머신러닝/딥러닝 스터디 1일차

PROBLEM -----(ALGORITHM) -----> ANS

solution

문제를 푸는 과정을 절차적으로 기술

ALGORITHM

1. Brute Force (완전 탐색)

 * BFS (Breadth-First search): 너비 우선 탐색

 * DFS (Depth-First Search): 깊이 우선 탐색

2. Divide & Conquer - 재귀로 구현

큰 문제를 작은 문제로 분할하여 해결

3. DP(Dynamic Programming) => table 을 만들어서 저장하기(top down & bottom up 둘 다 가능)

이름은 동적 계획법이지만 사실 동적이랑 관련은 없다. (멋있어 보이려 하는 이름)

겉보기에는 Divide & Conquer 와 비슷하지만 중복된 부분이 존재하여 Save table & Look up로 중복된 값은 다시 계산하지 않는다. 방정식을 찾는 것이 핵심.

4. Greedy ALGORITHM(탐욕적인 방법!)

 * Base performance

 * may be optimal

 * easy

 * 최적에는 반드시 증명이 필요하다.

5. Backtracking (되추적)

DFS 알고리즘으로 탐색을 하다가 답이 아닌 것이 나오면 더 이상 진행하지 않고 아래 가지를 절단하는 알고리즘

6. Branch and Bound (분기한정법)

backtracking이 depth 가 깊어지게 되면 탐색에 시간이 오래 걸리는 단점이 존재.

이를 보완하기 위해 BFS를 이용하는 알고리즘이다. 최대값을 구하려면 under estimation 을 하여 해당 예측보다 아래 가지는 다 쳐버린다. estimation 값은 탐색이 진행될수록 optimal에 가까워지게 되어 더 많은 가지를 친다.

7. Best First Branch and Bound

A* 알고리즘의 쉬운 형태로 B & B를 따르지만 Breadth 탐색 시 가장 가능성이 있는 곳 먼저 탐색하는 기법. 많은 가지를 효율적으로 친다.

8. NP/ NP hard

엄밀하게 NP를 정의하면 비결정 알고리즘으로 P time 안에 답이 맞는지 확인하는 것.

(Non-deterministic Polynomial time algorithm)

NP –hard 는 NP는 아니지만 NP만큼 어려운 문제

간단한 예로는 TSP문제, 0/1 Knapsack 같은 문제를 말한다.

TSP 문제: * 조 합 최 적 화 * (N! 의 시간을 줄이기)

Approximate algorithm: 시간을 줄이고 정확도를 버려서 근사값을 찾는다

보통 Greedy로 구현 하곤 함

9. Search algorithm

답을 정하고 그 답을 계속해서 변경해서 오차를 줄여나가는 과정

greedy 같은 알고리즘으로 근사를 하여 답을 찾고 random하게 search하여 오차를 줄여간다.