CS 310

“You can't cross the sea merely by standing and staring at the water.”

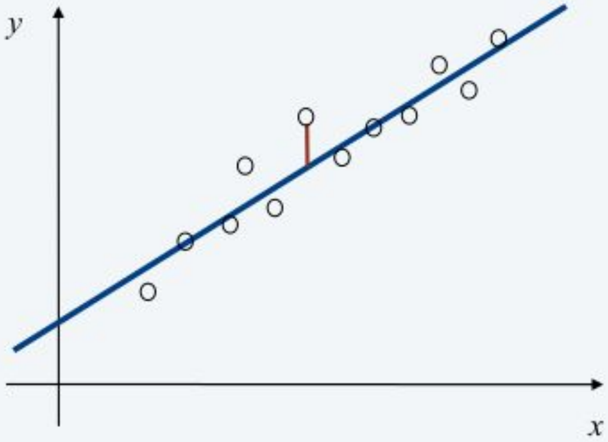
- *Rabindranath Tagore*

Segmented least squares

Least squares. Foundational problem in statistics.

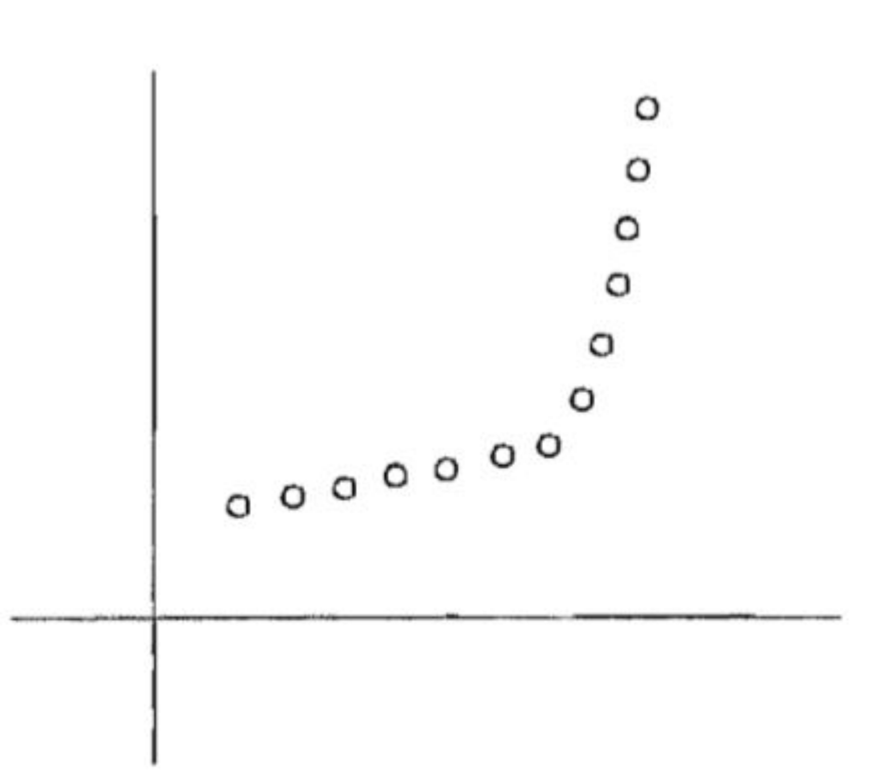
- Given n points in the plane: $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$.
- Find a line $y = ax + b$ that minimizes the sum of the squared error:

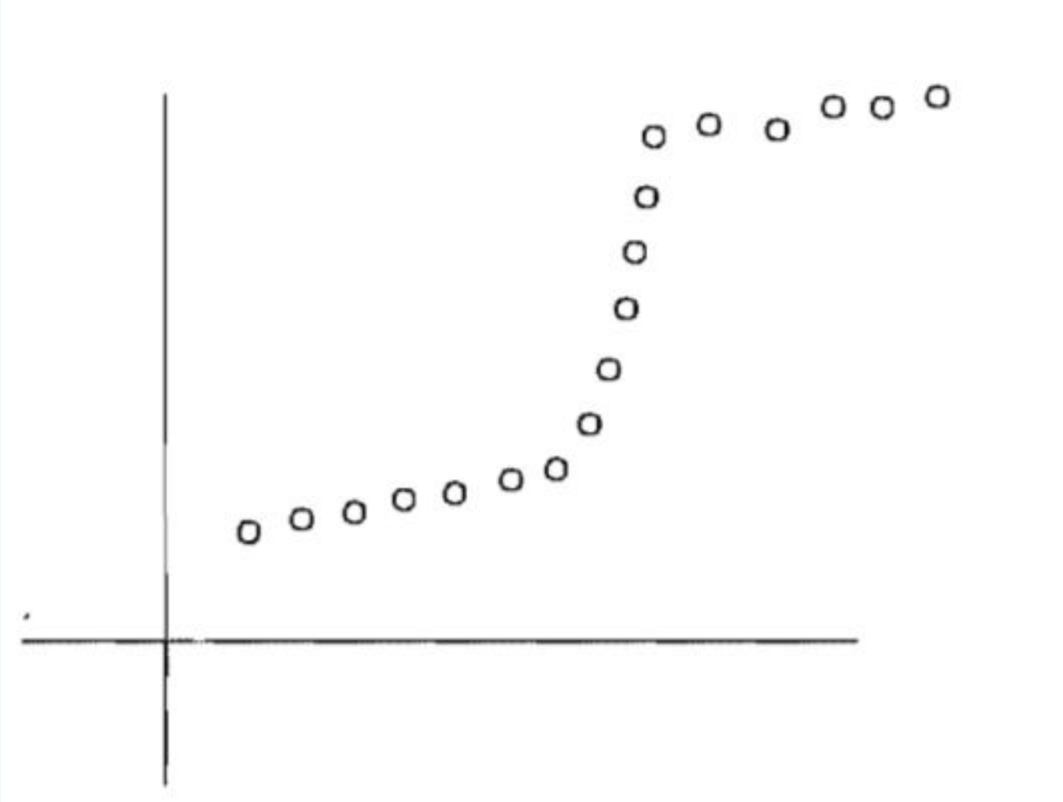
$$SSE = \sum_{i=1}^{n} (y_i - ax_i - b)^2$$



Best fit

Change detection





What should we optimize?

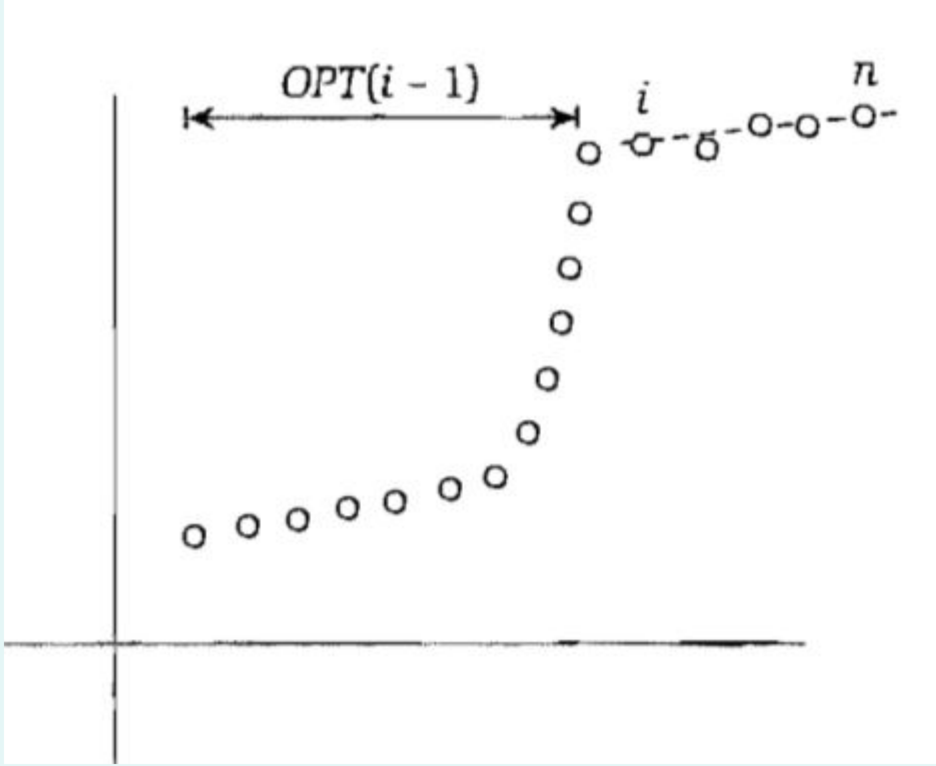
A set of lines that minimizes the error?

- Set of points P
- Partition P in some number of segments
  (segment is defined as contiguous set of x-coordinates)
- Define a penalty of partition.

Segmented least squares

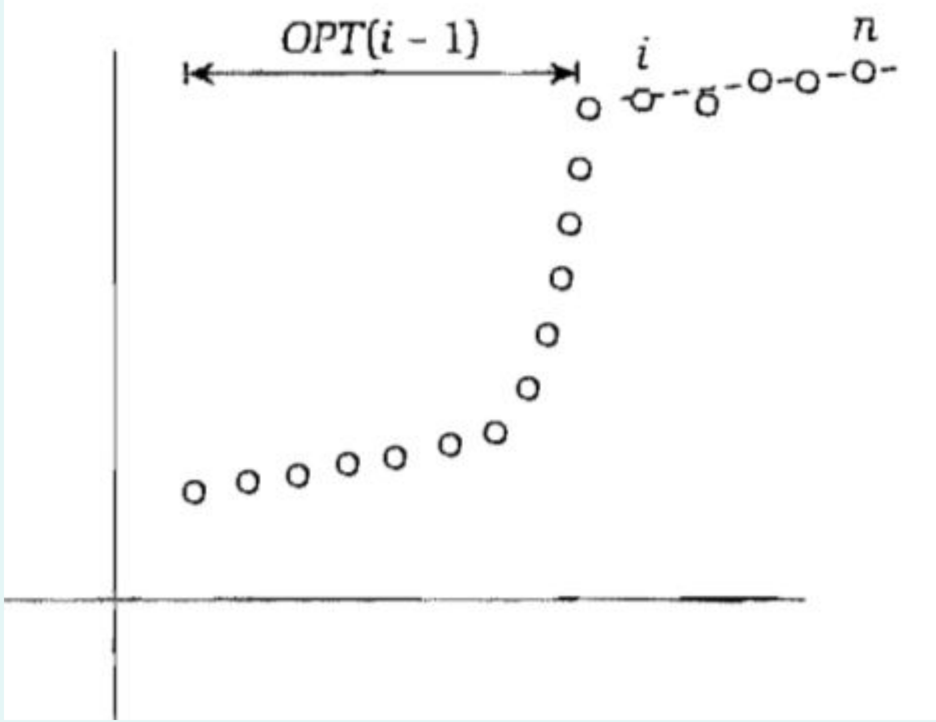
Given n points in the plane: $(x_1, y_1), (x_2, y_2)$, …, (x_n, y_n) with $x_1 < x_2 < ... < x_n$ and a constant $c > 0$, find a sequence of lines that minimizes $f(x) = E + c\,L$:

- E = the sum of the sums of the squared errors in each segment.
- L = the number of lines.


OPT(i - 1)
i
n

Notation.

- $OPT(j)$ = minimum cost for points $p_1, p_2, \ldots, p_j$.
- $e(i, j)$ = minimum sum of squares for points $p_i, p_{i+1}, \ldots, p_j$.



(6.6) *If the last segment of the optimal partition is $p_i, \ldots, p_n$, then the value of the optimal solution is* $\text{OPT}(n) = e_{i,n} + C + \text{OPT}(i-1)$.

Segmented least squares

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

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

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

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

RETURN $M[n]$.

Segmented least squares

Complexity?

Document layout problem

Dynamic programming

The beauty of such techniques is that the proof of correctness parallels the algorithmic structure.

Reference reading

Algorithm Design by Tardos et. al. 2006

Chapter 6: **§6.3** Segmented Least Squares

§6.4 Subset Sum and Knapsack

Introductions to Algorithms, 3rd Edition, by Cormen, et. al.

Chapter 15: **§15.4** Longest Common Subsequence

Network Flow

Ford - Fulkerson Algorithm

```
Initialize f_e=0 for all e ∈ E
repeat
    Search for an s-t path 'P' in the current residual graph G_f
    such that every edge of P has positive residual capacity.

if no such path 'P' then stop with current flow f_e for all e ∈ E

else
    Let Δ = min(residual capacity of e ∈ P in G_f)
    for all edges e of G whose forward edge is in P
        increase f_e by Δ
    for all edges e of G whose reverse edge is in P
        decrease f_e by Δ
```

Reference reading

Algorithm Design by Tardos et. al. 2006

Chapter 7: **§7.1** The Maximum-Flow Problem and the Ford-Fulkerson Algorithm