CS3000: Algorithms & Data
Jonathan Ullman

Lecture 6:

- Dynamic Programming: Segmented Least Squares

Jan 27, 2020

Dynamic Programming Recap

- **Recipe:**

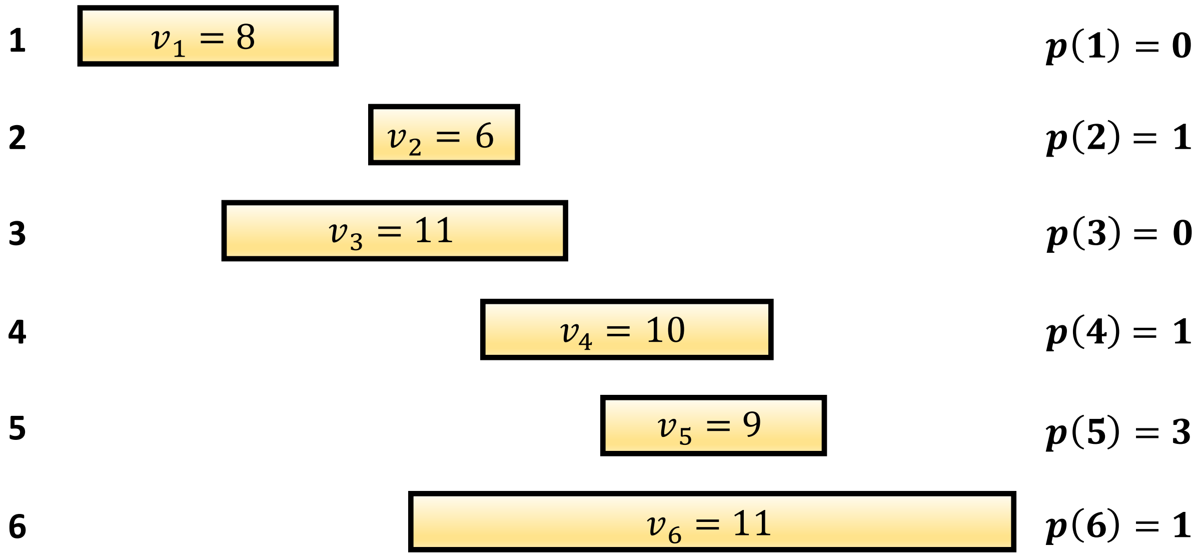
  (1) identify a set of **subproblems**

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

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

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

Dynamic Programming Recap



1 $v_1 = 8$ $p(1) = 0$

2 $v_2 = 6$ $p(2) = 1$

3 $v_3 = 11$ $p(3) = 0$

4 $v_4 = 10$ $p(4) = 1$

5 $v_5 = 9$ $p(5) = 3$

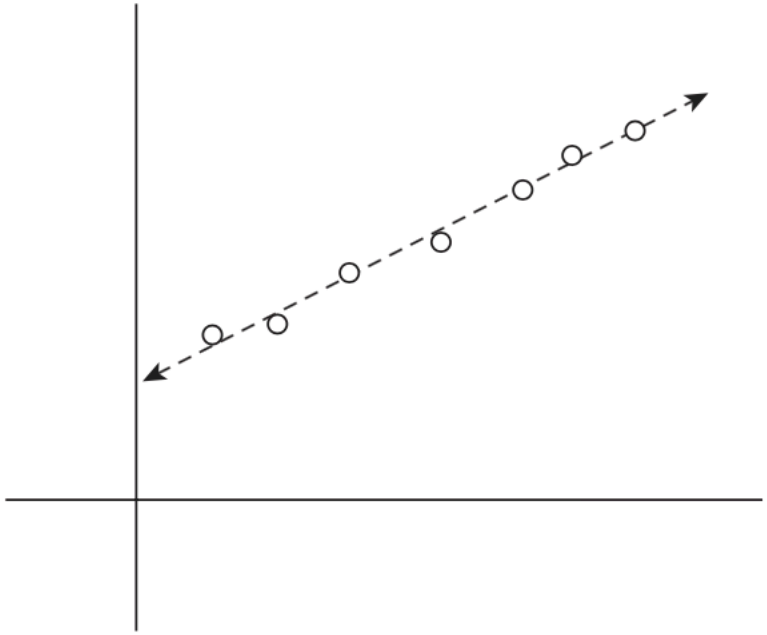
6 $v_6 = 11$ $p(6) = 1$

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

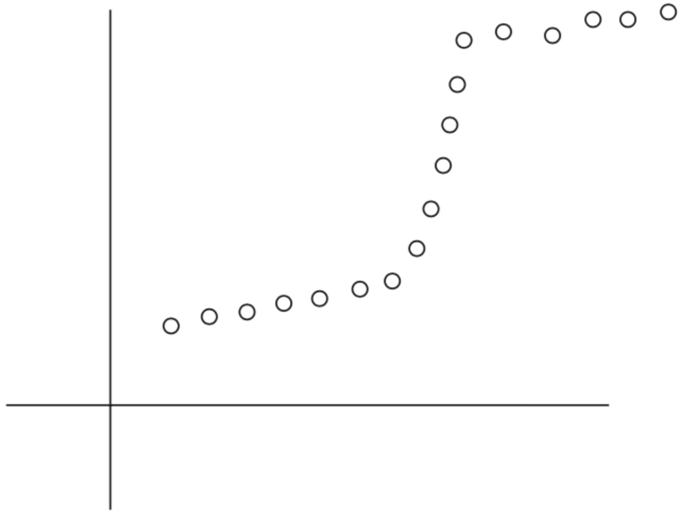


$$a = \frac{n\sum x_i y_i \ - (\sum x_i)(\sum y_i)}{n\sum x_i^2 - (\sum x_i)^2}$$

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

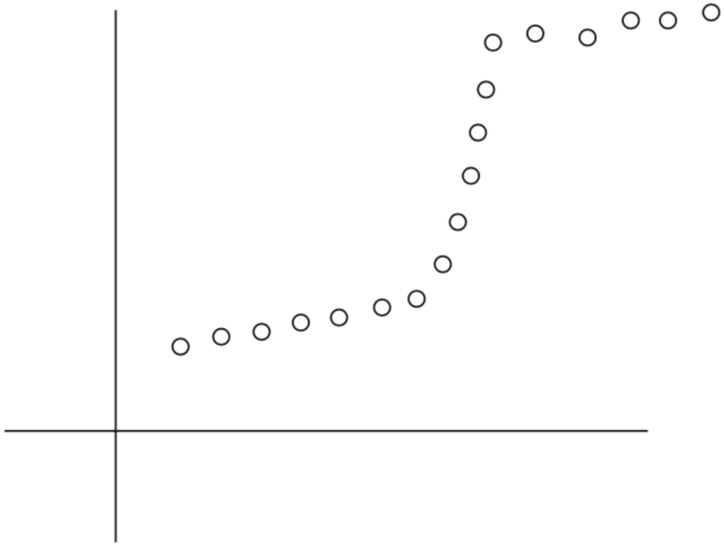
Segmented Least Squares

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



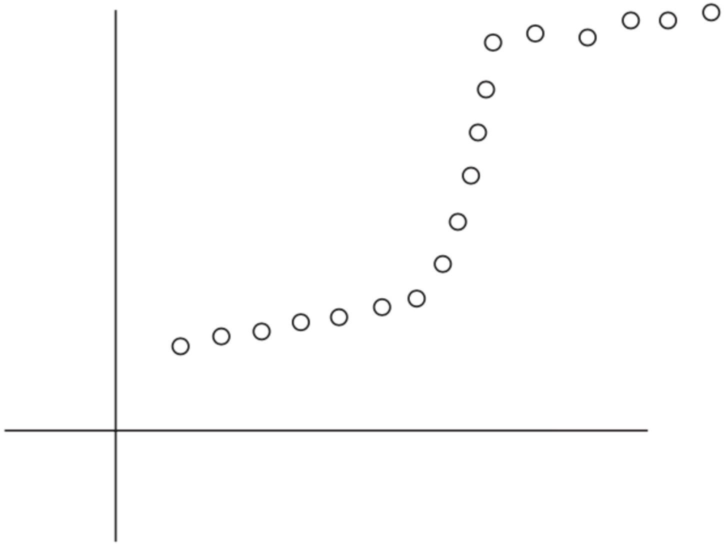
Segmented Least Squares

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



Segmented Least Squares

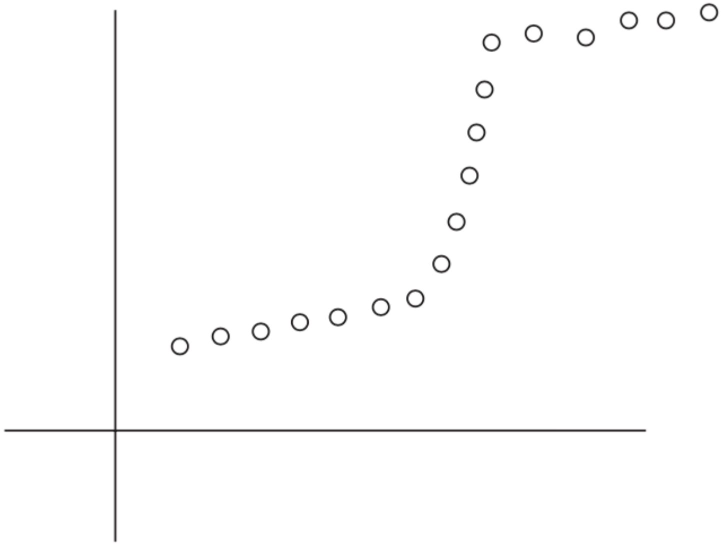
- **First observation:** for every segment S_j, L_j must be the (single) line of best fit for S_j
  - Let $L_{i,j}^*$ be the optimal line for $\{p_i, \dots, p_j\}$
  - Let $\varepsilon_{i,j} = \text{error}\left(L_{i,j}^*, \{p_i, \dots, p_j\}\right)$



SLS

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

- Let O be the **optimal** solution



SLS

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

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

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

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

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

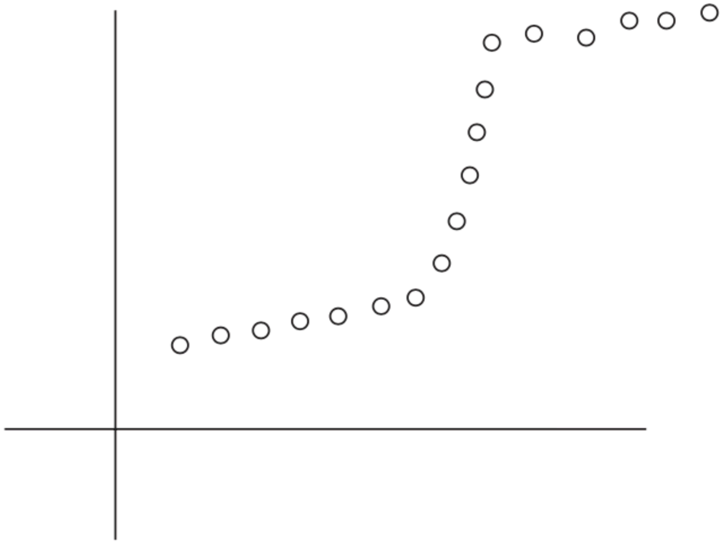
SLS: Take I

```
// 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")

```
// 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] ← $\min_{1 \le i \le n} \varepsilon_{i,n} + C + \text{FindOPT}(i-1)$
    return M[n]
```

SLS: Take III (“Bottom-Up”)



M[0]	M[1]	M[2]	…	M[i]	…	M[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):
    M[j] ← $\min_{1\leq i\leq j} \varepsilon_{i,j} + C + M[i-1]$
  return M[n]
```

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

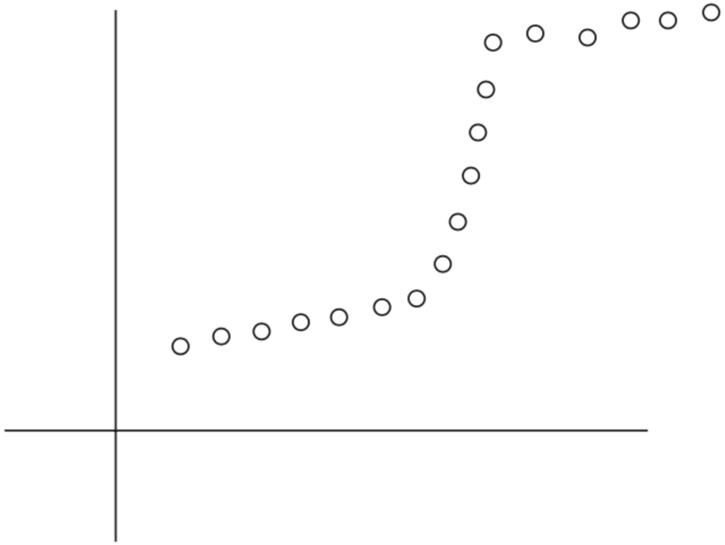
Finding Segments

```
// 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 $\leftarrow \mathrm{argmax}_{1 \le i \le n}\, \varepsilon_{i,n} + C + M[i-1]$:
    return {i,…,n} + FindSol(M,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 $\varepsilon_{i,j} = error\left(L^*_{i,j}, \{p_i, \ldots, p_j\}\right)$

- Let O be the optimal solution

SLSv.2

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

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

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

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,$\ell$] ← 0, M[j,0] ← $\infty$
FindOPT(n,k):
  if (M[n,k] is not empty): return M[n,k]
  else:
    M[n,k] ← $\min_{1 \le i \le n} \varepsilon_{i,n} + \text{FindOPT}(i-1, k-1)$
    return M[n,k]
```

SLSv.2: Take III (“Bottom-Up”)

	M[·,0]	M[·,1]	M[·,2]	M[·,3]	…	M[·,k]
M[0, ·]						
M[1, ·]						
M[2, ·]						
M[3, ·]						
…						
M[n, ·]						

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] ← $\min_{1 \le i \le j} \varepsilon_{i,j} + \mathbf{FindOPT}(j - 1, \ell - 1)$
  return M[n,k]
```

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 $\leftarrow \mathrm{argmax}_{1 \le i \le n}\, \varepsilon_{i,n} + M[i-1, k-1]$:
    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 ≈ total number of terms in all recurrences
  - Space ≈ total number of subproblems