Lecture 9: Machine Learning
Boosting

Big Data and Machine Learning en el Mercado Inmobiliario
Educación Continua

Ignacio Sarmiento-Barbieri

Universidad de los Andes

May 22, 2023

Agenda

1. Recap
   - Regularización
   - Más allá de la linealidad
2. Boosting
   - Motivation
   - Boosting Trees
3. Break

Recap: Regularization

$$min_{\beta} L(\beta) = \sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \alpha\lambda \sum_{j=1}^{p} |\beta_j| + (1-\alpha)\lambda \sum_{j=1}^{p} (\beta_j)^2 \quad (1)$$

- Si $\alpha = 1$ Lasso
- Si $\alpha = 0$ Ridge
- Como elegir (α, λ)? $\rightarrow$ Validación Cruzada Bidimensional

Más allá de la linealidad

- El objetivo es predecir Y dadas otras variables X. Ej: precio vivienda dadas las características

- Asumimos que el link entre Y and X esta dado por el modelo:

$$Y = f(X) + u \tag{2}$$

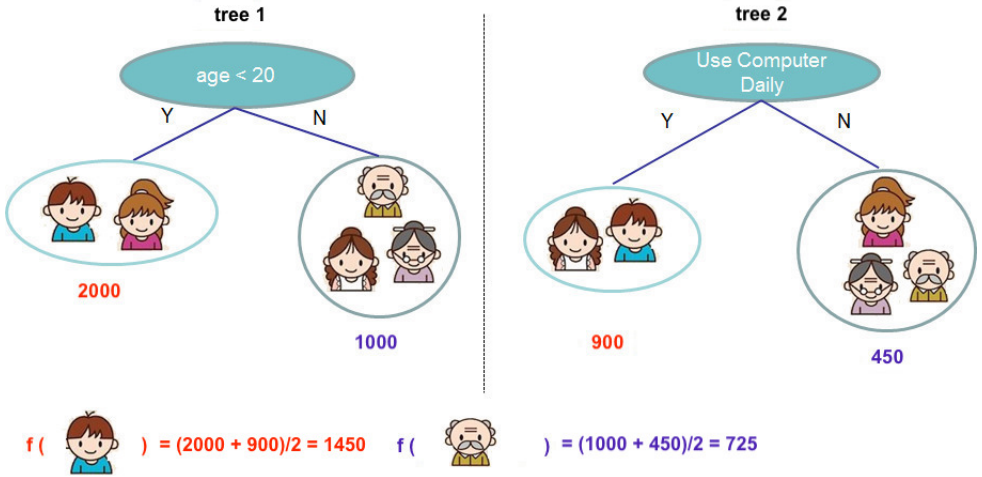
- Hasta ahora vimos modelos lineables o linealizables.
  - Regresión lineal, lasso, ridge, elastic net
- Árboles y Bosques
  - Modelo flexible e interpretable para la relación entre Y y X.
  - Para que? No-linealidades, interacciones.

Random Forests

Trees:

Random Forests:

Random Forests



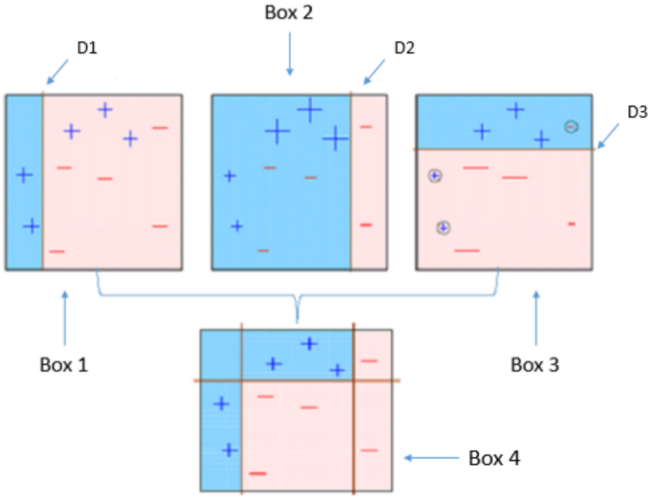
Boosting: Motivation

- 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 clasificadores débiles: clasificador marginalmente mejor que lanzar una moneda (tasa de error ligeramente mejor que .5)
- Ej.: CART con pocas ramas (dos ramas)
- Boosting: promedio ponderado de la sucesión de clasificadores débiles.

Boosting Trees: Algoritmo

- En el caso de los árboles ($\hat{f}(x)$) aprender su estructura es mucho mas difícil.
  - Necesitamos saber el tamaño del árbol.
  - Es intratable aprender todos los arboles al mismo tiempo.
- La estrategia de boosting es aprender iterativamente.
- Fijamos lo aprendido, y agregamos un nuevo árbol en cada paso.

Boosting Intuición



Source: https://www.analyticsvidhya.com/blog/2015/11/quick-introduction-boosting-algorithms-machine-learning/

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 arbol $\hat{f}^m(x)$ con d bifurcaciones ($d+1$ nodos) a los datos de entrenamiento (X, r)
   2. Actualizamos $\hat{f}(x)$ con una versión "shrunken" del nuevo arbol

$$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^m(x) \tag{3}$$

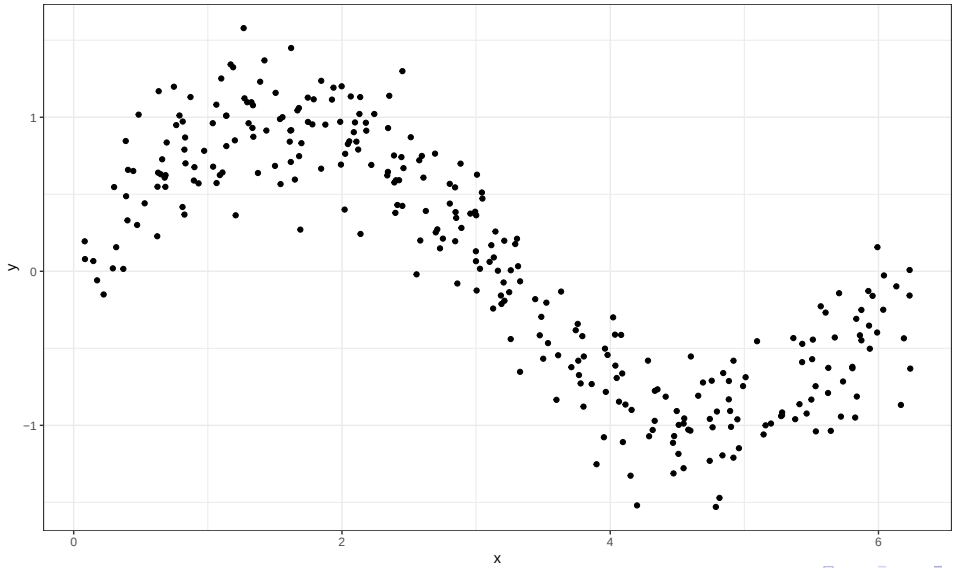
   3. Actualizamos los residuales

$$r_i \leftarrow r_i - \lambda \hat{f}^m(x) \tag{4}$$

3. El modelo final es

$$\hat{f}_{boost} = \sum_{m=1}^{M} \lambda \hat{f}^m(x) \tag{5}$$

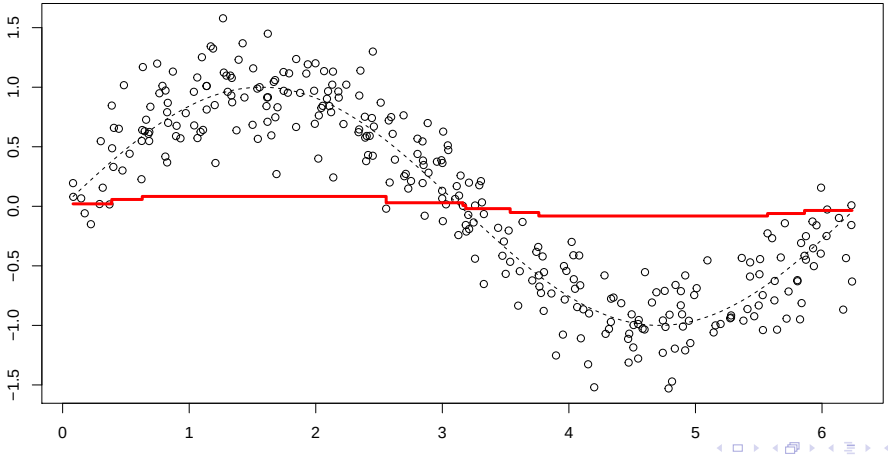
Boosting Trees: Demo



Boosting Trees: Example

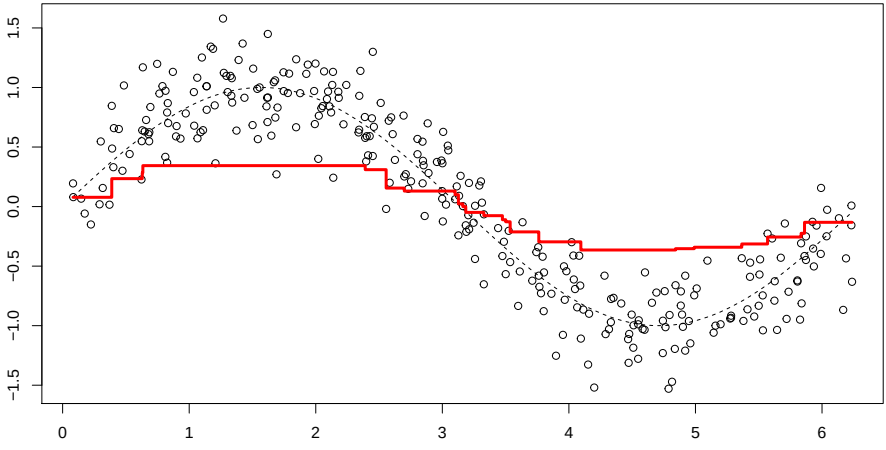
- Algorithm:

```
M<-2
```



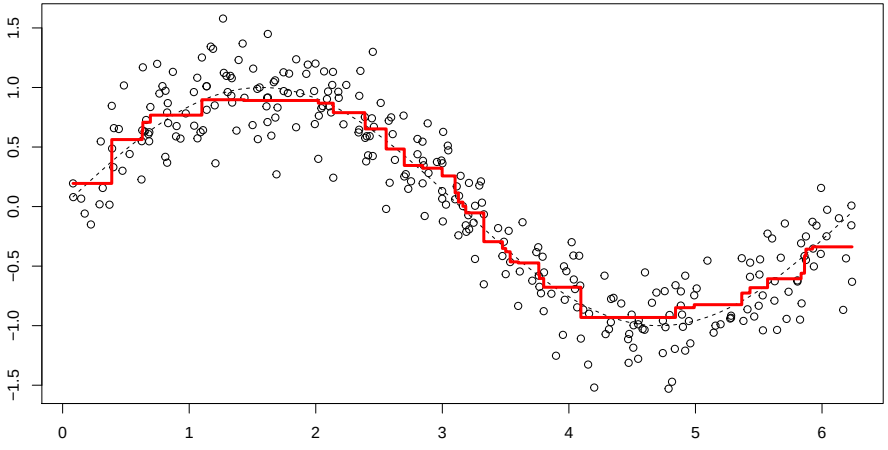
Boosting Trees: Example

```
M<-10
```



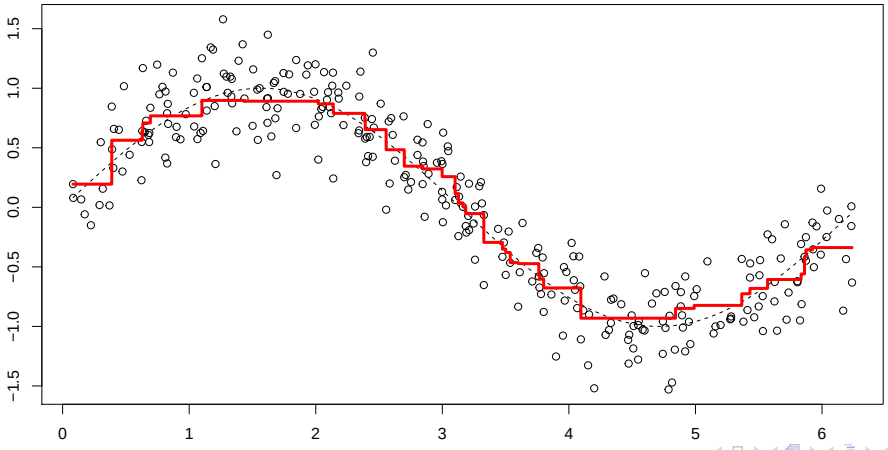
Boosting Trees: Example

```
M<-100
```



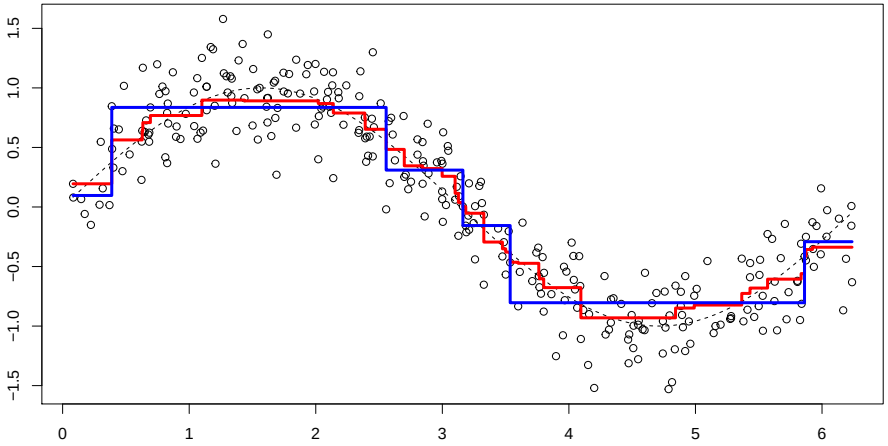
Boosting Trees: Example

```
M<-300
```



Boosting Trees: Example

- Simple tree (blue), boosted tree (red)



Boosting Trees: Iteraciones

- La primer pregunta es sobre 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 validacion cruzada

Boosting Trees: Iteraciones

- Los otros hyper-parámetros a fijar son
  - λ la tasa a la que aprende, los valores típicos son 0.01 o 0.001
  - d el número de bifurcaciones. A menudo $d = 1$ (stumps) funcionan bien
- Dependiendo de la implementación, puedo taambien utilizar subsamplig de filas y columnas
  - At each iteration we sample a fraction η of the training observations (without replacement), and grow the next tree using that subsample.
  - Reduces the computing time by the same fraction η, and some cases improves prediction
  - Same for columns and breaks high correlation (like forests)

XGBoost en un Boosting Tree

- ¿Qué árbol agregamos en cada paso?
- El que optimice la función objetivo

$$\mathcal{L} = \sum_{i=1}^{N} L(y_i, \hat{y}_i) + \sum_{k=1}^{m} \Omega(f_k) \tag{6}$$

- El segundo termino penaliza la complejidad del modelo,

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda ||\omega||_2 \tag{7}$$

XGBoost: palabras finales

- XGBoost es una extensión de boosting trees, hay muchas más.
- Su implementación fue diseñada específicamente para un rendimiento y velocidad óptimos.
- ¿Por qué hablar de XGBoost?
  - Entre las 29 soluciones ganadoras del desafío publicadas en página de Kaggle durante 2015, 17 soluciones utilizaron XGBoost.
  - Entre estas soluciones, ocho utilizaron exclusivamente XGBoost para entrenar el modelo, mientras que la mayoría de las demás combinaron XGBoost con redes neuronales. (El segundo método más popular, las redes neuronales profundas, se utilizó en 11 soluciones)
  - También se vio en la competencia de Minería de Datos y Descubrimiento de Conocimiento de 2015 organizada por ACM (Copa KDD), donde XGBoost fue utilizado por todos los equipos ganadores en el top-10.
  - Históricamente, XGBoost ha funcionado bastante bien para datos tabulares estructurados. Pero, si se trata de datos no estructurados como imágenes, las redes neuronales suelen ser una mejor opción.

Recap: Bagging, Forests, and Boosting

- We can improve performance a lot using either bootstrap aggregation (bagging), random forests, or boosting.
  - For each bootstrap sample, fit a regression tree $\hat{f}^b(x)$
    - Bagging: bootstraps of full sample
    - Random Forests: bootstrap of full sample and subset of predictors $\sqrt{(}p)$
    - Boosting Trees: learn sequentially from errors
  - Average across samples to get the predictor
  - Basically we are smoothing predictions.

Break