Paradigm of Dynamic Programming (Informal Description)

Partition of the initial problem **P(n)** into a set of subproblems

$P_1(n_1), P_2(n_2), \ldots, P_k(n_k)$ such that

- $n_i < n$ for all $i=1,\ldots,k$
- $k = poly(n)$
- $P(n)$ can be computed from $P_1(n_1), P_2(n_2), \ldots, P_k(n_k)$ in poly-time
- There is a natural *ordering* (from *smaller* to *bigger*) of the subproblems so that **recursion** can be applied efficiently.

6.3 Segmented Least Squares

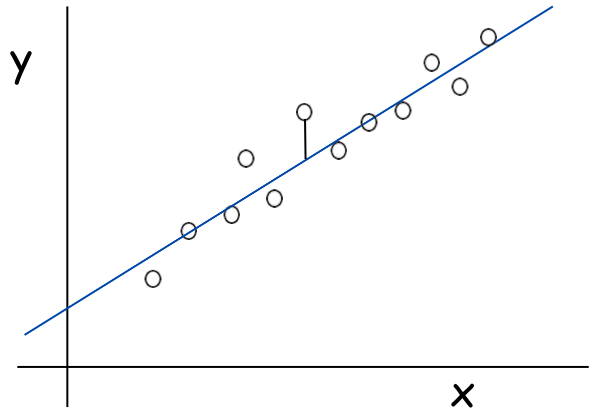
Segmented Least Squares

Least squares.

- Foundational problem in statistic and numerical analysis.
- Given **n** points in the plane: $I = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$.
- Find a line $y = ax + b$ that **minimizes** the **sum of the squared error**:

- (1)

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



Solution. Calculus $\Rightarrow$ **min** error SSE is achieved setting:

(2)

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

But for some sequence of points, the approximation given by just one line might be terrible....

Idea: find a good **trade-off** between

number of lines and **approximation quality**

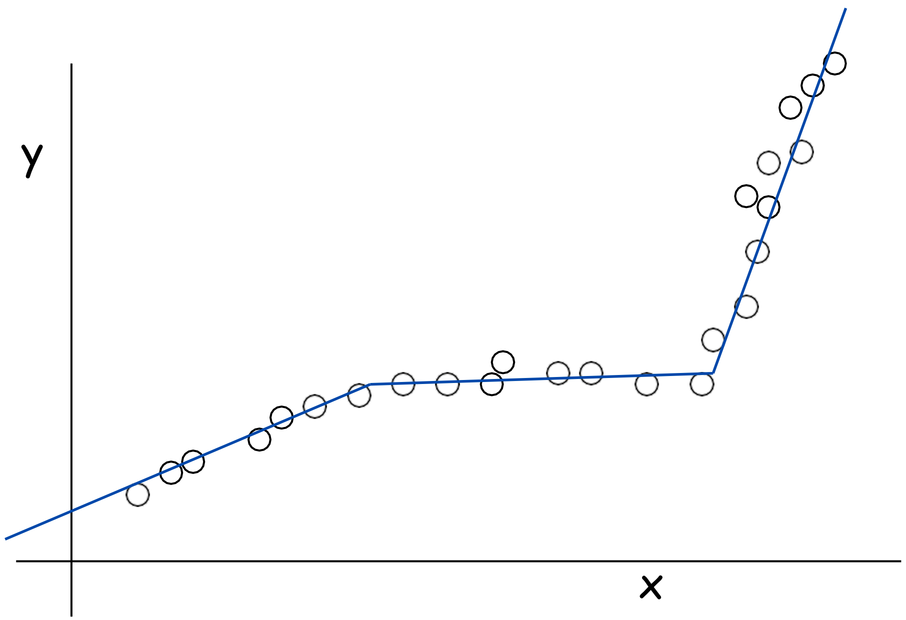
Segmented Least Squares

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

Q. What's a reasonable choice for $f(x)$ to balance accuracy and parsimony?

goodness of fit

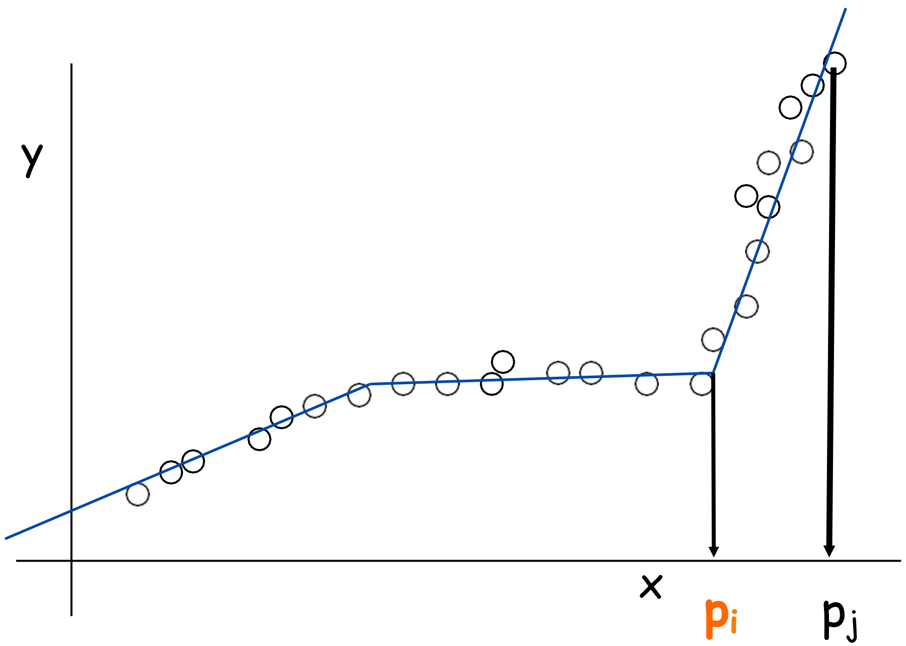
number of lines



Segmented Least Squares

Segmented least squares.

- Points lie roughly on a sequence of several line segments.
- Given n points in the plane $I = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ with
- $x_1 < x_2 < \ldots < x_n$, find a **sequence of lines** that minimizes:
  - the **sum** of the **sums** of the **squared errors** E in each segment: $E(i,j)=SSE(i,j)$
  - the **number** of lines L
- Tradeoff function: $E + c\,L$, for some constant $c > 0$.



Dynamic Programming: Multiway Choice

Notation.

- $OPT(j)$ = minimum cost for points $p_1, \dots, p_i, \dots, p_j$.
- $e(i, j)$ = minimum sum of squares for points $p_i, p_{i+1}, \dots, p_j$, according to Eq.s (1) and (2)

Observation (Optimal Structure).
Find a possible "recursion": let $[p_i, p_j]$ be the rightmost segment in $OPT(j)$, then

$$OPT(j) = OPT(i-1) + (1 \times C) + e(i,j) \quad \text{with } 1 \leq i \leq j$$

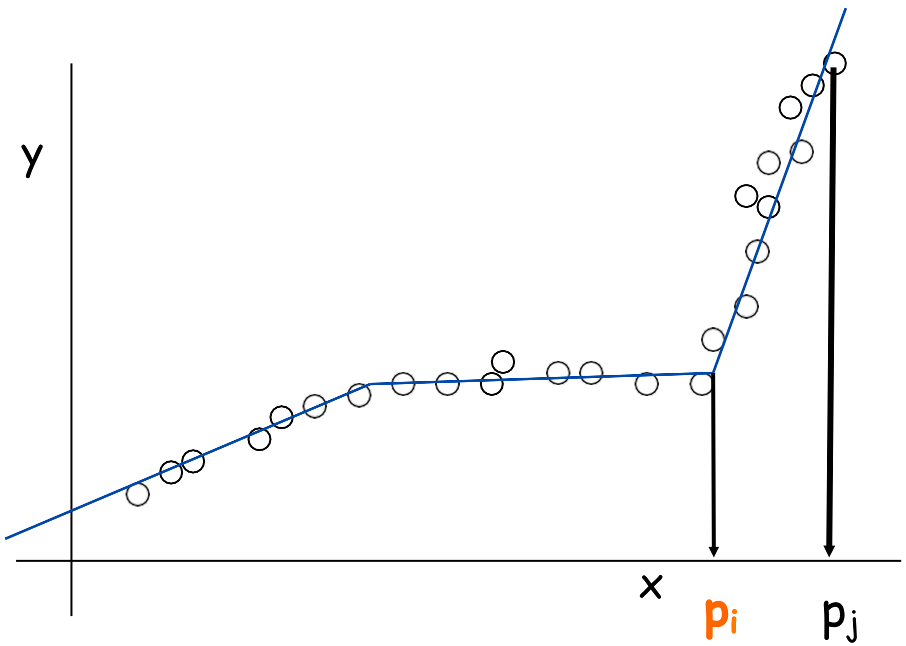
So the problem is to find the "optimal" i

Try All and get the best = Dynamic Programming!

Segmented Least Squares

Dyn. Prog. Idea: Assume an *Oracle* gives you the starting point p_i of the **rightmost** segment for the optimal solution of the instance $\mathbf{I} = \{p_1, \dots, p_j\}$
Then, you can split $\mathbf{OPT(j) := OPT(i-1) + c + e(i,j)}$

How finding point i ? TRY ALL!!!



Dynamic Programming: Multiway Choice

Notation.

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

  according to Eq.s (1): e(i,j) = $SSE(i,j) = \sum (y_i - ax_i - b)^2$
- and (2): **a=** ………………. ; **b =** ………………….

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: 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 ≤i ≤j (eij + c + M[i-1])

   return M[n]
}
```

Running time.

$O(n^3)$ (Prove as exercise and try to improve)

can be improved to $O(n^2)$ by pre-computing various statistics