An iterative Solution

Here is the Key equation again:

$$opt(n+1) = 0$$

$$opt(j) = \max\{OPT(j+1),\ V_j + OPT(p(j))\}.$$

for each j between 1 and n.

Our basic recursive algorithm was obtained by reading the equation as follows:

To Compute $Opt(j)$, we need to compute $opt(j+1)$ and $opt(p(j))$.

The iterative algorithm is obtained by reading the equation as follows:

~~Once~~ If we Know $Opt(j+1)$ and $opt(p(j))$, we can immediately deduce $opt(j)$.

The Key is that both $j+1$ and $p(j)$ are bigger than j. So we compute

$Opt(n+1)$, $Opt(n)$, ..., $Opt(j+1)$, $opt(j)$, ... $Opt(1)$

in that order. Thus when we wish to compute $Opt(j)$, both $opt(j+1)$ and $Opt(p(j))$ have already been computed.

Here is the iterative algorithm.

$M[n+1] \leftarrow 0$

for $j \leftarrow n$ down to 1

$\quad M[j] \leftarrow \max\{M[j+1],\ v_j + M[p(j)]\}$

A proof by induction that is based on the Key equation can be given to establish the correctness of this algorithm (if ~~needed~~ one is not completely sure of the correctness.)

The running time of the algo is clearly $O(n)$.

If we want the optimal solution and not just its ~~cost~~ value, this can be easily recovered.

Note that O_j, the optimal solution for $[j \ldots n]$ is better of two options:

(1) O_{j+1}

~~(2) v_j plus an optimal solution for~~

(2) v_j plus $O_{p(j)}$.

```
Print-Opt (j)
  If j = n+1 return.
  If M[j+1] > v_j + M[p(j)]
      Print-Opt (j+1)
  else
      Print (v_j)
      Print-Opt (p(j))
```

Segmented Least Squares

This problem is motivated by the issue of fitting multiple lines to a set of points. We assume a subroutine that given a set Q of points finds a line l that best fits Q. The quality of the fit is measured by an error term: if the error is 0, the fit is very good, and the larger it is, the worse is the fit.

Lets call this subroutine LineFit(Q). It returns $l(Q)$, the line that best fits Q, and $\text{error}(Q) \geq 0$, an error term measuring the quality of the fit.

The Input to the Segmented Least Squares problem is a sequence of points $P = \{p_1, p_2, \ldots, p_n\}$. The goal is to

partition the sequence into a set of "segments" so as to minimize a certain penalty function to be defined below. A segment is just a contiguous subsequence of the original sequence: It has the form $\{P_i, \ldots, P_j\}$ for some $i \leq j$.

The penalty of a partition is defined to be the sum of

(i) The number of segments in the partition times a given fixed multiplier $C > 0$.

(ii) for each segment of partition, ~~sum~~ error of the segment as computed by LineFit( ).

Another way of describing the penalty of a partition is the sum, over each segment, of C + error of that segment.

Some useful notation: let $e_{i,j}$ denote error $(\{p_i \dots p_j\})$.

Consider an optimal partition for $\{p_1, \dots p_n\}$ and suppose that its last segment is $\{p_i \dots p_n\}$. We can think of the optimal partition as being a partition Π of $\{p_1 \dots p_{i-1}\}$ followed by the segment $\{p_i \dots p_n\}$.

Thus, penalty of optimal partition

$$= \text{penalty of } \Pi + C + e_{i,n}.$$

<u>Claim</u>: Π must be optimal for $\{p_1 \dots p_{i-1}\}$.

The claim is easily proved by contradiction.

The claim says that if knew i, we could compute an optimal partition for $\{p_1 \dots p_{i-1}\}$, add segment $\{p_i \dots p_n\}$

and we would have an optimal partition for $\{p_1, \dots p_n\}$.

We don't know i, of course. But we can obtain an optimal partition for $\{p_1 \dots p_n\}$ by taking the best of the following n options:

For each $1 \le i \le n$,

the optimal partition of $\{p_1, \dots p_{i-1}\}$ followed by segment $\{p_i \dots p_n\}$.

~~(The~~

Let $Opt(j)$ denote the penalty of the optimal partition of $\{p_1 \dots p_j\}$.

Define $Opt(0) = 0$.

We have the following equation: ~~for~~

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

In general, for any $1 \le j \le n$,

$$\boxed{Opt(j) = \min_{1 \le i \le j} \left(Opt(i-1) + C + e_{i,j} \right).}$$

Let us translate this into an iterative algorithm. Once we have $opt(0), \ldots, Opt(j-1)$, we can easily compute $opt(j)$ using the above equation.

Array $M[0..n]$

$M[0] \leftarrow 0$

For each ~~$1 \le i \le j \le n$~~ (i,j) such that $1 \le i \le j \le n$,

Compute $e[i,j]$ using $LineFit(\{p_i, \ldots, p_j\})$.

For $j \leftarrow 1$ to n do

$M[j] \leftarrow \min_{1 \le i \le j-1} \left(M[0] + C + e[i,j] \right).$

// Thats a for loop where i ranges from 1 to $j-1$