Lecture 8:
• Dynamic Programming: Lines of best fit, Knapsack

Feb 2, 2018
Logistics

- Midterm I Feb 13th (Tuesday after next)
  - Divide-and-Conquer
  - Dynamic Programming
  - Asymptotic notation
  - Proofs by induction
  - Recurrences /
    - Master Theorem

- One page of notes
Recap

• Define the subproblems
• Write the recurrence
• Fill in the table of solutions

\[
\begin{align*}
1 &: v_1 = 8 & p(1) = 0 \\
2 &: \quad \quad v_2 = 6 & p(2) = 0 \\
3 &: \quad \quad v_3 = 11 & p(3) = 0 \\
4 &: \quad \quad \quad v_4 = 10 & p(4) = 2 \\
5 &: \quad \quad \quad v_5 = 9 & p(5) = 3 \\
6 &: \quad \quad \quad \quad v_6 = 15 & p(6) = 1 \\
\end{align*}
\]

• Find the opt. schedule by dynamic programming

\[\text{OPT} = 23 \quad \text{s}_{\text{OPT}} = \{1, 6\}\]
Recap

Step 0: Sort the intervals by end time.
Step 1: Define the subproblems.
Step 2: Write recurrence
\[
\text{OPT}(i) = \max \left\{ \text{OPT}(i-1), v_i + \text{OPT}(p(i)) \right\}
\]
Step 3: Build a table of solutions

<table>
<thead>
<tr>
<th></th>
<th>$v_1 = 8$</th>
<th>$v_2 = 6$</th>
<th>$v_3 = 11$</th>
<th>$v_4 = 10$</th>
<th>$v_5 = 9$</th>
<th>$v_6 = 15$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Subproblems:
\[
\text{OPT}(i) = \text{the value of the optimal schedule for } \{i_1, ..., i_k\} \quad i = 0, 1, ..., 6
\]
Today

• Dynamic programming
  • More Practice: lines of best fit, knapsack
  • More Tricks: selecting a suffix, adding variables
Lines of best fit
Warmup: Line of Best Fit

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

**Optimal Solution**

\[
\begin{align*}
a &= \frac{n \sum x_i y_i - (\sum x_i)(\sum y_i)}{n \sum x_i^2 - (\sum x_i)^2} \\
b &= \frac{\sum y_i - a \sum x_i}{n}
\end{align*}
\]

Can find optimal solution in \( O(n) \) time
Lines of Best Fit

• **Input:** \( n \) data points \( P = \{(x_1, y_1), \ldots, (x_n, y_n)\} \)

• What if the data does not look like a line?

\[
\text{one line has high error} \\
\text{three lines have \( \approx 0 \) error}
\]

Want to divide points into intervals, find one line per interval.

Problem does not make sense w/o some restriction on \# of lines.
**Lines of Best Fit**

> "cost of adding a new segment/line"

- **Input:** \( n \) data points \( P = \{(x_1, y_1), \ldots, (x_n, y_n)\} \),
  - cost parameter \( C > 0 \)
  - Assume \( x_1 < x_2 < \cdots < x_n \); write \( p_i = (x_i, y_i) \)
- **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"

\[
\text{cost}(S_1, \ldots, S_m, L_1, \ldots, L_m, P) = C_m + \sum_{j=1}^{m} \text{error}(L_j, S_j)
\]
Lines of Best Fit

Our algorithm “only” needs to find the segments $S_1, \ldots, S_m$

• First observation: for every segment $S_j$, $L_j$ will 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 $e_{i,j} = error(L^*_{i,j}, \{p_i, \ldots, p_j\})$

For each $i,j$, can find $L^*_{i,j}, e_{i,j}$ in $O(n)$ time

• $O(n^3)$ time total
• Can improve to $O(n^2)$ time
**Lines of Best Fit**

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

Let \( e_{i,j} = \text{error}(L_{i,j}^*, \{p_i, \ldots, p_j\}) \)

- Let \( O \) be the **optimal** solution
  - \( O \) is a set of segments \( S_1, \ldots, S_m \)
  - It has **some** final segment \( S_m = \{p_i, \ldots, p_n\} \)

  - \( O \) must use \( L_{i,n}^* \) for the last segment
  - \( O \) must contain the optimal solution for points \( \{p_1, \ldots, p_{i-1}\} \)

- \( \text{cost of } O = e_{i,n} + C + \text{cost of } \text{opt for } l_j, \ldots, i-1 \)
**Lines of Best Fit**

- Let $O$ be the **optimal** solution
- Let $\text{OPT}(n)$ be the cost of the optimal solution for points $p_1, ..., p_n$

\[
\text{OPT}(n) = \min_{1 \leq i \leq n} \left\{ e_{i,n} + C + \text{OPT}(i-1) \right\}
\]

$\text{OPT}(0) = 0$
$\text{OPT}(1) = C$
$\text{OPT}(2) = C$

- Only $n+1$ subproblems
- Recurrence for the subproblems
- Can evaluate subproblems "in order"
**Lines of Best Fit**

Let $L_{i,j}^*$ be the optimal line for $\{p_i, \ldots, p_j\}$

Let $e_{i,j} = \text{error}(L_{i,j}^*, \{p_i, \ldots, p_j\})$

---

Let $P = \{p_1, \ldots, p_n\}$ be the points

$C$ is the cost

$\text{LoBF}(n)$:

If $n = 0$: return 0

Else: return $\min_{1 \leq i \leq n} e_{i,n} + C + \text{LoBF}(i - 1)$
Let $P = \{p_1, \ldots, p_n\}$ be the points
Let $L[1, \ldots, n]$ be an array (initially empty)

MLoBF($i, \ldots$):

If $n = 0$: return 0
Else If ($L[n]$ not empty): return $L[n]
Else:

\[ M[n] \leftarrow \min_{1 \leq i \leq n} e_{i,n} + C + \text{LoBF}(i - 1) \]

return $M[n]$

- Every $1$ mate = recursive calls to fill in one subproblem
- There are $n$ subproblems

\[ \Rightarrow \text{Total # of calls is} \leq n \times n = O(n^2) \]

Total Time: $O(n^2)$
Lines of Best Fit

Let $O$ be the optimal solution

Let $OPT(i)$ be the cost of the optimal solution for points $p_1, ..., p_i$

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

$$= \min \left\{ e_{1,n} + C + OPT(0), \ldots, e_{n,n} + C + OPT(n-1) \right\}$$

• How do we find the actual segments?
  - One of the $n$ terms in the minimum is "best"
  - Call that $i$
  - Best final segment is $\{p_i, \ldots, p_n\}$
Let $P = \{p_1, \ldots, p_n\}$ be the points
Let $M[1, \ldots, n]$ be an array (initially empty)

FindLoBF($n$):

find $i \in \{1, \ldots, n\}$ s.t. $\sum_{i=1}^{n} e_{i,n} + C + M[i-1]

return ($\{p_i, \ldots, p_n\} + \text{FindLoBF}(i - 1)$)

$O(n^2)$ time to find the optimal set of segments once the table is filled.
**Bottom Up Approach**

- $OPT(i)$ only depends on $OPT(0), \ldots, OPT(i-1)$
Lines of Best Fit: Take II

• **Input:** $n$ data points $P = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, a maximum number of segments $k$
  - Assume $x_1 < x_2 < \cdots < x_n$; write $p_i = (x_i, y_i)$

• **Output:** a partition of $P$ into $k$ segments $S_1, \ldots, S_k$, lines $L_1, \ldots, L_k$, minimizing “cost”
Recap

• Can find the lines of best fit in time $O(n^2)$
  • Have to be careful about precomputing $e_{i,j}$

• New idea: find the best final segment
  • Compare to scheduling where we simply decided whether the final solution was in or out of the solution

• Many problems have the flavor of splitting maps into segments
• Let $O$ be the \textbf{optimal} solution
• Let $OPT(n)$ be the \textbf{cost} of the optimal solution for points $p_1, \ldots, p_n$.

\[ OPT(n) = \min_{1 \leq i \leq n} \{ e_i, n + OPT(i-1) \} \]

$O$ contains some segments $S_{i_1}, \ldots, S_{i_k}$.

$S_k$ is some segment $\{p_{i_1}, \ldots, p_{i_n}\}$.

(best line for $S_k$)

errors $e_{i, n}$

uses exactly $k$ segments

uses exactly $k$ segments

uses too many segments

$OPT(i-1)$
Key Idea: Adding Variables

• Let $O$ be the **optimal** solution

• Let $OPT(n,k)$ be the **cost** of the optimal solution for points $p_1, ..., p_n$ using $k$ segments

\[
OPT(n,k) = \min_{1 \leq i \leq n} \left\{ e_{i,n} + \text{OPT}(i-1,k-1) \right\}
\]

- $O$ uses $S_1, ..., S_k$
- $S_k$ is some $\{p_i, ..., p_n\}$

Diagram: Optimal solution with $k$ segments for points $p_1, ..., p_n$.
Let $L_{i,j}^*$ be the optimal line for \{p_i, \ldots, p_j\}
Let $e_{i,j} = \text{error}(L_{i,j}^*, \{p_i, \ldots, p_j\})$

Let $P = \{p_1, \ldots, p_n\}$ be the points
Let $M[1: n, 1: k]$ be an array (initially empty)

\begin{align*}
\text{MLoBF}(n, k): & \\
& \quad \text{If } (n = 0): \text{ return } 0; \text{ Else If } (n > 0, k = 0): \text{ return } \infty \\
& \quad \text{Else If } (M[n, k] \text{ not empty}): \text{ return } M[n, k] \\
& \quad \text{Else:} \\
& \quad \quad \text{fill one entry } M[n, k] \leftarrow \min_{1 \leq i \leq n} e_{i,n} + \text{LoBF}(i - 1, k - 1) \\
& \quad \quad \text{return } M[n, k]
\end{align*}

$\sim nk$ subproblems
$\sim n$ calls per problem

Total time is $O(n^2k)$
Bottom Up Approach

\[ \text{OPT}(i, s) = \min_{1 \leq j \leq n} \left( e_{j, i} + \text{OPT}(j-1, s-1) \right) \]

- \( \text{OPT}(i, s) \) only depends on \( \text{OPT}(j, s-1) \) for \( j \leq i \)
Recap

• Can find the $k$ lines of best fit in time $O(n^2k)$
  • Note: problem only makes sense for $1 \leq k < \frac{n}{2}$

• New idea: introduce a new variable
  • Use a larger set of subproblems
  • Gets easier with practice