Grid Search

Daphne S. González C.
dgonzalezc2104@alumno.ipn.mx

Edkir U. Nava M.
enavam2001@alumno.ipn.mx

1 Introducción

El Grid Search es una técnica para optimizar hiperparámetros evaluando combinaciones predefinidas con validación cruzada. Aunque eficaz, es computacionalmente costoso con muchas combinaciones. En esta práctica, se paraleliza el proceso para acelerar la búsqueda de hiperparámetros en modelos Random Forest y KNeighborsClassifier.

1.1 Descripción de los datos

El conjunto de datos empleado es el ***Vehicle Loan Database*** y contiene información clave de solicitantes de préstamos en la India, como edad, género, ingresos, puntaje crediticio, número de préstamos activos, historial crediticio, estado de residencia, ocupación, empleo y la relación préstamo-valor (LTV), con el objetivo de evaluar la solvencia crediticia mediante una puntuación de 0 a 100 (mayor puntuación, mejor perfil).

Inicialmente, el dataset contenía 279,857 registros, que se redujeron a 248,538 tras limpieza y preprocesamiento, eliminando valores nulos y fuera de rango. Todos los campos fueron convertidos a numéricos y la puntuación del perfil se transformó en binaria: 1 para puntuaciones de 80 o más, y 0 en caso contrario.

1.2 Descripción de las pruebas realizadas

Para las pruebas, se seleccionaron exclusivamente las primeras 100,000 muestras del conjunto de datos limpios y preprocesados. Cada proceso se ejecutó 10 veces con el fin de calcular métricas como la media y la desviación estándar. Las especificaciones de los equipos en los que se realizaron las pruebas se detallan en las tablas **1** y **2**.

Características	Equipo 1
Procesador	IntelCore i7-1165G7
RAM	16 GB
Frecuencia de Reloj	2.80 GHz
Núcleos Físicos	4
Núcleos Lógicos	8

Table 1: Características del Equipo utilizado para las pruebas de Random Forest

Características	Equipo 2
Procesador	AMD Ryzen 7 4700
RAM	16 GB
Frecuencia de Reloj	3.70 GHz
Núcleos Físicos	4
Núcleos Lógicos	8

Table 2: Características del Equipo utilizado para las pruebas de KNeighborsClassifier

Dado que ambos equipos tienen 8 núcleos físicos, las pruebas se realizaron desde 1 hasta 7 núcleos en incrementos de 2, dejando uno libre para otros procesos del sistema. Cada proceso genera un archivo log con los hiperparámetros y el accuracy del modelo.

2 Procedimiento

Para ambos modelos se siguieron los siguientes pasos generales:

- **Cargado del conjunto de datos:** Se carga un conjunto de datos previamente preprocesado desde un archivo CSV. Para la evaluación, se seleccionan las primeras 100,000 muestras del conjunto de datos.

- **Definición de la cuadrícula de hiperparámetros:** Se define una cuadrícula de combinaciones de hiperparámetros a ser evaluados. La cuadrícula varía según el modelo empleado, ya sea *Random Forest* o *K-Nearest Neighbors* (KNN).
- **Generación de combinaciones de hiperparámetros:** A partir de la cuadrícula definida, se generan todas las posibles combinaciones de hiperparámetros utilizando el método `itertools.product`, que obtiene el producto cartesiano de las listas de parámetros.
- **Nivelación de cargas:** La lista de combinaciones de hiperparámetros generada se divide equitativamente entre varios procesos (*threads*), con el fin de distribuir la carga de trabajo en función del número de que definamos en cada ejecución.
- **Entrenamiento y evaluación del modelo:** Para cada combinación de hiperparámetros, se entrena un modelo utilizando los datos de entrenamiento, distribuidos en un 80% para entrenamiento (*train*) y un 20% para prueba (*test*). El modelo se evalúa en el conjunto de prueba, y la métrica de rendimiento seleccionada es la precisión (`accuracy_score`).
- **Paralelización:** Definimos un número de procesos (`N_THREADS`) que ejecutan las combinaciones de hiperparámetros de manera paralela. Cada proceso se encarga de un subconjunto de combinaciones. Para evitar interferencias en la salida, se utiliza un bloqueo de exclusión mutua (*mutex lock*), lo cual asegura que los resultados se impriman correctamente.

De lo mencionado anteriormente se mide el tiempo total de ejecución de todos los procesos paralelizados.

2.1 Random Forest

Hiperparámetro	Valores
n_estimators	10 a 100 con un paso de 5
criterion	gini, entropy, log_loss
max_features	sqrt, log2
warm_start	True, False

Table 3: Hiperparámetros para Random Forest

Núm. Hilos	Media (s)	Desviación estándar (s)
1	989.5797	10.4996
2	752.3234	9.8589
4	480.2885	2.4288
6	392.5781	1.9996
7	364.3164	4.7505

Table 4: Tiempos Random Forest

2.2 Resultados KNN

Hiperparámetro	Valores
n Neighors	1 a 30 con pasos de 2
weights	uniform, distance
distance	eclidean, manhattan, minkowski

Table 5: Hiperparámetros para KNN

Núm. Hilos	Media (s)	Desviación estándar (s)
1	89.1205	0.6646
2	48.9299	0.8790
4	29.3976	0.5081
6	24.1367	0.2549
7	24.1176	1.3246

Table 6: Tiempos KNN

3 Conclusiones

La disminución de los tiempos de ejecución se observó en ambos algoritmos en relación con el número de hilos utilizados en la paralelización. Inicialmente, al pasar de 1 a 2 o 4 hilos, la reducción del tiempo de ejecución fue notable. Sin embargo, a medida que se incrementó el número de hilos, la mejora en los tiempos fue menos significativa. Esto sugiere que un mayor número de hilos no necesariamente optimizará el rendimiento de nuestro algoritmo. Por lo tanto, para cada proyecto, es importante determinar un número de hilos que ofrezca una mejora de tiempo significativa en comparación con el proceso secuencial.