 $\hat{f}(x)$

$$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^m(x) \tag{25}$$

   3. Actualizamos los residuales

$$r_i \leftarrow r_i - \lambda \hat{f}^m(x) \tag{26}$$

3. El modelo final es

$$\hat{f}_{boost} = \sum_{m=1}^{M} \lambda \hat{f}^m(x) \tag{27}$$

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

Gradient Boosting Trees

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

$$\hat{\Theta}_m = \underset{\Theta_m}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} L\left(y_i, f_{m-1} + T(x_i; \Theta_m)\right) \right\} \tag{28}$$

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

$$f_m = f_{m-1} - \epsilon \sum_{i=1}^{n} \nabla_{f_{m-1}} L(y_i, f_{m-1}(x)) \tag{29}$$

- 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

$$r_{im} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f=f_{m-1}} \tag{30}$$

   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

$$\gamma_{jm} = \underset{\gamma}{\text{argmin}} \sum_{x_i \in R_{jm}} L(y_i, f_{m-1} + \gamma) \tag{31}$$

   4. Update

$$f_m = f_{m-1} + \lambda \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm}) \tag{32}$$

3. El modelo final es

$$\hat{f}(x) = \lambda \sum_{m=1}^{M} \hat{f}_m(x_i) \tag{33}$$

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

- Cual es la innovación? Penaliza la optimización, en cada paso la función objetivo para construir el árbol es

$$\mathcal{L} = \sum_{i=1}^{N} L(y_i, \sum_{m=1}^{M} T(x_i; \Theta_m)) + \sum_{m=1}^{M} \Omega(T(x_i; \Theta_m)) \tag{34}$$

- El segundo termino penaliza la complejidad del modelo,

$$\Omega(.) = \zeta J + \frac{1}{2} \varphi \sum_{j=1}^{J} \gamma_j^2 \tag{35}$$

Example

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