Dynamic programming - II

Outline

Fitting piecewise functions to data

Simple Case

Given a set of points \( p_1, p_2, \ldots, p_n \) on a plane

Find a constant function \( f(x) = c \), s.t.

\[
\min_c \sum_{k=1}^{n} (c - p_k)^2
\]

Claim: \( c = \frac{1}{n} \sum_{k=1}^{n} p_k \) This is an average

Simple Case (Least Squares)

Proof.

Take a derivative and set it to 0:

\[
\frac{d}{dc} \sum_{k=1}^{n} (c - p_k)^2 = 0
\]

\[
\frac{d}{dc} \sum_{k=1}^{n} (c - p_k)^2 = \sum_{k=1}^{n} 2(c - p_k) = 2c - 2 \sum_{k=1}^{n} p_k = 0
\]

It follows, \( c = \frac{1}{n} \sum_{k=1}^{n} p_k \)

Piecewise Constant Segments

Rather than seek a single line, we allow a set of lines.

But this problem has a trivial solution:

Clearly we do not want to allow too many lines as well as too few.

Goal: Fund a set of constant functions \( f_1, \ldots, f_n \) s.t.

\[
\min_{f_1, \ldots, f_n} \left[ \sum_{k=1}^{n} (f_k - p_k)^2 + \alpha \sum_{k=1}^{n-1} \delta(f_k, f_{k+1}) \right]
\]

\( \delta(a, b) = \begin{cases} 0, & \text{if } a = b \\ 1, & \text{o.w.} \end{cases} \)

\( \alpha \) is a penalty

Fitting with penalty \( \alpha \) (given)

\[
\min_{f_1, \ldots, f_n} \left[ \sum_{k=1}^{n} (f_k - p_k)^2 + \alpha \sum_{k=1}^{n-1} \delta(f_k, f_{k+1}) \right]
\]

fidelity term smoothness term

If we have too few lines \( f_k \), we increase the fidelity term

As we increase the number of lines \( f_k \), we increase the penalty

There are exponentially many partitions (sets of lines), so we will be using DP.
Optimal Substructure in DP

Show that a solution to a problem consists of making a choice, which results in one or more subproblems.
Suppose that you are given the last choice that leads to an optimal solution.
Given this choice, determine which subproblems are to be solved.
Show that the solutions to the subproblems are optimal.

Subproblems: an intuition

Consider the optimal solution. The last point \( p_n \) belongs to a line that starts at some \( p_i \).
Then we remove these points from consideration and recursively solves the problem on the remaining points.

\[
\text{OPT}(n) = \text{OPT}(i-1) + \sum_{i=1}^{n} (c - p_i)^2 + \alpha
\]

Correctness

We need to prove that \( C(j) = \text{OPT}(j) \).
Clearly, \( \text{OPT}(j) \leq C(j) \).
Proof of \( C(j) \leq \text{OPT}(j) \) by strong induction
Base case \( j = 1 \). \( C(1) = 0 = \text{OPT}(1) \)
IH: true for \( j-1 \) points
We add \( j \)-th point.
Case 1). all points fit by a single line
Case 2). The last breakpoint is at \( p_i \).
which means that \( f_i \neq f_{i+1} \) and \( f_{i+1} = f_{i+2} = \ldots = f_j \).
\( \text{OPT}(j) = \text{OPT}(i) + \alpha + P(p_{i+1}, \ldots, p_j) \geq \)
by IH
\( \geq C(j-1) + \alpha + P(p_{i+1}, \ldots, p_j) = C(j) \)

Runtime

\[
C(j) = \min \left\{ \min_{i<j} \left( \frac{P(p_i, p_{i+1}, \ldots, p_j)}{f(i)} + \alpha + P(p_{i+1}, \ldots, p_j) \right) \right\}
\]

\[
P(p_{i+1}, \ldots, p_j) = \sum_{k=i}^{j} (c - p_k)^2
\]

For all \( i \)'s \( P(p_i, \ldots, p_j) = O(i) = O(j^2) \)
\( C(j) = \sum O(j^2) = O(n^3) \)

Can we do better?? Precompute \( P(p_i, \ldots, p_j) \)
Precompute all $P(p_i,...,p_j) = \sum_{k=i}^{j}(c_{ij} - p_k)^2$

1 \leq i < j \leq n

What is the runtime complexity of precomputing?

$O(n^3)$, hmm, this does not speed up the algorithm

How about using DP?

The main problem is how to compute $P$ for $j+1$ points knowing the result for $j$ points.

Proof of Claim2

Claim2: It takes $O(n^2)$ to compute all $P(p_i,...,p_j)$.

Def: The $k$-th moment $M_k$ is defined by $M_k = \sum_{j=1}^{n} p_j^k$

$M_0 = n$, $M_1 = p_1 + ... + p_n$, $M_2 = p_1^2 + ... + p_n^2$

Observe, if we know $M_k$ for $n$ points, we can compute $M_k$ for $(n+1)$ points in $O(1)$.

Proof of Claim1

Claim1: It takes $O(n^2)$ to compute all

$P(p_i,...,p_j) = M_2 - \frac{M_1^2}{M_0}$

We use DP to compute moments!!

$M_k(p_i,...,p_j) = \begin{cases} p_j^k, & \text{if } i = j \\ M_k(p_i,...,p_{j-1}) + p_j^k, & \text{o.w.} \end{cases}$

It has $O(n^2)$ time complexity.

Fitting in $L_1$

We considered fitting using least squares ($L_2$)

$$\min_{f_1,...,f_n} \sum_{j=1}^{n} (f_j - x_j)^2 + \alpha \sum_{j=1}^{n} \delta(f_j, x_j)$$

However in many practical cases a variation of the above achieves a better result, namely

$$\min_{f_1,...,f_n} \sum_{j=1}^{n} |f_j - x_j| + \alpha \sum_{j=1}^{n} \delta(f_j, x_j)$$
Let us start with the simplest case – a single line

\[
\text{MIN} \left[ \sum_{k=1}^{n} |f_k - p_k| + \alpha \sum_{k=1}^{n} \delta(f_k, f_{k+1}) \right]
\]

How do we find \( c \)?