Simple Random Forest Algorithm

Tabla de contenido

Raimundo Flores Cabra
dpto. Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla
Sevilla, España
Correo electrónico UVUS:
raiflocab@alum.us.es
Correo electrónico de contacto:
raimundokarate98@gmail.com

Vicente López Vázquez
dpto. Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla
Sevilla, España
Correo electrónico UVUS:
viclopvaz1@alum.us.es
Correo electrónico de contacto:
vlopezvazquez3@gmail.com

El objetivo de nuestro trabajo ha sido construir una versión simplificada del algoritmo de Bosques Aleatorios. Pero ¿en qué consiste este algoritmo? Un bosque aleatorio es un conjunto de árboles de decisión entrenados a partir de un conjunto de datos de entrenamiento con un conjunto aleatorio de atributos seleccionados para cada uno y tras la aplicación de Bootstrapping como técnica de muestreo. Esto nos permite construir un conjunto de árboles de decisión partiendo de un solo conjunto de datos de entrenamiento, ya que dichos datos son difíciles de conseguir y no muy abundantes. Por esta razón, se aplican técnicas de muestreo, en este caso Bootstrapping con reemplazo, para dar lugar a un nuevo conjunto de datos para cada árbol de decisión.

Esta técnica facilita el desarrollo de sistemas de predicción fiables y con mayor capacidad de predicción que un solo árbol de decisión. Debemos destacar que, a lo largo de este proyecto hemos comprendido que los datos con los que trabajamos no siempre son suficientes y tenemos que adaptarnos a ellos; que las técnicas de muestreo nos facilitan el trabajo con conjuntos de datos pequeños y con la construcción de varios árboles de decisión; y que un solo árbol no es tan eficaz como un conjunto promediado de ellos.

1. Introducción

La Inteligencia Artificial [1] tiene numerosos usos actualmente, desde sistemas de planificación automática o sistemas de decisión, a reconocimiento de la escritura o del habla. Nuestro caso trata sobre los árboles de decisión [2], un ejemplo de Aprendizaje Supervisado [3].

Un árbol de decisión es un modelo de predicción construido a partir de un conjunto de datos, de donde se extraen una serie de reglas, que sirven para representar y categorizar una serie de condiciones que ocurren de manera sucesiva, para la resolución del problema. En la siguiente figura podemos ver la estructura de un árbol de decisión:

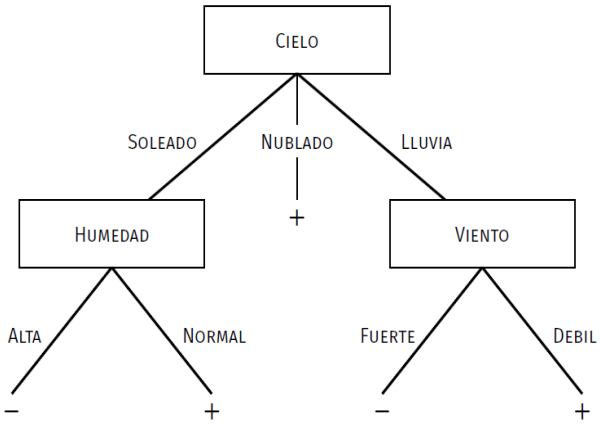

Figura 1 Árbol de decisión con 3 nodos de decisión.

Podemos ver que el árbol clasificará los distintos ejemplos teniendo en cuenta como primera forma de clasificación el atributo "Cielo". Dependiendo del valor de dicho atributo, el siguiente atributo usado para clasificar el ejemplo será "Humedad" o "Viento" o en caso de que el valor de "Cielo" sea nublado la clasificación es directa.

A grandes rasgos, nuestro proyecto trabaja con dichos árboles de decisión. Pero se complica al introducir técnicas de muestreo como Bootstrapping [4] y Random Subspace Method [5]. Además, trabajaremos con varios árboles con el objetivo de aumentar la precisión calculando la salida promedia. La aplicación de técnicas de muestreo para obtener nuevos conjuntos de datos a partir de uno y el uso de varios árboles entrenados con diferentes conjuntos de datos da lugar a una técnica llamada Random Forest (Bosque Aleatorio) [6].

El uso de Random Forest aumenta la precisión que el uso de un solo árbol de decisión proporciona y facilita el trabajo con conjuntos de datos pequeños o difíciles de conseguir. Por eso su uso está muy extendido y aceptado.

Como conclusión, nuestro proyecto ha consistido en diseñar y construir una forma simple del algoritmo del Random Forest, compararlo con el uso de árboles individuales, profundizar en el uso de las técnicas de muestreo arriba citadas y experimentar con distintos parámetros para llegar a los mejores valores de predicción posibles.

2. Preliminares

Antes de entrar de lleno en el trabajo, pasaremos a numerar y explicar en que consisten las técnicas que hemos usado durante el desarrollo del proyecto. Primero hablaremos de la técnica básica en la que se basa nuestro trabajo, para después entrar en las técnicas de muestreo utilizadas y finalmente en la técnica que se crea como conjunto de las anteriores:

- **Técnicas básicas:**
    - **Aprendizaje supervisado [3]:** El aprendizaje supervisado es una técnica usada para clasificar datos o ejemplos a partir de un conjunto de datos de entrenamiento. Los conjuntos de entrenamiento son vectores de datos donde cada dato está formado por unos atributos y por una clasificación o valor numérico.

      El objetivo del aprendizaje supervisado es el de dar lugar a una función de decisión que pueda responder correctamente con un margen de error a unos datos de entrada según el entrenamiento dado. Los árboles de decisión son un ejemplo de aprendizaje supervisado.
    - **Árboles de decisión [2]:** Un árbol de decisión es una técnica de aprendizaje automático, y como hemos podido ver arriba, forma parte del aprendizaje supervisado ya que recibe un conjunto de datos de entrenamiento con el fin de entrenar al árbol para que pueda responder con una probabilidad alta de acierto frente a nuevos datos de entrada. Esta técnica es en la que se basa nuestro trabajo, pero la complicamos al aplicar distintas técnicas de muestreo al conjunto de datos de entrenamiento.
- **Técnicas de muestreo:**
    - **Bootstrapping [4]:** Es una técnica de muestreo usada cuando tenemos un conjunto de datos limitado y no podemos obtener más. Dado un conjunto de datos de N filas, crearemos un nuevo conjunto de datos del mismo tamaño a partir de este.

      Dicho nuevo conjunto se crea recorriendo el conjunto de datos base; haciendo una selección con probabilidad de determinados ejemplos durante el recorrido; dichos ejemplos seleccionados sustituirlos por una copia de otro ejemplo cualquiera seleccionado con probabilidad. Así hasta generar un nuevo conjunto de datos de tamaño N o de un tamaño dado por el usuario.

      Es una técnica muy útil si, como hemos dicho antes, no contamos con un conjunto de datos abundante y tenemos necesidad de más datos que los que podemos conseguir.

- **Random Subspace Method [5]:** Otra técnica de muestreo usada en el ensamble de modelos. Se basa en la selección aleatoria de determinadas características del conjunto de datos de entrenamiento. El objetivo de esta técnica es que cada modelo de predicción se entrene con un conjunto de características concretas en vez de con el conjunto entero de estas, es decir, que cada modelo entrenado se especialice en un conjunto más pequeño de características para aumentar su precisión finalmente ensamblando todos los modelos y promediando la respuesta.

- **Técnica de ensamble:**
  - **Random Forest (Simplified) [6]:-** Es la técnica de ensamble que hemos usado. Hemos desarrollado una versión simplificada de Random Forest o Bosque Aleatorio traducido.

    Consiste en: partiendo de un conjunto de datos de entrenamiento y un número de árboles a entrenar, para cada árbol se aplican las técnicas de muestreo arriba descritas al conjunto de datos y se entrena el árbol. Tras entrenarlos todo, esta técnica devuelve un conjunto de árboles (Bosque) entrenados a partir de un mismo conjunto de datos, pero cada uno especializado en un conjunto de características (Aleatorio). Para las predicciones se promediarían las salidas del conjunto de árboles. Esta técnica proporciona una precisión mayor que un único árbol de decisión.

3. Metodología

En esta sección de la documentación se mostrarán y realizarán una explicación de todas las funciones realizadas en el trabajo. Primero enseñaremos la entrada y salida de la función, para continuar después con una explicación de la misma y para finalizar la utilidad en nuestro algoritmo.

- ***load_and_process_data***

  Entrada:

  - El nombre de los ficheros de entrenamiento y de prueba.

Salida:

- Dos **DataFrame.**

Algoritmo:

1. Usamos pandas para leer los archivos **csv** que contienen los conjuntos de datos, de entrenamiento y de prueba, y asignamos la lectura de dichos archivos a variables.
2. Se crean dos **DataFrame** vacios que almacenarán los datos de los archivos una vez se codifiquen.
3. Usaremos un **LabelEncoder** para cada columna a codificar. Usaremos la unión de los datos de los dos conjuntos al hacer **fit** al **LabelEncoder**. Tras esto, si la columna representa una variable categórica, será codificada. Si no, se queda como está y no se crea un **LabelEncoder** para la columna en cuestión.
4. Una vez acabado este bucle, se devuelven los **DataFrame** que tendrán los valores codificados.

Utilidad:

- Este algoritmo es usado para leer los archivos de los ficheros y poder codificar sus datos, ya que sino sería imposible trabajar con ellos.

- ***bootstrapping***

  Entrada:

  - **DataFrame** con el coinjunto de datos de entrenamiento.

  Salida:

  - **DataFrame** tras aplicar la técnica de muestreo **Bootstrapping**.

Algoritmo:

1. Se crearán índices aleatorios por medio de **np.random.randint**. La lista de índices será del tamaño del número de filas que tenga el conjunto de datos de entrada (El **DataFrame** de entrada).
2. Dicha lista de índices será usada para seleccionar valores del **DataFrame** de entrada y asignarlos al **DataFrame** de salida.

Utilidad:

- El algoritmo es una implementación de la técnica de muestreo conocida como **Bootstrapping**. Genera índices aleatorios para dar lugar a un nuevo conjunto de datos basado en el de entrada.

- ***divide_data***

  Entrada:

  - **DataFrame**.

  Salida:

  - Dos **DataFrame**: el conjunto de ejemplos y el conjunto de las clasificaciones de los ejemplos.

  Algoritmo:

  1. Se creará un **DataFrame** que contendrá todo el contenido del conjunto de datos de entrada excepto la última columna, que es la que contiene las **labels** o clasificaciones de los ejemplos.
  2. Se creará un segundo **DataFrame** que contendrá la clasificación de los ejemplos del conjunto de entrada.

Utilidad:

- Este algoritmo es usado para realizar la división del conjunto de entrenamiento y poder obtener todas las columnas con sus datos sin la variable de respuesta y, por otro lado, tener otro conjunto de datos únicamente con la variable respuesta o clasificación.

- ***random_subspace***

Entrada:

- Un **DataFrame** y un número real entre 0 y 1 (Representa un porcentaje).

Salida:

- Una lista de índices.

Algoritmo:

1. Se almacena el número de columnas y sus índices en dos variables distintas.
2. Se calcula el número de columnas que se seleccionarán, multiplicando el número real de entrada por el número de columnas.
3. Si el valor del real de entrada no está entre 0 y 1, se imprime por pantalla que debe estar entre ese rango.
4. Si se encentra en el rango de 0 a 1, se crearán un número de índices aleatorios igual al número de columnas que se van a seleccionar.
5. Dicha lista de índices se ordena según el orden natural.

Utilidad:

- El algoritmo implementa la técnica de muestreo **Random Subspace Method**.

- ***select_columns***

  Entrada:

  - Un **DataFrame** y una lista de índices.

  Salida:

  - **DataFrame** con las columnas seleccionadas.

  Algoritmo:

  1. Crea un diccionario que almacenará las columnas elegidas
  2. Mientras no se acaben de recorrer todos los índices de las columnas que se le pasan como entrada: Se guarda el valor de la columna que indique el índice del bucle y este valor se introduce en el diccionario del paso 1 en la posición que diga el índice el bucle.
  3. Por último, transforma el diccionario en un **DataFrame** y lo devuelve como salida.

  Utilidad:

  - El algoritmo implementa la selección de columnas a partir de una lista de índices y un conjunto de datos. Da lugar a un nuevo conjunto de datos solo con los datos correspondientes a las columnas seleccionadas.

- ***Meta-algorithm***

  Entrada:

  - Nombre del Archivo, de entrenamiento y de prueba, y tres enteros que representan, el número de árboles a entrenar, el número de características a tener en cuenta y la profundidad máxima de los árboles a entrenar.

  Salida:

  - Información por pantalla correspondiente a la Tasa de aciertos y a la Tasa de aciertos balanceada.

Algoritmo:

1. Usamos la función **load_and_process_data** para cargar y codificar los datos de los conjuntos de entrenamiento y de prueba.
2. Se crean variables para representar los bosques y los índices elegidos. Además, se crea una instancia del objeto **DecissionTreeClassifier**.
3. Se realiza un bucle **for** entre 0 y el número de árboles a entrenar.
    - **i.** Se dividen los datos del conjunto de entrenamiento en datos y en clasificación usando la función **divide_data**.
    - **ii.** Usamos la función **random_subspace** para ejecutar la técnica de muestreo y con la salida se ejecutamos la función **select_columns** para quedarnos solo con los datos de las columnas dadas.
    - **iii.** Con el conjunto de entrenamiento procesado y tras aplicarle las técnicas de muestreo pertinentes, entrenamos al árbol usando la función **fit** de **DecissionTreeClassifier**, incluyendo como parámetro la profundidad máxima que alcanzará el árbol.
    - **iv.** Introducimos en una lista la lista de índices antes creada para su posterior uso en las predicciones. A parte, introducimos el árbol ya entrenado en una lista.
4. Una vez finalizado el bucle se realiza de nuevo la función ***divide_data*** para los valores del conjunto de prueba. Entonces tenemos el conjunto de prueba dividido en datos y clasificación.
5. Se crea un diccionario que contendrá las predicciones y se realiza un bucle desde 0 hasta el tamaño del bosque.
    i. Se guarda el nombre del árbol y se llama a la función ***select_colums*** pasándole como parámetros los datos del conjunto de prueba y las columnas usadas para entrenar al árbol **i.**
    ii. Con el conjunto de datos de prueba ya dividido y con las columnas correspondientes a cada árbol, se ejecuta la función **predict** de **DecissionTreeClassifier** y las predicciones se guardan en el diccionario antes creado con llave **tree[i]** donde **i** es el número del árbol.

6. Al finalizar el bucle, transformamos el diccionario en un **DataFrame** y sacamos la moda de los resultados por fila, con el objetivo de promediar las predicciones.
7. Usando las funciones ***accuracy_score*** y ***balanced_accuracy_score***, pasándoles como parámetros a ambos las columnas de respuestas del conjunto de prueba y las predicciones promediadas del paso anterior, se calcula la Tasa de aciertos y la Tasa de aciertos balanceados.
8. Para finalizar, se ejecutan varios **print** con el objetivo de mostrar: el nombre de los archivos que contienen los conjuntos de entrenamiento y de prueba, el número de árboles a entrenar, el porcentaje de columnas a seleccionar, la profundidad máxima que pueden alcanzar los árboles, y finalmente las dos tasas de aciertos, la Tasa de aciertos y la Tasa de aciertos balanceada.

Utilidad:

- Esta es la implementación completa del meta-algoritmo pedido. Proporciona toda la información posible a partir de los parámetros introducidos.

4. Resultados

Podemos concluir que los resultados obtenidos son, tras muchos esfuerzos durante la implementación de todas las funciones necesarias y del meta-algoritmo completo, mejores que los resultados obtenidos mediante el uso de un solo árbol de decisión sin la aplicación de técnicas de muestreo. Los resultados se mostrarán en un NoteBook aparte llamado NoteBook de Resultados.

5. Conclusión

Como conclusión, hemos aprendido la utilidad de técnicas como los árboles de decisión o las técnicas de muestreo. Las técnicas de muestreo facilitan mucho la tarea cuando tenemos que trabajar con conjuntos de datos limitados. Además, el entrenamiento de varios árboles de decisión, junto a la aplicación de dichas técnicas de muestreo, conllevan una mejoría notable respecto al uso del árbol de decisión simple.

Dejamos aquí referencias a las páginas de consulta que hemos usado:

Información general sobre los árboles de decisión [7]. Información sobre el objeto **DecissionTreeClassifier** [8]. Información general sobre las clases en **Python** [9]. Información general sobre **Python** [10]. Documentación sobre el módulo **Pandas** [11]. Información sobre la técnica de muestreo **Bootstrapping** [12]. Información sobre el método **resample** [13]. Información sobre cómo definir una función en **Python** [14]. Información sobre como usar los selectores de índices [15].

Referencias

[1] https://es.wikipedia.org/wiki/Inteligencia_artificial

[2] https://es.wikipedia.org/wiki/%C3%81rbol_de_decisi%C3%B3n

[3] https://es.wikipedia.org/wiki/Aprendizaje_supervisado

[4] https://es.wikipedia.org/wiki/Bootstrapping_(estad%C3%ADstica)

[5] https://en.wikipedia.org/wiki/Random_subspace_method

[6] https://es.wikipedia.org/wiki/Random_forest

[7] https://scikit-learn.org/stable/modules/tree.html

[8] https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier

[9] https://www.w3schools.com/python/python_classes.asp

[10] https://docs.python.org/3.6/tutorial/

[11] http://pandas.pydata.org/pandas-docs/stable/getting_started/basics.html

[12] https://machinelearningmastery.com/a-gentle-introduction-to-the-bootstrap method/

[13] https://scikit-learn.org/stable/modules/generated/sklearn.utils.resample.html

[14] https://www.tutorialspoint.com/python/python_functions.htm

[15] https://www.shanelynn.ie/select-pandas-dataframe-rows-and-columns-using-iloc-loc-and-ix/

Figuras