Announcements

Hw1 is available on Gradescope. **Due Friday Jan 30.** All hw counts!! (no drops), but you have 6 late days to use

On all homework problems (expect for coding) you are **always asked to prove any statement you claim**. If you design an algorithm and claim (1) it is correct and (2) runs in polynomial time, you **must prove both statements**.

Solutions to section problems posted on canvas, video ~~will be~~ is posted ~~shortly~~ due to cancelled sections

Sections **attendance mandatory, will include a 10 min quiz** about previous hw.

Poll Everywhere: register with your Cornell netid

see Ed for instructions

hw2 will open after class on canvas

1 question coding

2 questions design & analyse algorithms

Dynamic Programming II summary

Smaller subproblems: smaller subproblems whose solution helps solving real problem

- need polynomially many subproblems

  e.g interval sceduling not all subset (2^n)

Algorithm: Iterative vs. Recursive (memorization: solve any subproblem only once)

build up solutions small ⟶ big

OK recursive, but store all solutions

Dynamic Programming II summary

Proof of correctness: induction proof

iterative alg. $\Rightarrow$ proof each step correct

Running time ✓

Extracting the solution, not only the OPT value

DP table

Opt value

Option 1: store also solution

may increase run time

Option 2: retrace from back

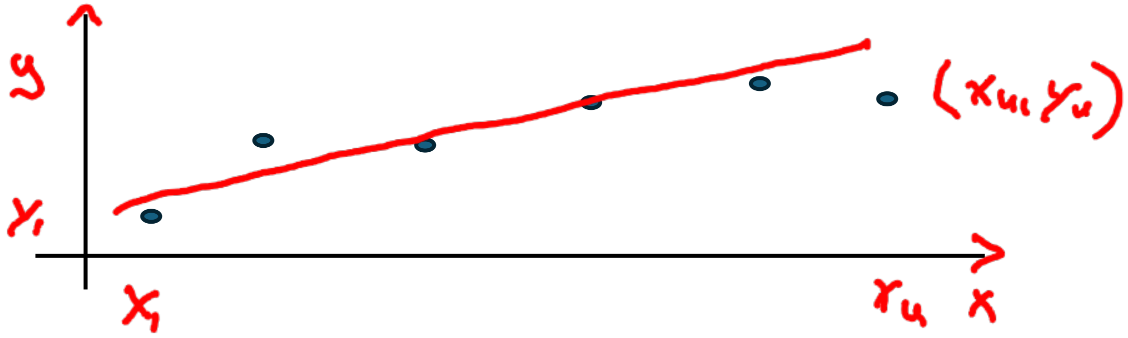
Opt(n)

$$Opt(n) = \max(Opt(n-1),\ v_n + Opt(p(n)))$$

Dynamic Programming II: Segmented Least Squares

Least Squares: fit line

$y = mx + b$ minimizing
$Err(m, b) = \sum (y_i - m x_i)^2$

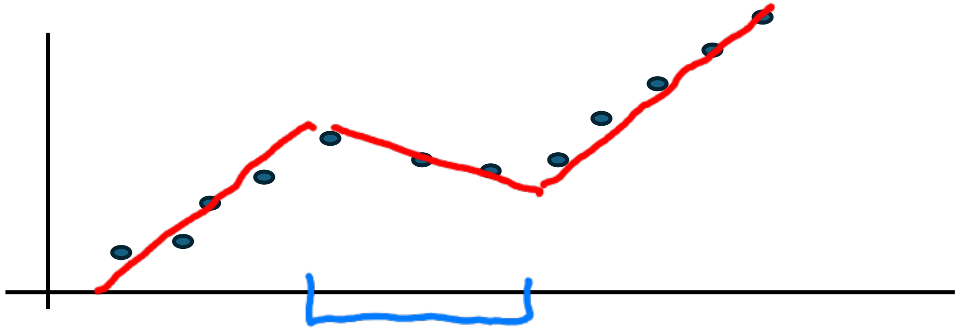


$y = b + mx$?

Solution: $m = \frac{n \sum_i x_i y_i - (\sum_i x_i)(\sum_i y_i)}{n \sum_i x_i - (\sum_i y_i)^2}$ and $b = \frac{\sum_i y_i - m \sum_i x_i}{n}$

Segmented Least Squares: penalty C for using additional segments

at times fitting multiple lines is better



3 subintervals

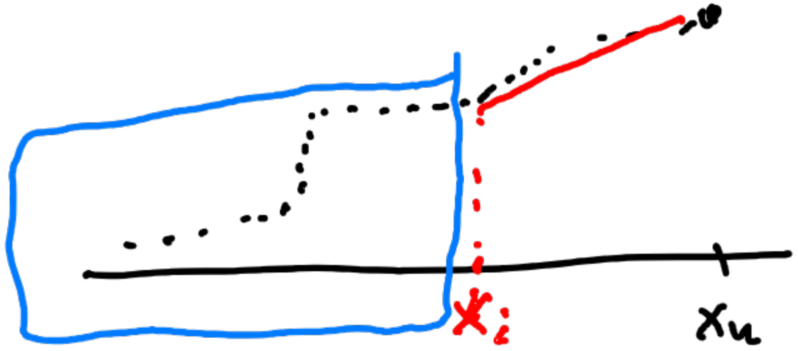
Let e_{ij} = error if single line fit $(x_i\ y_i) \ldots (x_j\ y_j)$

Find partition subintervals

min $\sum_{k=1}^{\ell} e_{i_k j_k} + C \cdot \ell$ ← the intervals

C parameter input

Subproblems, and the “final decision”



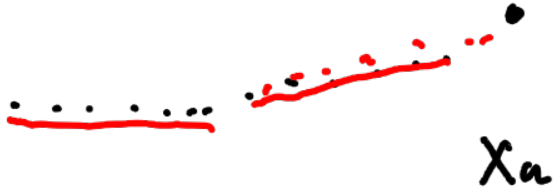
(1) (x_n, y_n) – should we start separate line, or continue with previous line

need $Opt(n-1)$ solution points $1..n-1$

(2) using

beginning of segment including last point $[i, n]$

$$Opt(n) = \min_{1 \le i \le n} Opt(i-1) + e_{in} + C$$



$Opt(i)$ = best solution for points $[1, \dots, i]$

$Opt(0) = 0$

For $j = 1$ to n

(*) $Opt(j) = \min_{1 \le i \le j} \Big[Opt(i-1) + \underbrace{e_{ij} + C}_{\text{cost of last segment}} \Big]$ multiway decision

endfor

Output $Opt(n)$

previous cost

Correctness of the Algorithm getting the Opt value

natural induction proof.

base $Opt(0) = 0$

induction step: explain why * correct assuming all previous Opt values correct

Note: last decision picks start x_i of last segment

Extracting the solution itself

last decision $\min_i \; Opt(i-1) + e_{i,n} + C$

$\Longrightarrow$ last interval $[i,n]$

next check equation for $Opt(i-1)$

$\vdots$

Join by Web PollEv.com/evatardos772

What is the running time of the algorithm we designed?

A. $O(n)$ time

B. $O(n \log n)$ time

C. $O(n^2)$ time

D. $O(n^3)$ time

E. $O(2^n)$ time

n iterations

O(n) per iteration

$\leq n$ options for i

if e_{ij} is pre-computed

computing e_{ij}

$O(n^3)$

$Opt(0)=0$

For $j=1\ldots n$

$Opt(j) = \min_{i<j} Opt(i-1)+C+\boxed{e_{ij}}$

end for

Output $Opt(n)$

trace back to find solution

OK to store solutions with Opt