Midterm I Wed Feb 12

CS3000: Algorithms & Data
Jonathan Ullman

Lecture 6:

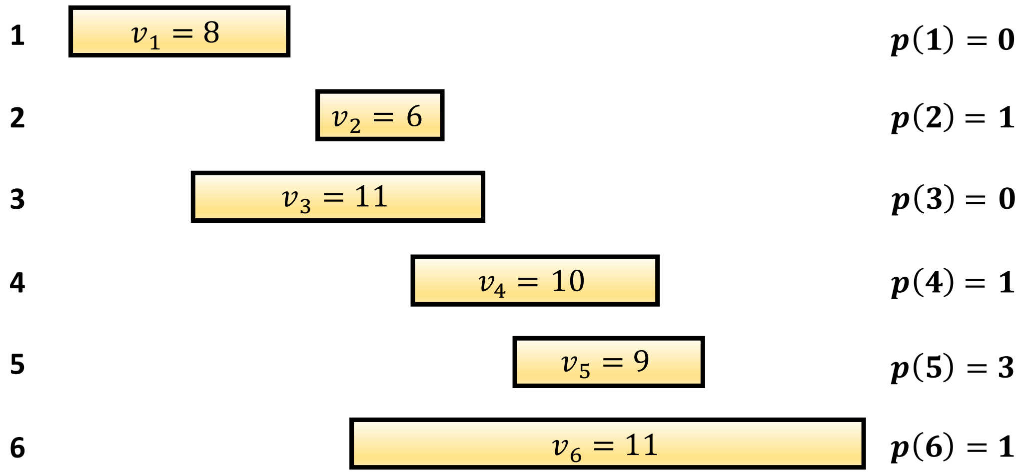
- Dynamic Programming: Segmented Least Squares

Jan 27, 2020

Dynamic Programming Recap

- **Recipe:**
  - creativity
  - (1) identify a set of **subproblems**
  - (2) relate the subproblems via a **recurrence**
  - automated
  - (3) find an **efficient implementation** of the recurrence
  - (4) **reconstruct the solution** from the DP table

Dynamic Programming Recap

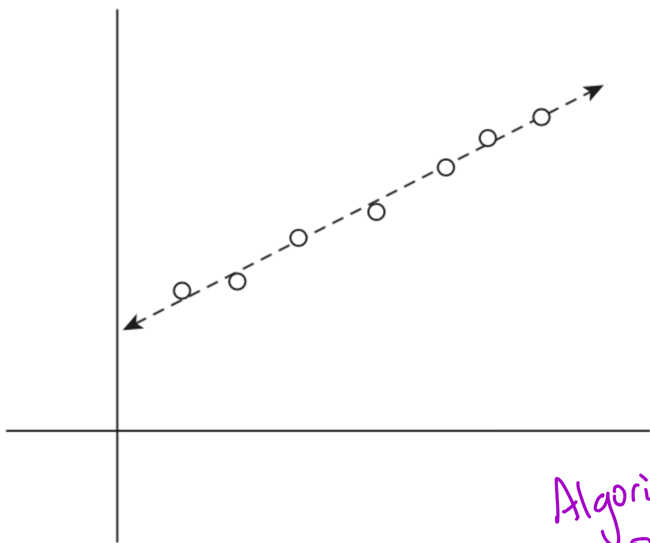


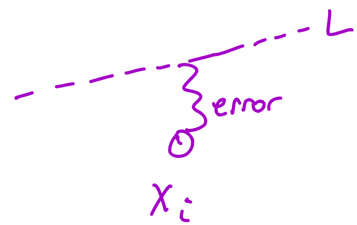
M[0]	M[1]	M[2]	M[3]	M[4]	M[5]	M[6]

Segmented Least Squares

Background: Least Squares

- **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 $\mathrm{error}(L, P) = \sum_i (y_i - ax_i - b)^2$





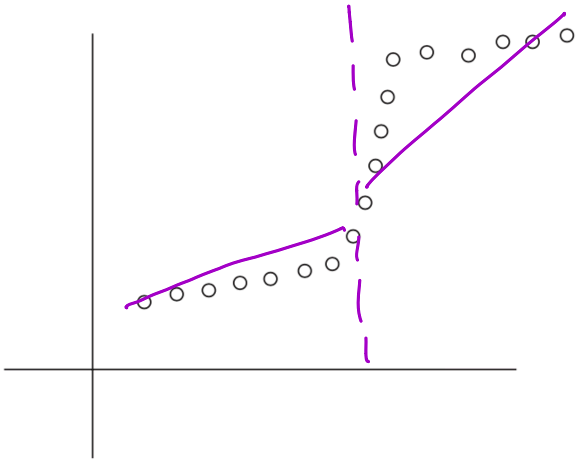
$$a = \frac{n\sum x_i y_i - (\sum x_i)(\sum y_i)}{n\sum x_i^2 - (\sum x_i)^2}$$

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

Algorithmically Boring!

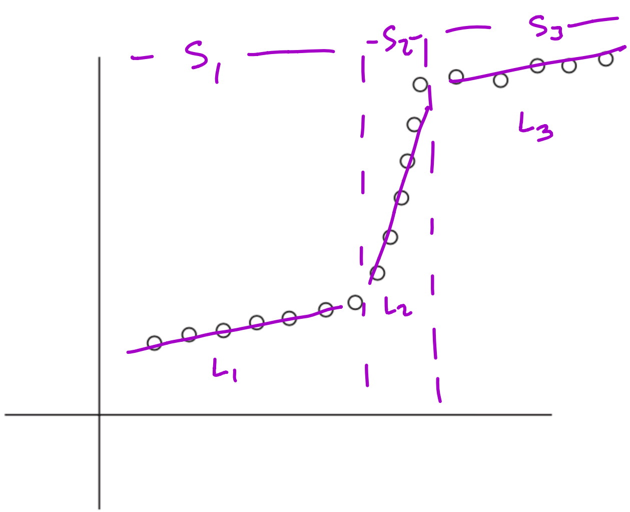
Segmented Least Squares

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



Segmented Least Squares

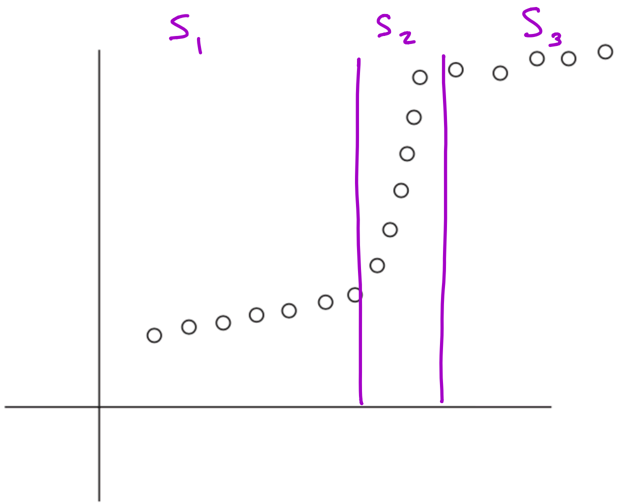
- **Input:** n data points $P = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, **cost parameter** $C > 0$
  - Assume $x_1 < x_2 < \cdots < x_n$
- **Output:** a partition into segments $S_1, S_2, \ldots, S_m$ (contiguous) and lines $L_1, L_2, \ldots, L_m$, minimizing "total cost"



$$\sum_{j=1}^{m} \mathrm{error}(S_j, L_j) + mC$$

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 $\{p_i, \dots, p_j\}$
  - Let $\varepsilon_{i,j} = \text{error}\left(L_{i,j}^*, \{p_i, \dots, p_j\}\right)$



SLS

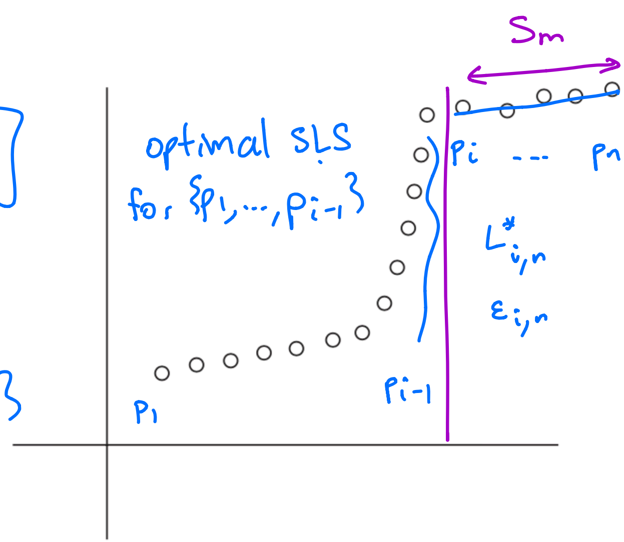
Let $L_{i,j}^*$ be the optimal line for $\{p_i, \dots, p_j\}$
Let $\varepsilon_{i,j} = error(L_{i,j}^*, \{p_i, \dots, p_j\})$

- Let O be the **optimal** solution (a partition into segments)

- n possible answers, could be any segment $\{p_i, \dots, p_n\}$ for $i = 1, 2, \dots, n$

What is the last segment?

- Case i: the last segment is $\{p_i, \dots, p_n\}$
  - optimal solutions is $\{p_i, \dots, p_n\}$ + [opt for $\{p_1, \dots, p_{i-1}\}$]



- Subproblems to consider:
  opt sol'n for all sets $\{p_1, \dots, p_i\}$

SLS

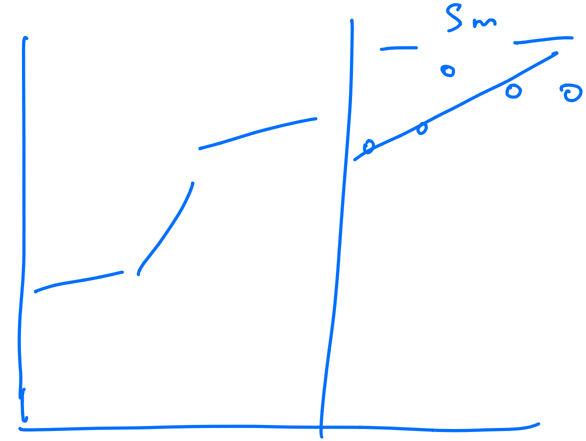
Let $L^*_{i,j}$ be the optimal line for $\{p_i, \ldots, p_j\}$
Let $\varepsilon_{i,j} = error(L^*_{i,j}, \{p_i, \ldots, p_j\})$

"total cost"

- 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 $L^*_{i,j}$ ∪ optimal sol. for $\{p_1, \ldots, p_{i-1}\}$
  - can use any $i \in \{1, \ldots, j\}$

- cost of the solution is

$$\varepsilon_{i,j} + C + \mathrm{OPT}(i-1)$$



SLS

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

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

Minimum over cost of all possible final segments

SLS: Take I

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

SLS: Take II ("Top-Down")

Only n+1 problems to solve

```
// 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]
```

$$M[n] \leftarrow \min_{1 \le i \le n} \varepsilon_{i,n} + C + \mathrm{FindOPT}(i-1)$$

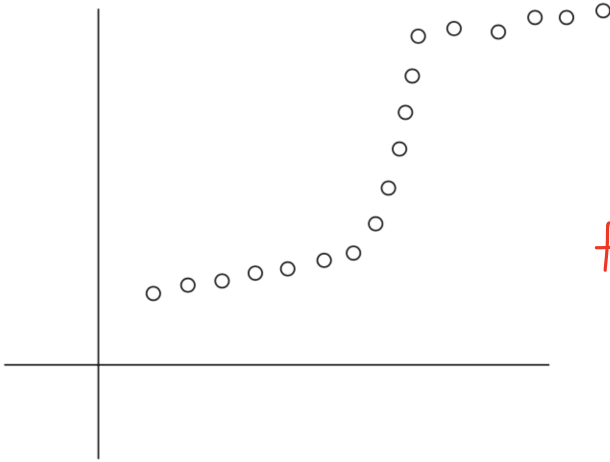
Each array entry gets filled once

× n recursive calls fill one elt

Total # of calls is $O(n^2)$

+ Time to compute $\varepsilon_{i,j}$ [Can be done in $O(n^2)$ time]

SLS: Take III ("Bottom-Up")



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

If minimum is achieved by $\varepsilon_{i,j} + C + OPT(i-1)$ the opt sol'n for $1,\dots,j$ uses $i,\dots,j$ as the final segment

M[0]	M[1]	M[2]	M[i-1] …	M[j]	…	M[n]
0	C	C	…		…	OPT(n)

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): //Loop through n-2 times
    M[j] ← min_{1≤i≤j} ε_{i,j} + C + M[i − 1]  // Can evaluate in O(n) time
  return M[n]
```

Total Time: $O(n^2)$

Finding Segments

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

- 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 $i \in \{1, \dots, j\}$

The final segment is $\{p_i, \dots, p_j\}$

$\Longleftrightarrow$ The cost is $\varepsilon_{i,j} + C + OPT(i-1)$

If $OPT(j) = \varepsilon_{i,j} + C + OPT(i-1)$

$\Longrightarrow \exists$ an optimal solution where $\{p_i, \dots, p_j\}$ is the final segment

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 ← argmax_{1≤i≤n} ε_{i,n} + C + M[i − 1]:
    return {i,…,n} + FindSol(M,i-1)
```

argmin

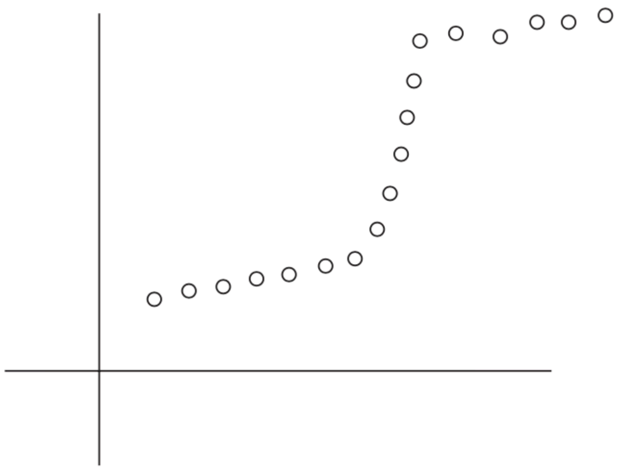
Find i s.t.

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

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 "total cost"

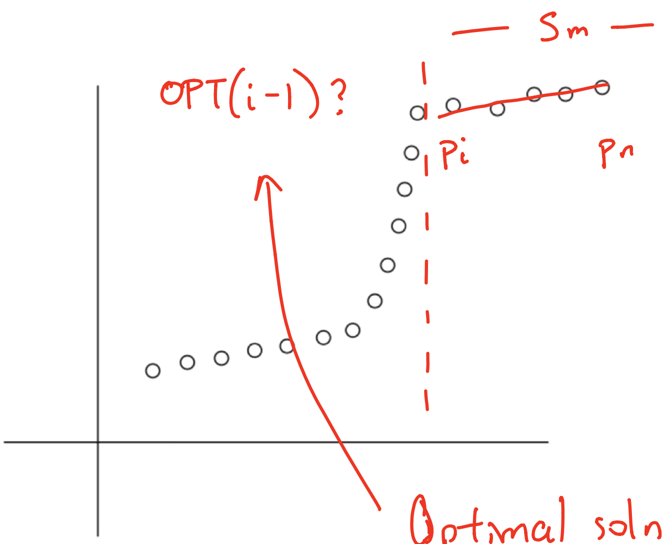


SLSv.2

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 O be the optimal solution
- What is the final segment?

Need to "record" how much of the budget is remaining



Optimal soln for {1, ..., i-1} could use k segments!

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

SLSv.2

Bigger set of subproblems

- Let $\mathrm{OPT}(j, \ell)$ be the optimal solution for points $\{1, \ldots, j\}$ using $\leq \ell$ segments
- **Case 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")

```
// 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]
```

$M[j,\ell] = OPT(j,\ell)$

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

	0 segments	1 segment	2 segments	- - -		k segments
	M[·,0]	M[·,1]	M[·,2]	M[·,3]	…	M[·,k]
∅ — M[0, ·]	0	0	0	0	0	0
$\{p_1\}$ — M[1, ·]	∞					
$\{p_1,p_2\}$ — M[2, ·]	∞					
M[3, ·]	∞			$OPT(j,\ell)$		
⋮ — …	∞					
$\{p_1,\dots,p_n\}$ — M[n, ·]	∞					$OPT(n,k)$

Fill 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): // Loop k-1 times
    for (j = 1,…,n): // Loop n-1 times
    M[j,ℓ] ← min_{1≤i≤j} ε_{i,j} + FindOPT(j − 1, ℓ − 1) // O(n) time
  return M[n,k]
```

Total Running Time: $O(kn^2)$

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:** break problem up by final segment
- **Version 2:** can solve SLS with a "hard cap" of k segments in time $O(n^2k)$ space $O(n^2 + nk)$
  - **New idea:** define subproblems using two variables
- Correctness follows from the recurrence
- Computational costs:
  - Running time $\approx$ total number of terms in all recurrences
  - Space $\approx$ total number of subproblems