中山大學
SUN YAT-SEN UNIVERSITY

Lecture 7
Dynamic Programming
Part III

Algorithm Design

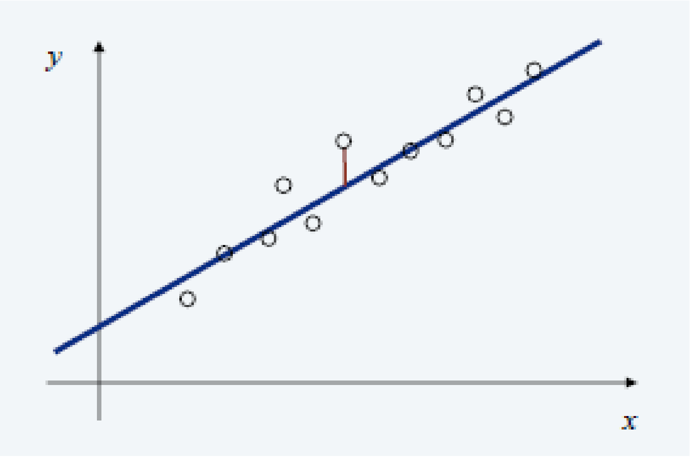
zhangzizhen@gmail.com

QQ group: 117282780

Segmented Least Squares

- Least squares. Foundational problem in statistics.
- Given *n* points in the plane: *(x1, y1), (x2, y2) , …, (xn, yn).*
- Find a line *y* = *ax* + *b* that minimizes the sum of the squared error:

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

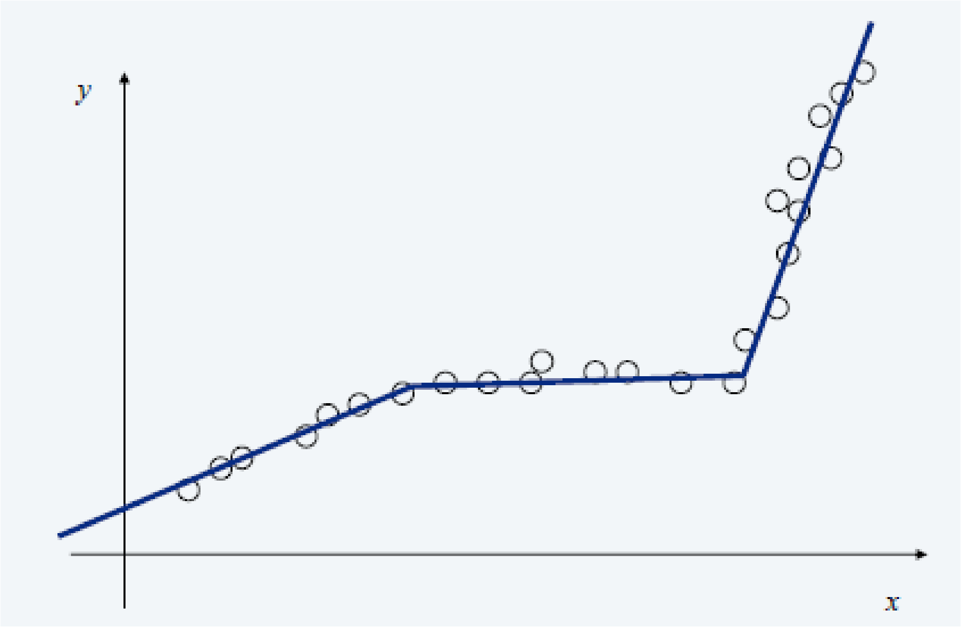


- Solution. Calculus ⇒ min error is achieved when

$$a = \frac{n \sum_i x_i y_i - (\sum_i x_i)(\sum_i y_i)}{n \sum_i x_i^2 - (\sum_i x_i)^2}, \quad b = \frac{\sum_i y_i - a \sum_i x_i}{n}$$

Segmented Least Squares

- Segmented least squares: Points lie roughly on a sequence of several line segments.
- Given *n* points in the plane: $(x_1, y_1), (x_2, y_2)$, …, (x_n, y_n) *with* $x_1 < x_2 < ... < x_n$, find a sequence of lines that minimizes $f(x)$.



Segmented Least Squares

- Given n points in the plane: $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ *with* $x_1 < x_2 < \ldots < x_n$, *and a* constant $c > 0$, find a sequence of lines that minimizes $f(x) = E + c\,L$:
- *E = the sum of the sums of the squared errors in each segment.*
- *L = the number of lines.*

Segmented Least Squares

- Notation:
  - $OPT(j)$ = minimum cost for points $p_1, p_2, \ldots, p_j$.
  - $e(i, j)$ = minimum sum of squares for points $p_i, p_{i+1}, \ldots, p_j$.
- To compute $OPT(j)$:
  - Last segment uses points $p_i, p_{i+1}, \ldots, p_j$ *for some i.*
  - Cost = $e(i, j) + c + OPT(i - 1)$.

$$OPT(j) = \begin{cases} 0 & \text{if } j = 0 \\ \min\limits_{1 \le i \le j} \{\ e(i,j) + c + OPT(i-1) \} & \text{otherwise} \end{cases}$$

Segmented Least Squares

SEGMENTED-LEAST-SQUARES $(n,\ p_1,\ \ldots,\ p_n\ ,\ c)$

FOR j = 1 TO n

FOR i = 1 TO j

Compute the least squares $e(i, j)$ for the segment $p_i, p_{i+1}, \ldots, p_j$.

$M[0] \leftarrow 0$.

FOR j = 1 TO n

$M[j] \leftarrow \min_{1 \le i \le j} \{ e_{ij} + c + M[i-1] \}$.

RETURN $M[n]$.

Segmented Least Squares

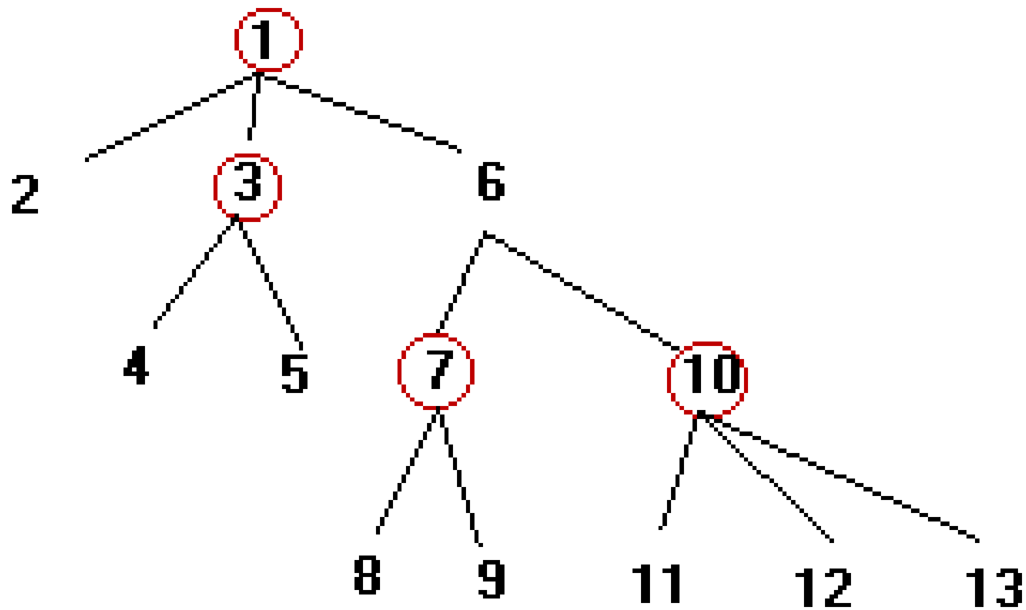
- The dynamic programming algorithm solves the segmented least squares problem in $O(n^3)$ *time and* $O(n^2)$ space.
- Bottleneck: computing $e(i, j)$ for $O(n^2)$ pairs.
- $O(n)$ per pair using formula.

$$a = \frac{n \sum_i x_i y_i - (\sum_i x_i)(\sum_i y_i)}{n \sum_i x_i^2 - (\sum_i x_i)^2}, \quad b = \frac{\sum_i y_i - a \sum_i x_i}{n}$$

- Can be improved to $O(n^2)$ time and $O(n)$ space by precomputing various statistics.

Minimum vertex cover of a tree

- 问题描述：给出一个n个结点的树，要求选出其中的一些顶点，使得对于树中的每条边(u, v)，u和v至少有一个被选中. 请给出选中顶点数最少的方案.



Minimum vertex cover of a tree

- 在树结构中，每个结点都是某棵子树的根
- 把n个结点分别用0~n-1编号
- 用ans[i][0]表示：在不选择结点i的情况下，以i为根的子树，最少需要选择的点数；
- 用ans[i][1]表示：在选择结点i的情况下，以i为根的子树，最少需要选择的点数.

Minimum vertex cover of a tree

- ans[i][0]表示：在不选择结点i的情况下，以i为根的子树，最少需要选择的点数；
- ans[i][1]表示：在选择结点i的情况下，以i为根的子树，最少需要选择的点数.
- 当i是叶子时，ans[i][0] = 0, ans[i][1] = 1；
- 否则，
  - ans[i][0] = Σans[j][1] (对于i的所有子结点j)
  - ans[i][1] = 1+Σmin(ans[j][0], ans[j][1]) (对于i的所有子结点j)

Traveling Salesman Problem

- Given *n* cities and the distances d_{ij} between any two of them, we wish to find the shortest tour going through all cities and back to the starting city. Generally, the TSP is given as a graph $G=(V,D)$ where $V=\{1,2, . . ., n\}$ is the set of cities, and *D* is the adjacency distance matrix, with $\forall i,j \in V$, $i \neq j$, $d_{ij}>0$, the problem is to find the tour with minimal distance weight, that starting at city 1 goes through all *n* cities and returns to city 1.

Traveling Salesman Problem

- The TSP is a well known NP-hard problem.
- There are n! feasible solutions. Enumerate them may take O(n!) time.
- What is the appropriate subproblem for the TSP?
  - Suppose we have started at city 1 as required, have visited a few cities, and are now in city *j*. What information do we need in order to extend this partial tour?
  - We need to know *j*, since this will determine which cities are most convenient to visit next.
  - We also need to know all the cities visited so far, so that we don't repeat any of them.

Traveling Salesman Problem

- For a subset of cities $S \subseteq \{1,2,\ldots,n\}$ that includes *1*, and $j \in S$, let *C(S,j)* be the length of the shortest path visiting each node in *S* exactly once, starting at *1* and ending at *j*.
- How to express *C(S,j)* in terms of smaller sub-problems.
- We need to start at 1 and end at j; what should we pick as the second-to-last city? It has to be some $i \in S$, so the overall path length is the distance from 1 to *i*, namely, $C(S-\{j\}, i)$, plus the length of the final edge, d_{ij}.
- We pick the best *i*, then

$$C(S,j) = \min_{i \in S : i \neq j} C(S - \{j\}, i) + d_{ij}$$

Traveling Salesman Problem

- There are at most 2^n*n subproblems.
- Each one takes linear time to solve.
- The total time complexity is $O(2^n*n^2)$.
- The sub-problems are ordered by |S|. (Use a queue to extend *S*))

C({1},1) = 0
for s = 2 to n:
  for all subsets S ⊆ {1,2,…,n} of size s and containing 1:
    C(S,1) = ∞
    for all $j \in S, j \neq 1$:
      C(S, j) = $min\{C(S-\{j\},i)+d_{ij} : i \in S, i \neq j\}$
return $min_j C(\{1,\ldots,n\},j)+d_{j1}$

Traveling Salesman Problem

- How to represent the set *S*?
- Usually, we can Represent a set of *n* elements as an *n* bit numbers.
- Example: (n=5)

$S=\Phi$ → $(00000)_2$ → 0

$S=\{0\}$ → $(00001)_2$ → 1

$S=\{1,3\}$ → $(01010)_2$ → 10

$S=\{0,1,2,3,4\}$→ $(11111)_2$ → 15

Traveling Salesman Problem

- Check if element *i* is present in set *S*
- Find the resulting set when we add *i* to set *S*
- Iterating through all the subsets of size <= n

Thank you!