Clase 2, Módulo 2: Complejidad y clasificación de problemas

Complejidad y clasificación de problemas

La idea de que un problema sea “resoluble” no alcanza para describir el mundo de la computación. Como vimos, todos los problemas computacionales tienen una entrada, un proceso y una salida. Pero eso no significa que todos puedan resolverse de manera práctica. Aquí es donde entra el estudio de la **complejidad computacional**, un área que analiza cómo crece el esfuerzo necesario para resolver un problema a medida que aumenta el tamaño de los datos de entrada.

Imaginemos que tenemos un algoritmo que tarda un segundo en procesar una lista de diez elementos. Si esa misma lista aumenta a veinte elementos y el tiempo de cálculo sube a cuatro segundos, podríamos decir que el problema escala razonablemente. Sin embargo, hay problemas que, al duplicar el tamaño de la entrada, hacen que el tiempo de resolución se multiplique de manera explosiva, llegando a volverse impracticables. La diferencia entre un crecimiento “razonable” y uno “explosivo” es lo que distingue a las distintas **clases de complejidad**.

La clase P: problemas tratables

La clase **P** agrupa todos aquellos problemas que se pueden resolver en **tiempo polinomial**. En términos sencillos, significa que la cantidad de pasos que requiere el algoritmo crece como una potencia del tamaño de la entrada: al cuadrado, al cubo, etc., pero siempre de manera controlada. Aunque resolver un millón de casos puede llevar más tiempo que resolver diez, la computadora sigue siendo capaz de hacerlo en un lapso útil.

Ejemplos clásicos de problemas en P son ordenar una lista de números o encontrar la ruta más corta entre dos puntos en un mapa. Son tareas que usamos todos los días sin pensarlo, y que las máquinas resuelven con gran eficacia.

La clase NP: problemas difíciles de resolver, fáciles de verificar

La clase **NP** es más desafiante. Contiene problemas para los cuales verificar una solución es rápido, pero encontrar esa solución puede ser extremadamente difícil. El Sudoku es un ejemplo perfecto: si alguien nos entrega un tablero ya resuelto, revisarlo lleva apenas segundos. Pero si debemos encontrar la solución desde cero, el esfuerzo de búsqueda puede volverse enorme.

Otro caso importante es la factorización de números grandes. Si nos dan un número primo y su factorización, verificar la multiplicación es inmediato. Pero descubrir por nosotros mismos los factores de un número de cientos de dígitos es tan costoso que forma la base de la criptografía moderna.

NP-Hard y NP-Complete

Dentro de NP encontramos dos categorías que marcan los límites de la dificultad:

- **NP-Hard** designa problemas al menos tan difíciles como los más complejos de NP. Resolver uno de ellos de manera eficiente equivaldría a tener la llave para resolver cualquier otro.
- **NP-Complete** son los problemas más representativos de NP. El clásico ejemplo es el **problema del viajante**: hallar la ruta más corta que recorra todas las ciudades y regrese al inicio. Verificar la longitud de una ruta propuesta es sencillo; lo complicado es encontrar la mejor entre miles de millones de combinaciones posibles.

Estos problemas son tan importantes que, si se descubriera un algoritmo eficiente para resolver uno de ellos, la frontera entre lo tratable y lo intratable se correría para siempre.

El dilema P vs. NP

La gran pregunta, aún sin respuesta, es si **P = NP**. Es decir, si acaso todo problema cuya solución puede verificarse rápidamente también puede resolverse de manera rápida. Resolver este enigma no es solo un desafío académico: tendría consecuencias enormes en la seguridad digital, en la planificación logística y en casi todos los campos donde la IA y la informática se aplican.

Por ahora, la mayoría de los investigadores cree que P no es igual a NP, pero nadie ha podido demostrarlo formalmente. Es uno de los problemas abiertos más famosos de la ciencia de la computación, y resolverlo tiene incluso un premio de un millón de dólares.

Conexión con la IA

La relevancia de estas categorías para la Inteligencia Artificial es clara. Muchos de los problemas con los que la IA trabaja cotidianamente —planificación, reconocimiento de patrones, optimización de rutas, juegos complejos— se encuentran en NP o son NP-Hard. Como no pueden resolverse de manera exacta en tiempos útiles, la IA recurre a heurísticas, aproximaciones y métodos de aprendizaje automático que permiten llegar a soluciones buenas, aunque no siempre perfectas.