Iteración eficiente sobre filas en Pandas

Ignacio Pedrero Muro

October 30, 2023

1 Introduction

Reflexiona sobre tus experiencias pasadas con el aprendizaje automático y el uso de Pandas. ¿Has creado funciones combinando columnas en tus conjuntos de datos? ¿Has enfrentado problemas de rendimiento al procesar grandes volúmenes de datos? Reconoce que a menudo, las técnicas utilizadas en las etapas iniciales del aprendizaje pueden no ser las más eficientes. Específicamente, considera las formas en las que has iterado sobre las filas en Pandas.

Investiga y compara las diferentes técnicas para iterar filas en un DataFrame de Pandas. Encuentra la manera más eficiente y entiende lo que sucede detrás de escena. Importa un conjunto de datos con el que hayas trabajado anteriormente en Pandas. Si no tienes uno propio, puedes usar un conjunto de datos de ejemplo.

Aplica la técnica optimizada que has aprendido para iterar sobre las filas del conjunto de datos. Observa las diferencias en rendimiento y comprensión en comparación con tus métodos anteriores. Reflexiona sobre tus hallazgos. ¿Cómo ha cambiado tu perspectiva sobre la iteración en Pandas? ¿Qué aprendizajes consideras más valiosos para tu futuro como científico de datos?

2 Reflexiona sobre tus experiencias pasadas con el aprendizaje automático y el uso de Pandas

2.1 ¿Has creado funciones combinando columnas en tus conjuntos de datos?

Si, en anteriores tareas como en el ironhack estube trabajando con pandas y con bases de datos grandes para poder sacar un modelo de prediccion y correlacionar columnas con el fin de reducir lo maximo nuestra base de datos para poder iterar sobre ella y obtener buenos resultados intentando maximizar la eficiencia.

2.2 ¿Has enfrentado problemas de rendimiento al procesar grandes volúmenes de datos?

Si, el ordenador tardaba en procesar algunas funciones y tardaba en devolverme graficas, ya que tenia que comparar todas las columnas, unas con otras para ver

hasta que punto tenian relacion unas con otras

3 Reconoce que a menudo, las técnicas utilizadas en las etapas iniciales del aprendizaje pueden no ser las más eficientes. Específicamente, considera las formas en las que has iterado sobre las filas en Pandas.

La manera que empleé para iterar con estas filas en pandas fueron el metodo for, el metodo replace y el metodo drop, que evidentemente no son las mas efectivas, esto se debe a que eran las unicas maneras que conocia para iterar sobre la base de datos y desconocia bastante sobre metodos de mejora de rendimiento de los programas y no estaba tan interesado en mejorar el rendimiento como en obtener un resultado del modelo independientemente de lo que fuera a tardar, incluso sin llegar a analizar el tamaño de la base de datos y lo que supondría en el futuro trabajar con una base de datos mucho mas grande.

4 Investiga y compara las diferentes técnicas para iterar filas en un DataFrame de Pandas. Encuentra la manera más eficiente y entiende lo que sucede detrás de escena.

Despues de leer la tarea he conseguido entender lo que hay detras de cada manera de iterar sobre las filas y columnas en pandas con grandes bases de datos. Esto es un resumen de lo que hace cada una y de como escribir el codigo para realizarlas: Iterrows: El método iterrows() en pandas es una forma de iterar a través de un DataFrame fila por fila. Sin embargo, se debe usar con precaución, ya que no es la forma más eficiente de iterar a través de un DataFrame, especialmente si el DataFrame es grande. En su lugar, se recomienda usar métodos vectorizados siempre que sea posible, ya que son más eficientes en términos de rendimiento.

1. El método iterrows() devuelve un iterador que produce pares de índice de fila y Series que representan cada fila del DataFrame. Algunas cosas a tener en cuenta sobre iterrows():

Es más lento que otras formas de iterar a través de un DataFrame debido a la creación de objetos Series en cada iteración.

Devuelve una vista de cada fila como una Serie, por lo que puede acceder a los valores de la fila mediante la notación de etiqueta, por ejemplo, row['A'], row['B'].

Puede ser útil cuando necesitas realizar operaciones más complejas en cada fila, pero generalmente no se recomienda para tareas sencillas, ya que existen métodos más eficientes como apply() y las operaciones vectorizadas de pandas.

En resumen, iterrows() es una opción cuando necesitas iterar fila por fila en un DataFrame, pero debes ser consciente de su potencial ineficiencia en comparación con otras técnicas de pandas. Si estás realizando operaciones simples en todo el DataFrame, es preferible evitar iterrows() y utilizar métodos vectorizados para mejorar el rendimiento.

2. Bucle for con loc o iloc:
Al igual que con el método anterior, las filas se convierten en objetos de la serie Pandas, lo que degrada el rendimiento.

Curiosamente, .iloc es más rápido que .loc. Tiene sentido ya que Python no tiene que verificar las etiquetas definidas por el usuario y mirar directamente dónde está almacenada la fila en la memoria.

En este caso, utilizamos un bucle for para iterar a través de las etiquetas de las columnas del DataFrame y accedemos a cada columna utilizando la notación de etiqueta.

3. Apply:
El apply() método es un bucle for disfrazado, por lo que el rendimiento no mejora tanto. El método apply es una función de pandas que te permite aplicar una función específica a lo largo de un eje (filas o columnas) de un DataFrame o una Serie. Puedes usarlo para realizar operaciones personalizadas en tus datos de manera eficiente y vectorizada pero no es mucho mas rapido que el primer metodo. El método apply puede utilizarse para realizar operaciones más complejas que solo aplicar una función matemática. Puedes usar funciones lambda o funciones personalizadas para realizar cualquier tipo de transformación en tus datos.

4. Itertuplica:
Según la documentación oficial , itera "sobre las filas de un DataFrame como tuplas nombradas de los valores". En la práctica, significa que las filas se convierten en tuplas , que son objetos mucho más livianos que Pandas Series.

Cada fila se devuelve como un objeto NamedTuple con atributos correspondientes a los nombres de las columnas. Puedes acceder a los valores de las columnas utilizando la notación de punto, por ejemplo, row.A para acceder al valor de la columna 'A' en la fila actual.

itertuples() es más eficiente que iterrows() cuando necesitas iterar a través de un DataFrame fila por fila, ya que no crea objetos Series en cada iteración. Por lo tanto, es una buena opción cuando deseas un rendimiento óptimo en este tipo de operaciones.

5. Compresión de listas:
Las comprensiones de lista son una forma elegante de iterar sobre una lista como

una sola línea.

Por ejemplo, [print(i) for i in range(10)] imprime números del 0 al 9 sin ningún bucle for explícito . Digo "explícito" porque Python en realidad lo procesa como un ciclo for si miramos el código de bytes.

Entonces, ¿por qué es más rápido? Sencillamente porque no llamamos al .append()método en esta versión. Sin embargo, ten en cuenta que las comprensiones de listas son adecuadas para operaciones simples y pueden no ser la opción más eficiente cuando se realizan operaciones más complejas o cuando se necesita modificar el DataFrame original. Para operaciones más complejas, puede ser más adecuado utilizar el método apply() u otras funciones de pandas que permiten un mayor control y eficiencia en el manejo de los datos.

6. Vectorizacion de pandas:
Hasta ahora, todas las técnicas utilizadas simplemente suman valores únicos. En lugar de agregar valores únicos, ¿por qué no agruparlos en vectores para resumirlos? La diferencia entre sumar dos números o dos vectores no es significativa para una CPU , lo que debería acelerar las cosas.

Además de eso, Pandas puede procesar objetos de Series en paralelo , ¡utilizando todos los núcleos de CPU disponibles!

La sintaxis también es la más simple imaginable: esta solución es extremadamente intuitiva. Debajo del capó, Pandas se encarga de vectorizar nuestros datos con un código C optimizado usando bloques de memoria contiguos.

La vectorización en pandas se refiere a la capacidad de realizar operaciones en un conjunto de datos (por ejemplo, una Serie o un DataFrame) de manera eficiente, sin necesidad de recorrer los elementos uno por uno en un bucle. En lugar de utilizar bucles for, la vectorización aprovecha las capacidades internas de optimización de pandas y las bibliotecas subyacentes, como NumPy, para realizar operaciones en bloques de datos, lo que conduce a un mejor rendimiento.

La vectorización en pandas se logra principalmente a través de operadores y funciones que actúan sobre objetos Series o DataFrames en su totalidad, en lugar de elementos individuales. Algunos ejemplos de operaciones vectorizadas en pandas incluyen:

Operaciones matemáticas: Puedes realizar operaciones matemáticas en una Serie o un DataFrame directamente.

Funciones estadísticas: Pandas proporciona una variedad de funciones estadísticas que se aplican a columnas enteras, como mean(), sum(), min(), max(), etc.

Funciones de manipulación de datos: Puedes aplicar funciones de manipulación de datos a través de métodos como apply(), que permite aplicar una función personalizada a una columna o fila específica.

Operaciones de filtro y selección: Puedes filtrar filas o seleccionar subconjuntos de datos de acuerdo con condiciones específicas.

La vectorización es esencial en pandas, ya que mejora significativamente el rendimiento al trabajar con grandes conjuntos de datos. Utiliza las funciones vectorizadas proporcionadas por pandas siempre que sea posible en lugar de bucles for, ya que los bucles pueden ser mucho más lentos y menos eficientes.

Es importante tener en cuenta que pandas se basa en la biblioteca NumPy, que es altamente optimizada para operaciones vectorizadas. Por lo tanto, cuando trabajas con pandas, estás aprovechando la eficiencia de NumPy detrás de escena para mejorar el rendimiento.

7. Vectorización NumPy:
NumPy está diseñado para manejar computación científica. Tiene menos gastos generales que los métodos de Pandas, ya que las filas y los marcos de datos se convierten en archivos np.array. Se basa en las mismas optimizaciones que la vectorización de Pandas.

La vectorización en NumPy se refiere al uso de operaciones de matriz en lugar de bucles explícitos para realizar cálculos en arrays (vectores y matrices) de manera eficiente. NumPy está diseñado para trabajar con arrays multidimensionales de datos y proporciona una amplia variedad de funciones y operadores que se pueden aplicar directamente a estos arrays, lo que permite realizar cálculos de manera más rápida y eficiente en comparación con los bucles for. Las ventajas de la vectorización en NumPy incluyen:

Eficiencia: NumPy está altamente optimizado y está escrito en C, lo que significa que las operaciones en arrays se realizan de manera eficiente.

Claridad del código: El código vectorizado suele ser más claro y conciso que el código que utiliza bucles explícitos, lo que facilita la lectura y el mantenimiento del código.

Facilidad de uso: NumPy proporciona una amplia variedad de funciones y operadores que son fáciles de usar para realizar operaciones matemáticas y estadísticas en arrays.

La vectorización es una parte fundamental de NumPy y es una de las razones por las que NumPy es ampliamente utilizado en la comunidad científica y de análisis de datos. Permite realizar operaciones complejas en grandes conjuntos de datos de manera eficiente y con un código más legible.

5 Reflexiona sobre tus hallazgos. ¿Cómo ha cambiado tu perspectiva sobre la iteración en Pandas? ¿Qué aprendizajes consideras más valiosos para tu futuro como científico de datos?

Ha cambiado mi perspectiva el saber que existen muchas maneras de pedirle ciertas cosas al ordenador y que es importante conocerlas todas para saber en que momento usar cada una y asi mejorar el rendimiento de los programas en función de las necesidades del programa. La capacidad de escalar el trabajo a conjuntos de datos más grandes y complejos es esencial en la ciencia de datos. La iteración eficiente a través de operaciones vectorizadas y la comprensión de las capacidades de pandas permiten a los científicos de datos manejar datos más grandes sin sacrificar el rendimiento. En resumen, cambiar la perspectiva

sobre la iteración en pandas implica reconocer la importancia de la eficiencia, la claridad del código y la utilización de las herramientas disponibles. Los aprendizajes obtenidos en este sentido son fundamentales para el éxito y la efectividad de un científico de datos en el manejo y análisis de datos en la actualidad.