6.3 Segmented Least Squares
Least squares.

Foundational problem in statistic and numerical analysis.
Given $n$ points in the plane: $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$.

Q. How to find a line $y = ax + b$ that fits these points?
Least squares.

Foundational problem in statistic and numerical analysis.
Given \( n \) points in the plane: \((x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\).

Q. How to find a line \( y = ax + b \) that fits these points?

Find a line \( y = ax + b \) that minimizes the sum of the squared error:

\[
SSE = \sum_{i=1}^{n} (y_i - ax_i - b)^2
\]

Solution. Calculus \( \Rightarrow \) min error is achieved when

\[
a = \frac{n \sum_i x_i y_i - (\sum_i x_i) (\sum_i y_i)}{n \sum_i x_i^2 - (\sum_i x_i)^2}, \quad b = \frac{\sum_i y_i - a \sum_i x_i}{n}
\]
Segmented least squares.

Points lie roughly on a sequence of several line segments. Given \( n \) points in the plane \((x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\) with \( x_1 < x_2 < \ldots < x_n \), find a sequence of line segments that minimizes \( f(x) \).

Q. What's a reasonable choice for \( f(x) \) to balance accuracy and parsimony?
Segmented least squares.

Points lie roughly on a sequence of several line segments. Given $n$ points in the plane $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ with $x_1 < x_2 < \ldots < x_n$, find a sequence of line segments that minimizes:

- the sum of the sums of the squared errors $e$ in each segment
- the number of lines $L$

Tradeoff function: $e + cL$, for some constant $c > 0$. 
Dynamic Programming: Multiway Choice

Notation.

\[ \text{OPT}(j) = \text{minimum cost for points } p_1, p_2, \ldots, p_j. \]

\[ e(i, j) = \text{minimum sum of squared errors for points } p_i, p_{i+1}, \ldots, p_j. \]

Reason backward, computing \( \text{OPT}(j) \) using subproblems

Q. How can value of \( \text{OPT}(j) \) be expressed based on subproblems? (1 min)

Q. What are the options here?
Dynamic Programming: Multiway Choice

Notation.

\[ \text{OPT}(j) = \text{minimum cost for points } p_1, p_2, \ldots, p_j. \]
\[ e(i, j) = \text{minimum sum of squared errors for points } p_i, p_{i+1}, \ldots, p_j. \]

Reason backward, computing \( \text{OPT}(j) \) using subproblems

Q. How can value of \( \text{OPT}(j) \) be expressed based on subproblems? (1 min)

Q. What are the options here?

A. The start \( i \) of the last segment \( (1 \leq i \leq j) \).
   Last segment uses points \( p_i, p_{i+1}, \ldots, p_j \) for some \( i \).
   Cost = \( e(i, j) + c + \text{OPT}(i-1) \).
Dynamic Programming: Multiway Choice

OPT(j) call graph
Dynamic Programming: Multiway Choice

Notation.

\( \text{OPT}(j) = \text{minimum cost for points } p_1, p_2, \ldots, p_j. \)

\( \text{e}(i, j) = \text{minimum sum of squared errors for points } p_i, p_{i+1}, \ldots, p_j. \)

Reason backward, computing \( \text{OPT}(j) \) using subproblems

Q. How can value of \( \text{OPT}(j) \) be expressed based on subproblems? (1 min)

Q. What are the options here?

A. The start \( i \) of the last segment.

Last segment uses points \( p_i, p_{i+1}, \ldots, p_j \) for some \( i \).

Cost = \( \text{e}(i, j) + c + \text{OPT}(i-1) \).

\[
\text{OPT}(j) = \begin{cases} 
0 & \text{if } j = 0 \\
\min_{1 \leq i \leq j} \left\{ e(i, j) + c + \text{OPT}(i-1) \right\} & \text{otherwise}
\end{cases}
\]

Choose \( i \in [1, j] \) Cost of this choice
Segmented Least Squares: Algorithm

**INPUT**: n, p₁,…,pₙ, c

```
Segmented-Least-Squares() {
    M[0] = 0
    for j = 1 to n
        for i = 1 to j
            compute the least square error eᵢⱼ for the segment pᵢ,…, pⱼ

    for j = 1 to n
        M[j] = \min_{1 \leq i \leq j} (eᵢⱼ + c + M[i-1])

    return M[n]
}
```

Q. What is the running time? (1 min)
Segmented Least Squares: Algorithm

**INPUT:** $n$, $p_1, \ldots, p_N$, $c$

```
Segmented-Least-Squares() {
    M[0] = 0
    for $j = 1$ to $n$
        for $i = 1$ to $j$
            compute the least square error $e_{ij}$ for the segment $p_i, \ldots, p_j$
        for $j = 1$ to $n$
            $M[j] = \min_{1 \leq i \leq j} (e_{ij} + c + M[i-1])$
    return $M[n]$
}
```

**Q.** What is the running time? (1 min)

**A.** $O(n^3)$. (but can be improved to $O(n^2)$ by pre-computing various statistics)

Bottleneck = initialization: computing $e(i, j)$ for $O(n^2)$ pairs, $O(n)$ per pair using previous formula.
Example exam exercise

6.5 (Chinese) word segmentation problem
Given a string \( x_1x_2...x_n \), give an efficient algorithm to split \( x \) into words (substrings) such that the sum of the quality of these words is maximized.

Example. “mogenzeslapen”:

<table>
<thead>
<tr>
<th>word</th>
<th>quality</th>
</tr>
</thead>
<tbody>
<tr>
<td>mogen</td>
<td>4</td>
</tr>
<tr>
<td>enz</td>
<td>1</td>
</tr>
<tr>
<td>gen</td>
<td>2</td>
</tr>
<tr>
<td>sla</td>
<td>2</td>
</tr>
<tr>
<td>pen</td>
<td>2</td>
</tr>
<tr>
<td>slapen</td>
<td>5</td>
</tr>
<tr>
<td>ze</td>
<td>1</td>
</tr>
<tr>
<td>en</td>
<td>1</td>
</tr>
</tbody>
</table>

rules for this example:
• number of letters-1
• punish uncommon words