Bagging and Random Forest

El Random Forest es uno de los algoritmos de aprendizaje automático más populares y poderosos, perteneciente a la categoría de algoritmos de conjunto conocida como Bootstrap Aggregation o bagging. Este capítulo explora el algoritmo de conjunto Bagging y el algoritmo Random Forest para modelado predictivo. Después de leer este capítulo, se adquirirá conocimiento sobre:

- El método de bootstrap para estimar cantidades estadísticas a partir de muestras.

- El algoritmo Bootstrap Aggregation para crear múltiples modelos diferentes a partir de un solo conjunto de datos de entrenamiento.

- El algoritmo Random Forest, que realiza pequeños ajustes a Bagging y resulta en un clasificador muy potente.

Este capítulo proporcionará una comprensión de estos conceptos fundamentales en el contexto de Random Forest y Bagging para el modelado predictivo.

28.1 Método Bootstrap

Antes de adentrarnos en Bagging, echemos un vistazo rápido a una técnica fundamental importante llamada bootstrap. El bootstrap es un método estadístico potente para estimar una cantidad a partir de una muestra de datos. Es más fácil de entender si la cantidad es una estadística descriptiva como la media o la desviación estándar. Supongamos que tenemos una muestra de 100 valores (x) y nos gustaría obtener una estimación de la media de la muestra. Podemos calcular la media directamente a partir de la muestra como:

$$mean(x) = \frac{1}{100} \times \sum_{i=1}^{100} x_i \qquad (28.1)$$

Sabemos que nuestra muestra es pequeña y que nuestra media tiene un error. Podemos mejorar la estimación de nuestra media utilizando el procedimiento de bootstrap:

1. Crear muchos (por ejemplo, 1000) submuestras aleatorias de nuestro conjunto de datos con reemplazo (lo que significa que podemos seleccionar el mismo valor varias veces).

2. Calcular la media de cada submuestra.

3. Calcular el promedio de todas nuestras medias recopiladas y usarlo como nuestra media estimada para los datos.

Por ejemplo, supongamos que utilizamos 3 remuestreos y obtuvimos los valores medios 2.3, 4.5 y 3.3. Tomando el promedio de estos, podríamos considerar la media estimada de los datos como 3.367. Este proceso se puede utilizar para estimar otras cantidades como la desviación estándar e incluso cantidades utilizadas en algoritmos de aprendizaje automático, como los coeficientes aprendidos.

28.2 Bootstrap Aggregation (Bagging)

Bootstrap Aggregation, o Bagging en resumen, es un método de conjunto simple pero poderoso. Los métodos de conjunto combinan predicciones de múltiples algoritmos de aprendizaje automático para mejorar la precisión en comparación con modelos individuales. Bagging es especialmente útil para reducir la varianza en algoritmos con alta varianza, como los árboles de decisión, como los Árboles de Clasificación y Regresión (CART).

Los árboles de decisión, especialmente CART, son sensibles a los datos de entrenamiento específicos que encuentran. Cambiar los datos de entrenamiento puede dar lugar a árboles de decisión y predicciones significativamente diferentes. Bagging aplica el procedimiento de Bootstrap a algoritmos de aprendizaje automático con alta varianza, típicamente árboles de decisión. Supongamos un conjunto de datos de 1000 instancias y el algoritmo CART; Bagging para CART procedería de la siguiente manera:

1. Crear múltiples (por ejemplo, 100) submuestras aleatorias del conjunto de datos con reemplazo.

2. Entrenar un modelo CART en cada submuestra.

3. Para un nuevo conjunto de datos, calcular la predicción promedio de cada modelo.

Por ejemplo, si cinco árboles de decisión con bagging predicen las clases "azul", "azul", "rojo", "azul" y "rojo" para una instancia de entrada, se predice la clase más frecuente (azul). En Bagging con árboles de decisión, se presta menos atención a que los árboles individuales sobreajusten los datos de entrenamiento. Por lo tanto, los árboles de decisión individuales se desarrollan en profundidad (pocos ejemplos de entrenamiento en cada nodo hoja) y no se podan, lo que resulta en alta varianza y baja sesgo. Estas características son esenciales al combinar predicciones mediante bagging.

El único parámetro a considerar al bagging con árboles de decisión es la cantidad de árboles a crear. Este parámetro se elige aumentando incrementalmente la cantidad de árboles hasta que la precisión deja de mejorar (por ejemplo, en una prueba de validación cruzada). Crear un gran número de árboles de decisión puede llevar tiempo, pero no sobreajustará los datos de entrenamiento. Bagging, al igual que los árboles de decisión, es aplicable tanto a problemas de clasificación como de regresión.

28.3 Bosque Aleatorio

Los Bosques Aleatorios son una mejora sobre los árboles de decisión con bagging. Un problema con los árboles de decisión como CART es que son ávidos. Eligen qué variable dividir utilizando un algoritmo ávido que minimiza el error. Como tal, incluso con Bagging, los árboles de decisión pueden tener muchas similitudes estructurales y, por ende, resultar en una alta correlación en sus predicciones. La combinación de predicciones de múltiples modelos en conjuntos funciona mejor si las predicciones de los submodelos no están correlacionadas o, en el mejor de los casos, están débilmente correlacionadas.

Random Forest cambia el algoritmo para la forma en que se aprenden los subárboles para que las predicciones resultantes de todos los subárboles tengan menos correlación. Es un ajuste simple. En CART, al seleccionar un punto de división, se permite que el algoritmo de aprendizaje examine todas las variables y todos los valores de variables para seleccionar el punto de división más óptimo. El algoritmo de bosque aleatorio cambia este procedimiento para que el algoritmo de aprendizaje esté limitado a una muestra aleatoria de características para buscar. El número de características que se pueden examinar en cada punto de división

(m) debe especificarse como un parámetro del algoritmo. Puedes probar diferentes valores y ajustarlo utilizando validación cruzada.

- For classification a good default is: $m = \sqrt{p}$.
- For regression a good default is: $m = \frac{p}{3}$.

Donde m es la cantidad de características seleccionadas aleatoriamente que se pueden examinar en un punto de división y p es la cantidad de variables de entrada. Por ejemplo, si un conjunto de datos tuviera 25 variables de entrada para un problema de clasificación, entonces:

$$m = \sqrt{25}$$
$$m = 5 \qquad (28.2)$$

28.4 Rendimiento Estimado

Para cada muestra bootstrap tomada del conjunto de datos de entrenamiento, habrá muestras dejadas atrás que no fueron incluidas. Estas muestras se llaman muestras Fuera de la Bolsa o OOB. El rendimiento de cada modelo en sus muestras omitidas, cuando se promedia, puede proporcionar una estimación de precisión de los modelos con bagging. Esta estimación de rendimiento se llama comúnmente la estimación OOB. Estas medidas de rendimiento son una estimación confiable del error de prueba y correlacionan bien con las estimaciones de error de validación cruzada.

28.5 Importancia de Variables

A medida que se construyen los árboles de decisión con bagging, podemos calcular cuánto disminuye la función de error para una variable en cada punto de división. En problemas de regresión, esto podría ser la disminución en el error cuadrático total y, en clasificación, podría ser el puntaje de Gini. Estas disminuciones en el error se pueden promediar en todos los árboles de decisión y producir una estimación de la importancia de cada variable de entrada. Cuanto mayor sea la disminución cuando se elige la variable, mayor será la importancia. Estas salidas pueden ayudar a identificar subconjuntos de variables de entrada que pueden ser más o menos relevantes para el problema y sugerir posibles experimentos de selección de características donde se eliminan algunas características del conjunto de datos.

28.6 Preparación de Datos para CART con Bagging

CART con Bagging no requiere ninguna preparación especial de datos, aparte de una buena representación del problema.

28.7 Resumen

En este capítulo, descubriste el algoritmo de conjunto de aprendizaje automático llamado Bagging y la variación popular llamada Bosque Aleatorio. Aprendiste:

- Cómo estimar cantidades estadísticas a partir de una muestra de datos.

- Cómo combinar las predicciones de múltiples modelos de alta varianza utilizando bagging.

- Cómo ajustar la construcción de árboles de decisión al aplicar bagging para descorrelacionar sus predicciones, técnica llamada Bosques Aleatorios.

Ahora conoces el algoritmo de conjunto de bagging, los árboles de decisión con bagging y los bosques aleatorios. En el próximo capítulo, descubrirás cómo implementar árboles de decisión con bagging desde cero.

Boosting y AdaBoost

Boosting es una técnica de conjunto que intenta crear un clasificador fuerte a partir de varios clasificadores débiles. En este capítulo, descubrirás el método de conjunto AdaBoost para el aprendizaje automático. Después de leer este capítulo, sabrás:

- Qué es el método de conjunto Boosting y cómo funciona de manera general.

- Cómo aprender a potenciar árboles de decisión utilizando el algoritmo AdaBoost.

- Cómo realizar predicciones utilizando el modelo AdaBoost aprendido.

- Cómo preparar mejor tus datos para usar con el algoritmo AdaBoost.

30.1 Método de Conjunto Boosting

Boosting es un método de conjunto general que crea un clasificador fuerte a partir de varios clasificadores débiles. Esto se logra construyendo un modelo a partir de los datos de entrenamiento, luego creando un segundo modelo que intenta corregir los errores del primer modelo. Se agregan modelos hasta que el conjunto de entrenamiento se predice perfectamente o se agrega un número máximo de modelos. AdaBoost fue el primer algoritmo de boosting realmente exitoso desarrollado para la clasificación binaria. Es el mejor punto de partida para comprender el boosting. Los métodos modernos de boosting se basan en AdaBoost, especialmente las máquinas de boosting de gradiente estocástico.

30.2 Aprendizaje de un Modelo AdaBoost a partir de Datos

AdaBoost se utiliza mejor para mejorar el rendimiento de los árboles de decisión en problemas de clasificación binaria. AdaBoost se llamaba originalmente AdaBoost.M1 por los desarrolladores de la técnica. Más recientemente, puede referirse como AdaBoost discreto porque se utiliza para la clasificación en lugar de la regresión. AdaBoost se puede utilizar para mejorar el rendimiento de cualquier algoritmo de aprendizaje automático. Se utiliza mejor con aprendices débiles.

Estos son modelos que logran precisión ligeramente superior al azar en un problema de clasificación. El algoritmo más adecuado y, por lo tanto, más comúnmente utilizado con AdaBoost son los árboles de decisión con un solo nivel. Debido a que estos árboles son tan cortos y contienen solo una decisión para la clasificación, a menudo se les llama "decision stumps" (pilares de decisión). Cada instancia en el conjunto de datos de entrenamiento tiene un peso. El peso inicial se establece en:

Wieght(xi) = 1/n

Donde x_i es la i-ésima instancia de entrenamiento y n es el número de instancias de entrenamiento.

30.3 Cómo entrenar un modelo

Un clasificador débil (pilar de decisión) se prepara en los datos de entrenamiento utilizando las muestras ponderadas. Solo se admiten problemas de clasificación binaria (dos clases), por lo que cada pilar de decisión toma una decisión sobre una variable de entrada y produce un valor +1.0 o -1.0 para el primer o segundo valor de clase. Se calcula la tasa de clasificación incorrecta para el modelo entrenado. Tradicionalmente, se calcula de la siguiente manera:

$$error = \frac{correct - N}{N}$$

Donde el error es la tasa de clasificación incorrecta, correcto es el número de instancias de entrenamiento clasificadas correctamente por el modelo y N es el número total de instancias de entrenamiento. Por ejemplo, si el modelo predijo correctamente 78 de 100 instancias de entrenamiento, la tasa de error o clasificación incorrecta sería (78-100)/100 = 0.22\). Esto se modifica para usar el peso de las instancias de entrenamiento:

$$error = \frac{\sum_{i=1}^{n}(w_i \times perror_i)}{\sum_{i=1}^{n} w}$$

Que es la suma ponderada de la tasa de clasificación incorrecta, donde \(w\) es el peso para la instancia de entrenamiento \(i\) y P_error es el error de predicción para la instancia de entrenamiento \(i\), que es 1 si está mal clasificado y 0 si está clasificado correctamente. Por ejemplo, si tuviéramos 3 instancias de entrenamiento con los pesos 0.01, 0.5 y 0.2. Los valores predichos fueron -1, -1 y -1, y las variables de salida reales en las instancias fueron -1, 1 y -1, entonces los valores de P_error serían 0, 1 y 0. La tasa de clasificación incorrecta se calcularía como:

$$error = \frac{0.01 \times 0 + 0.5 \times 1 + 0.2 \times 0}{0.01 + 0.5 + 0.2}$$
$$error = 0.704$$

Se calcula un valor de etapa (stage value) para el modelo entrenado, el cual proporciona un peso para cualquier predicción que haga el modelo. El valor de etapa para un modelo entrenado se calcula de la siguiente manera:

$$stage = ln(\frac{1 - error}{error})$$

Donde "stage" es el valor de etapa utilizado para ponderar las predicciones del modelo, ln() es el logaritmo natural y "error" es el error de clasificación errónea para el modelo. El efecto del peso de la etapa es que los modelos más precisos tienen más peso o contribución a la predicción final. Los pesos de entrenamiento se actualizan dando más peso a las instancias

incorrectamente predichas y menos peso a las instancias correctamente predichas. Por ejemplo, el peso de una instancia de entrenamiento (w) se actualiza utilizando:

$$w = w \times e^{stage \times perror}$$

Donde \(w \) es el peso para una instancia de entrenamiento específica, \(e \) es la constante numérica número de Euler elevada a una potencia, \(\text{stage}\) es la tasa de error de clasificación para el clasificador débil y \(\text{perror}\) es el error que el clasificador débil cometió al predecir la variable de salida para la instancia de entrenamiento, evaluado como:

. perror = 0 IF y == p
. perror = 1 IF y != p

Donde \(y \) es la variable de salida para la instancia de entrenamiento y \(p \) es la predicción del clasificador débil. Esto tiene el efecto de no cambiar el peso si la instancia de entrenamiento fue clasificada correctamente y aumentar ligeramente el peso si el clasificador débil clasificó incorrectamente la instancia.

30.4 Ensemble AdaBoost

Los modelos débiles se agregan secuencialmente, entrenados utilizando los datos de entrenamiento ponderados. El proceso continúa hasta que se hayan creado un número predeterminado de clasificadores débiles (un parámetro del usuario) o no se pueda realizar más mejora en el conjunto de datos de entrenamiento. Una vez completado, se obtiene un conjunto de clasificadores débiles, cada uno con un valor de etapa.

30.5 Realización de predicciones con AdaBoost

Las predicciones se realizan calculando el promedio ponderado de los clasificadores débiles. Para una nueva instancia de entrada, cada clasificador débil calcula un valor predicho como +1.0 o -1.0. Los valores predichos se ponderan según el valor de etapa de cada clasificador débil. La predicción para el modelo de conjunto se toma como la suma de las predicciones ponderadas. Si la suma es positiva, se predice la primera clase; si es negativa, se predice la segunda clase.

Por ejemplo, 5 clasificadores débiles pueden predecir los valores 1.0, 1.0, -1.0, 1.0, -1.0. Según una votación mayoritaria, parece que el modelo predecirá un valor de 1.0 o la primera clase. Estos mismos 5 clasificadores débiles pueden tener los valores de etapa 0.2, 0.5, 0.8, 0.2 y 0.9 respectivamente. Calcular la suma ponderada de estas predicciones da como resultado una salida de -0.8, que sería una predicción del conjunto de -1.0 o la segunda clase.

30.6 Preparación de Datos para AdaBoost

Esta sección enumera algunas heurísticas para preparar mejor sus datos para AdaBoost.

- **Datos de Calidad**: Dado que el método de conjunto continúa intentando corregir las clasificaciones erróneas en los datos de entrenamiento, es necesario asegurarse de que los datos de entrenamiento sean de alta calidad.

- **Outliers (Valores Atípicos):** Los valores atípicos llevarán al conjunto a trabajar arduamente para corregir casos que son poco realistas. Estos podrían eliminarse del conjunto de datos de entrenamiento.

- **Datos Ruidosos:** Los datos ruidosos, especialmente el ruido en la variable de salida, pueden ser problemáticos. Si es posible, intente aislar y limpiar estos datos de su conjunto de datos de entrenamiento.

30.7 Resumen

En este capítulo, has descubierto el método de conjunto Boosting para el aprendizaje automático. Aprendiste sobre:

- Boosting y cómo es una técnica general que sigue agregando aprendices débiles para corregir errores de clasificación.
- AdaBoost como el primer algoritmo exitoso de boosting para problemas de clasificación binaria.
- Cómo aprender el modelo AdaBoost ponderando las instancias de entrenamiento y los propios aprendices débiles.
- Cómo predecir con AdaBoost mediante la ponderación de las predicciones de los aprendices débiles.
- Dónde buscar más antecedentes teóricos sobre el algoritmo AdaBoost.

Ahora conoces el método de conjunto Boosting y el algoritmo de aprendizaje automático AdaBoost. En el próximo capítulo, descubrirás cómo implementar el algoritmo AdaBoost desde cero.