

Selección de Características en GPUs para Inferir Redes Genéticas de Forma Energéticamente Eficiente

Jorge González-Domínguez, Javier Darriba

Universidade da Coruña
jgonzalezd@udc.es, javier.darriba@udc.es

Abstract

La inferencia de redes genéticas permite comprender procesos celulares complejos. MRNET es una herramienta ampliamente utilizada para inferir dichas redes, pero su coste computacional limita su escalabilidad y eficiencia energética. Este trabajo presenta una versión paralela de MRNET optimizada para GPUs. Gracias al uso de estas arquitecturas de memoria compartida y a la librería Parallel-FST, el algoritmo resultante no solo es más rápido, sino también más eficiente energéticamente, lo que lo hace adecuado para entornos de computación sostenible.

1. Introducción

Históricamente, diversas técnicas computacionales han sido desarrolladas para abordar la inferencia de redes de regulación genética, las cuales son esenciales para comprender las interacciones entre genes que definen numerosos procesos biológicos y patológicos. Una de las más destacadas es el método MRNET [4] (Minimum Redundancy Networks), basado en el algoritmo de selección de características mRMR [3] (Maximum Relevance Minimum Redundancy), que tiene como objetivo minimizar la redundancia entre las características seleccionadas mientras maximiza su relevancia para las características objetivo, proporcionando así modelos más precisos y fácilmente interpretables. En este escenario, mRMR consigue trabajar sobre conjuntos biológicos interpretando los genes como características. Sin embargo, MRNET se enfrenta a limitaciones notables en cuanto a eficiencia computacional. La complejidad cúbica de este método en relación con el número de genes implica que su aplicación en estudios a gran escala puede ser computacionalmente prohibitiva.

El presente trabajo estudió el uso de implementaciones paralelas de mRMR disponibles en la librería Parallel-FST [2] para desarrollar versiones optimizadas de MRNET. Concretamente, nos centramos en las versiones para GPUs, que permiten no solo acelerar la selección de características sino hacerlo a un menor coste energético.

2. Implementación paralela

Para llevar a cabo la llamada de la función que implementa el algoritmo mRMR en CUDA de la biblioteca Parallel-FST es necesario proporcionar cuatro parámetros fundamentales. El primer parámetro es el número de características a seleccionar, que en este caso, debido al método MRNET, corresponde a todas las características disponibles. El segundo es una instancia de la clase *Dataset*, que representa la matriz de características. Es importante destacar que la última columna de esta matriz se considera como la clase objetivo. El tercer parámetro es un vector donde se almacenan los resultados, compuesto por pares que indican el índice de

la característica y su respectiva puntuación. Finalmente, se requiere una instancia de la clase *Config*, la cual permite ajustar varios parámetros como la precisión del resultado, el número de streams, el número de bloques en la GPU y el número de hilos por bloque, el uso o no de memoria compartida en la GPU, la activación o desactivación del perfilado en la ejecución y la GPU a utilizar mediante un índice. Estos parámetros pueden tener una gran influencia en el rendimiento [1].

Teniendo esto en cuenta la implementación de MRNET con CUDA consiste en un bucle principal realizando iteraciones sobre cada característica, considerándola como clase objetivo de forma secuencial. Este procedimiento se inicia desde la última característica hasta la primera, intercambiando la columna de la clase objetivo actual por la de la característica seleccionada. En cada iteración, se actualiza la matriz de características para que la nueva clase objetivo sea efectivamente la característica actual, se ejecuta el algoritmo mMRM, se ordenan los resultados por el índice de característica y se escribe el resultado en el fichero de salida.

3. Evaluación experimental

Se ha comparado la velocidad de análisis de la herramienta MRNET original con la nueva versión basada en Parallel-FST ejecutada sobre dos tipos de GPUs de NVIDIA: por un lado una GPU NVIDIA T4¹, basada en la arquitectura Turing, y por otro una GPU NVIDIA A100², con arquitectura Ampere.

Se ha empleado un conjunto de datos genómico real (GDS1083)³ formado por 1124 genes y 108 muestras. La versión secuencial de MRNET, requirió en una máquina con un Intel Xeon Ice Lake 23 horas 56 minutos para completar su ejecución. Cabe destacar que este tiempo se debe principalmente al cálculo en el algoritmo mRMR, ya que sumando el resto de las partes del programa, como la lectura del fichero de entrada (28 ms) o la discretización de los datos (6 ms), apenas llegan a sumar un par de segundos.

En relación con la implementación en CUDA, se ajustó la configuración en el archivo *Config.h* para evaluar diferentes números de streams, precisiones, número de bloques y de hilos por bloque. Los resultados mostraron que la aceleración respecto a la ejecución secuencial, en el mejor de los casos, es de 26.5 usando la GPU T4 y de 33.4 con la GPU A100. Esto indica que la arquitectura Ampere, más moderna y con capacidades superiores, es algo más rápida que la arquitectura Turing.

References

- [1] Bieito Beceiro, Jorge González-Domínguez, Laura Morán-Fernández, Verónica Bolón-Canedo, and Juan Touriño. CUDA Acceleration of MI-based Feature Selection Methods. *Journal of Parallel and Distributed Computing*, 190:104901, 2024.
- [2] Bieito Beceiro, Jorge González-Domínguez, and Juan Touriño. Parallel-FST: A Feature Selection Library for Multicore Clusters. *Journal of Parallel and Distributed Computing*, 169:106–116, 2022.
- [3] Chris Ding and Hanchuan Peng. Minimum Redundancy Feature Selection from Microarray Gene Expression Data. *Journal of Bioinformatics and Computational Biology*, 3(02):185–205, 2005.
- [4] Patrick E Meyer, Kevin Kontos, Frederic Lafitte, and Gianluca Bontempi. Information-Theoretic Inference of Large Transcriptional Regulatory Networks. *EURASIP Journal on Bioinformatics and Systems Biology*, 2007(1):79879, 2007.

¹<https://www.nvidia.com/es-es/data-center/tesla-t4/>

²<https://www.nvidia.com/es-es/data-center/a100/>

³<https://www.ncbi.nlm.nih.gov/sites/GDSbrowser?acc=GDS1083>