Combinación de modelos

Jordi Casas Roma
Julià Minguillón Alfonso

1 crédito
xxx/xxxxx/xxx

Índice

Introducción

La combinación de clasificadores o combinación de modelos (también conocido como "ensemble", en inglés) es una técnica que combina múltiples modelos de aprendizaje automático para mejorar el rendimiento y la generalización del sistema. En lugar de confiar en un único modelo muy potente para realizar la predicción, un *ensemble* utiliza un conjunto de modelos más débiles (y muy diferentes entre sí) en un esfuerzo por aumentar la precisión y robustez del sistema global.

La combinación de modelos se ha demostrado como una estrategia efectiva para mejorar el rendimiento predictivo y la robustez de los modelos de aprendizaje automático, y es ampliamente utilizado en diversas aplicaciones y contextos diferentes.

Objetivos

En los materiales didácticos de este módulo encontraremos las herramientas indispensables para asimilar los siguientes objetivos:

1. Comprender el concepto de combinación de modelos y sus principales características.
2. Conocer la combinación paralela de clasificadores, así como sus principales ventajas e inconvenientes.
3. Conocer la combinación secuencial de clasificadores, así como sus principales ventajas e inconvenientes.
4. Saber aplicar algunos de los principales modelos de este tipo, como por ejemplo Random Forest, AdaBoost y Gradient Boosting.

1. Contextualización

La combinación de clasificadores o combinación de modelos (también conocido como "ensembler", en inglés) es una técnica que combina múltiples modelos de aprendizaje automático para mejorar el rendimiento y la generalización del sistema. En lugar de confiar en un único modelo para realizar la predicción, un *ensembler* utiliza un conjunto de modelos más débiles o diferentes en un esfuerzo por aumentar la precisión y robustez del sistema global.

Dado un conjunto de M modelos de aprendizaje, donde cada modelo se denota por $M_i, 1 \leq i \leq B$, un *ensembler* combina las predicciones de cada modelo individual M_i para obtener una predicción final en base a las predicciones de los modelos base.

Para un problema de clasificación, el *ensembler* puede combinar las predicciones utilizando técnicas como votación mayoritaria, votación ponderada, o promediado. Por ejemplo, en el caso de votación mayoritaria, el *ensembler* seleccionaría la clase que haya sido predicha por la mayoría de los modelos base, resolviendo los posibles empates de alguna manera.

En el caso de un problema de regresión, el *ensembler* podría promediar (de forma ponderada o no) las predicciones de los modelos base para obtener la predicción final.

En relación con la notación de este texto, emplearemos la siguiente notación para referirnos al conjunto de entrada de datos:

$$D_{n,m} = \begin{pmatrix} d_1 = & a_{1,1} & a_{1,2} & \cdots & a_{1,m} & c_1 \\ d_2 = & a_{2,1} & a_{2,2} & \cdots & a_{2,m} & c_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ d_n = & a_{n,1} & a_{n,2} & \cdots & a_{n,m} & c_n \end{pmatrix}$$

donde:

- n es número de elementos o instancias,
- m el número de atributos del conjunto de datos o también su dimensionalidad,
- d_i indica cada una de las instancias del juego de datos,
- $a_{i,j}$ indica cada uno de los atributos descriptivos, y

- c_i indica el atributo objetivo o clase a predecir para cada elemento.

Es importante destacar que el atributo objetivo o clase a predecir (c_i) unicamente existe (o aplica) en los métodos supervisados. En el caso de los métodos no supervisados, el conjunto de datos sigue el mismo patrón y notación, excepto para el atributo de clase (c_i) que no existe (o no se emplea) en el caso de los conjuntos no etiquetados.

2. Esquemas conceptuales

En la actualidad, el estado del arte en minería de datos combina dos aproximaciones diferentes. La primera consiste en el desarrollo de nuevos algoritmos (lo cual ocurre muy ocasionalmente), o bien en el ajuste fino de los parámetros de algún algoritmo conocido, habitualmente para adaptarlo a la naturaleza concreta de unos datos o del problema a resolver. En este sentido, los avances en el área de minería de datos están asegurados y a lavez limitados por el llamado "*No free lunch theorem*" (Wolpert, 1996), el cual postula que no existe un algoritmo *a priori* que sea superior el resto para cualquier conjunto de datos.

Lectura complementaria

Wolpert, David H. (1996). "The lack of a priori distinctions between learning algorithms". Neural computation, Vol. 8(7) pp. 1341-1390

La segunda aproximación para crear mejores modelos consiste en combinar clasificadores más o menos sencillos para crear uno mucho más complejo, de forma que la decisión tomada sea una combinación de cientos o miles de decisiones parciales. Obviamente, los clasificadores usados como base para construir el clasificador combinado deben ser lo más diversos posible, con la esperanza de que los errores cometidos por un clasificador base concreto sean minoritarios con respecto al resto, de forma que los errores puntuales no alteren una decisión basada en la opinión correcta de la mayoría. Una revisión reciente de los trabajos en esta área de investigación puede encontrarse en el trabajo de Wozniak *et al.*, 2014.

Lectura complementaria

Michal Wozniak, Manuel Graña, Emilio Corchado. (2014). "A survey of multiple classifier systems as hybrid systems". Information Fusion, Vol. 16 pp. 3-17. https://doi.org/10.1016/j.inffus.2013.04

Existen dos opciones para la construcción de clasificadores combinados. La primera, que combina clasificadores trabajando en paralelo, es cuando todos los clasificadores base utilizan el mismo modelo o algoritmo (p.e. árboles de decisión), así que será necesario manipular el conjunto de entrenamiento original para generar diferentes clasificadores base y poder tomar una decisión conjunta a partir de la decisión parcial de cada uno de ellos. La segunda opción consiste en combinar clasificadores base muy diferentes (p.e. árboles de decisión y redes neuronales) de forma secuencial, de forma que cada clasificador utilice los resultados de un clasificador anterior, intentando capturar alguna característica clave de la naturaleza de los datos o del problema a resolver.

2.1. Combinación paralela de clasificadores base similares

Como se ha comentado, una opción es generar una gran cantidad de clasificadores base construidos a partir de la alteración del conjunto de entrenamiento original. Todos estos clasificadores base son muy parecidos, al estar basados en ligeras variaciones del conjunto de entrenamiento, pero no son idénticos, proporcionando diversidad al clasificador combinado, el cual combina (mediante un esquema de votación) las clasificaciones parciales para tomar una decisión.

Existen dos técnicas básicas para la creación del clasificador combinado, en función de cómo se genera el conjunto de entrenamiento de cada clasificador parcial y del peso asignado a cada uno de ellos en la votación final, llamadas *bagging* y *boosting*.

2.1.1. Bagging

La idea básica del *bagging* es utilizar el conjunto de entrenamiento original para generar centenares o miles de conjuntos similares usando muestreo con reemplazo. Es decir, de un conjunto D de n elementos se pueden escoger $n' \leq n$ al azar, existiendo la posibilidad de escoger un mismo elemento más de una vez (de ahí "con reemplazo"). Normalmente $n' = n$, por lo que en cada nuevo conjunto generado existirán elementos repetidos del conjunto original.

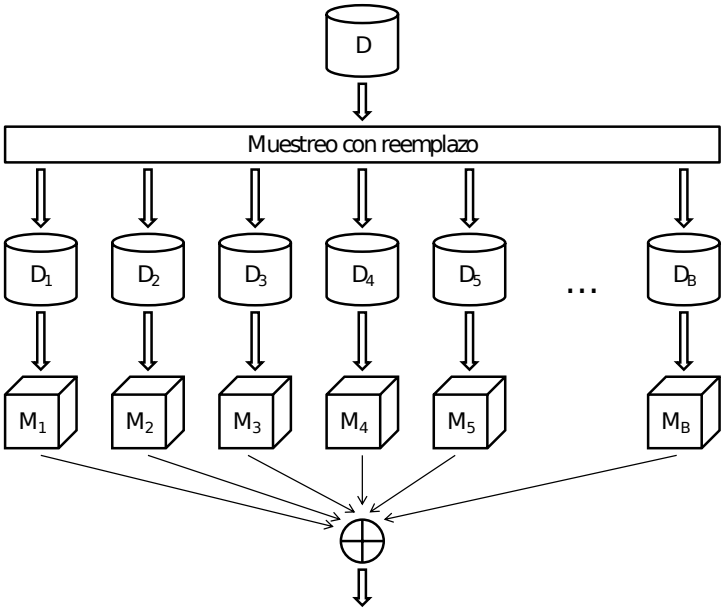
Aunque no es tan habitual, los conjuntos generados también pueden usar una dimensionalidad $m' \leq m$, de forma que no todas las mismas variables disponibles existan en los conjuntos generados. Esto puede ayudar a reducir el grado de colinearidad entre variables, haciendo emerger variables relevantes que pueden quedar siempre descartadas en frente de otra variable.

Una vez se han construido los conjuntos de entrenamiento parciales, se construye un clasificador para cada uno de ellos, siendo los árboles de decisión la opción más típica. De hecho, teniendo en cuenta la naturaleza del clasificador combinado, no es necesario crear clasificadores base muy precisos, ya que el objetivo es que los errores cometidos por cada clasificador base queden minimizados en frente de la decisión de la mayoría. Por lo tanto, en el caso de árboles de decisión, lo que es habitual es crear clasificadores más pequeños (es decir, limitados en profundidad) reduciendo el coste computacional de todo el conjunto.

En el caso del *bagging*, la decisión final se toma por mayoría, dando el mismo peso a todas las decisiones parciales. Es decir, la clase resultante del clasificador combinado es aquella que aparece más veces entre las decisiones parciales tomadas por los clasificadores base. La Figura 1 representa el proceso de creación del clasificador combinado usando B clasificadores parciales.

Una manera de medir el error cometido por el clasificador combinado se conoce como "Out-Of-Bag", dado que el error estimado es el promedio de todos los errores parciales cometidos por cada clasificador base para todos aquellos elementos no usados en el conjunto de entrenamiento usado para construirlo. De esta manera no es necesario separar los datos de entrada en un conjunto de entrenamiento y otro de test, sino que se usan todos para construir el clasificador combinado. No obstante, si se dispone de suficientes datos, es siempre recomendable utilizar un conjunto de test para validar el clasificador combinado.

Figura 1. Diagrama de un clasificador combinado basado en *bagging*



2.1.2. Boosting

La idea del boosting es ligeramente diferente. Se parte también de clasificadores base muy sencillos (también llamados débiles) de los cuales se supone que, al menos, cometen menos errores que aciertos, es decir, que funcionan ligeramente mejor que un clasificador aleatorio.

Se empieza construyendo un primer clasificador base usando el conjunto de entrenamiento original. Como el clasificador es débil (por ejemplo, un árbol de decisión de profundidad limitada), se cometen unos cuantos errores para el conjunto de entrenamiento. Entonces, para construir el siguiente clasificador base, lo que se hace es ponderar los elementos del conjunto de entrenamiento, de forma que aquellos que han sido clasificados erróneamente por el clasificador anterior tengan más peso y, de una manera u otra, tengan mayor peso también en el proceso de creación del siguiente clasificador base. El clasificador combinado es la combinación de la predicción del primer clasificador base más la del segundo, ponderadas de acuerdo a algún esquema de pesos que tenga en cuenta la precisión de cada clasificador. Este clasificador combinado también cometerá unos errores que se usarán para dar más peso a aquellos elementos erróneamente clasificados, construyendo un tercer clasificador combinado, etc.

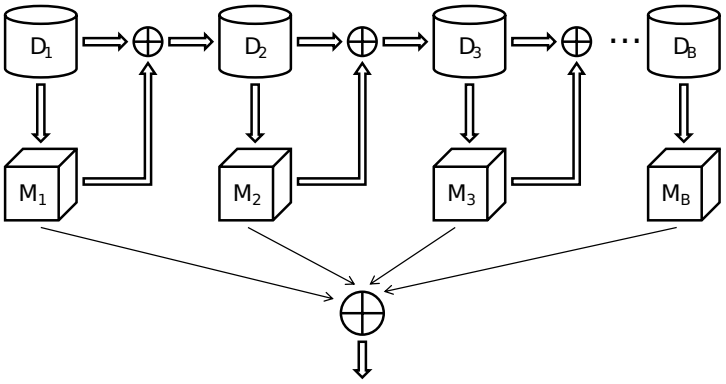
Es decir, en cada etapa se construye un clasificador nuevo que combina una serie de decisiones sucesivas que tienden a enfocarse en aquellos elementos más difíciles de clasificar, intentando que cada etapa corrija los errores de la anterior, pero sin estropear las decisiones correctas ya tomadas. Para ello, la secuencia de pesos de los sucesivos clasificadores parciales suele ser descendiente, de forma que el procedimiento se para

Lectura complementaria

Yoav Freund, Robert E. Schapire. (1996). "Experiments with a new boosting algorithm". In Proc. of the 13th Int. Conf. on Machine Learning, Vol. 96, pp. 148-156

cuando el siguiente clasificador ya no aporta nada respecto al anterior. Este esquema, mostrado en la Figura 2, permite múltiples variaciones, siendo la más conocida la descrita por Freund and Schapire en 1996, llamada AdaBoost.

Figura 2. Diagrama de un clasificador combinado basado en *boosting*



Es importante destacar que el proceso de creación del clasificador combinado es secuencial, dado que para realizar una nueva iteración y obtener un nuevo clasificador combinado es necesario haber evaluado los B clasificadores base anteriores. No obstante, una vez se ha alcanzado el clasificador combinado final, la evaluación se realiza también en paralelo, ya que en una primera etapa se genera la decisión parcial para cada uno de los clasificadores base y en la segunda se obtiene la clasificación final ponderando todas la decisiones parciales.

Uno de los problemas de *boosting* es la posibilidad de sobreentrenar el clasificador combinado, por lo que es necesario disponer de un conjunto de test e ir evaluando en cada etapa el resultado obtenido sobre el mismo, deteniendo el proceso cuando el error en el conjunto de test empieza a aumentar.

2.2. Combinación secuencial de clasificadores base diferentes

En este caso el objetivo es construir un clasificador que combina clasificadores base muy diferentes, con el objetivo de incrementar la diversidad de predicciones e intentar aprovechar todo el conocimiento explícito que se tenga sobre el problema a resolver o los datos disponibles. Por ejemplo, si se utiliza un árbol de decisión como clasificador, se pueden construir unos cuantos clasificadores que funcionen como una etapa previa, de forma que alimenten el árbol de decisión con información extraída de los datos u

otras decisiones parciales.

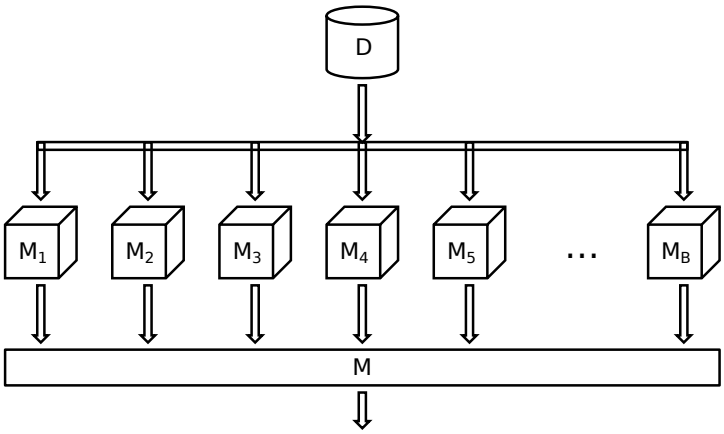
Se trata, entonces, de un proceso secuencial, dado que se generan unas cuantas decisiones parciales que posteriormente son usadas para tomar la decisión final. En función de la información que recibe el clasificador combinado de los clasificadores parciales, se distinguen dos métodos, llamados *stacking* y *cascading*.

Estas técnicas son adecuadas cuando la naturaleza del problema a resolver también presenta una estructura secuencial. Por ejemplo, en el diagnóstico no invasivo de tumores cerebrales mediante el uso de espectroscopia de resonancia magnética (Minguillón *et al.*, 2002), en lugar de intentar predecir el tipo de tumor detectado directamente (lo cual es muy complicado porque existen muchos tipos de tumores diferentes y de diferente grado de malignidad), lo habitual es establecer una secuencia de decisiones parciales intentando atacar el problema desde una primera decisión muy sencilla (p.e. establecer simplemente si se trata de un tumor o no), seguida de una segunda decisión que determina si se trata de un tumor benigno o maligno, seguida de una tercera que determina el grado de malignidad. Es decir, en cada paso se utiliza la decisión tomada anteriormente, empezando por una primera decisión sencilla que se va refinando en una secuencia de decisiones cada vez más específicas.

Lectura complementaria

Minguillón, J., Tate, A.R., Arús, C., Griffiths, J.R. (2002). "Classifier Combination for In Vivo Magnetic Resonance Spectra of Brain Tumours". In: Roli, F., Kittler, J. (eds) Multiple Classifier Systems. MCS 2002. Lecture Notes in Computer Science, vol 2364. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-45428-4_28

Figura 3. Diagrama general de un clasificador combinado basado en *cascading* y *stacking*



En general, se puede pensar en la combinación secuencial como una manera de abordar el problema, intentando resolver primero los casos más sencillos con un clasificador también sencillo, dejando el resto a una secuencia de clasificadores cada vez más complejos y específicos.

2.2.1. Stacking

La idea básica de *stacking* es construir diferentes clasificadores base de forma que cada uno de ellos genere una decisión parcial para cada elemento de entrada del conjunto de entrenamiento. Entonces se construye un nuevo clasificador usando como datos de entrada todas las predicciones parciales, en lugar de los datos originales de entrada. Este segundo clasificador suele ser un árbol de decisión, una red neuronal sencilla o una regresión logística (en el caso de que el número de clases sea dos).

Aunque no es habitual, esta estructura puede repetirse en diferentes niveles, combinando decisiones de otros clasificadores combinados, de ahí la idea de apilamiento o *stacking*. El problema de siempre es la tendencia a crear un clasificador combinado demasiado específico para el conjunto de entrenamiento que no generalice bien ante nuevos datos.

2.2.2. Cascading

El caso de *cascading* es parecido al de *stacking* pero utilizando no solamente las predicciones parciales de los clasificadores base, sino también los datos originales e incluso otros datos que se hayan podido generar durante la toma de decisiones. La idea básica es alimentar al clasificador combinado con decisiones parciales, así como los motivos que han llevado a tomar dichas decisiones.

Por ejemplo, cuando un árbol de decisión ha asignado una clase a un elemento del conjunto de entrada, dicha clase es el resultado de una serie de decisiones internas que llevan a una hoja, la cual tiene una profundidad en el árbol, representa una región que contiene un porcentaje de los datos de entrada y se conoce el error cometido al asignar dicha clase como representante de todos los datos de entrada contenidos en la región que representa dicha hoja. O por ejemplo, si uno de los clasificadores base es un algoritmo de clustering, se puede usar como información adicional a la clase o cluster asignado a un elemento del conjunto de entrada cada una de las distancias a cada uno de los centroides de los clusters, como medida de la fuerza que tiene dicha decisión.

2.3. Resumen

El uso de clasificadores combinados permite, en muchas ocasiones, mejorar la capacidad predictiva de un modelo, siendo posible también la inclusión de conocimiento relativo a la naturaleza del problema a resolver, especialmente con los clasificadores que operan secuencialmente. Su construcción es sencilla, utilizando en muchas ocasiones los mismos algoritmos y técnicas (reemplazo con muestreo, sistemas de votación por mayoría simple, etc.), siendo el caso de *boosting* el más complejo, aunque existen diferentes algoritmos que lo implementan.

No obstante, los clasificadores combinados también presentan una serie de problemas, algunos ya mencionados a lo largo de este módulo didáctico. El primer punto a tener en cuenta es el coste computacional que puede tener el clasificador combinado, tanto por lo que respecta a su creación (especialmente) como a su ejecución. No obstante, es posible aprovechar la estructura paralela de los clasificadores combinados en caso de disponer de infraestructura tecnológica que permita la paralelización de procesos, de forma que cada clasificador base se ejecuta en un nodo en paralelo (a la vez) a todos los otros clasificadores base, reduciendo así el tiempo de ejecución necesario.

Un aspecto a tener en cuenta es que no por aumentar el número de clasificadores base involucrados se va a conseguir siempre una mejora de la precisión. Es mucho más importante la diversidad de los clasificadores base que su número. Como la diversidad depende del conjunto de entrenamiento usado para cada clasificador base, es importante que el conjunto de entrenamiento original sea suficientemente grande y diverso y que el proceso de muestreo asegure un buen grado de aleatoriedad, generando conjuntos de entrenamiento parciales también diversos.

Por otra parte, la interpretabilidad de los resultados es mucho más complicada, ya que exige la interpretación de cientos o miles de clasificadores base parciales y su posterior combinación. No obstante, es posible medir la importancia relativa de cada variable en el conjunto mediante diferentes técnicas (p.e. contando cuantas veces aparece cada variable en cada clasificador base con su peso en el clasificador combinado), lo que puede proporcionar cierto conocimiento al respecto.

3. Algoritmos y métodos

En este capítulo veremos algunos de los principales algoritmos de combinación de modelos, aunque una revisión exhaustiva de los mismos queda fuera del alcance de este texto.

3.1. Random Forest

Random Forest (Breiman, 2001) es un algoritmo *ensembler* de tipo *bagging*, que se utiliza principalmente para tareas de clasificación y regresión.

Lectura complementaria

L. Breiman. (2001). "Random forests". Machine learning, Vol. 45(1):5-32

Este modelo combina múltiples árboles de decisión independientes y se utiliza un muestreo tanto de los elementos del conjunto original de entrenamiento como de sus variables. El clasificador combinado se conoce como "Random forest" , dado que se trata precisamente de un conjunto (o "bosque") de árboles que han sido creados mediante un proceso aleatorio (por lo que respecta al conjunto de entrenamiento usado para cada uno de ellos).

La práctica habitual consiste en generar versiones diferentes del conjunto de entrenamiento usando muestreo con reemplazo, tal y como define el método de *bagging*. Entonces, durante el proceso de construcción de cada árbol de decisión se selecciona aleatoriamente un subconjunto de las variables del conjunto de datos, dando opciones a variables que normalmente quedarían eclipsadas por otras que tuvieran mayor relevancia. Este procedimiento permite medir la importancia relativa de cada variable, estimando el error cometido por el clasificador combinado cuando se altera dicha variable, permutando aleatoriamente sus valores en el conjunto de test. Este error se mide para cada uno de los clasificadores parciales, promediando el error cometido en todos ellos para cada una de las variables. El porcentaje de error se compara al estimado con el conjunto de test sin dicha permutación aleatoria, de forma que es posible medir el impacto de dicha variable, dado que si el error aumenta cuando se permuta una variable, esto quiere decir que la variable es relevante para el problema que se está resolviendo.

El proceso para el entrenamiento de un modelo Random Forest se detalla en el Algoritmo 1.

El conjunto de datos de entrada D consta de n muestras y m características: $D = \{(d_1, c_1), (d_2, c_2), ..., (d_n, c_n)\}$, donde d_i es el vector de características de la muestra i y c_i es la etiqueta correspondiente. Los parámetros que recibe el algoritmo incluyen el número de árboles en el bosque (T), el número de características a considerar al

Algoritmo 1 Pseudocódigo del algoritmo Random Forest

Require: Training dataset $D = \{(d_1, c_1), (d_2, c_2), ..., (d_n, c_n)\}$

Require: Number of trees T, number of features to consider m', number of samples to consider n'

for $t = 1$ to T **do**

Randomly select n' samples with replacement to form D_t

Randomly select m' features from the available m features

Build a decision tree using D_t and the selected features

Repeat until the tree is fully developed

end for

return Set of T trained decision trees

buscar la mejor división en cada nodo del árbol (normalmente $m' \leq \sqrt{m}$ o $m' \leq \log_2 m$), que se denota como m', y el número de muestras a considerar para entrenar cada árbol (seleccionadas aleatoriamente con reemplazo), indicado como n', el cual es habitualmente n.

Como se puede apreciar en el detalle del algoritmo, el proceso que se sigue para entrenar el modelo es el siguiente:

1) Para $t = 1$ hasta T:

a) Seleccionar aleatoriamente n' muestras del conjunto de entrenamiento con reemplazo para formar un conjunto de datos D_t.

b) Seleccionar aleatoriamente m' características de las m' características disponibles.

c) Construir un árbol de decisión utilizando el conjunto de datos D_t y las características seleccionadas en el paso anterior. Cada división se elige buscando la característica y el umbral que maximizan la ganancia de información (o reducción de impureza) en los nodos resultantes.

d) Repetir los pasos anteriores hasta que el árbol esté completamente desarrollado (se haya alcanzado una profundidad máxima o se haya cumplido otro criterio de detención).

2) El conjunto final de árboles de decisión forma el bosque aleatorio.

Entre las principales ventajas o puntos fuertes del algoritmo Random Forest, podemos destacar el aumento de precisión en los resultados, dado que generalmente produce modelos más precisos que un solo árbol de decisión. También es notable destacar la reducción del sobreajuste (*overfitting*) que se suele producir en este modelo debido al uso de múltiples árboles en subconjuntos aleatorios de datos y características. Otro punto importante es que este modelo es capaz de proporcionar medidas de importancia de características, lo que puede ayudar a entender qué características son más relevantes para la tarea. Por último, pero no por ello menos relevante, cabe destacar que es posible la construcción de árboles en paralelo, lo que acelera el proceso de entrenamiento.

Asimismo, continua aportando algunas de las principales ventajas de los árboles de decisión, como son la robustez frente a datos ruidosos y valores atípicos, y la posibilidad de trabajar con características numéricas y/o categóricas sin necesidad de un preprocesamiento exhaustivo de los datos.

Por el contrario, también debemos mencionar que el consumo de recursos de este modelo es notablemente superior a otros, ya que se deben construir y entrenar múltiples árboles de decisión. Por este mismo motivo, el tiempo de predicción puede ser más largo debido a la combinación de las predicciones de los múltiples árboles.

Aunque hemos comentado que este modelo es capaz de proporcionar medidas de importancia de características, es cierto que la interpretación de un modelo Random Forest puede ser más costosa que la interpretación de un solo árbol de decisión ya que se deben inspeccionar todos los modelos construidos.

3.2. AdaBoost

Lectura complementaria

Yoav Freund, Robert E. Schapire. (1996). "Experiments with a new boosting algorithm". In Proc. of the 13th Int. Conf. on Machine Learning, Vol. 96, pp. 148-156

AdaBoost (*Adaptive Boosting*) fue propuesto por Freund y Schapire (1995) y consiste en crear varios predictores sencillos en secuencia, de tal manera que cada predictor intenta corregir los errores que ha cometido el predictor anterior. Es decir, el segundo predictor intentar ajustar correctamente los errores que el primer predictor, mientras que el tercero realiza la misma tarea sobre los errores cometidos por el segundo predictor, y así sucesivamente.

El modelo AdaBoost es un claro ejemplo de *boosting*, y sigue el esquema mostrado en la Figura 2.

El proceso para el entrenamiento de un modelo AdaBoost se detalla en el Algoritmo 2.

Algoritmo 2 Pseudocódigo del algoritmo AdaBoost

Require: Training dataset $D = \{(d_1, c_1), (d_2, c_2), ..., (d_n, c_n)\}$

Require: Number of weak learners T

Initialize sample weights: $w_1 = \frac{1}{N}, w_2 = \frac{1}{N}, ..., w_N = \frac{1}{N}$

for $t = 1$ to T **do**

Train a weak learner $f_t(d)$ using weights $w_1, w_2, ..., w_n$

Calculate weighted error: $err_t = \sum_{i=1}^{n} w_i \cdot \mathbb{1}(f_t(d_i) \neq c_i)$

Calculate importance coefficient: $\alpha_t = \frac{1}{2} \ln\left(\frac{1-err_t}{err_t}\right)$

Update sample weights: $w_i = w_i \cdot \exp\left(-\alpha_t \cdot c_i \cdot f_t(d_i)\right)$

Normalize sample weights: $w_1, w_2, ..., w_n = \frac{w_1}{\sum_{i=1}^{n} w_i}, \frac{w_2}{\sum_{i=1}^{n} w_i}, ..., \frac{w_n}{\sum_{i=1}^{n} w_i}$

end for

Construct weighted combined model: $F(d) = \sum_{t=1}^{T} \alpha_t f_t(d)$

return Combined model $F(d)$

El conjunto de datos etiquetados de entrada D consta de n muestras y m características: $D = \{(d_1, c_1), (d_2, c_2), ..., (d_n, c_n)\}$, donde d_i es el vector de características de la

muestra i y c_i es la etiqueta correspondiente (podemos suponer que $c_i \in \{-1, +1\}$). Los parámetros que recibe el algoritmo se limitan al número de modelos débiles (generalmente árboles de decisión) que se utilizarán en el ensamblado, y que denotan como T en el pseudocódigo anterior.

Como se puede apreciar en el detalle del algoritmo, el proceso que se sigue para entrenar el modelo es el siguiente:

1) Inicializar los pesos de las muestras: $w_1 = \frac{1}{n}, w_2 = \frac{1}{n}, ..., w_n = \frac{1}{n}$.

2) Para $t = 1$ hasta T:

a) Entrenar un modelo débil $f_t(d)$ en el conjunto de datos D ponderado por los pesos $w_1, w_2, ..., w_n$.

b) Calcular el error ponderado del modelo $f_t(d)$: $err_t = \sum_{i=1}^{n} w_i \cdot \mathbb{1}(f_t(d_i) \neq c_i)$, donde $\mathbb{1}$(condición) es la función indicadora que devuelve 1 si la condición es verdadera y 0 en caso contrario.

c) Calcular el coeficiente de importancia del modelo $f_t(d)$: $\alpha_t = \frac{1}{2} \ln\left(\frac{1-err_t}{err_t}\right)$.

d) Actualizar los pesos de las muestras: $w_i = w_i \cdot \exp\left(-\alpha_t \cdot y_i \cdot f_t(d_i)\right)$ para normalizar las muestras correctamente.

e) Normalizar los pesos de las muestras: $w_1, w_2, \ldots, w_n = \frac{w_1}{\sum_{i=1}^{n} w_i}, \frac{w_2}{\sum_{i=1}^{n} w_i}, \ldots, \frac{w_n}{\sum_{i=1}^{n} w_i}$.

3) Construir el modelo combinado ponderado: $F(d) = \sum_{t=1}^{T} \alpha_t f_t(d)$

Entre las principales ventajas de este modelo encontramos algunas que ya hemos descrito en la sección anterior y que son compartidas con el modelo Ranfom Forest, como por ejemplo, la mejora del rendimiento respecto a los árboles de decisión simples, el manejo de características numéricas y/o categóricas, y la información relativa a la importancia de características o atributos, que es importante y puede ayudar a la interpretación del modelo.

Además, este modelo aporta otras características relevantes, como el uso de un número de parámetros muy reducido, lo que facilita la tarea de ajuste de hiperparámetros. Adicionalmente, este modelo se puede emplear con varios algoritmos base (*weak learners*) para adaptarse mejor a diferentes tipos de datos y problemas.

Entre sus principales inconvenientes, podemos destacar la sensibilidad del modelo a los datos con ruido y con valores atípicos (*outliers*) y al desequilibrio de clases, lo que puede llevar a una clasificación sesgada. Finalmente, destacar también que el proceso de ajustar los pesos y entrenar múltiples modelos puede ser computacionalmente costoso, especialmente con un gran número de modelos base.

3.3. Gradient Boosting

Lectura complementaria

J. Friedman, (2001). "Greedy Function Approximation: A Gradient Boosting Machine", en The Annals of Statistics, vol. 29(5), pp. 1189-1232.

El modelo Gradient Boosting fue propuesto por Friedman (2001) y consiste en crear varios predictores en secuencia, específicamente modelos aditivos, mediante la combinación de varios modelos base más simples. El primer predictor usa la media de la variable $a_{.,j}$ para predecir, luego el segundo predictor explica los errores del primer predictor, el tercer predictor explicar los errores del segundo predictor y así sucesivamente.

El proceso para el entrenamiento de un modelo Gradient Boosting se detalla en el Algoritmo 3.

Algoritmo 3 Pseudocódigo del algoritmo Gradient Boosting

Require: Training dataset $D = \{(d_1, c_1), (d_2, c_2), ..., (d_n, c_n)\}$

Require: Number of base learners T

Require: Learning rate η

Initialize predictions: $F_0(d) = 0$

for $t = 1$ to T **do**

Calculate residuals: $r_{ti} = c_i - F_{t-1}(d_i)$ for $i = 1, 2, ..., n$

Train a base learner $h_t(d)$ on the residuals r_{ti}

Find the coefficient α_t by optimizing a loss function, e.g., minimizing squared error: $\alpha_t = \arg\min_\alpha \sum_{i=1}^{n} (\alpha h_t(d_i) - r_{ti})^2$.

Update predictions: $F_t(d) = F_{t-1}(d) + \eta \alpha_t h_t(d)$

end for

return Combined model $F(d) = \sum_{t=1}^{T} \eta \alpha_t h_t(d)$

El conjunto de datos etiquetados de entrada D consta de n muestras y m características: $D = \{(d_1, c_1), (d_2, c_2), ..., (d_n, c_n)\}$, donde d_i es el vector de características de la muestra i y c_i es la etiqueta correspondiente. Los parámetros que recibe el algoritmo incluyen el número de modelos base que se utilizarán en el ensamblaje (T) y una colección de funciones base (por lo general, árboles de decisión débiles) que componen el ensamblaje ($\{h_t(d)\}$). La salida es un modelo combinado $F(d)$ que es la suma ponderada de los modelos base $h_t(d)$: $F(d) = \sum_{t=1}^{T} \alpha_t h_t(d)$.

Como se puede apreciar en el detalle del algoritmo, el proceso que se sigue para entrenar el modelo es el siguiente:

1) Inicializar las predicciones: $F_0(d) = 0$.

2) Para $t = 1$ hasta T:

a) Calcular los residuos: $r_{ti} = c_i - F_{t-1}(d_i)$ para $i = 1, 2, \ldots, n$.

b) Ajustar un modelo base $h_t(d)$ a los residuos r_{ti}.

c) Calcular el coeficiente de aprendizaje α_t mediante una técnica de búsqueda de línea, como mínimos cuadrados o descenso de gradiente, minimizando una función de pérdida entre las predicciones $F_{t-1}(d)$ y los residuos r_{ti}.

d) Actualizar el modelo combinado: $F_t(d) = F_{t-1}(d) + \alpha_t h_t(d)$.

3) Construir el modelo combinado final: $F(d) = \sum_{t=1}^{T} \alpha_t h_t(d)$.

Entre las principales ventajas o puntos fuertes de este modelo encontramos algunas que ya hemos descrito en las secciones anteriores, y que son compartidas con los modelos Ranfom Forest y AdaBoost. Entre ellas, destacar el aumento de precisión respecto a los modelos simples, posibilidad de manejar características numéricas y categóricas sin necesidad de un preprocesamiento exhaustivo, y posibilidad de obtener información relativa a la importancia de las características o atributos, lo que puede ayudar en la interpretación del modelo.

Por contra, algunas de sus debilidades son: el tiempo de entrenamiento, dado que el entrenamiento de múltiples modelos base puede ser computacionalmente costoso, especialmente en conjuntos de datos grandes y con modelos base complejos. También la cierta sensibilidad a valores atípicos y al ruido de los datos, y la necesaria configuración de los hiperparámetros, como el número de modelos base y la tasa de aprendizaje, para lograr un buen rendimiento.

Resumen

En este módulo didáctico nos hemos centrado en la combinación de clasificadores, también conocidos como modelos "ensembler". Estos modelos se basan en la composición en serie o paralelo de diferentes modelos más simple que permiten generar una respuesta conjunta para un determinado problema.

En la primera parte del texto hemos revisado los esquemas conceptuales, que básicamente incluyen la combinación paralela de clasificadores base similares, y la combinación secuencial de clasificadores base diferentes.

En la segunda parte del módulo hemos enumerado algunos de los principales modelos "ensembler", como pueden ser Random Forest, AdaBoost y Gradient Boosting.

Glosario

Ensembler Modelo construido combinando otros modelos , ya sea de forma secuencial (en serie) o en paralelo.

Procesamiento en paralelo: cuando diferentes modelos se crean a la vez (usando por ejemplo pequeñas variaciones del conjunto de datos de entrada) y los resultados son combinados para obtener un resultado (p.e. mediante una votación por mayoría simple).

Procesamiento en serie: cuando el resultado de un modelo se utiliza como entrada del modelo siguiente.

Sobreajuste (*overfitting*): el sobreajuste se produce cuando el algoritmo usado para crear un modelo captura en exceso la naturaleza intrínseca del conjunto de datos usado en el proceso de creación, imposibilitando que el modelo generalice bien para datos nunca observados con anterioridad, creando un sesgo.

Bibliografía

Géron, Aurélien (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Editio*n. O'Reilly Media, Inc.

Cichosz, Pawel (2015). *Data mining algorithms. Explained using R.* Wiley.

J. Gironés Roig, J. Casas Roma, J. Minguillón Alfonso, R. Caihuelas Quiles (2017). *Minería de datos: Modelos y algoritmos.* Barcelona: Editorial UOC.