Bagging, Random Forests y Boosting

Walter Sosa-Escudero

Universisad de San Andres y CONICET

CART

- Forma inteligente de representar no linealidades.
- Arriba: variables mas relevantes.
- Muy facil de comunicar. Reproduce proceso decisorio humano.
- Si la estructura es lineal, CART no anda bien.
- Lineal: las variables importan siempre (no dentro de un nodo)
- Poco robusto

Bagging, random forests y boosting al rescate (ideas minimas).

Bagging

- Bagging: bootstrap aggregation.
- Problema con CART: varianza alta.
- Bagging: boostrap training sets: tomar como prediccion el promedio de las predicciones boostrap:

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

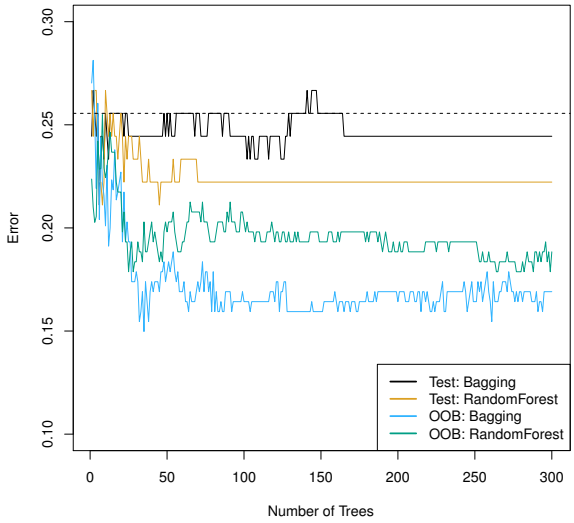
- Idea: la varianza del promedio es menor que la de una prediccion sola.

- Clasificacion: voto por mayoria.
- $B \sim 100$
- *Out of bag prediction:* en vez de CV, usar las observaciones que no entraron sorteadas al boostrap! Luego tomar promedio.

Pregunta (molesta): por que no hacer bagging con regression en econometria?

Random Forests

- Problema con bagging: si hay un predictor fuerte, distintos arboles son muy similares entre si. Alta correlacion. Realmente se reduce la varianza?
- Forests: bajar la correlacion entre los arboles en el boostrap.
- Si hay p predictores, en cada particion usar solo $m < p$ predictores, elegidos al azar.
- Bagging: forest con $m = p$ (usa todos los predictores en cada particion).
- Tipicamente $m = \sqrt{p}$


Error
0.10
0.15
0.20
0.25
0.30
0
50
100
150
200
250
300
Number of Trees
Test: Bagging
Test: RandomForest
OOB: Bagging
OOB: RandomForest

Boosting

- Problema con CART: alta varianza. Inestabilidad.
- Weak classifier: clasificador marginalmente mejor que tirar una moneda. Tasa de error apenas mejor que 0.5.
- Ej: CART con pocas ramas ('stump', dos ramas).

Boosting: promedio ponderado de una sucesion de clasificadores debiles. Notable mejora.

- $y \in -1, 1$ (por simplicidad), X vector de predictores.
- $\hat{y} = G(X)$ (clasificador)
- err $= \frac{1}{N} \sum_{i=1}^{N} I(y_i \neq G(x_i))$

AdaBoost.M1

1. Comenzar con pesos $w_i = 1/N$
2. Para $m = 1$ hasta M:
   1. Ajustar $G_m(x)$ usando ponderadores w_i.
   2. Computar error de prediccion

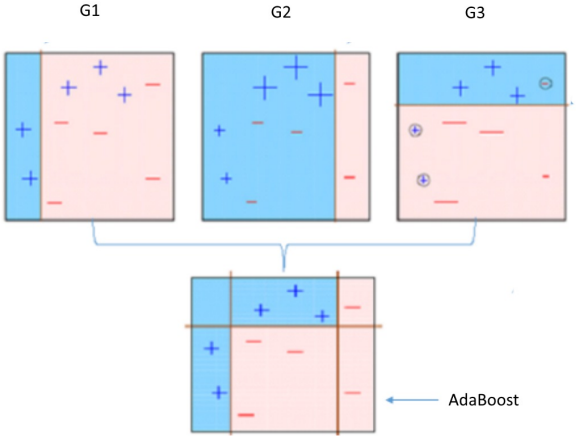
$$\text{err}_m = \frac{\sum_{i=1}^{N} w_i \, I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}$$

   3. Computar $\alpha_m = \ln \left[(1 - \text{err}_m)/\text{err}_m \right]$
   4. Actualizar ponderadores: $w_i \leftarrow w_i c_i$,

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

3. Output: $G(x) = \text{sgn} \left[\sum_{m=1}^{M} \alpha_m \, G_m(x) \right]$

- $c_i = \exp\left[\alpha_m \, I(y_i \neq G_m(x_i))\right]$
- Si i estuvo correctamente predicha, $c_i = 1$. No ajuste.
- Caso contrario, $c_i = \exp(\alpha_m) = (1 - \text{err}_m)/\text{err}_m > 1$ (por que?)
- En cada paso el metodo da mas importancia relativa a las observaciones mal predichas.
- Paso final: promedio ponderado de predicciones en cada paso.



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