Áboles de Decisión, Bosques Aleatorios y Adaboost

Felipe Sánchez Soberanis

27 de octubre de 2022

Índice

Árboles de decisión **2**
¿Qué es un árbol de decisión? . . . 2
¿Por qué necesitamos un árbol de decisión? . . . 2

Bosques aleatorios **3**
¿Por qué no usar árboles de decisión? . . . 3
Bosques aleatorios . . . 3

Adaboost **4**

¿Cómo funciona AdaBoost? **4**
¿Cómo codificar AdaBoost en Python? . . . 4
Algoritmo Adaboost . . . 4

Árboles de decisión

Un árbol de decisión es una de las herramientas visuales de clasificación y predicción más sencillas, pero muy eficaz, que se utiliza en la toma de decisiones. El árbol toma un problema o situación de raíz y explora todos los posibles escenarios relacionados con él con base en numerosas decisiones. Dado que los árboles de decisión son muy ingeniosos, desempeñan un papel crucial en diferentes sectores.

¿Qué es un árbol de decisión?

Al principio, un árbol de decisión aparece como una estructura en forma de árbol con diferentes nodos y ramas. Cuando se observa un poco más de cerca, se puede ver que ha distribuido un problema o una situación en detalle. Se basa en los principios de clasificación que predicen el resultado de una decisión, dando lugar a diferentes ramas de un árbol. Parte de una raíz, que gradualmente tiene diferentes nodos de decisión. La estructura tiene nodos de terminación al final.

Un árbol de decisión puede utilizarse en casi todos los sectores. Esto se debe a que podemos tomar cualquier caso real o hipotético y representarlo mediante un diagrama de árbol de decisión. Para entender mejor qué es un árbol de decisión, consideremos este ejemplo. Plantea una pregunta sencilla: ¿comprar o no un nuevo software? Si compramos una nueva herramienta, entonces nos lleva a la comparación entre las dos opciones. Si no, podemos seguir utilizando el software actual o pedir prestada la herramienta de un amigo.

¿Por qué necesitamos un árbol de decisión?

Dado que un árbol de decisión proporciona un mapa sistemático del escenario actual y de las opciones disponibles, tiene ciertamente una amplia gama de aplicaciones. A continuación, se exponen algunas de las ventajas y razones para utilizar un diagrama de árbol de decisión.

- Con la ayuda de este diagrama, podemos resolver un problema abarcando todos los aspectos posibles.
- Desempeña un papel crucial en la toma de decisiones ya que nos ayuda a evaluar los pros y los contras de las diferentes opciones, así como su impacto a largo plazo.
- No se necesita ningún cálculo para crear un árbol de decisión, lo que los hace idóneos en todos los sectores.
- Estos diagramas pueden utilizarse para representar todo tipo de escenarios tanto categóricos como continuos que pueden ser difíciles de representarse de otro modo.
- En la actualidad, los árboles de decisiones desempeñan un papel fundamental en tecnologías futuristas como el aprendizaje automático y la inteligencia artificial.
- Pueden representar tanto los datos cuantitativos como los cualitativos de una manera visualmente atractiva y, además, sin aplicar demasiados cálculos.
- La mejor parte de los diagramas de árbol de decisión es que son extremadamente fáciles de crear y no requieren una gran formación.

Bosques aleatorios

¿Por qué no usar árboles de decisión?

Tienen un gran problema: el compromiso entre el bias (o sesgo) y la varianza.

Los árboles de decisión se caracterizan por ser capaces de ajustarse bastante bien a los datos de entrenamiento, es decir que tienen lo que se conoce como bias bajo. Pero desafortunadamente el error al momento de realizar la predicción (lo que se conoce como varianza) es generalmente alto.

Y esto se debe a que son muy sensibles a los datos de entrenamiento: un ligero cambio en tan sólo algunos de ellos puede dar origen a un árbol totalmente diferente, lo que dificulta aún más hacer predicciones con datos nuevos.

Bosques aleatorios

Los bosques aleatorios permiten resolver este problema, preservando lo positivo de los árboles de decisión (es decir un bias bajo) pero a su vez logrando reducir su varianza.

Esto se logra introduciendo dos variantes a estos árboles de decisión:

La primera es que en lugar de entrenar un único árbol se entrenan varios, usualmente decenas o cientos, de ahí precisamente el término "bosques".

Pero esto no es suficiente, por que si se entrena cada árbol del bosque con el mismo set de datos se seguiría teniendo el mismo problema de la varianza alta.

Así que la segunda parte de la solución es entrenar cada árbol del bosque con una parte distinta del set de datos original. Y acá es donde viene una característica de suma importancia: este subset de entrenamiento se selecciona aleatoriamente. De ahí precisamente el nombre "bosques aleatorios".

Con el bosque aleatorio ya entrenado, se puede hacer la predicción para un dato nuevo, y para esto se deben agregar los resultados individuales de los árboles. Es decir, que si se va a clasificar, la categoría asignada al dato será simplemente la categoría votada por la mayoría de los árboles, mientras que si la tarea es de regresión el valor asignado al dato será simplemente el promedio de las predicciones hechas por cada árbol.

Con la aleatoriedad cada árbol será entrenado con un subset diferente, y por tanto, si el set de entrenamiento tiene ruido este afectará probablemente sólo a unos cuantos árboles, pero no a la totalidad del bosque. Además, al agregar los resultados para generar la predicción, los árboles que no funcionan tan bien no tendrán un impacto significativo en el resultado final.

Así que al combinar estos dos elementos (la aleatoriedad y la agregación) se logra reducir la varianza de los árboles individuales.

Entonces un bosque aleatorio es como el equipo de jueces que evalúa a los clavadistas en los juegos olímpicos. Habrá jueces poco exigentes o demasiado estrictos (es decir que generarán una alta varianza) que asignarán puntajes o muy altos o muy bajos a los participantes. Pero al tener 7 u 8 jueces y obtener la puntuación de forma conjunta (es decir agregando los resultados) se evita tener puntajes excesivamente altos o demasiado bajos.

Adaboost

El algoritmo AdaBoost, abreviatura de Adaptive Boosting, es una técnica de Boosting utilizada como método de ensemble en el aprendizaje automático. Se denomina Boosting Adaptativo ya que los pesos se reasignan a cada instancia, con pesos más altos asignados a las instancias incorrectamente clasificadas. El Boosting se utiliza para reducir el sesgo y la varianza en el aprendizaje supervisado. Funciona según el principio de crecimiento secuencial de los aprendices. Excepto el primero, cada uno de los aprendices subsiguientes crece a partir de los aprendices anteriores. En palabras sencillas, los aprendices débiles se convierten en fuertes. El algoritmo AdaBoost funciona según el mismo principio que el boosting, con una ligera diferencia. Vamos a discutir esta diferencia en detalle.

¿Cómo funciona AdaBoost?

Hace "n" número de árboles de decisión durante el periodo de entrenamiento de los datos. Cuando se hace el primer árbol de decisión/modelo, se da prioridad al registro incorrectamente clasificado en el primer modelo. Sólo estos registros se envían como entrada para el segundo modelo. El proceso continúa hasta que especifiquemos un número de aprendices base que queramos crear. Recuerde que la repetición de registros está permitida con todas las técnicas de boosting.

El registro que se clasifica incorrectamente se utiliza como entrada para el siguiente modelo. Este proceso se repite hasta que se cumpla la condición especificada. Hay un número "n" de modelos realizados tomando los errores del modelo anterior. Así es como funciona el boosting. Los modelos 1, 2, 3, ..., N son modelos individuales que pueden conocerse como árboles de decisión. Todos los tipos de modelos de boosting funcionan según el mismo principio.

Cuando se utiliza el bosque aleatorio, el algoritmo hace un número "n" de árboles. Hace árboles propios que consisten en un nodo de inicio con varios nodos de hoja. Algunos árboles pueden ser más grandes que otros, pero no hay una profundidad fija en un bosque aleatorio. Con AdaBoost, sin embargo, el algoritmo sólo hace un nodo con dos hojas, conocido como Stump.

Estos tocones son aprendices débiles y las técnicas de boosting lo prefieren. El orden de los tocones es muy importante en AdaBoost. El error del primer muñón influye en cómo se hacen los demás muñones.

¿Cómo codificar AdaBoost en Python?

En Python, la codificación del algoritmo AdaBoost toma sólo 3-4 líneas y es fácil. El clasificador AdaBoost debe ser importado de la biblioteca sci-kit learn. Antes de aplicar AdaBoost a cualquier conjunto de datos, uno debe dividir los datos en entrenamiento y prueba. Después de dividir los datos en entrenamiento y prueba, los datos de entrenamiento están listos para entrenar el modelo AdaBoost. Estos datos tienen tanto la entrada como la salida. Después de entrenar los datos, el algoritmo intentará predecir el resultado en los datos de prueba. Los datos de prueba consisten sólo en las entradas. El modelo no conoce la salida de los datos de prueba. La precisión puede comprobarse comparando la salida real de los datos de prueba y la salida predicha por el modelo. Esto puede ayudar a concluir cómo está funcionando el modelo y qué grado de precisión puede considerarse, dependiendo del planteamiento del problema. Si se trata de un problema médico, la precisión debe ser superior al 90 %. Por lo general, un 70 % de precisión se considera bueno. La precisión también depende de otros factores además del tipo de modelo.

Algoritmo Adaboost

El algoritmo Adaboost es una buena técnica de conjunto y puede utilizarse tanto para problemas de clasificación como de regresión. En la mayoría de los casos, se utiliza para problemas de clasificación. Es mejor que cualquier otro modelo, ya que mejora la precisión del modelo que se puede comprobar yendo en secuencia. Primero se pueden probar los árboles de decisión y luego ir por el bosque aleatorio para finalmente aplicar el boost e implementar AdaBoost. La precisión sigue aumentando a medida que seguimos la secuencia anterior.

La técnica de asignación de pesos después de cada iteración hace que el algoritmo AdaBoost sea diferente de todos los demás algoritmos de boosting y eso es lo mejor que tiene.