EST-25134: Aprendizaje Estadístico

Profesor: Alfredo Garbuno Iñigo — Primavera, 2023 — Extensiones Arboles.
Objetivo: En esta sesión estudiaremos una extensión de los árboles de decisión como mecanismos iterativos de partición del espacio de atributos. Estudiaremos un método reciente, que resulta más robusto a selecciones de hiper-parámetros. Además conectaremos esto con una técnica estadística de mucha utilidad en la era del *Big Data*, pruebas de hipótesis múltiples..
Lectura recomendada: Capítulo 3 de [2]. Pruebas de hipótesis múltiples pueden ser consultadas en el libro de Efron and Hastie [1].

1. INTRODUCCIÓN

Los árboles de decisión por construcción son insensibles a:

- Variables continuas con *outliers*.
- Variables continuas con diferentes escalas.
- Variables categóricas (no se necesitan transformar a *dummies*).

El algoritmo de ajuste de un árbol de decisión, en regresión por ejemplo, busca minimizar la función de pérdida

$$R_\alpha(T) = \sum_{m=1}^{|T|} \sum_{i:x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha|T|\,, \tag{1}$$

donde $\hat{y}_{R_m} = \sum_{i:x_i \in R_m} y_i / n_m$ es el promedio de las respuestas en la región m-ésima, a tráves de ir buscando secuencialmente las regiones R_m que logren la máxima disminución de RSS.

La búsqueda se puede lograr por medio de un algoritmo voraz (*greedy*) que escoge variables y puntos de corte.

1.0.1. Para pensar: En nuestra última discusión sobre árboles de decisión es que tienen un sesgo al utilizar variables categóricas con muchas etiquetas. ¿Por qué?

1.1. Motivación

Consideremos el experimento siguiente. Tenemos una colección de 7 atributos y una respuesta. Los datos son generados de manera aleatoria.

```
set.seed(108727)
generate_data(nsamples = 100) ▷
    print(n = 3)
```

```
# A tibble: 100 × 8
      y    ch2 m2    m4    m10   m20      nor      uni
  <dbl> <dbl> <fct> <fct> <fct> <fct>  <dbl>    <dbl>
1 2.08   1.86 1     2     10    16     0.107  0.00758
2 0.804  2.40 2     3     9     4     -0.309  0.804
3 0.313  1.29 2     2     4     20     1.18   0.658
# ... with 97 more rows
#  Use 'print(n = ...)' to see more rows
```

De manera artificial consideraremos que la respuesta y está relacionada con los demás atributos (sabemos que no es cierto).

```
fit_tree ← function(engine){
  data_train ← generate_data()

  tree_spec ← decision_tree(tree_depth = 2) ▷
    set_engine(engine) ▷
    set_mode("regression")

  tree_spec ▷
    fit(y ∼ ., data = data_train) ▷
    extract_fit_engine() ▷
    vi() ▷
    filter(Importance > 0) ▷
    mutate(rank = min_rank(desc(Importance)))
}
```

Repetimos este proceso (generar datos aleatorios y ajustar un árbol) un número determinado de veces y registramos cuántas veces cada atributo fue utilizado en el nodo raíz.

En contraste, con un modelo denominado `conditional trees` somos capaces de evitar ese sesgo. Incluso podemos evitar escoger variables de corte que no tienen asociación con la respuesta.

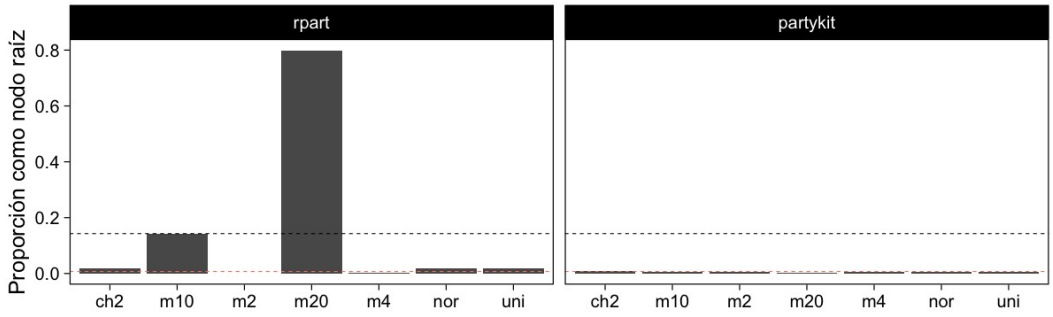

FIGURA 1. *La línea negra representa la probabilidad de seleccionar una variable al azar como nodo raíz ($p = 1/7$) y la cruva salmón representa la tolerancia de error (tasa de falsos positivos) ajustada ($\alpha = 0.05/7$).*

2. INFERENCIA CONDICIONAL

Recordemos que lo que necesitamos para construir un árbol de decisión es iterar:

1. Seleccionar una variable;
2. Seleccionar un punto de corte.

La variable se puede escoger primero por medio de comparaciones estadísticas entre un atributo X_j y la respuesta Y:

1. Hacer una prueba de χ^2 si ambas son variables nominales.
2. Hacer una prueba de correlación si ambas son continuas.

El punto de corte se puede escoger con una métrica de impureza.

Consideremos una colección de datos $(x_i, y_i) \overset{\text{iid}}{\sim} \mathbb{P}_{\mathcal{X},\mathcal{Y}}$ con $n = 1, \ldots, n$. Lo que queremos contrastar es

$$H_0 : \qquad \mathbb{P}(Y|X) = \mathbb{P}(Y)\,. \tag{2}$$

La prueba adecuada se puede escoger de acuerdo a las características de X y Y.

En general podemos utilizar un mecanismo general por medio de un vector de estadísticas

$$T = \mathsf{vec}\left(\sum_{i=1}^{n} g(x_i)h(y_i)^\top\right) \in \mathbb{R}^{pq}\,, \tag{3}$$

donde el operador `vec` transforma matrices en vectores (por arreglo), $g : \mathcal{X} \to \mathbb{R}^p$ es una `transformación de atributos` y $h : \mathcal{Y} \to \mathbb{R}^q$ es un `función de influencia`.

Cómo escogemos g y h es lo que nos permite realizar las pruebas de hipótesis adecuadas:

- Pruebas de correlación;
- Pruebas de muestras pareadas;
- Pruebas de K-muestras similares a pruebas ANOVA;
- Pruebas de independencia en tablas de contingencia.

Independientemente de la prueba, para poder utilizar este mecanismo debemos de saber la `distribución de muestreo` de T bajo la hipótesis nula.

La ventaja que tenemos es que la estructura del problema de contraste de hipótesis nos permite dejar fijos los atributos y realizar permutaciones $\sigma \in S$ sobre la respuesta. Es decir,

$$((x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)) \mapsto \left((x_1, y_{\sigma(1)}), (x_2, y_{\sigma(2)}), \ldots, (x_n, y_{\sigma(n)})\right)\,, \tag{4}$$

Denotemos por μ_h el valor esperado de la función de influencia condicional en la permutación σ y por Σ_h la matriz de varianzas-covarianzas asociada a ese valor esperado.

Esto nos permite calcular el valor esperado y matriz de varianzas covarianzas del estadístico T condicional en la permutación $\sigma \in S$, los cuales denotamos μ y Σ respectivamente.

Finalmente con esto podemos calcular un estadístico de prueba para H_0. Esto se puede lograr a través de una forma cuadrática o un estadístico de orden

$$c_{\mathsf{quad}} = (T - \mu)^\top \Sigma^{-1} (T - \mu)\,, \tag{5}$$

$$c_{\mathsf{max}} = \text{máx} \left| \frac{T - \mu}{\mathsf{diag}(\Sigma)^{1/2}} \right|\,. \tag{6}$$

Lo ideal es poder tener conocimiento de la distribución de muestreo de c_{quad} y c_{max}. Esto con la intención de poder construir valores p para contrastar H_0. Por ejemplo, podemos calcular

$$\mathbb{P}\left(c(T, \mu, \Sigma) \leq z | S\right)\,, \tag{7}$$

la cual se puede calcular como el número de permutaciones que tienen un estadístico menor que el nivel z dividido por el número total de permutaciones. Y evaluar si se satisface un umbral, definido por el analista, para controlar el nivel de significancia estadística con

$$\alpha = \mathbb{P}\{\text{rechazar } H_0 \text{ cuando es verdadera}\}\,. \tag{8}$$

Utilizar remuestreo (*bootstrap*) nos puede ayudar a calcular aproximaciones hasta un nivel de tolerancia dado (mas detalles de esto en mi curso de simulación (`EST-24107: Simulación`)).

O podemos argumentar por algún resultado asintótico para determinar distribuciones de muestreo que permitan un cálculo mas eficiente. Por ejemplo, en el caso $pq = 1$ podemos utilizar

$$c_{\mathsf{quad}} \sim \chi^2_1\,, \qquad c_{\mathsf{max}} \sim \mathsf{N}(0, 1)\,. \tag{9}$$

2.1. Ejemplos de pruebas de hipótesis de independencia

Bajo el caso de dos variables continuas podemos utilizar el mecanismo de inferencia condicional para probar independencia estadística. Por ejemplo, regresemos a nuestro ejemplo de jugadores de *baseball*.

```
library(coin)
independence_test(Salary ~ Years, data = hitters, teststat = "quadratic")
```

```

        Asymptotic General Independence Test

data:  Salary by Years
chi-squared = 42, df = 1, p-value = 9e-11
```

Incluso también podríamos aplicar lo mismo para nuestro conjunto de datos ficticio.

```
fake_data ← generate_data()
independence_test(y ~ nor, data = fake_data, teststat = "quadratic")
```

```

        Asymptotic General Independence Test

data:  y by nor
chi-squared = 0.3, df = 1, p-value = 0.6
```

Por supuesto, también podríamos hacer una comparación para atributos categóricos. Por ejemplo, regresando a nuestro ejemplo de *Scooby Doo*. El que el monstruo del capítulo sea real o no, ¿tiene relación con que *Scooby* haya sido capturado en el episodio?

```
independence_test(monster_real ~ caught_scooby,
                  data = scooby_data, teststat = "quadratic")
```

```

        Asymptotic General Independence Test

data:  monster_real by caught_scooby (caught, not caught)
chi-squared = 13, df = 1, p-value = 3e-04
```

Adicionalmente, podríamos contrastar con una aproximación por remuestreo a la distribución de permutaciones:

```
independence_test(monster_real ~ caught_scooby,
                  data = scooby_data, teststat = "quadratic",
                  distribution = approximate(nresample = 10000))
```

```

        Approximative General Independence Test

data:  monster_real by caught_scooby (caught, not caught)
chi-squared = 13, p-value = 1e-04
```

3. ÁRBOLES CONDICIONALES

El objetivo es: construir árboles de decisión con una estrategia de particionado recursivo sin sesgo basada inferencia condicional.

En cada paso de selección de variables lo que hacemos es realizar una prueba de hipótesis por cada atributo. Es decir, buscamos

$$H_0^j : \qquad \mathbb{P}(Y|X_j) = \mathbb{P}(Y)\,, \qquad j = 1, \ldots, m\,. \tag{10}$$

Dado que cada estadístico asociado $c(T^j, \mu^j, \Sigma^j)$ puede tener un escala muy distinta necesitamos estandarizar todos a la misma. Es por esto, que contrastar los valores p asociados a cada prueba nos ayudan a tener control sobre la escala de las comparaciones.

Adicionalmente, por diseño de las pruebas de hipótesis podríamos aceptar algunos falsos positivos. Esto puede ocurrir con una tasa de α (el nivel de tolerancia de error de las pruebas). Así que lo que buscamos es buscar que la colección de pruebas que realicemos tenga una tasa de error controlada

$$\mathsf{FWER} \leq \alpha\,. \tag{11}$$

Esto lo podemos lograr, por ejemplo, con la corrección de Bonferroni, al considerar cada umbral como α/m.

El argumento es

$$\begin{aligned}
\mathsf{FWER} &= \mathbb{P}\{\text{reject any true } H_0^j\} && (12)\\
&= \mathbb{P}\left\{\cup_{j\in I_0} \text{reject } H_0^j \text{ when it's true}\right\} && (13)\\
&= \mathbb{P}\left\{\cup_{j\in I_0} \left(p_j \leq \frac{\alpha}{m}\right)\right\} && (14)\\
&\leq \sum_{j=1}^{m} \mathbb{P}\left(p_j \leq \frac{\alpha}{m}\right) = \alpha\,, && (15)
\end{aligned}$$

donde I_0 denota el conjunto de hipótesis nulas verdaderas.

> Decimos que la variable aleatoria X es una variable aleatoria `super-uniforme` cuando domina estocásticamente una variable aleatoria uniforme. Es decir, cuando
>
> $$\mathbb{P}\{X \leq u\} \leq u \quad \text{ para toda } u \in [0, 1]\,. \tag{16}$$

Con la tasa de error controlada, y una vez seleccionada la variable para realizar el corte entonces procedemos a proponer puntos de corte para dicho atributo. El cual nos lleva a contrastar hipótesis para las distribuciones condicionales

$$\{y_i | x_{ij} < s\} \qquad \text{y} \qquad \{y_i | x_{ij} \geq s\}\,. \tag{17}$$

4. CONCLUSIONES

El modelo de `CTree` (disponible a través de `partykit`) es un modelo:

1. que utiliza pruebas de hipótesis para determinar variables y puntos de corte;
2. tiene un mecanismo de selección insesgado;
3. no requiere mucho post-procesamiento (poda).
4. el nivel α es un hiper-parámetro para estos modelos.

De acuerdo a Greenwell [2] , aún cuando `CTree` tiene mejores propiedades estadística que `CART` hay un uso generalizado por el último debido a herramientas de código abierto.

Además, de acuerdo a Loh [3] mientras un árbol se escoja por cuestiones predictivas y no por inferencia tiene un riesgo bajo al utilizar procedimientos con sesgo. Por otro lado, validación cruzada puede ayudar a eliminar ramas redundantes durante el proceso de poda (mientras tengamos pocos atributos y un número suficiente de datos).

REFERENCIAS

[1] B. Efron and T. Hastie. *Computer Age Statistical Inference*. Cambridge University Press, 2016. 1

[2] B. M. Greenwell. *Tree-Based Methods for Statistical Learning in R*. Chapman and Hall/CRC, Boca Raton, first edition, may 2022. ISBN 978-1-00-308903-2. . 1, 6

[3] W.-Y. Loh. Fifty Years of Classification and Regression Trees. *International Statistical Review / Revue Internationale de Statistique*, 82(3):329–348, 2014. ISSN 0306-7734. 6