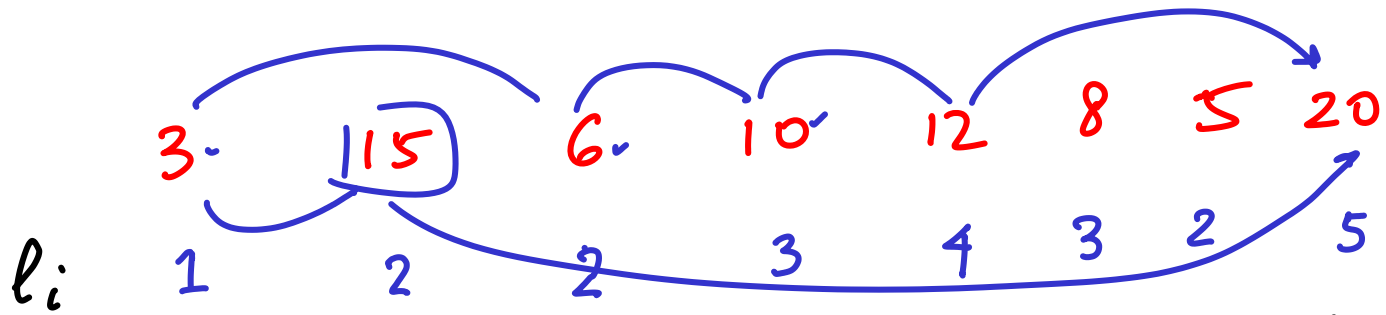
COL 702 Lecture 20 Oct 11

$x_1 \; x_2 \; x_3 \; x_4 \; \cdots x_i \; x_{i+1} \cdots x_n$

LMS: $i_1 < i_2 < i_3 \cdots i_k$

$x_{i_1} \le x_{i_2} \le x_{i_3} \cdots x_{i_k}$

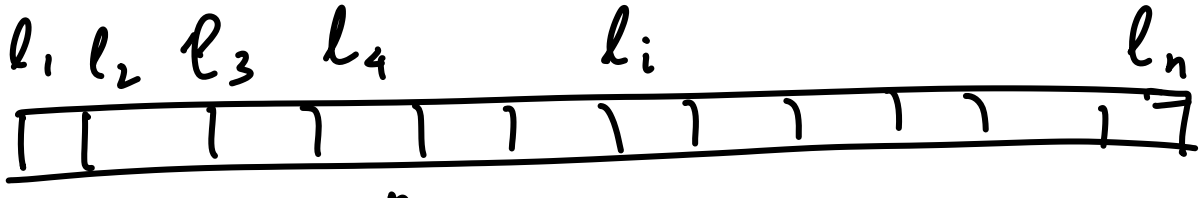
length K LMS — longest such sequence



LMS ending at position i, say ℓ_i

$$\ell_i = \max_{\substack{j<i \\ x_j \le x_i}} \{\ell_j\} + 1$$

$$\text{LMS} : \max_{1 \le i \le n} \{\ell_i\}$$



Running time $\sum_{i=1}^{n} O(i) = O(n^2)$

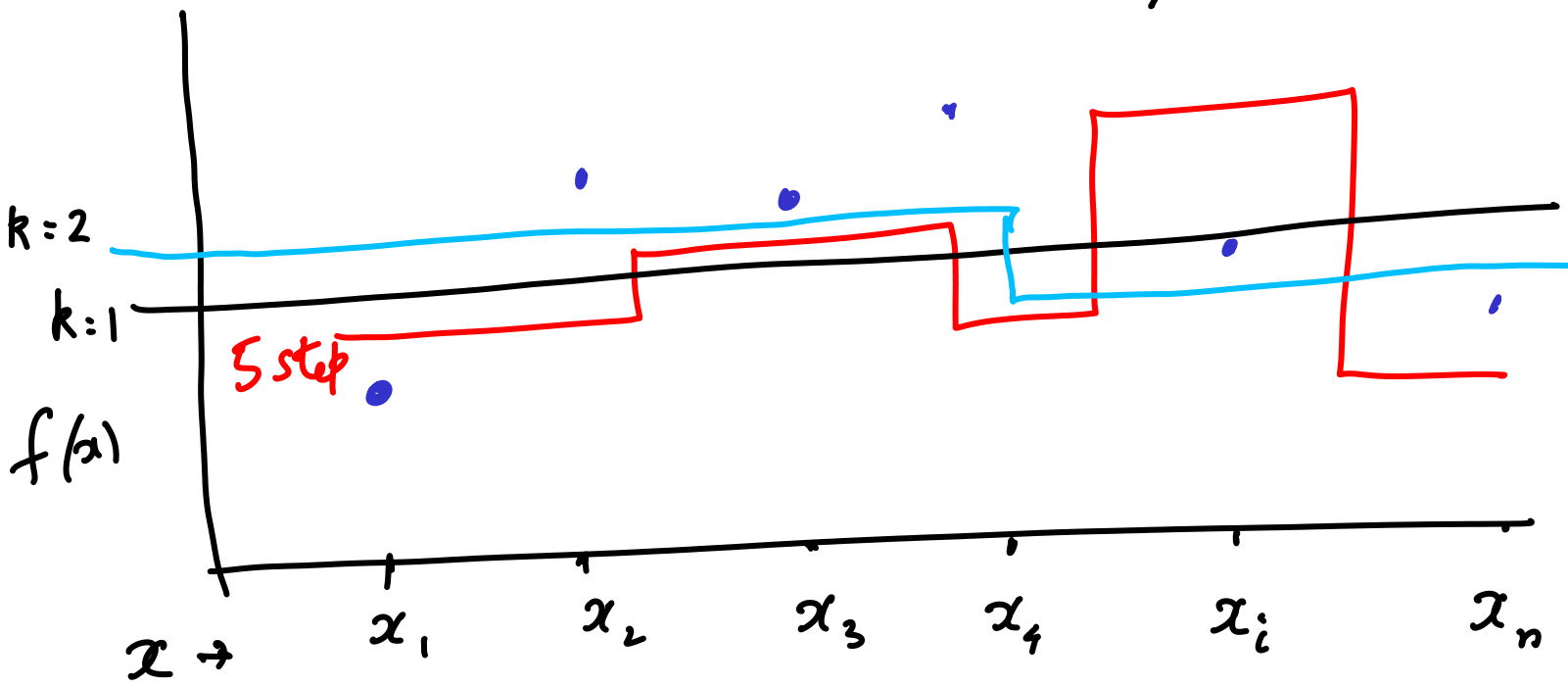
Can the $O(n^2)$ running time be improved?

Possibility 1 : Different recurrence

Possibility 2 : Some clever data structure (priority heap)

$O(n \log n)$ solns possible

Function approximation/compression



$(x_i, f(x_i))$ for all $1 \le i \le n$

$f(x) = x^2$ is easy to describe

Function approximation : involves finding "best" say f^* - according to some error metric

Possible metrics: $\Delta_i := |f^*(x_i) - f(x_i)|$

① $\sum \Delta_i$: sum of errors

② $\sum \Delta_i^2$ " " squares

③ $\max_i \Delta_i$. L_∞ metric

families of functions by which we want to approximate

Eg Linear function : $f^*: ax + b$

Degree k function : $a_{k-1} x^k + \dots$

given k k-step function according to sum of square error

Simple observations

① If $k = n$, then trivial, since the steps can be chosen at $x_i = f(x_i)$

$\Delta_i = 0$ for all i

(2) $K = 1$ Constant function

$$f(x) = \alpha$$

$$\text{Error} : \sum_{i=1}^{n} \left(f(x_i) - \alpha\right)^2$$

For what α is the error minimized

$\alpha = \frac{1}{n} \sum_{i=1}^{n} f(x_i)$ is known to minimize the error

(3) The steps can be restricted to one of $\{x_1, x_2 \dots x_n\}$ and not the entire Real line

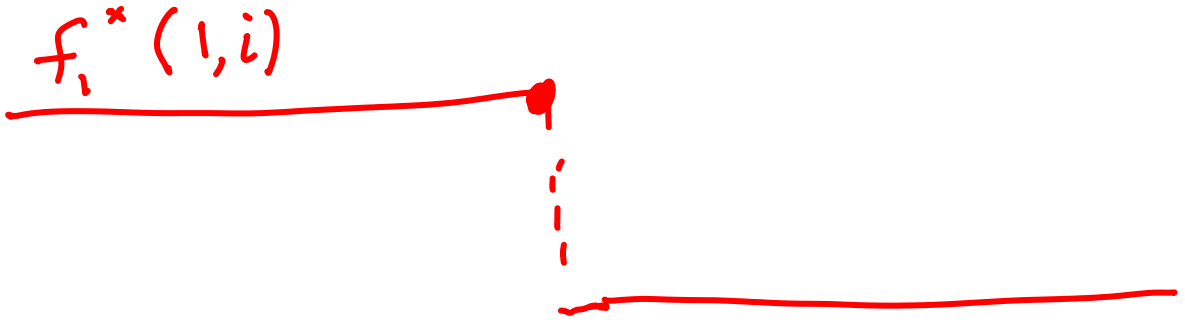
For the best possible 2 step function we only have to decide where the step is chosen by the optimal 2 step function f_2^* f_i^* : optimal i step function

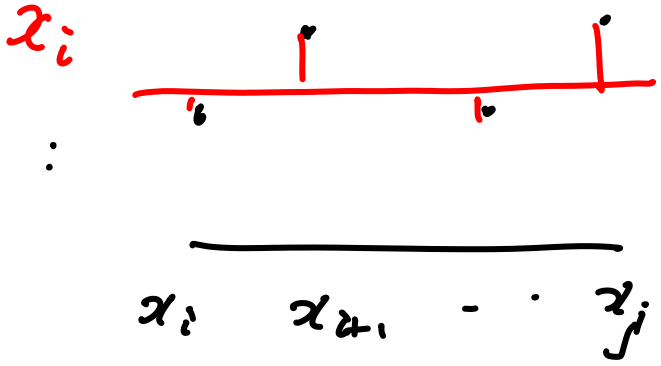
f_1^* : optimal 1 step function for the entire domain

$g^*(i,j)$: optimal 1-step function error of the for $[x_i\ x_{i+1} \cdots x_j]$

$$f_2^*(1,n) = \underset{g^*(1,i)+g^*(i+1,n)}{\operatorname{argmin}(i)} \; f_1^*(1,i) \odot f_1^*(i+1,n)$$

$\odot$ ← joined with





Computation of $g^*(i,j)$:

Time to compute $g^*(i,j)$

$i < j$: $O((j-i))$

Precompute $g^*(i,j)$ $\forall (i,j)$: $O(n^3)$

$\Rightarrow$ f_2^* can be computed in $O(n)$-time (given $g^*(i,j)$)

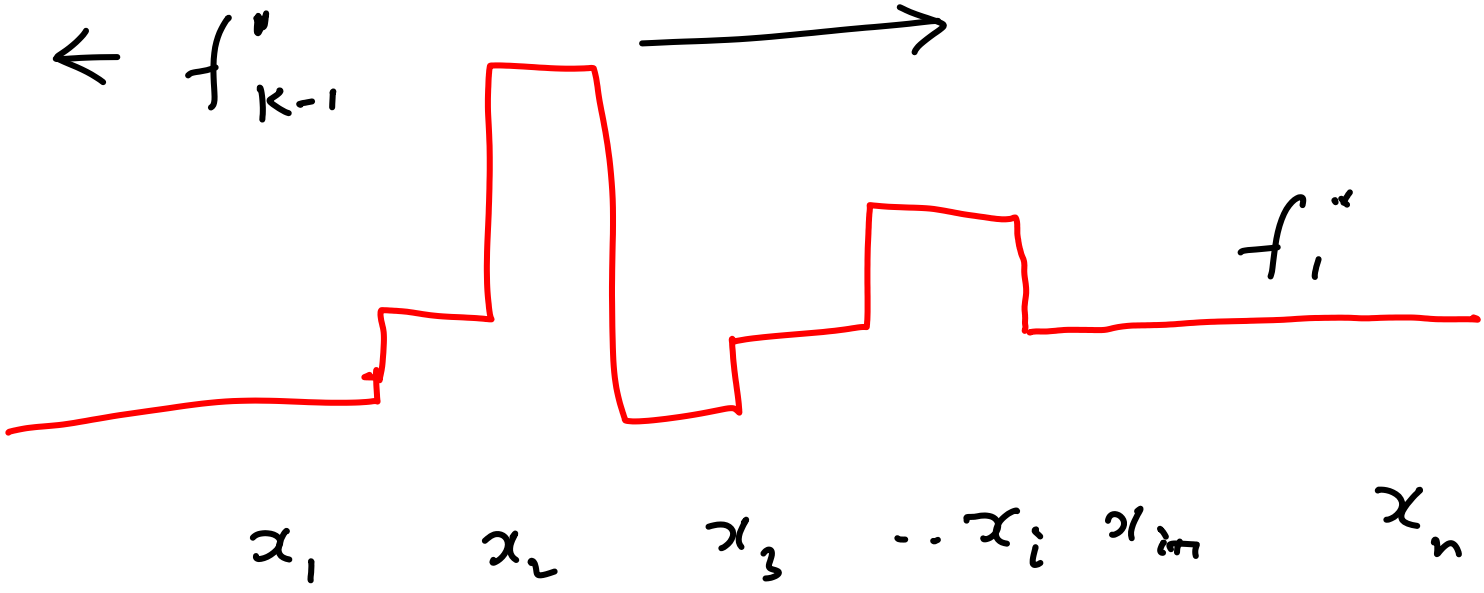
f_k^* : determined by two intervals

say $[1, x_j]$ $[x_{j+1}, n]$

$f_{k_1}^*$ $f_{k_2}^*$ $k_1 + k_2 = k$

It is easier to partition f_k^* into

f_1^* and f_{k-1}^*



The general problem is

$f_j^*(i)$: the best j step optimal function for the interval $[1 .. i]$ for $1 \le j \le k$, $1 \le i \le n$

Table size : kn

What is the time to compute each entry (given $g^*(i,j)$ are precomputed)

To compute $f_j^*(i)$ we need $f_{j-1}^*(i')$ for all $1 \le i' \le i$

Total time $\le n \cdot (kn) \sim O(kn^2)$