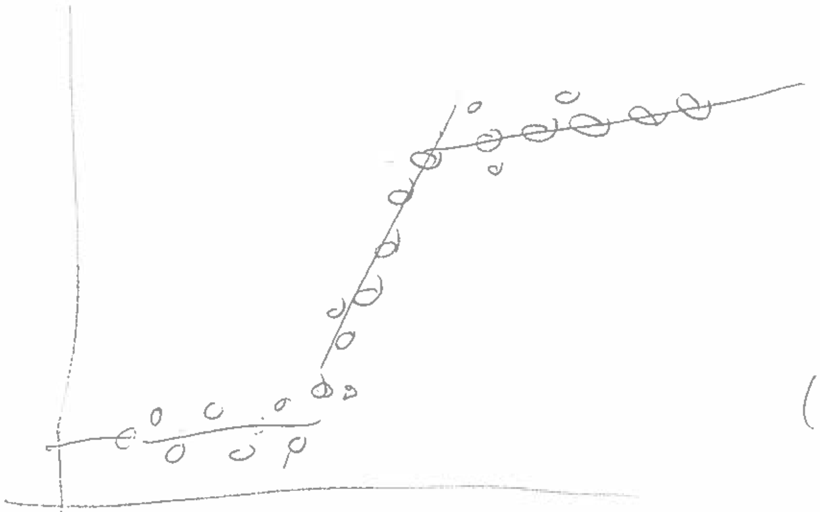
Segmented Least Squares



(DEMO)

Input: sequence of points $p_1, \dots, p_n$

$p_i = (x_i, y_i)$

$x_1 < x_2 < \dots < x_n$

Goal: partition points into segments $S_1, S_2, \dots, S_k$

$$\underbrace{p_1, p_2, p_3}_{S_1}\ \underbrace{p_4\ p_5 \dots}_{S_2} \dots\dots\dots \underbrace{\quad p_n}_{S_k}$$

In each segment S_ℓ, fit a line L_ℓ by OLS + let $\text{Error}(L_\ell, S_\ell)$ be the error (SSE).

$\boxed{\text{calculus} \quad \Theta(n)}$

Minimize

$$\underbrace{\sum_{\ell=1}^{k} \text{Error}(L_\ell, P_\ell)}_{\text{total error}} + \underbrace{C \cdot k}_{C \times (\#\text{ segments})}$$

Idea for recurrence: consider starting point p_i of final segment $P_k = p_i p_{i+1} \cdots p_n$

Value of optimal solution is

(*) $\quad \text{Error}(L_k, P_k) + C + OPT(i-1)$

value of optimal solution on points $p_1, p_2, \ldots, p_{i-1}$

(To determine final segment, minimize (*) over all choices of p_i)

Subproblems:

Let $OPT(j)$ = value of optimal solution on points $p_1, \ldots, p_j$

Let $e_{i,j} = \text{Error}(L^*, \{p_i, p_{i+1}, \ldots, p_j\})$ be error of best fit line on $p_i, \ldots, p_j$

Recurrence:

$$OPT(j) = \min_{1 \le i \le j} \left(e_{i,j} + C + OPT(i-1) \right)$$

all choices of p_i

error of final segment

penalty for one segment

optimal soln to subproblem

Subset Sum

Example problem where we need to "add a variable" to define subproblems.

Input:

Items $i = 1, 2, \ldots, n$
Weights: w_i (integers)
Capacity W

Goal: select subset $S \subset \{1, 2, \ldots, n\}$ of all items to maximize total weight $\sum_{i \in S} w_i$ without exceeding capacity W.

$$\boxed{\max_{\substack{S \subset \{1, \ldots, n\},\\ \sum_{i \in S} w_i < W}} \sum_{i \in S} w_i}$$ BIG

Example ($W = 6$, $n = 3$)

i	1	2	3
w_i	2	2	3

S	weight	
$\{1,2,3\}$	$2+2+3=7$	
$\{1,2\}$	$2+2=4$	
$\{1,3\}$	$2+3=5$	OPTIMAL

Applications: packing, scheduling
(Many variations of this problem with different applications)

False Start: try to define subproblems in same way as Weighted Interval Scheduling

Let $OPT(i)$ = best possible solution using only items $\{1, 2, \dots, i\}$

$$= \max_{\substack{S \subset \{1, \dots, i\} \\ \sum_{i \in S} w_i \le W}} \sum_{i \in S} w_i$$

Let $\mathcal{O}$ be an optimal solution + consider whether or not $n \in \mathcal{O}$

Case 1: $n \notin \mathcal{O} \Rightarrow OPT(n-1) = OPT(n)$

Case 2: $n \in \mathcal{O} \Rightarrow OPT(n-1) = w_n + \underbrace{???}$

problem: we only have $W' = W - w_n$ weight remaining
Don't know how to express this

Idea: introduce extra variable w = amount of weight remaining. Define

$$OPT(i, w) = \max_{\substack{S \subseteq \{1, 2, \dots, i\} \\ \sum_{j \in S} w_j \le w}} \sum_{j \in S} w_j$$

Value of best solution on items $\{1, \dots, i\}$ with capacity w

Exercise: what is solution to original problem in terms of OPT(i, w)?

Overall optimal value is OPT(n, W)

Now, consider same two cases:

Case 1: $n \in \mathcal{O} \Rightarrow OPT(n, W) = OPT(n-1, W)$

Case 2: $n \notin \mathcal{O} \Rightarrow OPT(n, W) = w_n + OPT(n-1, W - w_n)$

So, recurrence is:

$$OPT(i, w) = \begin{cases} \max\left(OPT(i-1, w),\ w_i + OPT(i-1, w-w_i)\right) & \text{if } w_i \le w \\ OPT(i-1, w) & \text{if } w_i > w \end{cases}$$

$OPT(i, 0) = 0$

Slide with Pseudocode

```
Subset-Sum(n, W)
    Set M[0, w] = 0 for w = 0,..,W
    For i = 1,...,n   // items
        For w = 0,...,W   // weights
            if w_i > w
                M[i, w] = M[i-1, w]
            else
                M[i, w] = max(M[i-1, w], w_i + M[i-1, w-w_i])
            end
```

Subset-Sum running time

$\Theta(nW)$

NOT polynomial in input size

(E.g. $W = 1{,}000{,}000$)

Example

OPT

i								weight
3	0	0	2	3	5	5	5	3
2	0	0	2	2	4	4	4	2
1	0	0	2	2	2	2	2	2
0	0	0	0	0	0	0	0	
	0	1	2	3	4	5	6	

w

i

w

(Integer) Knapsack Problem

Items $1, 2, \ldots, n$

Weights w_i

Values v_i

Capacity W

Goal: $\max\limits_{\substack{S \subset \{1,\ldots,n\} \\ \sum_{i \in S} w_i \leq W}} \sum_{i \in S} v_i$ maximize total value subject to weight constraint

Slight modification of Subset Sum

Recurrence?

$$OPT(i, w) = \max\left(OPT(i-1, w),\ v_i + OPT(i-1, w_i - w_i)\right)$$