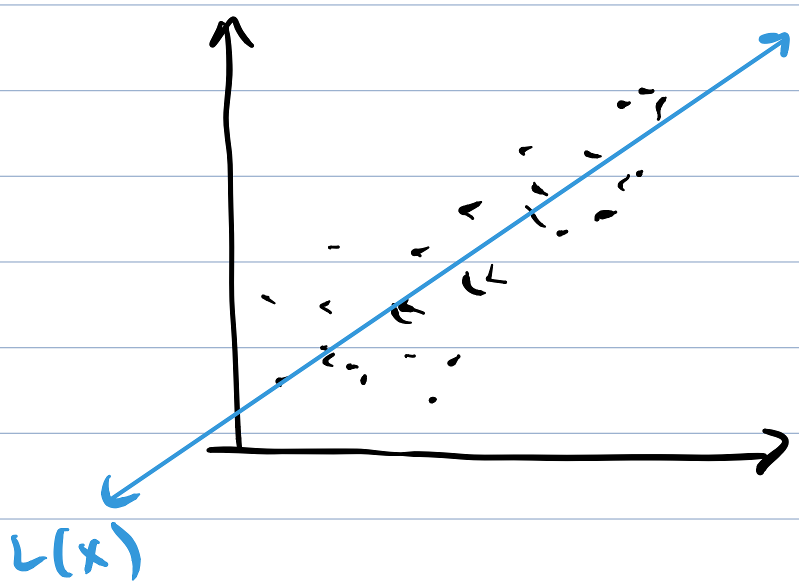
Mon, March 15 Lecture #21 /42

Last time: WIS

Example #2: Segmented Least Squares
Linear Regression:

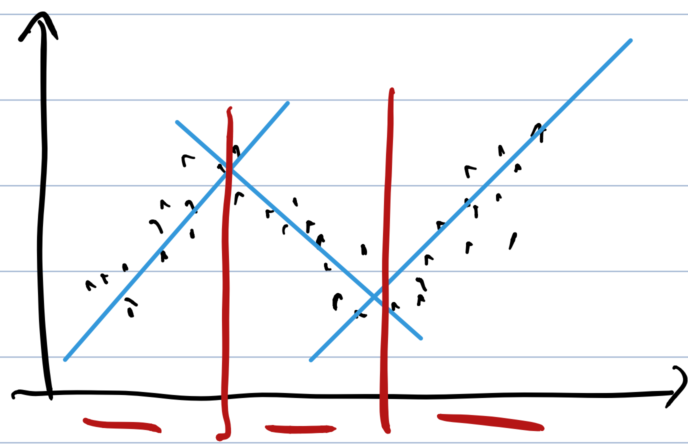


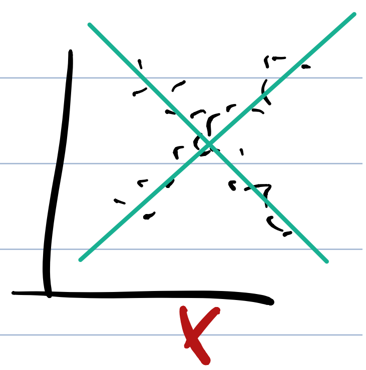
points $(x_1, y_1) \dots (x_n, y_n)$

goal: minimize

$$\sum_{i=1}^{n} (L(x_i) - y_i)^2$$

Bad:





Segmented Least Squares:
- Split the data points into k consecutive blocks

- Find the least squares line for each block separately
- Minimize: some combination of # of lines and the errors of each line

Score: $C \cdot (\#\text{ of lines}) +$ sum of errors of each line

where $C > 0$ is a parameter that we can tweak to affect the optimal solution.

<u>Search Space</u>: all ways of splitting the points into consecutive blocks.

<u>Ex</u>: $\underbrace{1 \quad 2 \quad 3 \quad 4}, \underbrace{5 \quad 6}, \underbrace{7 \quad 8 \quad 9}, \underbrace{10}$

$= 4$ blocks

Each of these candidates corresponds to ~~an~~ ~~"integer partition"~~ "compositions"

$10 = 4 + 3 + 2 + 1$

$4 + 2 + 3 + 1$

The # of these is exponential. 2^n

Dynamic Programming:

Subproblems

$\left(\begin{array}{ccc} \text{WIS: each new request is either} \\ \text{in} & \text{or} & \text{out} \\ v_\ell + O(p(R_\ell)) & & O(R_{\ell-1}) \end{array} \right)$

Each solution has:
- a final block of points $p_i, \ldots, p_n$
- an optimal solution on $p_1, p_2, \ldots, p_{i-1}$

$n = 50$

Define: * $e_{i,j}$ to be the least squares error for points $p_i, p_{i+1}, \ldots, p_j$.
* $O(i)$ to be the optimal score on $p_1, \ldots, p_i$.

We want $O(n)$.

What is the cost of having the last block go $p_i, \ldots, p_n$?

$$\text{cost} = \underbrace{e_{i,n}}_{\text{error of last line}} + \overbrace{C}^{\text{cost per line}} + \underbrace{O(i-1)}_{\text{optimal solution on } p_1, \ldots, p_{i-1}}$$

So, the optimal score for $p_1, \dots, p_n$ is the one where i leads to a minimum cost.

Recurrence:
$$\begin{cases} O(0) = 0 \\ O(j) = \min\limits_{1 \le i \le j} \left(e_{i,j} + C + O(i-1) \right) \end{cases}$$

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

$$= \min\left(e_{1,n} + C + O(0),\ e_{2,n} + C + O(1),\ \dots,\ e_{n,n} + C + O(n-1) \right)$$

1 2 3 4 5 6 $n=6$

one line

WIS: top-down
Better: bottom-up

memo = dict()
memo[0] = 0
compute $e_{i,j}$ for all pairs $i \le j$.   all pairs $O(n^2)$
for $j = 1, \ldots, n$:
    $\text{memo}[j] = \min(e_{i,j} + C + \text{memo}[i-1],$
                    $i = 1, \ldots, j)$ $O(n)$
return memo[n]

"iterative"

Runtime? $O(n^2)$

The key to d.p. is figuring out what you need to know.

In SLS, we only need to know the optimal cost for each possible endpoint $O(i-1)$. The actual composition on $p_1, \ldots, p_{i-1}$ has no effect on the cost of adding the next block $p_i, \ldots, p_j$.
So we can forget it.

C = cost per line