CS3000: Algorithms & Data
Jonathan Ullman

Lecture 7:

- Dynamic Programming: Segmented Least Squares

Sep 28, 2018

Dynamic Programming Recap

smaller versions of the same problem (or similar)

- **Recipe:**

"small"

cleverness

(1) identify a set of **subproblems**

(2) relate the subproblems via a **recurrence**

almost independent of the problem

(3) find an **efficient implementation** of the recurrence

(4) **reconstruct the solution** from the DP table

Dynamic Programming Recap

OPT(i) = the value of the optimal sched using only $\{1, \ldots, i\}$

$$OPT(i) = \max\{\, OPT(i-1),\ v_i + OPT(p(i)) \,\}$$

$OPT(0) = 0$

$OPT(1) = v_1$

1	$v_1 = 8$	$\boldsymbol{p(1) = 0}$
2	$v_2 = 6$	$\boldsymbol{p(2) = 0}$
3	$v_3 = 11$	$\boldsymbol{p(3) = 0}$
4	$v_4 = 10$	$\boldsymbol{p(4) = 2}$
5	$v_5 = 9$	$\boldsymbol{p(5) = 3}$
6	$v_6 = 15$	$\boldsymbol{p(6) = 1}$

$S = \{6, 1\}$

M[0]	M[1]	M[2]	M[3]	M[4]	M[5]	M[6]
0	8	8	11	18	20	23

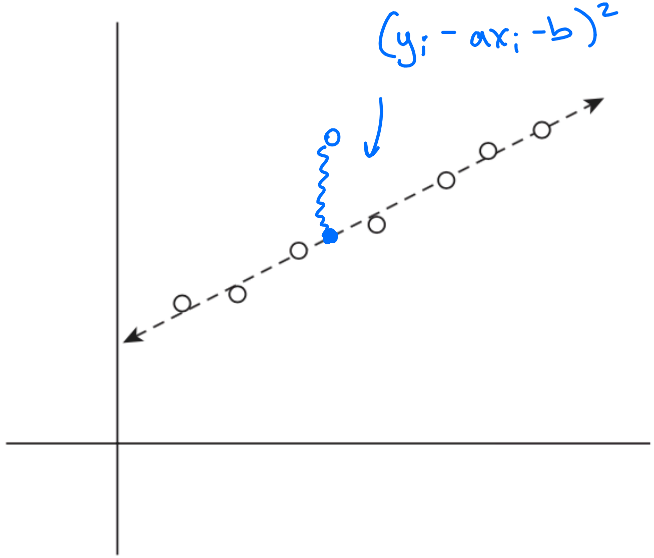
Segmented Least Squares

Background: Least Squares

years of schooling   income

- **Input:** n data points $P = \{(x_1, y_1), \ldots, (x_n, y_n)\}$
- **Output:** the line L (i.e. $y = ax + b$) that fits **best**
  - **best** = minimizes $error(L, P) = \sum_i (y_i - ax_i - b)^2$

$(y_i - ax_i - b)^2$



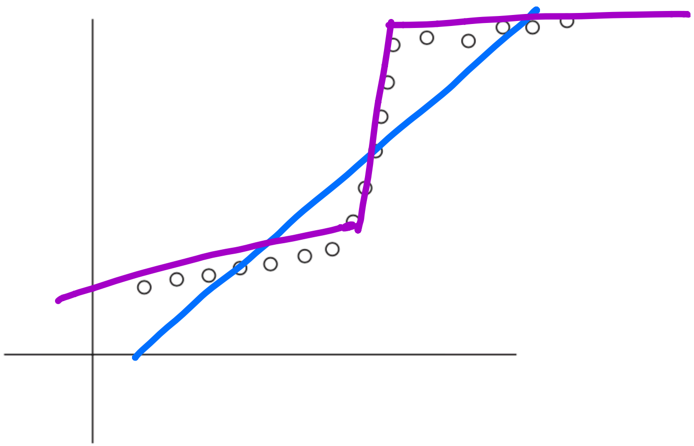
$$a = \frac{n \sum x_i y_i - (\sum x_i)(\sum y_i)}{n \sum x_i^2 - (\sum x_i)^2}$$

$$b = \frac{\sum y_i - a \sum x_i}{n}$$

There is an $O(n)$ time algorithm for finding the line of best fit

Segmented Least Squares

- **Input:** n data points $P = \{(x_1, y_1), \ldots, (x_n, y_n)\}$
- What if the data does not look like a line?

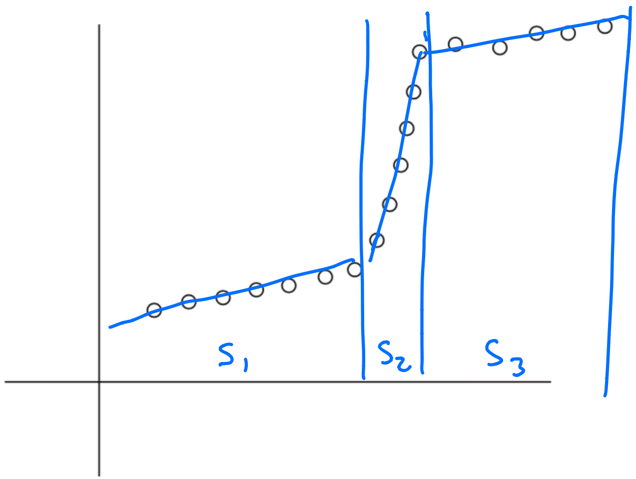


I can describe some data better using > 1 line

Using $\frac{n}{2}$ lines defeats the purpose

Segmented Least Squares

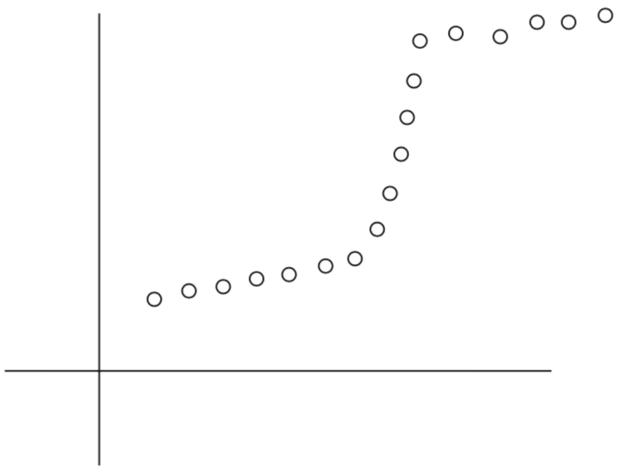
- **Input:** n data points $P = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, **cost parameter** $C > 0$
  - Assume $x_1 < x_2 < \cdots < x_n$ (If not, you could sort yourself)
- **Output:** a partition of P into contiguous segments $S_1, S_2, \ldots, S_m$, lines $L_1, L_2, \ldots, L_m$, minimizing "cost"



$$\text{cost}(S_1 \ldots S_m, L_1, \ldots, L_m) = \sum_{j=1}^{m} \text{error}(L_j, S_j) + mC$$

Segmented Least Squares

- **First observation:** for every segment S_j, L_j must be the (single) line of best fit for S_j
  - Let $L^*_{i,j}$ be the optimal line for $\{p_i, \ldots, p_j\}$
  - Let $\varepsilon_{i,j} = error\left(L^*_{i,j}, \{p_i, \ldots, p_j\}\right)$



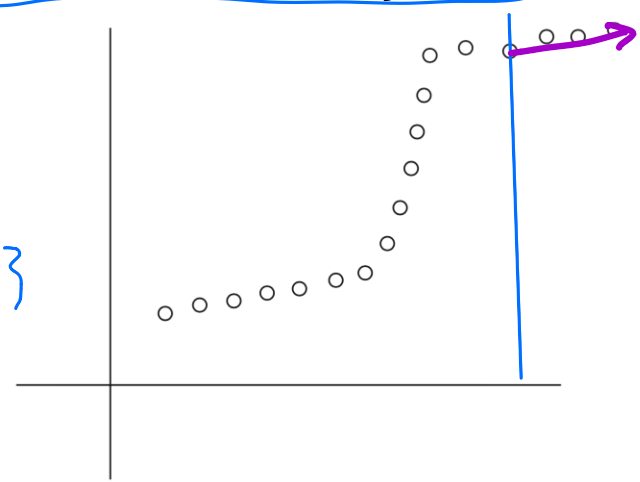
- Easy to compute $\{\varepsilon_{i,j}\}$ in $O(n^3)$
- Can also do it in $O(n^2)$ using more cleverness

SLS

Let $L^*_{i,j}$ be the optimal line for $\{p_i, \dots, p_j\}$
Let $\varepsilon_{i,j} = error(L^*_{i,j}, \{p_i, \dots, p_j\})$

- Let O be the **optimal** solution
- What is the final segment in O? $S_m = \{p_i, \dots, p_n\}$
  $L_m = L^*_{i,n}$
- If final segment is $\{p_i, \dots, p_n\}$, then the optimal solution for $\{p_1, \dots, p_n\}$ is the optimal sol. for $\{p_1, \dots, p_{i-1}\}$ + $\{p_i, \dots, p_n\}$

Subproblems:
Find optimal solution for $\{p_1 \dots p_j\}$
$0 \le j \le n$



Ask the Audience

- OPT(j) is the total error of the optimal SLS solution for points $\{p_1, \ldots, p_j\}$
- Case i: the final segment is $\{p_i, \ldots, p_j\}$

some $1 \le i \le j$

total cost is ① + ② + ③

① error on $\{p_i \ldots p_j\}$

② cost C of using one segment

③ optimal cost for $\{p_1 \ldots p_{i-1}\}$

Recurrence for OPT(j)

$$\boxed{\begin{gathered} OPT(j) = \min_{i:\, 1 \le i \le j} \varepsilon_{i,j} + C + OPT(i-1) \\ OPT(0) = 0 \qquad OPT(1) = C \qquad OPT(2) = C \end{gathered}}$$

SLS

Let $L^*_{i,j}$ be the optimal line for $\{p_i, \dots, p_j\}$
Let $\varepsilon_{i,j} = error(L^*_{i,j}, \{p_i, \dots, p_j\})$

- Let $\mathrm{OPT}(j)$ be the **value** of the optimal solution for points $\{p_1, \dots, p_j\}$
- **Case *i*:** final segment is $\{p_i, \dots, p_j\}$
  - optimal solution is $L^*_{i,j}$ ∪ optimal sol. for $\{p_1, \dots, p_{i-1}\}$
  - can use any $i \in \{1, \dots, j\}$

SLS

Let $L_{i,j}^*$ be the optimal line for $\{p_i, \ldots, p_j\}$
Let $\varepsilon_{i,j} = error\left(L_{i,j}^*, \{p_i, \ldots, p_j\}\right)$

- Let $\mathrm{OPT}(j)$ be the **value** of the optimal solution for points $\{p_1, \ldots, p_j\}$
- **Case i:** final segment is $\{p_i, \ldots, p_j\}$
  - optimal solution is $L_{i,j}^* \cup$ optimal sol. for $\{p_1, \ldots, p_{i-1}\}$
  - can use any $i \in \{1, \ldots, j\}$

Recurrence: $\mathrm{OPT}(j) = \min_{1 \leq i \leq j} \varepsilon_{i,j} + C + \mathrm{OPT}(i-1)$

Base cases: $\mathrm{OPT}(0) = 0$
$\mathrm{OPT}(1) = \mathrm{OPT}(2) = C$

SLS: Take I

```
// All inputs are global vars
FindOPT(n):
  if (n = 0): return 0
  elseif (n = 1,2): return C
  else:
    return
```
$$\min_{1\le i\le n} \varepsilon_{i,n} + C + \mathrm{FindOPT}(i-1)$$

$$T(n) = \sum_{i=0}^{n-1} T(n-1)$$

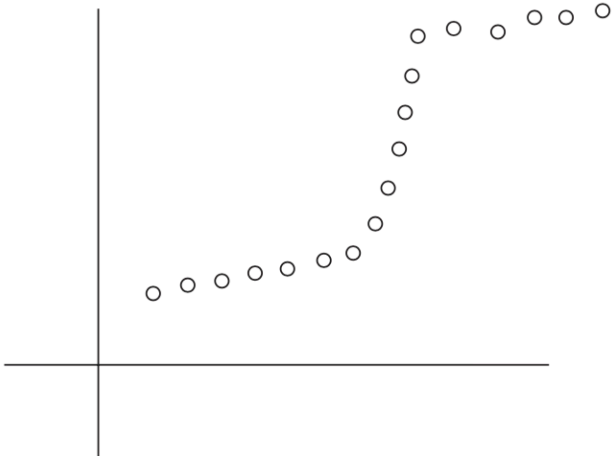
$$T(n) \geq T(n-1) + T(n-2) \qquad T(n) \doteq \Omega(1.62^n)$$

SLS: Take II ("Top-Down")

```
// All inputs are global vars
M ← empty array, M[0] ← 0, M[1] ← C, M[2] ← C
FindOPT(n):
  if (M[n] is not empty): return M[n]
  else:
    M[n] ← min_{1≤i≤n} ε_{i,n} + C + FindOPT(i − 1)
    return M[n]
```

- Have to fill M[3] ... M[n]
- To fill M[i] we make i recursive calls
- \# of recursive calls is $\sum_{i=3}^{n} i = \Theta(n^2)$

SLS: Take III ("Bottom-Up")



Dynamic Programming Table

M[0]	M[1]	M[2]	…	M[i]	…	M[n]
0	c	c	…		…	

SLS: Take III ("Bottom-Up")

```
// All inputs are global vars
FindOPT(n):
  M[0] ← 0, M[1] ← C, M[2] ← C
  for (j = 3,…,n):
    M[j] ← min_{1≤i≤j} ε_{i,j} + C + M[i − 1]
  return M[n]
```

n-2 iterations

O(n) per iteration

Running Time is $\Theta(n^2)$ + time to compute $\{\varepsilon_{i,j}\}$

Rule of Thumb: $O($# of pairs i,j s.t. $M[i]$ depends on $M[j]$ $)$

Finding Segments

Let $L_{i,j}^*$ be the optimal line for $\{p_i, \dots, p_j\}$
Let $\varepsilon_{i,j} = error(L_{i,j}^*, \{p_i, \dots, p_j\})$

O_j = optimal segments for $\{p_1, \dots p_j\}$

- Let $\mathrm{OPT}(j)$ be the **value** of the optimal solution for points $\{p_1, \dots, p_j\}$
- **Case i:** final segment is $\{p_i, \dots, p_j\}$
  - optimal solution is $L_{i,j}^* \cup$ optimal sol. for $\{p_1, \dots, p_{i-1}\}$
  - can use any $i \in \{1, \dots, j\}$

If $i \in \underset{i:\, 1 \le i \le j}{\operatorname{argmin}} \; \varepsilon_{i,j} + C + \mathrm{OPT}(i-1)$

then O_j is $\{p_i, \dots, p_j\} + O_{i-1}$

exists an optimal O_j

Finding Segments

```
// All inputs are global vars
// M[0:n] contains solutions to subproblems
FindSol(M,n):
  if (n = 0): return ∅
  elseif (n = 1): return {1}
  else:
    Let i ← argmin_{1≤i≤n} ε_{i,n} + C + M[i − 1]:
    return {i,…,n} + FindSol(M,i-1)
```

$$T(n) \le Cn + T(i-1)$$

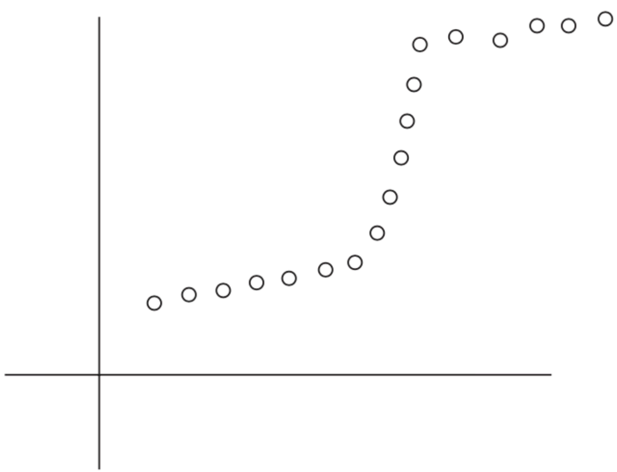
$$\le Cn + T(n-1)$$

$$T(n) = \Theta(n^2)$$

Segmented Least Squares v.2

Segmented Least Squares v.2

- **Input:** n data points $P = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, parameter $1 \leq k \leq n$
  - Hard upper bound on the number of segments
- **Output:** a partition of P into $\leq k$ contiguous segments $S_1, S_2, \ldots, S_k$ minimizing "cost"



$$\text{cost}(S_1 \ldots S_k, L_1 \ldots L_k)$$

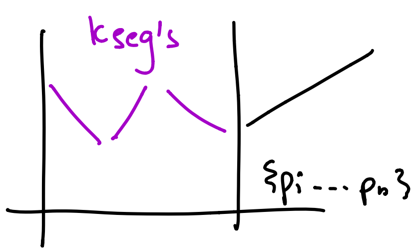
$$= \sum_{i=1}^{k} \text{error}(S_i, L_i)$$

SLSv.2

Let $L^*_{i,j}$ be the optimal line for $\{p_i, \dots, p_j\}$
Let $\varepsilon_{i,j} = error(L^*_{i,j}, \{p_i, \dots, p_j\})$

- Let O be the optimal solution
- O has some final segment $\{p_i, \dots, p_n\}$
- If the final segment is $\{p_i, \dots, p_n\}$, then O should be $opt(\{p_i, \dots, p_n\}) + opt(\{p_1, \dots, p_{i-1}\})$

$$OPT(n) = \min_{1 \le i \le n} \varepsilon_{i,n} + OPT(i-1)$$



Problem: optimal soln for $\{p_1, \dots, p_{i-1}\}$ might use k segments

SLSv.2

$(n+1)(k+1)$ subproblems

Let $L^*_{i,j}$ be the optimal line for $\{p_i, \ldots, p_j\}$
Let $\varepsilon_{i,j} = error(L^*_{i,j}, \{p_i, \ldots, p_j\})$

- Let $\mathrm{OPT}(j, \ell)$ be the optimal solution for points $\{1, \ldots, j\}$ using $\leq \ell$ segments
- **Case $\boldsymbol{i}$:** final segment is $\{p_i, \ldots, p_j\}$
  - optimal solution is $L^*_{i,j}$ ∪ optimal solution for points $\{p_1, \ldots, p_{i-1}\}$ using $\leq \ell - 1$ segments
  - can use any $i \in \{1, \ldots, j\}$

Recurrence: $\mathrm{OPT}(j, \ell) = \min_{1 \leq i \leq j} \varepsilon_{i,j} + \mathrm{OPT}(i-1, \ell-1)$

Base cases: $\mathrm{OPT}(0, \ell) = 0 \quad \forall \ell \geq 0$

$\mathrm{OPT}(j, 0) = \infty \quad \forall j \geq 1$

SLSv.2: Take II ("Top-Down")

2D array

```
// All inputs are global vars
M ← empty array, M[0,ℓ] ← 0, M[j,0] ← ∞
FindOPT(n,k):
  if (M[n,k] is not empty): return M[n,k]
  else:
    M[n,k] ← min_{1≤i≤n} ε_{i,n} + FindOPT(i−1, k−1)
    return M[n,k]
```

SLSv.2: Take III ("Bottom-Up")

$(n+1)(k+1)$
subproblems

$M[j,\ell]$ = best value for $\{p_1, \dots p_j\}$ using $\le \ell$ segments

	M[·,0]	M[·,1]	M[·,2]	M[·,3]	…	M[·,k]
M[0, ·]	0	0	0	0	0	0
M[1, ·]	∞	0				
M[2, ·]	∞	0				
M[3, ·]	∞					
…	∞				$M[j,\ell]$	
M[n, ·]	∞					

Fill this out by going down one column at a time

SLSv.2: Take III ("Bottom-Up")

```
// All inputs are global vars
FindOPT(n,k):
  M[0,ℓ] ← 0, M[j,0] ← ∞
  for (ℓ = 1,…,k):
    for (j = 1,…,n):
    M[j,ℓ] ← min_{1≤i≤j} ε_{i,j} + FindOPT(j − 1, ℓ − 1)
  return M[n,k]
```

k

n

$O(n)$

Total Running Time: $O(n^2k)$

Space: $O(nk)$

SLSv.2: Finding Segments

```
// All inputs are global vars
// M[0:n,0:k] contains solutions to subproblems
FindSol(M,n,k):
  if (n = 0): return ∅
  elseif (n = 1): return {1}
  else:
    let i ← argmax_{1≤i≤n} ε_{i,n} + M[i − 1, k − 1]:
    return {i,…,n} + FindSol(M,i-1,k-1)
```

SLS Wrapup

- **Version 1:** can solve SLS with a "segment cost" in time $O(n^2)$ space $O(n^2)$
  - New idea: multiway case analysis for the final segment
- **Version 2:** can solve SLS with a "hard cap" of k segments in time $O(n^2 k)$ space $O(n^2 + nk)$
  - New idea: introducing additional variables to expand the set of subproblems
- Correctness follows from the recurrence
- Computational costs:
  - Running time $\approx$ total number of terms in all recurrences
  - Space $\approx$ total number of subproblems