Árboles de decisión y churn rate

Escuela de Matemáticas Bourbaki

Mayo 2020

Contents

1 Introducción

Las siguientes notas son la bitácora de un curso de 9 horas que impartimos en una alianza con Data Science Institute. Además de este documento los invitamos a consultar el Github del curso en este link.

El curso es una invitación al uso de los árboles de decisión para problemas de clasificación binaria, el curso está dividido en clases de la siguiente manera:

1. ¿Qué es machine learning y qué son los árboles de decisión? (una hora)
2. Un vistazo a Python (una hora).

3. Descripción formal de los algoritmos que aprende árboles de decisión (dos horas).
4. Implementación de los árboles de decisión para la predicción de churn rate (dos horas).
5. Aspectos avanzados de los árboles de decisión (tres horas).

1.1 ¿Qué es Machine Learning y los árboles de decisión?

Machine learning es un conjunto de técnicas que nos permiten hacer predicciones utilizando información del pasado. Sin embargo algunas veces hacer predicciones correctas no es suficiente pues nos gustaría tener información de por qué cierta predicción se ha hecho para enriquecer nuestro entendimiento del problema, en machine learning aquellas técnicas que nos proporcionen dicha información se dice que son algoritmos transparentes.

En este curso hablaremos con detalle de los árboles de decisión los cuáles se construyen con algoritmos donde el concepto de entropía (orden en la información) es escencial. Describiremos los detalles de esta familia de algoritmos y después hablaremos sobre el proceso de interpretación de los resultados utilizando árboles de decisión así como métodos más avanzados. Una diferencia fundamental con el curso de redes neuronales es que esta vez las entradas de nuestra base de datos serán binarias, eso corresponde al caso cuando la información que tenemos en cada uno de nuestros atributos solo puede tomar dos valors distintos.

1.2 Nociones matemáticas

- Un subconjunto importante de los números reales es $\mathbb{N}$ llamada el conjunto de los números naturales y consiste en los siguientes números $\mathbb{N} = \{1, 2, 3, \ldots\}$.
- Si $d \in \mathbb{N}$ es un número natural entonces denotaremos por $\{0,2\}^d$ al conjunto de vectores $(x_1, x_2, \ldots, x_d)$ de tamaño d con entradas en $\{0,2\}$. Por ejemplo $\{0,2\}^2 = \{(0,0), (0,2), (2,0), (2,2)\}$.
- El logaritmo de base 2 es la función definida únicamente para valores positivos y que satisface la siguiente ecuación: $log_2(x) = n$ si y solo si $2^n = x$. Notemos que:
  1. $log_2(1) = 0$
  2. Si $0 < x < 1$ entonces $log_2(x) < 0$
  3. Si $x \leq y$ entonces $log_2(x) \leq log_2(y)$.
- Si Ω es un conjunto y $A, B \subseteq \Omega$ son dos subconjuntos, denotaremos por:
  1. $A \cup B$ a la unión entre A y B.
  2. $A \cap B$ a la intersección entre A y B.
  3. A^c al complemento de A.
- Fijemos un conjunto finito Ω. Una distribución de probabilidad es una asignación numérica $\mathbb{P}$ a cada elemento $A \subseteq \Omega$ tal que si A, B son dos subconjuntos:
  1. $0 \leq \mathbb{P}(A) \leq 1 = \mathbb{P}(\Omega)$
  2. $\mathbb{P}(A^c) = 1 - \mathbb{P}(A)$
  3. $\mathbb{P}(A \cup B) = \mathbb{P}(A) + \mathbb{P}(B)$, si $A \cap B = \emptyset$.

  A la pareja $(\Omega, \mathbb{P})$ se le llamará un espacio de probabilidad.
- (Distribución uniforme) supongamos que Ω es un conjunto de tamano m. Definamos la siguiente ley de probabilidad $\mathbb{P}_{unif}(\omega_1, \ldots, \omega_i) = \frac{i}{m}$. Para fijar ideas se puede pensar en este ejemplo cuando $m = 6$, esto corresponde a la probabilidad de obtener algún resultado cuando lanzamos un dado.

- (Distribución de Bernoulli) Supongamos que Ω es un conjunto de tamaño 2 i.e. $\Omega = \{\omega_1, \omega_2\}$. Sea $0 \leq p \leq 1$ un número arbitrario. Definimos $\mathbb{P}_{Bernoulli}(\omega_1) = 1 - p$ y $\mathbb{P}_{Bernoulli}(\omega_2) = p$. Para fijar notación supongamos que tenemos una moneda cargada a sol (digamos ω_1) tal que de cada 10 lanzamientos, 8 son sol, en ese caso $p = \frac{2}{10}$.
- Sea $(\Omega, \mathbb{P})$ un espacio de probabilidad. Sean $A, B \subseteq \Sigma$ son dos eventos aleatorios tales que $\mathbb{P}(B) > 0$. Definimos una nueva ley de probabilidad $\mathbb{P}(A|B) = \mathbb{P}\frac{(A \cap B)}{\mathbb{P}(B)}$ lo cual se lee como la probabilidad del evento A una vez que haya ocurrido el evento B.

2 Instalación de Python y un entorno de trabajo para Machine-Learning

La forma más rápida y eficiente de instalar Python, manejar sus librerías y evitar problemas de dependencia(por actualizaciones y versiones) es mediante el uso de **Miniconda** que además de la instalación de Python, instala conda[1] un sistema gestor de paquetes y de entornos virtuales [1] [2].

Una forma alternativa es utilizar Anaconda, que al igual que Miniconda, es soportada por la misma compañía Anaconda. Ambas alternativas instalan la misma versión de Python y de conda. La razón por la que se recomienda Miniconda es porque permite instalar solamente lo que se requiere para este curso, con la opción de descargar otros paquetes cuando se requieran. Anaconda por el contrario instala por defecto, una cantidad excesiva de paquetes que raramente son utilizados, lo cual requiere demasiado espacio y tiempo.

2.1 Instalación de Miniconda [3]

Descarga aquí el archivo correspondiente a tu sistema operativo, ya sea que sea que MacOS, Windows y Linux, eligiendo la versión de Python 3.7.

2.1.1 Instalación para Windows

1. Da doble clic en el archivo descargado.
2. Sigue las instrucciones que se muestren aceptando las opciones que se proponen por defecto (si es necesario se pueden cambiar después).
3. Desde el menú de Inicio de Windows abre el programa Anaconda Prompt.

2.1.2 Instalación para MacOS

1. En la Terminal de tu Mac, navega hasta el directorio en donde descargaste el instalador y corre la siguiente línea: *bash Miniconda3-latest-MacOSX-x86_ 64.sh*
2. Sigue las instrucciones, aceptando las opciones por defecto (si es necesario se pueden cambiar después)
3. Cierra la Terminal y vuélvela a abrir, para que los cambios sean actualizados.

[1] Los entornos virtuales permiten controlar las versiones de software (en nuestro caso python y sus librerías) usado para análisis o aplicaciones

2.1.3 Instalación para Linux

1. En la Terminal de Linux, navega hasta el directorio en donde descargaste el instalador y corre la siguiente línea: *bash Miniconda3-latest-Linux-x86_ 64.sh*

2. Sigue las instrucciones, aceptando las opciones por defecto (si es necesario se pueden cambiar después)

3. Cierra la Terminal y vuélvela a abrir, para que los cambios sean actualizados.

Para todos los sistemas operativos:
4. Para ver todos los paquetes que instalados por defecto en la distribución de Miniconda, corre la siguiente línea: *conda list*

2.2 Creación de un entorno virtual para análisis de datos

Un entorno virtual es un ambiente de trabajo aislado, lo que permite instalar determinadas librerías o versiones de librerías sin que afecte al resto del sistema principal o de otros ambientes.

Por defecto, Miniconda crea un entorno llamado base que estará activo cuando abramos la terminal. Este entorno contiene todos los programas enlistados cuando se utilizó el comando conda list. Aunque podríamos usar este entorno, una mejor práctica es crear un entorno nuevo. Para ello es conveniente cambiar la configuración de conda para que no abra automáticamente el entorno base.

2.2.1 Desactivar el entorno base

• En Mac y Linux, se debe correr la linea que se muestra a continuación y enseguida cerrar y volver a a abrir la terminal: *conda config –set auto_ activate_ base false*

• Los usuarios de Windows deberán desactivar manualmente el entrono base utilizando *conda deactivate*

2.2.2 Creación de un entrono específico para análisis de datos mediante conda-forge

El entorno que vamos a crear lo titularemos MachineLearning, en éste instalaremos las librerías que necesitaremos para nuestros análisis.

1. Crea el ambiente de trabajo mediante la siguiente línea: *conda create -n MachineLearning*

2. Actívalo con la siguiente instrucción: *conda activate MachineLearning*

3. Agrega el canal [2] conda config - - env - -add channels conda-forge

   Se recomienda utilizar el canal conda-forge, ya que es resultado de un esfuerzo colectivo con una gran variedad de librearías y paquetes que son cuidadosamente actualizados y donde se asegura tener versiones compatibles para macOS, Linux y Windows[4].

   Puedes ver los canales instalados mediante la siguiente línea: *conda config - -show channels*

   Cada canal tiene al menos una dirección URL asociada donde se localizan los repositorios de paquetes y librerías. Puedes ver estas direcciones utilizando: *conda info*

[2]Los repositorios desde donde podemos descargar las librerías de Python se llaman canales, Anconda, Inc provee por defecto el canal llamado default, sin embargo permite a cualquier usuario crear su propio canal si necesita otros paquetes que no están incluidos en el canal default, por esta razón existe una una variedad de canales disponibles en la web desde donde se pueden descargar librerías de Python.

4. Instala las librerías Pandas[3], Scikit-learn[4], Pydotplus [5], Seaborn, Matplotlib[6] y Jupyter Notebooks[7] copiando el siguiente comando (es importante copiarlo en una sola línea, separando mediante un espacio, cada librería a ser instalada):

   conda install pandas matplotlib notebook jupyter_ contrib_ nbextensions pydotplus scikit-learn pydotplus seaborn

3 Árboles de decisión y sus algoritmos de entrenamiento

A lo largo del curso supondremos que estamos en un problema clásico de aprendizaje supervisado tipo clasificación binaria, esta vez con entradas en $\{0,2\}^d$, buscamos entrenar a nuestro modelo con un conjunto de ejemplos

$$S = \{(x_1, z_1), \ldots, (x_N, z_N)\}$$

cuando $x_i \in \{0,2\}^d, z_i \in \{-1,+1\}$.

En este curso además supondremos que nuestros datos están ordenados i.e. los ejemplos clasificados como -1 están primero y después el resto:

$$S = \{(x_1,-1), (x_2,-1), \ldots, (x_m,-1), (x_{m+1},1), (x_{m+2},1), \ldots, (x_N,1)\}$$

3.1 Árboles de decisión

Definition 3.1. Un árbol de decisión de profundidad r y dimensión d es un conjunto ordenado de r-preguntas respecto a un vector $x = (x_1, \ldots, x_d) \in \{0,2\}^d$:

- x_{i_1} es igual a cero o a dos?
- x_{i_2} es igual a cero o a dos?
- x_{i_3} es igual a cero o a dos?

...

- x_{i_r} es igual a cero o a dos?

Remark 3.1. *Notemos que no hay garantía de que después de haber respondido a todas las preguntas de un árbol de profundidad $r < d$ sea posible responder si un ejemplo $x \in \{0,2\}^d$ se clasifica como positivo o negativo como lo muestra la segunda parte del siguiente ejemplo.*

Exercise 3.2. *Sea $S = \{(0,0,1), (0,2,-1), (2,0,-1), (2,2,1)\}$. Encontrar el árbol de clasificación que prediga la clasificación en S. ¿Es posible encontrar un árbol de clasificación de profundidad uno que también prediga la clasificación en S?*

Definition 3.2. (Voto mayoritario) En este curso dado un árbol de decisión M, formado por preguntas correspondientes a atributos $i_1, i_2, \ldots, i_r$, dado un ejemplo x el árbol de decisión hará una predicción para x siguiendo la regla del voto mayoritario i.e. $M(x) = -1$ si y solo si $\frac{S_{M,x,-1}}{S_{M,x}} \geq \frac{S_{M,x,+1}}{S_{M,x}}$ donde $S_{M,x,-1} = \{(x',-1) \in S : x_i = x'_i, \forall i \in \{i_1, i_2, \ldots, i_s\}\}$, $S_{M,x,+1} = \{(x',+1) \in S : x_i = x'_i, \forall i \in \{i_1, i_2, \ldots, i_s\}\}$ y $S_{M,x} = \{(x',z) \in S : x_i = x'_i, \forall i \in \{i_1, i_2, \ldots, i_s\}\}$.

[3]Pandas es la principal librería de python utilizada Analisis de datos. Al instalar Pandas, por defecto se instala Numpy, sobre la cual funciona Pandas. Numpy es ampliamente usada en Ciencia de Datos porque permite el fácil manejo de matrices y sus operaciones.

[4]Scikit-learn es una importante librería para machine learning

[5]Provee una interface para Python del lenguaje Graphviz's Dot que utilizaremos para graficar árboles de decisión

[6]Matplotlib es las principales librerías de Python usadas para graficar y visualizar información

[7]Jupyter notebook es una aplicación que nos ayudará a hacer nuestros análisis paso a paso, crear visualizaciones e incluir comentarios, como si se tratara de nuestro cuaderno de apuntes.

La ventaja y al mismo tiempo el gran problema de los árboles de decisión es la siguiente proposición:

Proposition 3.3. *Toda función $f : \{0,2\}^d \to \{-1,+1\}$ corresponde con un árbol de decisión con profundidad d y viceversa.*

Lo anterior genera un gran problema debido a la posibilidad de incurrir en sobre-ajuste, lo cual podría convertir machine learning en una experiencia desagradable.

Definition 3.3. Si hemos fijado un tiempo t y una base de datos S_t, supongamos que utilizaremos un algoritmo A para predecir un modelo M, se dice que el el modelo M ha incurrido en sobre-ajuste relativo a la información anterior cuando M incluye sistemáticamente el ruido estocástico (aleatorio) de la base S_t.

Para explicar con formalidad las desventajas causadas por ese sobre-ajuste podemos enunciar el siguiente resultado.

Proposition 3.4. *La cantidad de ejemplos necesaria para entrenar un árbol de decisión sin incurrir en sobre-ajuste crece de manera proporcional a 2^d.*

El resultado anterior se puede entender como una maldición de la dimensión pues a medida que d crezca, la cantidad de ejemplos necesarios 2^d podría ser excesivamente grande.

Una manera posible para resolver el problema de la clasificación es utilizando la llamada navaja de Occman la cuál se puede enunciar de la siguiente manera.

Theorem 3.5. *(Navaja de Occman no formal) Para cada tiempo t consideramos una base de datos S_t, supongamos que utilizaremos un algoritmo A para predecir un modelo M y además utilizando el mismo algoritmo habríamos podido llegar también a otra predicción M'. El modelo que tiene menos probabilidad de incurrir en overfitting es aquel que sea más simple de enunciar.*

Remark 3.6. *Vale la pena mencionar que el enunciado anterior es cierto en general para algoritmos supervisados en machine learning.*

Con el fin de que el enunciado anterior no sea muy complicado de entender no lo hemos escrito en el lenguaje formal matemático, sin embargo es posible hacerlo y demostrarlo forlamente.

En particular no hemos descrito formalmente lo que significa la frase "aquel que sea más simple de enunciar", sin embargo en nuestro caso particular de árboles de decisión es posible darle un sentido simple descrito en el siguiente corolario.

Corollary 3.7. *Si nuestro algoritmo A genera dos árboles de decisión M y M' de tal forma que ambos cometen el menor error posible en el conjunto de entrenamiento S, aquel que tenga una menor profundidad es el que menos posibilidad tiene de incurrir en sobre-ajuste.*

Remark 3.8. *Existe una contradicción entre dos objetivos utilizados en el corolario anterior: por un lado queremos que el algoritmo genere un arbol de decisión con el menor error posible en el conjunto de entrenamiento y por otro queremos que su longuitud sea lo menor posible. Una manera de solucionar esta contradicción es planteando un problema de optimización, desafortunadamente este problema resulta ser muy complicado. En la práctica no intentamos solucionar este problema teórico sino plantearemos algoritmos de entrenamiento que localmente funcionan correctamente, estos algoritmos en la práctica resultan ser suficientemente exitosos, están basados conceptos provenientes de la teoría de la información.*

3.2 Algoritmo de aprendizaje y el concepto de entropía

Hasta el momento no hemos hablado del algoritmo que entrena de manera eficiente un árbol de decisión, en esta sub-sección describiremos el algoritmo ID3 el cuál utiliza fuertemente el concepto de entropía que introduciremos a continuación.

Definition 3.4. Con la notación anterior de la base de datos S, definimos las siguientes dos cantidades: $p = \frac{m}{N}, q = \frac{N-m}{N}$. La entropía de nuestra base de datos se define de la siguiente manera:

$$E(S) = -p \cdot log_2(p) - q \cdot log_2(q)$$

Remark 3.9. *La definición anterior de entropía es un caso particular de una definición mucho más general para espacios de probabilidad (de hecho se puede definir también para variables aleatorias). Para el lector familiarizado con el concepto de espacio de probabilidad, en nuestro caso estamos considerando un espacio de probabilidad Donde el conjunto es* $\Omega = \{+1, -1\}$ *y la distribución de probabilidad es* $\mathbb{P}(+1) = q, \mathbb{P}(-1) = p$. *En ese caso nuestra definición de entropía se denota como* $E(\Omega, \mathbb{P}) = -p \cdot log_2(p) - q \cdot log_2(q)$.

Exercise 3.10. *Graficar la función anterior como una función de* m.

Remark 3.11. *Debemos interpretar la función de entropía como una medida de desorden, si todos los puntos en* S *están ya sea clasificados como* -1 *o* $+1$ *entonces diremos que el conjunto* S *está ordenado, el caso opuesto es cuando están clasificados en proporciones iguales.*

Antes de definir el algoritmo $ID3$ que permite entrenar árboles de decisión es necesario introducir la siguiente funció de ganancia.

Definition 3.5. Utilizando la noción anterior comenzamos definiendo las siguientes cantidades:

- $c_{j,-1}$ es la cantidad de puntos $x_i \leq N$ que:
  1. $(x_i, -1) \in S$
  2. $x_{i,j} = 0$
- $d_{j,-1}$ es la cantidad de puntos $x_i \leq N$ que:
  1. $(x_i, -1) \in S$
  2. $x_{i,j} = 2$
- $c_{j,1}$ es la cantidad de puntos $x_i \leq N$ que:
  1. $(x_i, 1) \in S$
  2. $x_{i,j} = 0$
- $d_{j,1}$ es la cantidad de puntos $x_i \leq N$ que:
  1. $(x_i, 1) \in S$
  2. $x_{i,j} = 2$
- $c_j = c_{j,-1} + c_{j,+1}, d_j = d_{j,-1} + d_{j,+1}$.

Para cada $j \leq d$ definimos la función de ganancia con respecto a la cualidad j en la base de datos S:

$$Gain(S, j) = E(S) - \frac{c_j}{N}\left(-\frac{c_{j,-1}}{c_j} log_2\left(\frac{c_{j,-1}}{c_j}\right) - \frac{c_{j,+1}}{c_j} log_2\left(\frac{c_{j,+1}}{c_j}\right)\right) - \frac{d_j}{N}\left(-\frac{d_{j,-1}}{d_j} log_2\left(\frac{d_{j,-1}}{d_j}\right) - \frac{d_{j,+1}}{d_j} log_2\left(\frac{d_{j,+1}}{d_j}\right)\right)$$

Remark 3.12. *Utilizando el concepto de espacio de probabilidad mencionado en la observación 3.9 es posible aligerar la notación de la función Gain, comencemos definiendo los siguientes espacios de probabilidad.*

- $\Omega_{C,j} = \{C_{j,-1}, C_{j,+1}\}$, $\mathbb{P}(C_{j,-1}) = c_{j,-1}, \mathbb{P}(C_{j,+1}) = c_{j,+1}$

- $\Omega_{D,j} = \{D_{j,-1}, D_{j,+1}\}$, $\mathbb{P}(D_{j,-1}) = d_{j,-1}, \mathbb{P}(D_{j,+1}) = d_{j,+1}$

Entonces podemos re-escribir la función de ganancia de la siguiente forma: $Gain(S,i) = E(S) - \frac{c_j}{N} E(\Omega_{C,j}) - \frac{d_j}{N} E(\Omega_{D,j})$.

Remark 3.13. *Una manera de interpretar lo anterior es como la diferencia del error que comente nuestro árbol de decisión antes y después de hacer la pregunta ¿la entrada* j *es* 0 *o* 2*?, de hecho es posible demostrar lo siguiente cuando la función* E *es la entropía respecto al parámetro* m*:*

$$Gain(S,j) = E(\mathbb{P}(z=-1)) - (\mathbb{P}(x_j=0) E(\mathbb{P}(z=-1|x_j=0)) + \mathbb{P}(x_j=1) E(\mathbb{P}(z=-1|x_j=1)))$$

Ahora por fin estamos listos de describir nuestro algoritmo de aprendizaje con detalle:

Definition 3.6. (ID3) Sea $S' \subseteq S$ un subconjunto de nuestra base de datos y $D \subseteq \{1,2,\ldots,d\}$, definimos inductivamente el algoritmo $ID3(S',D)$ de la siguiente manera:

- Si todos los ejemplos en S' están clasificados como $+1$ o -1 regresar el árbol de decisión constante que corresponda.
- Si $D = \emptyset$ regresar el árbol de decisión constante correspondiente a aquella clasificación mayoritaria en S'
- Si no, sea $j = \underset{i \in D}{argmax}(Gain(S',i))$
- Agregar al árbol de decisión la pregunta: ¿la entrada j es 0 o 2?
- Agregar los árboles de decisión correspondientes a $ID3(\{(x,z) \in S' : x_j = 0\}, D \backslash \{j\})$ y $ID3(\{(x,z) \in S' : x_j = 2\}, D \backslash \{j\})$.

Remark 3.14. *Un problema de este algoritmo es que no hemos definido ninguna restricción respecto al tamaño del arbol de decisión que entrena, es posible arbitrariamente definir un límite en su profundidad y eso mejorará en general el desempeño del algoritmo. En el capítulo* 5 *hablaremos de otra técnica llamada pruning que permite restringuir la profundidad de nuestros árboles de decisión.*

4 Classificador binario de del churn rate para compañías de telecomunicaciones

4.1 Churn rate

El churn rate es una medida de desapego de los clientes relativo a algún producto en particular, normalmente se define respecto a algún periodo de tiempo, digamos un trimestre.

Definition 4.1. Si al inicio de un trimestre T había n clientes que comsumían nuestro producto P y al final había m de ellos que lo seguían consumiendo, se define el churn rate relativo al producto P y al trimestre T como $churn(T,P) = \frac{m}{n}$.

Esto significa que si $churn(T,P) = 1$ entonces el periodo T ha sido muy exitoso pues no se han perdido ninguno de los clientes. Por el contrario si $churn(T,P)$ es cercano a cero entonces ha sido un mal periodo.

También es posible definir el churn relativo a un cliente en particular, diremo que su churn rate será igual a cero si se ha ido e igual a uno si ha permanecido. En estas notas utilizaremos esa definición.

4.2 Descripción de la base de datos

Relativo a una empresa de telecomunicaciones buscamos entender las características más relevantes para la retención de clientes de una empresa de telecomunicaciones. El conjunto de datos se tomó de Kaggle Kaggle. Éste cuenta con 7,043 renglones y 21 columnas.

Cada renglón representa un cliente y cada columna un atributo. Entre los atributos se incluyen:

Características socidemograficas: sexo, si tienen pareja y dependientes económicos Información de la cuenta: antiguedad como cliente en la compañía, contrato, forma de pago, monto del pago mensual, etc Servicios que recibe: telefono, si tiene multiples líneas, internet, seguridad, respaldos, TV, etc Clientes que se abandonaron la compañía en el último mes ('Churn') En la fase de preprocesamiento, se elimina el ID del cliente, se sustituyen los valores nulos por 0 (solo 11, presentes en la variable TotalCharges), se realiza una transformación de las variables categóricas en binarias mediante la codificación One Hot Encoding, y se balancean los casos. De esta manera, al final del preprocesamiento, el conjunto para usar en el árbol de decision tiene 3,738 renglones y 34 columnas.

5 Temas selectos en árboles de decisión

Existen diversos métodos para mejorar la capacidad estadística o computacional de los árboles de decisión, en esta sección estudiaremos algunas de estas ideas.

5.1 Índice de impureza Gini

En la sección 3.2 describimos un algoritmo de aprendizaje que utiliza el concepto de entropía para elegir el feature $j \leq d$ que mejor separa a la información sin embargo existen otras maneras de hacerlo, en esta sección introducimos el índice Gini de impureza, el algoritmo de entrenamiento en este caso se le conoce como CART.

Definition 5.1. Sea S una base de datos binaria como en el capítulo 3, definimos el índice Gini y el índice de Gini relativo a $j \leq d$ de la siguiente manera.

- $Gini\,(S) = 1 - \left(\frac{m}{N}\right)^2 - \left(\frac{N-m}{N}\right)^2$.
- $Gini\,(S,j) = \frac{c_j}{N}\left(1 - \left(\frac{c_{j,-1}}{c_j}\right)^2 - \left(\frac{c_{j,+1}}{c_j}\right)^2\right) + \frac{d_j}{N}\left(1 - \left(\frac{d_{j,-1}}{d_j}\right)^2 - \left(\frac{d_{j,+1}}{d_j}\right)^2\right)$

La diferencia entre entropía e índice Gini es sutil como lo indica el siguiente resultado

Proposition 5.1.

- $Gini\,(S) = E\,(S) = 0$ *si y solo si* $m = 0$ *o* $m = N$.
- $Gini\,(S) = .5$ *y* $E\,(S) = 1$ *si y solo si* $\frac{N}{2} = m$
- $Gini\,(S) \leq E\,(S)$.

En este artículo se hace una comparación entre ambos índices, las diferencias son muy sutiles.

5.2 Random forests v.s. sobre-ajuste

Con el objetivo de reducir el sobre-ajuste existe un método introducido por Breiman en 2001 llamado random forest.

Definition 5.2. (Random Forest) Dada una base de datos supervisada S de un problema binario,

1. Para algún número $N' \leq N$ tomemos un sub-muestreo aleatorio (utilizando la ley de distribución uniforme sobre $\{1, 2, \ldots, N\}$ y de manera independiente) $S' \subseteq S$ de tamaño N'.

2. Para algún $k \leq d$ construyamos una familia de subconjuntos de tamaño k, $I_t \subseteq \{1, 2, \ldots, d\}$ construidos de manera independiente e idénticamente distribuidos respecto a la ley de probabilidad uniforme en $\{1, \ldots, d\}$.

3. Utilizando el algoritmo 3.6, construyamos una familia de árboles de decisión que correspondan a $ID3\,(S', I_t)$.

4. Si fijamos $t \leq T$ para algún T, predecir respecto a los árboles de decisión anteriores utilizando la regla del voto mayoritario 3.2.

Remark 5.2. *A medida que k sea suficienteme pequeño será más plausible prevenir sobre-ajuste.*

Es importante mencionar que existen diversas variantes de este método y en esa sección solo hemos descrito una de ellas.

5.3 Pruning

Pruning es una técnica que permite restringuir la profundidad de los árboles de decisión y por tanto mejorar la eficacia computacional durante el proceso de entrenamiento.

Esta ténica nació del célebre problema de crear un algoritmo capaz de vencer a un Gran Maestro de ajedrez. Después del trabajo fundamental de Shannon utilizando equilibrios de Nash para proponer un algoritmo utilizando árboles de decisión para crear una estrategia ganadora, los científicos de IBM se enfrentaron al problema de que los árboles eran demasiado profundos y por tanto la búsqueda irracionalmente costosa en términos computacionales. La mejora vino precisamente del llamado $\alpha\beta$-pruning para reducir su profundidad.

6 Referencias

[1] «Conda — conda 4.8.3.post5+125413ca documentation». https://docs.conda.io/projects/conda/en/latest/ (accedido mar. 26, 2020).

[2] Anaconda is Bloated - Set up a Lean, Robust Data Science Environment with Miniconda and Conda-Forge.

[3] «Miniconda — Conda documentation». https://docs.conda.io/en/latest/miniconda.html (accedido mar. 26, 2020).

[4] «A brief introduction — conda-forge 2019.01 documentation». https://conda-forge.org/docs/user/introduction.html (accedido mar. 26, 2020).