Dynamic Programming

- The Knapsack Problem
- Dynamic Programming Formulation
- Implementation
- Fractional Knapsack and Subset Sum Problem

Imdad ullah Khan

Knapsack Problem: Dynamic Programming

Input: A set $\mathcal{U}$ of objects $\{a_1, \ldots, a_n\}$ with

- integral weights $\{w_1, \ldots, w_n\}$ and
- positive values $\{v_1, \ldots, v_n\}$ and
- a positive integer C (capacity)

Output: A subset $S \subset \mathcal{U}$ with total weight $\leq C$ and maximum total value

- Fix an order on objects $a_1, \ldots, a_n$
- $\text{OPT-SET}(k)$ is the max value feasible subset of $\mathcal{U}[1 \ldots k]$
- $\text{OPT-VAL}(k)$ is the total value of $\text{OPT-SET}(k)$

- Out goal is to find $\text{OPT-VAL}(n)$ (and $\text{OPT-SET}(n)$)

Knapsack Problem: Dynamic Programming

Argue about structure of the solution (though we can't compute it)

See how solution is composed of that to smaller subproblems

- Either a_n is not part of the solution, $\textsc{opt-set}(n)$
  - v_n is not counted in $\textsc{opt-val}(n)$
  - Some subset of $a_1, \ldots, a_{n-1}$ is $\textsc{opt-set}(n)$
  - Analyze $\textsc{opt-set}(n-1)$
- Or a_n is part of the solution, $\textsc{opt-set}(n)$
  - v_n is counted in $\textsc{opt-val}(n)$
  - Some subset of $a_1, \ldots, a_{n-1}$ is in $\textsc{opt-set}(n)$ in addition to a_n
  - We don't even know if it is valid to include a_n in $\textsc{opt-set}(n)$
- Need more subproblems?

Knapsack Problem: Dynamic Programming

Input: A set $\mathcal{U}$ of objects $\{a_1, \ldots, a_n\}$ with

- integral weights $\{w_1, \ldots, w_n\}$ and
- positive values $\{v_1, \ldots, v_n\}$ and
- a positive integer C (capacity)

Output: A subset $S \subset \mathcal{U}$ with total weight $\leq C$ and maximum total value

- Fix an order on objects $a_1, \ldots, a_n$
- $\text{OPT-SET}(k, c)$ is the max value feasible subset of $\mathcal{U}[1 \ldots k]$ and c
- $\text{OPT-VAL}(k, c)$ is the total value of $\text{OPT-SET}(k, c)$
- Out goal is to find $\text{OPT-SET}(n, C)$ (and $\text{OPT-VAL}(n, C)$)

Knapsack Problem: Dynamic Programming

Argue about structure of the solution (though we can't compute it)

See how solution is composed of that to smaller subproblems

- Either a_n is part of the solution, OPT-SET(n, C)
  - v_i is counted in OPT-VAL(n, C)
  - Remaining items in OPT-SET(n, C) are from $a_1, \ldots, a_{n-1}$
  - Remaining capacity is $C - w_n$
  - Analyze OPT-SET$(n - 1, C - w_n)$
- Or a_n is not part of the solution, OPT-SET(n, C)
  - v_n is not counted in OPT-VAL(n)
  - Items in OPT-SET(n, C) are from $a_1, \ldots, a_{n-1}$
  - Remaining capacity is C
  - Analyze OPT-SET$(n - 1, C)$

Analyze OPT-SET$(n - 1, C - w_n)$ and OPT-SET$(n - 1, C)$?

Knapsack Problem: Dynamic Programming

Analyze $\text{OPT-SET}(n-1, C-w_n)$ and $\text{OPT-SET}(n-1, C)$?

- $a_n \notin \text{OPT-SET}(n, C)$
  - $\text{OPT-SET}(n, C)$ is a solution of $[a_1, \ldots, a_{n-1}], C$
  - $\text{OPT-SET}(n, C)$ is a max value solution of $[a_1, \ldots, a_{n-1}], C$
  - A bigger value solution for $[a_1, \ldots, a_{n-1}], C$ is also good for $[a_1, \ldots, a_n], C$
- Or $a_n \in \text{OPT-SET}(n, C)$
  - $\text{OPT-SET}(n, C) \setminus \{a_n\}$ is a solution for $[a_1, \ldots, a_{n-1}], C - w_n$
  - $\text{OPT-SET}(n, C) \setminus \{a_n\}$ is a best solution for $[a_1, \ldots, a_{n-1}], C - w_n$
  - If $O \subset \{a_1, \ldots, a_{n-1}\}$ has total weight $\leq C - w_n$ and value $\geq \text{OPT-VAL}(n-1, C-w_n)$
  - Then $O \cup \{a_n\}$ has weight $\leq C - w_n + w_n$ and value $value(O) + v_n \geq \text{OPT-VAL}(n-1, C-w_n) + v_n = \text{OPT-VAL}(n, C)$

Knapsack Problem: Dynamic Programming

Analyze $\textsc{opt-set}(n-1, C-w_n)$ and $\textsc{opt-set}(n-1, C)$?

- A max value subset of items $\{a_1, \ldots, a_n\}$ with total weight $\leq C$ is either a max value subset of items $\{a_1, \ldots, a_{n-1}\}$ with total weight $\leq C$
- or it is a_n union with a max value subset of items $\{a_1, \ldots, a_{n-1}\}$ with total weight at most $C - w_n$

$$\textsc{opt-val}(n, C) = \begin{cases} \textsc{opt-val}(n-1, C-w_n) + v_n & \text{if } a_n \in \textsc{opt-set}(n, C) \\ \textsc{opt-val}(n-1, C) & \text{if } a_n \notin \textsc{opt-set}(n, C) \end{cases}$$

In general

$$\textsc{opt-val}(k, c) = \begin{cases} \textsc{opt-val}(k-1, c-w_k) + v_k & \text{if } a_k \in \textsc{opt-set}(k, c) \\ \textsc{opt-val}(k-1, c) & \text{if } a_k \notin \textsc{opt-set}(k, c) \end{cases}$$

Knapsack Problem: Dynamic Programming

Recursion ?

- If $a_k \in \textsc{opt-set}(k, c)$, then find max value subset of $\{a_1, \ldots, a_{n-1}\}$ with weight at most $c - w_k$
- If $a_k \notin \textsc{opt-set}(k, c)$, then find max value subset of $\{a_1, \ldots, a_{n-1}\}$ with weight at most c

We don't know bases cases and which branch to take

Try both branches and select the one bigger ?

$$\textsc{opt-val}(k, c) = \max \begin{cases} 0 & \text{if } k = 0 \\ 0 & \text{if } c = 0 \\ \textsc{opt-val}(k-1, c - w_k) & \text{if } a_k \in \textsc{opt-set}(k, c) \\ \textsc{opt-val}(k-1, c) & \text{if } a_k \notin \textsc{opt-set}(k, c) \end{cases}$$