Lecture 4:
Arboles y Bosques
Aprendizaje y Minería de Datos para los Negocios

Ignacio Sarmiento-Barbieri

Universidad de los Andes

October 24, 2021

Agenda

Más allá de la linealidad

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

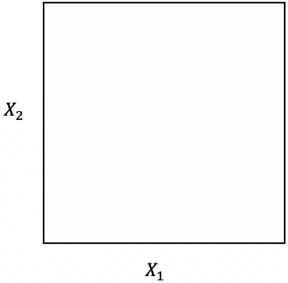
- Asumimos que el link entre Y and X esta dado por el modelo:

$$Y = f(X) + u \tag{1}$$

- Hasta ahora vimos modelos lineables o linealizables.
  - Regresión lineal, polinomial, escalonadas, splines, regresión local
- Árboles (CARTs)
  - Modelo flexible e interpretable para la relación entre Y y X.
  - Para que? No-linealidades, interacciones.

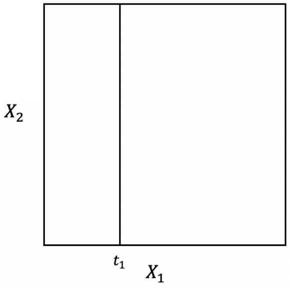
Árboles: que hacen?

1. Y es la variable a predecir, los insumos son X_1 y X_2
2. Partimos el espacio (X_1, X_2) en dos regiones, en base a una sola variable (particion horizontal o vertical).



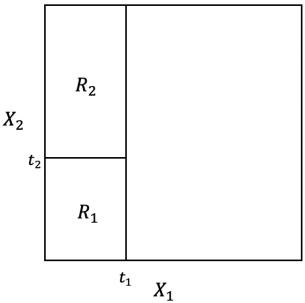
Árboles: que hacen?

1. Y es la variable a predecir, los insumos son X_1 y X_2
2. Partimos el espacio (X_1, X_2) en dos regiones, en base a una sola variable .
3. Dentro de cada región proponemos como predicción la media muestral de Y en cada región.
4. Punto: elegir la variable y el punto de partición de manera optima (mejor ajuste global).



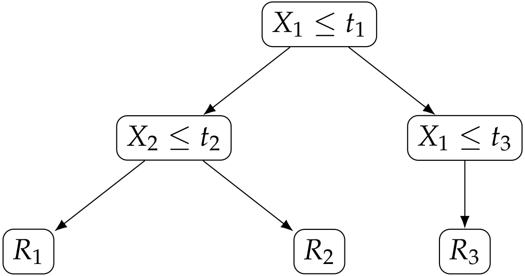
Árboles: que hacen?

1. Y es la variable a predecir, los insumos son X_1 y X_2
2. Partimos el espacio (X_1, X_2) en dos regiones, en base a una sola variable .
3. Dentro de cada región proponemos como predicción la media muestral de Y en cada región.
4. Punto: elegir la variable y el punto de partición de manera optima (mejor ajuste global).

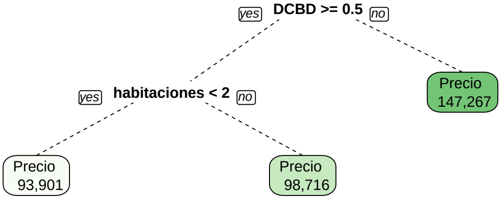


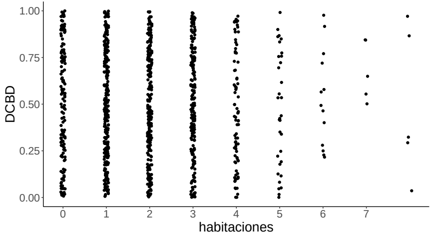
Árboles: que hacen?

1. Y es la variable a predecir, los insumos son X_1 y X_2
2. Partimos el espacio (X_1, X_2) en dos regiones, en base a una sola variable (partición horizontal o vertical).
3. Dentro de cada región proponemos como predicción la media muestral de Y en cada región.
4. Punto: elegir la variable y el punto de partición de manera optima (mejor ajuste global).
5. Continuamos partiendo



Árboles: que hacen?





Árboles: cómo lo hacen?

- Tenemos datos Y $n \times 1$ (precio) y X $n \times p$ (características)
- Definiciones
  - j es la variable que parte el espacio y s es el punto de partición
  - Defina los siguientes semiplanos

$$R_1(j,s) = \{X|X_j \leq s\} \;\; \& \;\; R_2(j,s) = \{X|X_j > s\} \tag{2}$$

  - El problema se reduce a buscar la variable de partición X_j y el punto s de forma tal que

$$\min_{j,s}\left[\min_{c_1}\sum_{x_i \in R_1(j,s)}(y-c_1)^2 + \min_{c_2}\sum_{x_i \in R_2(j,s)}(y-c_2)^2\right] \tag{3}$$

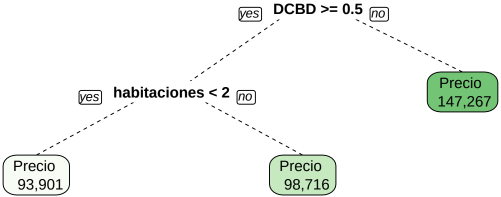
Árboles: cómo lo hacen?

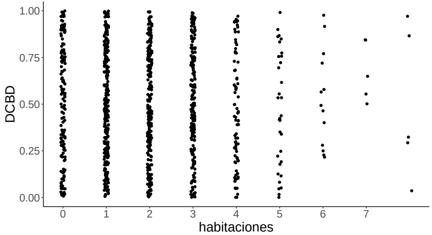
- Para cada variable y punto, la minimización interna es la media

$$\hat{c}_m = \frac{1}{n_m} \sum (y_i | x_i \in R_m) \tag{4}$$

- El proceso se repite para todas las regiones

Árboles: cómo lo hacen?





Árboles: cómo lo hacen?

- Para cada variable y punto, la minimización interna es la media

$$\hat{c}_m = \frac{1}{n_m} \sum (y_i | x_i \in R_m) \tag{4}$$

- El proceso se repite para todas las regiones
- El árbol final tiene M regiones

$$\hat{f}(x) = \sum_{m=1}^{M} \hat{c}_m I(x \in R_m) \tag{5}$$

Árboles: cómo lo hacen?

- El árbol creció, como lo paramos?
- Si el árbol es muy grade, tenemos overfit
- Un árbol mas chico, puede tener menos regiones. Esto puede llevar a una varianza menor y mejor interpretación al costo de un poco sesgo.
- Solución: Pruning (poda)
  - Dejar crecer un árbol muy grande T_0
  - Cortarlo te quedas con un sub-árbol (*subtree*)
  - Como determinamos la mejor forma de cortarlo? $\rightarrow$ menor error de predicción usando cross-validation

Árboles: cómo lo hacen?

- Desventaja, calcular el error de predicción usando cross-validation para cada sub-árbol posible es demasiado (muchos sub-árboles posbiles)
- Solución: *Cost complexity pruning (cortar las ramas mas débiles)*
  - Indexamos los arboles con T.
  - Un sub-árbol $T \in T_0$ es un árbol que se obtuvo colapsando los nodos terminales de otro árbol (cortando ramas).
  - $[T]$ = número de nodos terminales del arbol T

Árboles: cómo lo hacen?

- Cost complexity del árbol T

$$C_\alpha(T) = \sum_{m=1}^{[T]} n_m Q_m(T) + \alpha[T] \tag{6}$$

  - donde $Q_m(T) = \frac{1}{n_m} \sum_{x_i \in R_m} (y_i - \hat{c}_m)^2$ para los árboles de regresión
  - $Q_m(T)$ penaliza la heterogeneidad dentro de la regresión y el número de regiones
  - Objetivo: para un dado α, encontrar el pruning óptimo que minimice $C_\alpha(T)$

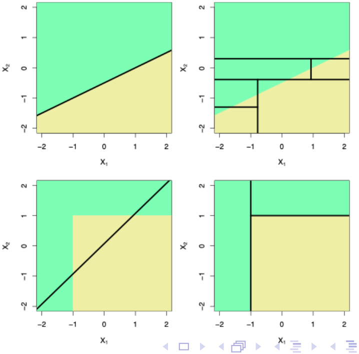
Árboles: cómo lo hacen?

- Mecanismo de búsqueda para T_α (pruning óptimo dado α).
  - Resultado: para cada α hay un sub-árbol único T_α que minimiza Cα (T).
  - Ramas mas débiles: eliminar sucesivamente las ramas que producen un aumento mínimo en $\sum_{m=1}^{[T]} n_m Q_m(T)$
  - Idea: remover ramas es colapsar, esto aumenta la heterogenidad, ergo, colapsamos las particiones menos necesarias.
  - Esto eventualmente colapsa hasta el nodo inicial (stump) pero va a través de una sucesión de árboles, que va del mas grande al mas pequeño cortando las ramas mas débiles.
  - Breiman et al. (1984): T_α pertenece a esta sequencia.
  - Uno puede enfocar la búsqueda en esta sucesión de sub-árboles.
  - Elección de α: cross validation.

Árboles vs. Modelos Lineales

- Cuál modelo es mejor?
  - Si la relación entre los predictores y la respuesta es lineal, los modelos lineales clásicos, como la regresión lineal, superan a los árboles de regresión.
  - Por otro lado, si la relación entre los predictores no es lineal, los árboles de decisión superarían a los enfoques clásicos.

- Arriba: el límite es lineal
  - Izquierda: modelo lineal (bueno)
  - Derecha: árbol
- Abajo: el límite es no-lineal
  - Izquierda: linear model
  - Derecha: arbol (good)



Ventajas y Desventajas de los Árboles

- Pros:
  - Los árboles son muy fáciles de explicar a las personas (probablemente incluso más fáciles que la regresión lineal)
  - Los árboles se pueden trazar gráficamente y son fácilmente interpretados incluso por no expertos. Variables más importantes en la parte superior
  - Funcionan bien en problemas de clasificación y regresión.
- Cons:
  - Los árboles no son muy precisos o robustos (ensamblados, bosques aleatorios y boosting al rescate)
  - Si la estructura es lineal, CART no funciona bien

Bagging

- Problema con CART: varianza alta.
- Podemos mejorar mucho el rendimiento mediante la agregación
- Bagging:
  - Obtenga repetidamente muestras aleatorias $(X_i^b, Y_i^b)_{i=1}^N$ de la muestra observada.
  - Para cada muestra de arranque, ajuste un árbol de regresión $\hat{f}^b(x)$
  - Promedie las muestras de bootstrap

$$\hat{f}_{bag} = \frac{1}{B} \sum_{b=1}^{B} \hat{f}^b(x) \tag{7}$$

- Básicamente estamos suavizando las predicciones.
- Idea: la varianza del promedio es menor que la de una sola predicción.

Random Forests

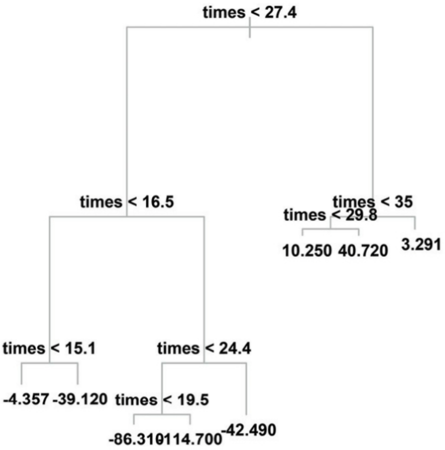
- Problema con el bagging: si hay un predictor fuerte, diferentes árboles son muy similares entre sí. Si hay alta correlación, ¿está realmente reduciendo la varianza?
- Bosques (forests): reduzca la correlación entre los árboles en el boostrap.
- Si hay p predictores, en cada partición use solo $m < p$ predictores, elegidos al azar.
- Bagging es forests con $m = p$ (usando todo los predictores en cada partición).
- Tipicamente $m = \sqrt{(}p)$

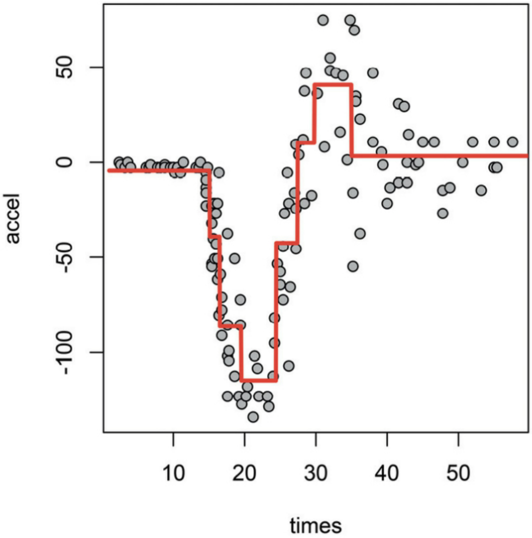
Random Forests

Trees:

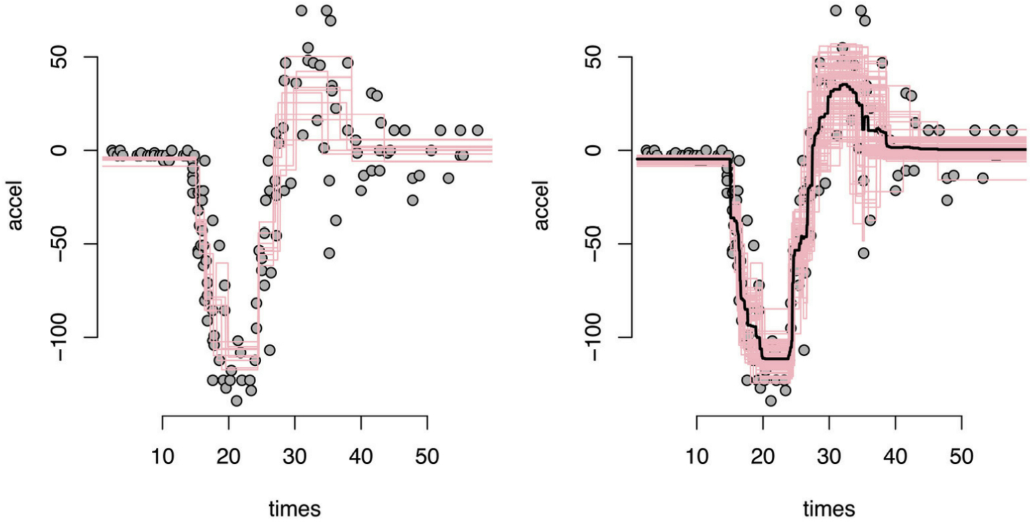
Random Forests:

Random Forests

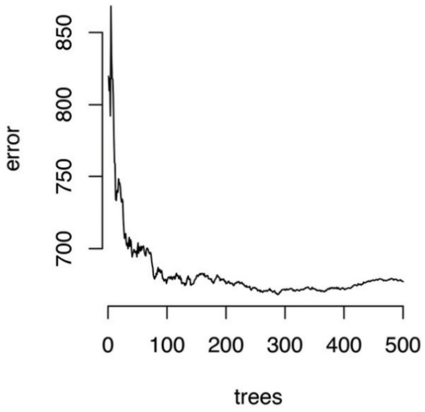




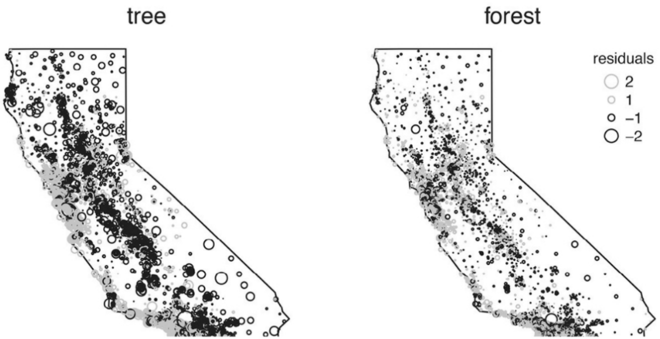
Random Forests



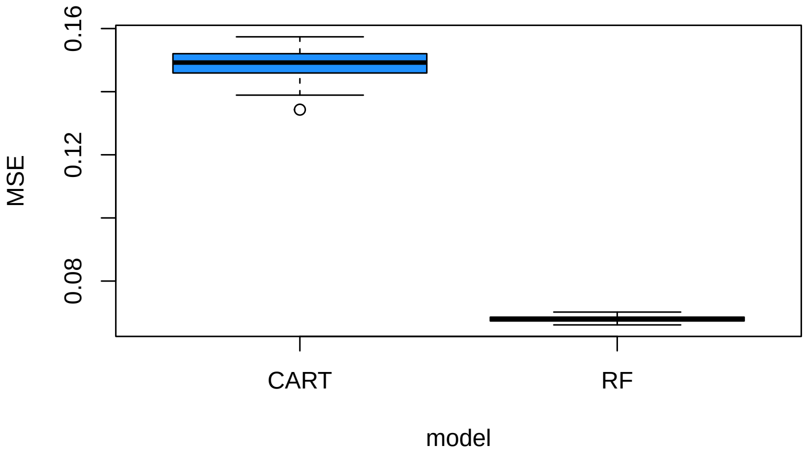
Random Forests



Residuales en muestra



MSE Fuera de Muestra



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

- El boosting viene de la computación y tiene un lenguaje ligeramente diferente.
- Fue desarrollado para problemas de clasificación pero se puede extender fácilmente a regresión.
- Vocabulario:
  - $y \in -1, 1$, X vector de predictores.
  - $y = G(X)$ (clasificador)
  - $err = \frac{1}{N} \sum_i^N I(y_i \neq G(x_i))$
- Para fijar ideas veamos `AdaBoost`

AdaBoost

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

$$err_m = \frac{\sum_{i=1}^{N} I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i} \tag{8}$$

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

$$c_i = exp\left[\alpha_m I(yi \neq G_m(x_i))\right] \tag{9}$$

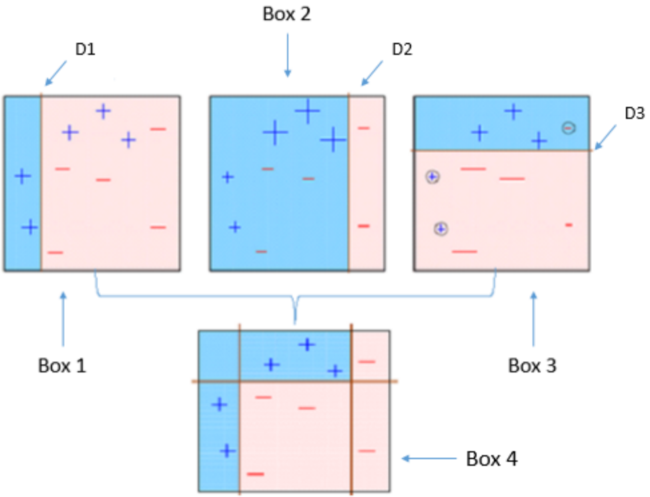
3. Resultado: $G(x) = sgn[\sum_{m=1}^{M} \alpha_m G_m(x)]$

AdaBoost

- $c_i = exp\left[\alpha_m I(y_i \neq G_m(x_i))\right]$
- 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

$$G(x) = sgn[\sum_{m=1}^{M} \alpha_m G_m(x)] \tag{10}$$

Boosting Intuición



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

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.
- Escribimos el valor de la predicción en cada paso m como $\hat{y}_i^m$.

Boosting Trees: Algoritmo

- La estrategia de boosting es aprender iterativamente.
- Fijamos lo aprendido, y agregamos un nuevo árbol en cada paso.
- Escribimos el valor de la predicción en cada paso m como $\hat{y}_i^m$.
- Entonces tenemos

$$
\begin{aligned}
\hat{y}_i^0 &= 0 \\
\hat{y}_i^1 &= \hat{y}_i^0 + f_1(x_i) \\
\hat{y}_i^2 &= \hat{y}_i^1 + f_2(x_i) \\
\ldots & \\
\hat{y}_i^M &= \sum_{m=1}^{M} f_m(x_i) = \hat{y}_i^{m-1} + f_m(x_i)
\end{aligned}
\tag{11}
$$

Boosting Trees: Algoritmo

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

$$obj^m = \sum_{i=1}^{N} L(y_i, \hat{y}_i^{(m)}) \tag{12}$$

$$= \sum_{i=1}^{N} L(y_i, \hat{y}_i^{m-1} + f_m(x_i)) \tag{13}$$

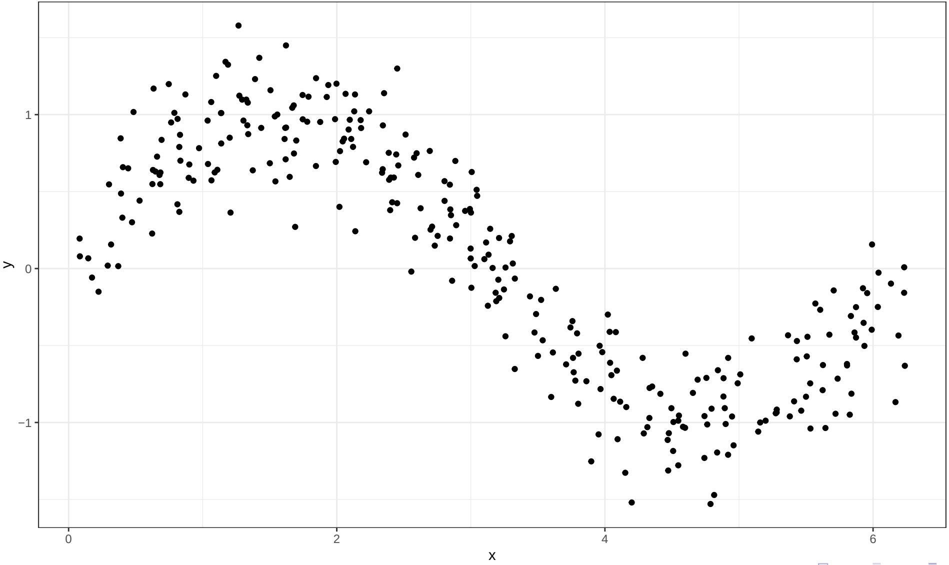
- Si usamos MSE como L la función objetivo es

$$\sum_{i=1}^{N} (y_i - \hat{y}_i^{m-1} + f_m(x_i))^2 \tag{14}$$

Boosting Trees: Iteraciones

- La pregunta se vuelve cuantas iteraciones (M) usar?
  - Cada iteración generalmente reduce el riesgo de entrenamiento $L(.)$, de modo que para M lo suficientemente grande este riesgo 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 estimar $M^{\star}$ es calcular el error de predicción como una función de M en una muestra de validación. El valor de M que minimiza este riesgo se toma como una estimación de M.

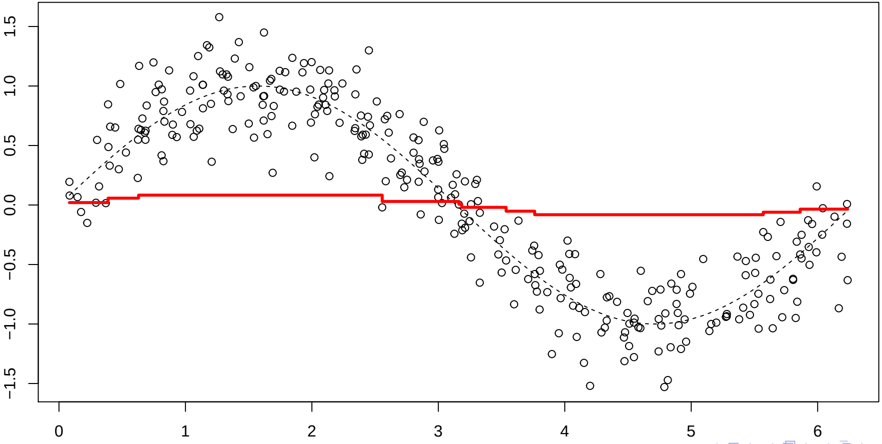
Boosting Trees: Demo



Boosting Trees: Example

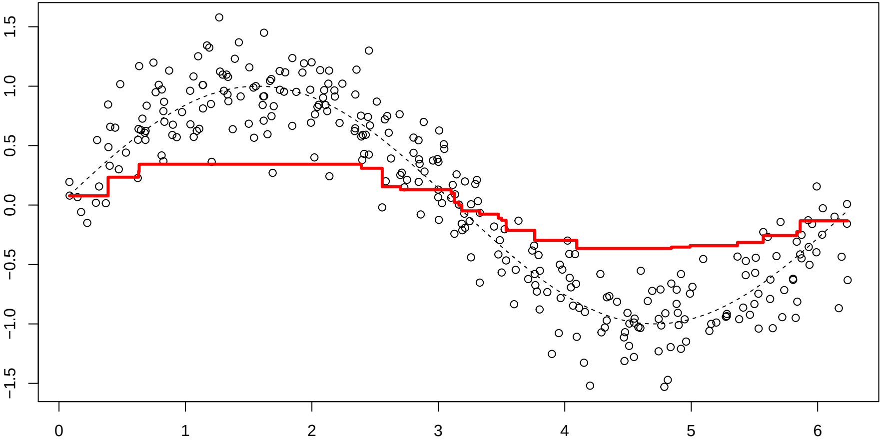
- Algorithm:

```
M<-2
```



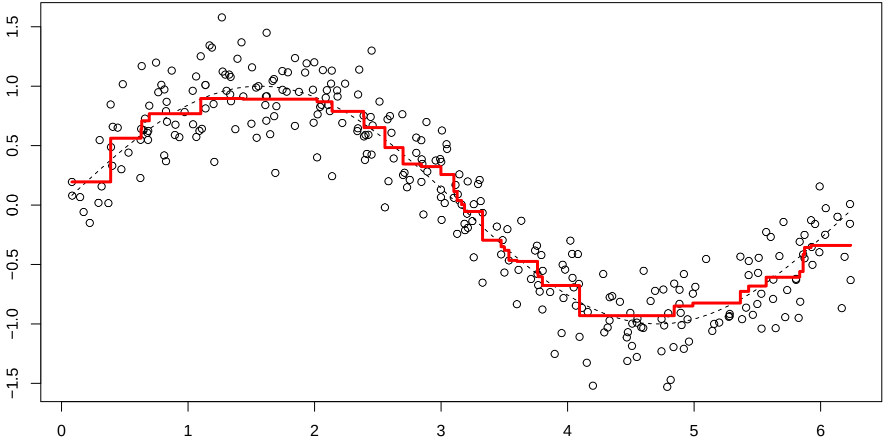
Boosting Trees: Example

```
M<-10
```



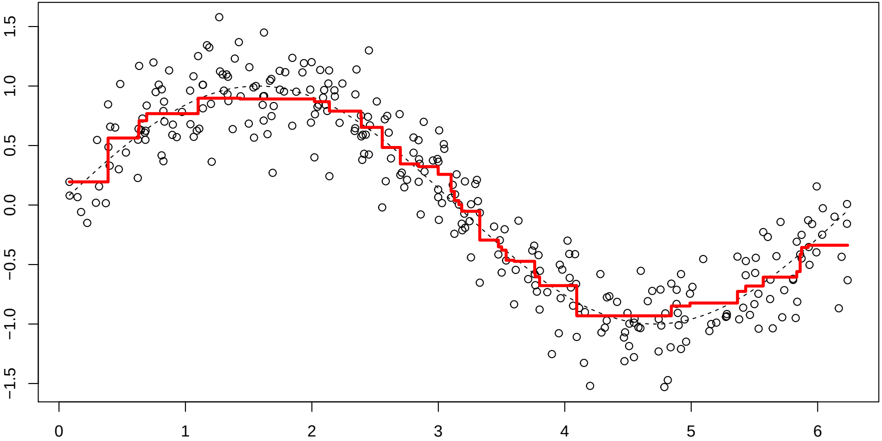
Boosting Trees: Example

```r
M<-100
```



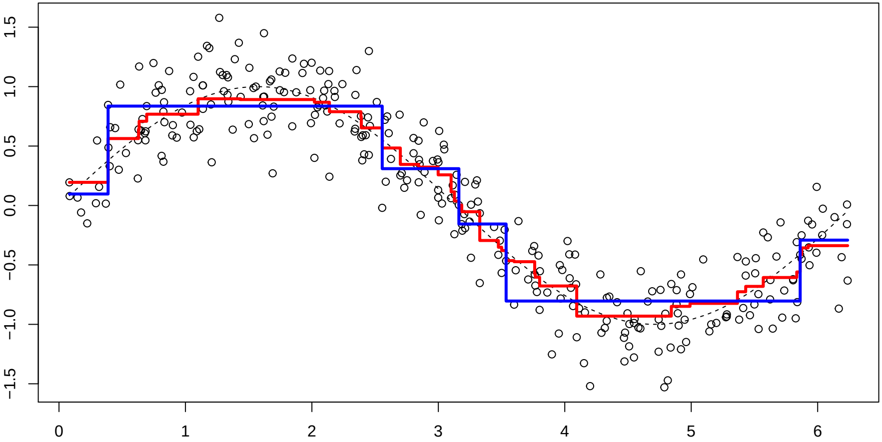
Boosting Trees: Example

```
M<-300
```



Boosting Trees: Example

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



XGBoost en un Boosting Tree

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

$$\mathcal{L} = \sum_{i=1}^{N} L(y_i, \hat{y}_i) + \sum_{k=1}^{m} \Omega(f_k) \tag{15}$$

- El segundo termino penaliza la complejidad del modelo,

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda ||\omega||_2 \tag{16}$$

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.

Volvemos en 15 mins con R

R para ML

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