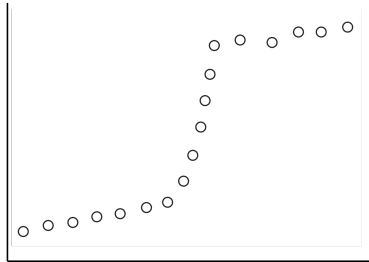


## Dynamic Programming III: Segmented Least Squares

### The Problem

We are given a set of points  $\{p_1 = (x_1, y_1), p_2 = (x_2, y_2), \dots, p_n = (x_n, y_n)\}$  sorted by  $x$ -coordinate. Our goal is to fit a (segmented) line to  $P$  with least squares error.



What is “error” here? We use square error (SSE) from any line we use. That is, if our line is determined by slope  $a$  and  $y$ -intercept  $b$ , then our SSE would be

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

Using calculus, we can derive that this is minimized when we set

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

But what if we can use as many segments as we want, just with a penalty  $c$  for each additional segment? How should we decide on the number of segments, and on what the segments should look like?

Our goal is to partition  $P$  into some  $C$  contiguous segments with minimal least squares error when there is a penalty  $c$  for each segment.

### Making the Key Observation

The last point  $p_n$  belongs to a single segment which must begin somewhere. Where does it begin? In each case, what does the optimal solution look like?

### Step 1: The Subproblem

Let  $\text{OPT}(i)$  denote the optimum solution for the points  $p_1, \dots, p_i$ , and  $\text{OPT}(0) = 0$ . Let  $e_{j,i}$  denote the minimum error of any line with respect to points  $p_j, \dots, p_i$  (i.e., find the line using the calculus solution up above).

## Step 2: The Recurrence

If the last segment of the optimal partition is  $p_i, \dots, p_n$ , then:  $\text{OPT}(n) = e_{i,n} + c + \text{OPT}(i - 1)$ .

Hence

$$\text{OPT}(i) = \min_{1 \leq j \leq i} \{e_{j,i} + c + \text{OPT}(j - 1)\}.$$

where we use the segment  $p_j, \dots, p_i$  if and only if  $j \in \text{argmin}$  of the above.

**Step 3: Prove that your recurrence is correct.** In the optimal solution on  $i$  points,  $i$  must be in a segment that starts at the  $j \in \text{argmin}$  of the above.  $\text{OPT}(j - 1)$  gives the optimal segmented SSE for the first  $i - 1$  points and  $e_{j,i}$  gives the optimal SSE for the segment from  $j$  to  $i$ , so adding these two error terms plus the penalty of  $c$  for using the one additional segment from  $j$  to  $i$  is the valid cost of this solution. If  $i$  instead was in a different segment that started at a different  $j'$ , then for the same reasons, the cost of this solution would be  $\text{OPT}(j' - 1) + c + e_{j',i}$ , but this term did not minimize the above which is why it was not selected, hence it cannot have optimal (minimal) error. Hence the above recurrence is correct.

**Step 4: State and prove your base cases.**  $\text{OPT}(0) = 0$  is enough to get us off the ground. Notice that  $\text{OPT}(1)$  is then well-defined.

**Step 5: State how to solve the original problem.** This is once again  $\text{OPT}(n)$ .

## Step 6: The Algorithm

---

**Algorithm 1** SegmentedLeastSquares( $p_1, \dots, p_n$ )

---

**Input:** Set of  $n$  points  $p_1, \dots, p_n$ .

Initialize memo array  $M$  of length  $n + 1$  with  $M[0] = 0$ .

**for** all pairs  $j \leq i$  **do**

    Compute  $e_{j,i}$  for the segment  $p_j, \dots, p_i$

**end for**

**for**  $i = 1, \dots, n$  **do**

$M[i] = \min_{1 \leq j \leq i} \{e_{j,i} + c + M[j - 1]\}$

**end for**

**return**  $M[n]$

---

## Step 7: Running Time

The recurrence optimizes over  $n$  things, and we loop over  $n$  entries, so filling the memo takes  $O(n^2)$  time. Solving for  $e_{j,i}$  takes  $O(n)$  time and there are  $O(n^2)$  pairs of  $(j, i)$ , so this takes  $O(n^3)$  time, which is the dominant term for the algorithm.

---

**Algorithm 2** FindSegments( $i$ )

---

**Input:** Number of points  $n$ .

Initialize memo array  $M$  of length  $n + 1$  with  $M[0] = 0$ .

**if**  $i = 0$  **then**

**return** *Null*

**else**

    Find an  $j$  that minimizes  $e_{j,i} + c + M[j - 1]$

**return** the segment  $\{p_j, \dots, p_i\}$  and FindSegments( $i - 1$ )

**end if**

---