XGBoost

Análisis de Datos 3

20 de diciembre de 2023

Métodos de *gradient boosting*

- *Gradient Boosting = Gradient Descent + Boosting*
- Boosting : buscar un modelo, de manera iterativa, como suma de otros modelos :

$$H(x) = \sum_t \rho_t \, h_t(x)$$

- En cada etapa se incluye (suma) un nuevo modelo para reducir los errores (residuos).
- El *adaboost* asocia los errores con las modificaciones en los pesos de ciertos puntos.
- En los métodos de *gradient boosting* los errores se identifican con los gradientes.
- Ambos puntos de vista se emplean para mejorar los modelos.

Gradient boosting para regresión

Dada una muestra $\big((x_1, y_1), \ldots, (x_n, y_n)\big)$ se puede usar la regresión simple para predecir de forma sencilla:

$$\hat{y} = F_1(x) = \bar{y} + \frac{\mathrm{Cov}(x, y)}{S_x^2}(x - \hat{x})$$

La idea es buscar un nuevo modelo F_2 para estimar los errores del modelo previo:

$$\Big(\big(x_1, y_1 - F_1(x_1)\big), \ldots, \big(x_n, y_n - F_1(x_n)\big)\Big)$$

El modelo resultante de esas dos etapas será:

$$\hat{y} = F_1(x) + F_2(x)$$

El proceso se puede repetir hasta cumplir algún criterio de parada.

Gradient boosting para regresión

En la regresión se emplea el criterio de mínimos cuadrados, es decir, se trata de buscar el mínimo de la función de pérdida

$$L(y, F(x)) = \frac{(y - F(x))^2}{2}$$

$$J = \sum_{i=1}^{n} L(y_i, F(x_i)) = \frac{1}{2} \sum_{i=1}^{n} (y_i - F(x_i)^2$$

Considerando cada $F(x_i)$ como un parámetro y derivando, se tiene:

$$\frac{\partial J}{\partial F(x_i)} = \frac{\partial \sum_j L(y_j, F(x_j))}{\partial F(x_i)} = \frac{\partial L(y_i, F(x_i))}{\partial F(x_i)} = F(x_i) - y_i$$

El gradiente cambiado de signo se puede identificar con los errores:

$$y_i - F(x_i) = -\frac{\partial J}{\partial} = -g(x_i)$$

Gradient boosting para regresión

La regresión lineal se puede expresar como un método *gradient boosting*.

- Se inicia el proceso estimando y mediante

$$\hat{y} = F_1(x) = \bar{y} = \frac{\sum_{i=1}^{n} y_i}{n}$$

- Se calculan los menos gradientes $-g(x_i) = y_i - F_1(x_i)$
- Se busca una nueva función F_2 para ajustar $-g(x_i)$

$$-\hat{g}(x) = F_2(x)$$

- El modelo final es $\hat{y} = F_1(x) + F_2(x)$

eXtreme Gradient Boosting (XGBoost)

- Pertenece a la familia de métodos *gradient boosting*.
- Se basa en modelos de árboles.
- Ganó varias competiciones de `kaggle`.
- Se puede usar desde R.

XGBoost

En la etapa t, el modelo predictivo es $F(x) = \sum_{k=1}^{t} F_k(x)$

Los predictores F_k están asociados a la función de pérdida elegida. Las más usuales son:

- Cuadrática

$$L = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

- Clasificación binaria (logística)

$$L = -\sum_{i=1}^{n} \Big(y_i \log p_i + (1 - y_i) \log \big(1 - p_i\big) \Big)$$

- Clasificación múltiple

$$L = -\sum_{i=1}^{n} \sum_{j=1}^{c} y_{i,j} \log p_{i,j}$$

XGBoost

En la búsqueda del modelo se incluye un término de regularización, que penaliza la complejidad e impide el sobreajuste. Cuando se usan árboles, se puede definir mediante la expresión

$$\Omega = \gamma T + \lambda \sum_{j=1}^{p} w_j^2$$

donde T = número de hojas y w_j = puntuación de la hoja j-ésima.
La función objetivo que se maneja para construir el modelo es

$$\text{Objetivo} = \text{Obj}(y, \hat{y}) = L + \Omega$$

En cada etapa se calcula $\dfrac{\partial \text{Obj}(y, \hat{y})}{\partial \hat{y}}$ y se modifican las predicciones siguiendo la dirección del gradiente.

XGBoost

En la iteración t, la búsqueda del nuevo predictor se basa en

- La expresión de cada predicción:

$$\hat{y}_i^t = \hat{y}_i^{t-1} + F_t(x_i)$$

- La función objetivo

$$\text{Obj}^t = \sum_{i=1}^{n} L\left(y_i, \hat{y}_i^{t-1} + F_t(x_i)\right) + \sum_{j=1}^{t} \Omega(F_j)$$

Si la función de pérdida es cuadrática se tiene:

$$\text{Obj}^t = \sum_{i=1}^{n} \left((y_i - \hat{y}_i^{t-1}) - F_t(x_i)\right)^2 + \sum_{j=1}^{t-1} \Omega(F_j) + \Omega(F_t)$$

$$= \sum_{i=1}^{n} \left(-2\,(y_i - \hat{y}_i^{t-1})\,F_t(x_i) + F_t(x_i)^2\right) + \Omega(F_t) + \text{C}^{\text{te}}$$

XGBoost

La búsqueda de la solución se hace usando una aproximación de Taylor de orden dos de la función objetivo.

$$f(x+\Delta x) \approx f(x) + f'(x)\,\Delta x + \frac{1}{2} f''(x)\,\Delta x^2$$

$$\text{Obj}^t = \sum_{i=1}^{n} \Big(L(y_i, \hat{y}_i^{t-1}) + g_i F_t(x_i) + \frac{1}{2} h_i F_t^2(x_i) \Big) + \Omega(F_t) + \text{C}^{\text{te}}$$

$$\text{con } g_i = \frac{\partial L(y_i, \hat{y}_i^{t-1})}{\partial \hat{y}_i^{t-1}} \text{ y } h_i = \frac{\partial^2 L(y_i, \hat{y}_i^{t-1})}{\partial (\hat{y}_i^{t-1})^2}.$$

Si las funciones predictoras se construyen usando árboles, la función objetivo se puede reescribir de manera sencilla:

$$\text{Obj}^t \approx \sum_{i=1}^{n} g_i F_t(x_i) + \frac{1}{2} h_i F_t^2(x_i) + \Omega(F_j) + \text{C}^{\text{te}} =$$

$$= \sum_{i=1}^{n} g_i F_t(x_i) + \frac{1}{2} h_i F_t^2(x_i) + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 + \text{C}^{\text{te}} =$$

$$= \sum_{j=1}^{T} \left(\sum_{i \in I_j} g_i w_j + \frac{1}{2} \sum_{i \in I_j} (h_i + \lambda) w_j^2 \right) + \gamma T$$

donde $I_j = \{i \mid i \in \text{hoja } j\}$

Datos perdidos:

1. En un árbol binario, se asignan todos los casos con datos perdidos
   - al nodo izquierdo, y se calcula la mayor ganancia.
   - al nodo derecho, y se calcula la mayor ganancia.
2. Se elige la dirección con mayor ganancia.

Al final, todo nodo tiene una *dirección por omisión* para datos con valores perdidos.

XGBoost

El manejo del XGBoost depende de muchos parámetros que se utilizan para resolver los siguientes problemas:

- Evitar el sobreajuste.
- Muestras muy desbalanceadas.

Los parámetros se pueden agrupar en tres clases:

- Generales
- Relacionados con el *boosting*
- Parámetros operativos

Parámetros generales

booster	árbol o modelo lineal (casi siempre es mejor el árbol)
nthread	número de hilos de ejecución

Parámetros de *boosting*

eta	tasa de aprendizaje (más o menos conservadora)
gamma	reducción mínima para dividir
lambda	coeficiente de suma de pesos^2
max_depth	máxima profundidad del árbol
min_child_weight	peso mínimo para dividir
subsample	proporción de entrenamiento

Parámetros operativos

objective función de pérdida para el aprendizaje (por omisión, regresión de mínimos cuadrados)

- `"reg:linear"` (por omisión)
- `"binary:logistic"` (0 y 1)
- `"multi:softmax"`
  `num_class`=$k \Longrightarrow (0, \ldots, k-1)$

eval_metric medida para validación (por omisión, raíz cuadrada del error cuadrático medio)

- `rmse`
- `error` (proporción)
- auc

XGBoost

Bibliografía

- https://hastie.su.domains/Papers/ESLII.pdf
  Elements of Statistical Learning
- https://github.com/dmlc/xgboost/

Paquete xgboost de R

```
install.packages("xgboost")
library("xgboost")
?agaricus.train # referencia: label = poisonous
data(agaricus.train) ; atr <- agaricus.train
data(agaricus.test)
str(atr)
atr$data$Dimnames[[2]] # nombres de variables
b <- xgboost(atr$data, atr$label, nrounds=100)
b <- xgboost(atr$data, atr$label, nrounds=100,
             nthread=parallel::detectCores(),
             objective="binary:logistic")
table(agaricus.test$label,
      predict(b, agaricus.test$data) > .5)
```

Paquete xgboost de R

```
download.file(paste0(
 "http://bellman.ciencias.uniovi.es/~carleos",
 "/master/manadine/curso1/AnalisisDatos1",
 "/4-neuronas/dat/mnist.rda"),"/tmp/mnist.rda")
load("/tmp/mnist.rda")
Me <- M[1:60000,] # entrenamiento
b <- xgboost(as.matrix(Me[-785]), # -Label
        as.numeric(as.character(Me$Label)) - 1,
        nrounds=100, num_class=10,
        objective="multi:softmax")
Mv <- M[60001:70000,]
t <- table(Mv$Label,
           predict(b, as.matrix(Mv[-785])))
sum(diag(t)) # 98%
```