In *Computing Science and Statistics* **32** (2000).

Significantly revised version: Unimodal Regression via Prefix Isotonic Regression,
Computation Statistics and Data Analysis 53 (2008), 289–297.

Optimal Algorithms for Unimodal Regression

Quentin F. Stout

University of Michigan
Ann Arbor, MI 48109-2122

Abstract

This paper gives optimal algorithms for determining real-valued univariate unimodal regressions, that is, for determining the optimal regression which is increasing and then decreasing. Such regressions arise in a wide variety of applications. They are a form of shape-constrained nonparametric regression, closely related to isotonic regression. For the L_2 metric our algorithm requires only $\Theta(n)$ time for regression on n points, while for the L_1 metric it requires $\Theta(n \log n)$ time. Previous algorithms only considered the L_2 metric and required $\Omega(n^2)$ time. All previous algorithms used multiple calls to isotonic regression, and our major contribution is to organize these into a prefix isotonic regression, whereby one computes the regression on all initial segments. The prefix approach utilizes the solution for one initial segment to aid in the solution of the next, which considerably reduces the total time required. Our prefix isotonic regression algorithm for the L_1 metric also supplies the first $\Theta(n \log n)$ algorithm for L_1 isotonic regression.

Keywords and phrases: univariate unimodal regression, umbrella ordering, isotonic or monotonic regression, prefix operation, scan, pool adjacent violators (PAV), L_1 regression, median regression, mergeable heaps, persistent data structure

1 Introduction

Given n univariate real data values (x_i, y_i) with nonnegative real weights w_i, $i = 1, \ldots, n$, where $x_1 < \cdots < x_n$, and given a $p \in [1, \infty]$, the L_p *increasing isotonic regression* of the data is the set $\{(x_i, \hat{y}_i) : i = 1, \ldots, n\}$ that minimizes

$$\begin{array}{ll} \sum_{i=1}^{n} w_i |y_i - \hat{y}_i|^p & \text{if } 1 \le p < \infty \\ \max_{i=1}^{n} w_i |y_i - \hat{y}_i| & \text{if } p = \infty \end{array} \qquad (1)$$

subject to the increasing isotonic constraint that

$$\hat{y}_1 \le \hat{y}_2 \cdots \le \hat{y}_n.$$

Note that the values are merely required to be nondecreasing, rather than strictly increasing. The L_p *unimodal regression* of the data is the set $\{(x_i, \hat{y}_i) : i = 1, \ldots, n\}$ that minimizes Equation (1) subject to the unimodal constraint that there is an $m \in \{1, \ldots, n\}$ such that

$$\hat{y}_1 \le \hat{y}_2 \cdots \le \hat{y}_m \ge \hat{y}_{m-1} \cdots \ge \hat{y}_n,$$

i.e., such that $\{\hat{y}_i\}$ is increasing isotonic on $1 \ldots m$ and decreasing isotonic on $m \ldots n$. Note that the unimodal constraint is also known as umbrella ordering, and that isotonic regression is also known as monotonic regression.

By the *norm of a regression* we mean the quantity in Equation (1). This is a slight misuse of the term because we are not taking p^{th} roots, but since the only role norms and distances play in the algorithms is to compare to determine which is larger, the roots are unnecessary.

Both isotonic regression and unimodal regression are examples of nonparametric shape-constrained regression. Our interest in unimodal regression was motivated by dose-response problems with competing failure modes [6], but more generally such regressions are of use in a wide range of applications when there is prior knowledge about the shape of a response function but no assumption of a parametric form.

In Section 2 we examine previous work on the problem of determining unimodal regression, all of which was for L_2 regression. The previously published algorithms required time that ranged from $\Theta(n^2)$ to $\Theta(n2^n)$. In Section 3 we introduce the notion of prefix isotonic regression, and in Section 3.1 show how to efficiently determine it for the L_2 metric in $\Theta(n)$ time. In Section 3.2 we show that L_1 prefix isotonic regression is slightly more challenging, and show how to determine it in $\Theta(n \log n)$ time. Section 4 contains an immediate corollary of the results on prefix isotonic regression, namely that unimodal regression can be computed in the same time bounds. Section 5 concludes with some final remarks.

Throughout, we assume that we are given the data in order of increasing independent variable. If the data is not so ordered, then an initial sorting step, taking $\Theta(n \log n)$ time, is needed.

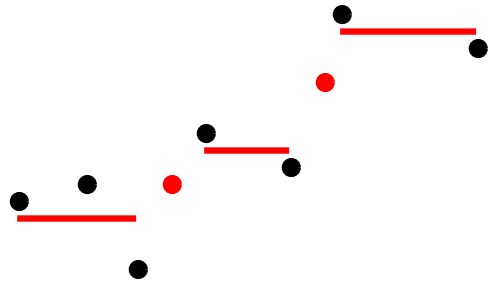

Figure 1: L_2 Increasing Isotonic Regression

2 Previous Work

Isotonic regression does not yield a smooth curve, but rather a collection of level sets where the regression is constant. For example, Figure 1 shows the L_2 increasing isotonic regression of a set of data with equal weights, where the black dots represent data values and the red/gray lines represent the level sets. Dots that are red/gray represent data values that are also regression values. (Equivalently, these are dots with no lines directly above or below.) It can occur that none of the data values are also regression values.

It is well-known that the L_2 increasing isotonic regression can be determined in $\Theta(n)$ time. Several optimal algorithms use some variation of the "pair adjacent violators" (PAV) approach [1]. In this approach, initially each data value is viewed as a level set. At each step, if there are two adjacent level sets that are out of order (i.e., the left level set is above the right one) then the sets are combined and the weighted L_2 mean of the data values becomes the value of the new level set. It can be shown that no matter what order is used to combine level sets, once there are no level sets out of order the correct answer has been produced [12].

Apparently all previous work on unimodal regression has concentrated on L_2 regression, though the basic approach can be applied to arbitrary metrics. Previous researchers solved the problem by trying each possible i as the location of the maximum. For each i they fit increasing isotonic and decreasing isotonic curves to the remaining points, and determine the distance between the resulting fit and the data. The smallest distance attained corresponds to the solution of the problem.

Testing each new value of i results in new calls to procedures to determine isotonic fits. The fastest and most straightforward approach, used in [4, 5, 7, 11, 13] and given in Figure 2, fits an increasing curve to the values corresponding to $x_1 \ldots x_i$ and a decreasing curve to the values corresponding to $x_i \ldots x_n$. Since L_2 isotonic regression of m points can be determined in $\Theta(m)$ time, this approach to determining unimodal regression takes

$$\Theta\left(\sum_{i=1}^{n} C + \Theta(i-1) + \Theta(n-i)\right) = \Theta(n^2)$$

time.

{mode: location of mode of best unimodal fit}

do $i = 1, n$
    distl(i) = norm_increasing_iso_regres($x_1 \ldots x_i$)
    distr(i) = norm_decreasing_iso_regres($x_i \ldots x_n$)
enddo

mode=$\arg\min$ {distl(i)+distr(i): $1 \le i \le n$}

Figure 2: Best Previous Unimodal Regression Algorithm

A somewhat different approach was used in [8]. He noted that for any unimodal function, the values on both sides of the maximum can be rearranged into a single decreasing sequence. For example, if $n = 5$ and if x_3 is the location of the maximum, then the remaining values could be in the decreasing order corresponding to one of the sequences (x_2, x_1, x_4, x_5), (x_2, x_4, x_1, x_5), (x_2, x_4, x_5, x_1), (x_4, x_2, x_1, x_5), (x_4, x_2, x_5, x_1), or (x_4, x_5, x_2, x_1). For each such sequence he determined the isotonic regression, and the one with minimal norm was the one that corresponded to the best unimodal fit with x_3 as maximum. Varying i through all possible locations of the maximum would then give the globally best fit.

By combining them into a single order, instead of utilizing the fact that the best regression on each side is independent of the other side, the number of isotonic regressions required by [8] was far greater than those used by others. It can be shown that exactly 2^{n-1} isotonic regressions are required (see Appendix A), and thus the total time of this approach is $\Theta(n2^n)$. This is substantially worse than the approach in Figure 2, and is feasible for only small values of n.

In general, the mode of the best unimodal fit is not unique. For example, if the weighted data values are {(1,1,1), (2,0,1), (3,1,1)}, as in Figure 3, then for any norm, one best unimodal fit is {(1,1), (2,0.5), (3,0.5)}, with mode at 1, and another best unimodal fit is {(1,0.5), (2,0.5), (3,1)}, with mode at 3. All of the previously published algorithms, and the ones herein, can locate all of the modes that correspond to best fits, and some secondary criteria could be applied to select among them. The algorithms in this paper do not apply such criteria, but the modifications to do so are straightforward.

Despite the nonuniqueness of the optimum, it is easy to show that for any L_p metric with $p < \infty$, for any optimum mode x_m, the value at x_m of its optimum fit is the original

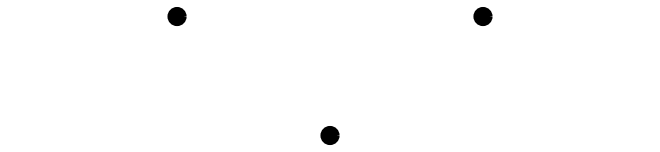

Figure 3: Data Values with Nonunique Mode

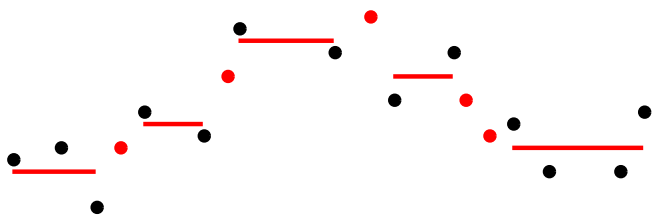

Figure 4: A Unimodal Regression

data value y_m. It is also easy to see that the increasing isotonic regression on $x_1 \ldots x_m$ has value y_m at x_m, as does the decreasing isotonic regression on $x_m \ldots x_n$. Figure 4 shows a unimodal regression where all the data have equal weights. The dots represent data values, and the red/gray lines represent the level sets. Red/gray dots represent data values that are also regression values. The mode of the regression is the tenth dot from the left.

3 Prefix Isotonic Regression

Given real data values $\{(x_i, y_i)\}$ with nonnegative real weights $\{w_i\}, i = 1, \ldots, n$, and given a metric μ on the reals, the μ *prefix isotonic regression problem* is to determine, for all m, $1 \leq m \leq n$, the μ increasing isotonic regression on the subset consisting of weighted values with indices $\leq m$. That is, it is to determine the μ increasing isotonic regression for all initial portions of the data.

Note that prefix isotonic regression determines exactly the set of increasing isotonic regression problems examined by [4, 5, 7, 11, 13]. However, the critical observation is that determining all of them should be approached as a single integrated problem, rather than merely as a collection of calls to subroutines to solve each subproblem. Prefix operations, also called *scan operations* (the term that was used for them in the computer language APL), are utilized as building blocks for various efficient algorithms in computer science. In parallel computing, prefix operations are also known as *parallel prefix operations* since often all values can be determined concurrently.

The basic prefix isotonic regression algorithm is given in Figure 5. The outermost loop on i goes through the points

{left(i): left endpoint of level set containing x_i}
{mean(i): mean value of level set containing x_i}
{dist(i): norm of increasing isotonic regression on $x_1 \ldots x_i$}

mean(0) = -∞
left(0) = 0
dist(0) = 0
do $i = 1, n$
    $\hat{y}$ = y_i
    ℓ = i
    while ($\hat{y} \leq$ mean(ℓ-1)) do
        {merge with level set to left}
        ℓ = left(ℓ-1)
        $\hat{y}$ = weighted mean of $(y_\ell, w_\ell) \ldots (y_i, w_i)$
    endwhile
    mean(i) = $\hat{y}$
    left(i) = ℓ
    newdist = weighted dist. $\hat{y}$ to $(y_\ell, w_\ell) \ldots (y_i, w_i)$
    dist(i) = newdist+dist(ℓ-1)
enddo

Figure 5: Prefix Isotonic Regression

in increasing indexing order, adding them to the previous solution. The "loop invariant" is that at the start of the loop, the increasing isotonic regression of the points $x_1 \ldots x_{i-1}$ has been determined. In right to left order, it consists of the level set containing points with indices left$(i-1) \ldots i-1$, with value mean$(i-1)$, the level set containing points with indices left(left$(i-1)-1) \ldots$left$(i-1)-1$, with value mean(left$(i-1)-1$), the level set containing points with indices left(left(left$(i-1)-1)-1) \ldots$left(left$(i-1)-1)-1$ with value mean(left(left$(i-1)-1)-1$), and so on. Further, the norm of this regression is dist$(i-1)$.

More generally, we have the following fact:

> **Observation 1:** After the algorithm in Figure 5 has completed, for any index m, $1 \leq m \leq n$, the increasing isotonic regression on the leftmost m values can be recovered in $\Theta(m)$ time from the values stored in left() and mean(), and has norm dist(m).

The recovery proceeds exactly as above, in right-to-left order. Note that when a new point at index i is added, it updates only the left(i), mean(i), and dist(i) entries, leaving the earlier entries unchanged since values for other indices within the merged level set will never be referred to again and hence are not needed. The left() and mean() arrays form what is called a *persistent data structure*, allowing one to rapidly recreate the intermediate regressions. Most data structures typically destroy intermediate states.

If the value of the new point, y_i, is greater than the mean of the level set containing x_{i-1}, then it is easy to see that the increasing isotonic regression of the points with indices $1 \ldots i$ is merely that of the points $1 \ldots i-1$ unioned with a new level set consisting only of x_i with value y_i. However, if y_i is less than or equal to the mean of the level set containing x_{i-1}, then they are out of order and must be merged. This new merged level set is then compared to the level set to its left. If they are in order, i.e., if the mean of the left level set is less than the mean of the right level set, then the process is done, while if their means are out of order they are merged and the process of comparing to the left is repeated. This is accomplished in the while-loop.

The fact that this merging process correctly determines the increasing isotonic regression for the new initial segment follows immediately from the PAV property.

To apply the algorithm in Figure 5 to a specific metric, one needs to determine how to do the operations inside the while-loop, i.e., how to determine the mean and distance of the merged level sets. As will be shown in Sections 3.1 and 3.2, efficiently implementing these operations heavily depends upon the metric.

> **Observation 2:** If the operations of determining the mean and distance in the while-loop can be accomplished in constant time, then the entire algorithm requires only $\Theta(n)$ time. This is because the total number of iterations of the while-loop can be at most $n-1$. This may not be obvious since the while-loop may be iterated $\Theta(n)$ times for a single value of i, and the loop is encountered n times. However, every time the loop is iterated, two disjoint nonempty level sets have been merged. One can view the data set as initially being n disjoint sets, and these can only be merged at most $n-1$ times. It is clear that all of the other operations within the while-loop take constant time per iteration, and that the operations outside the while-loop take constant time per iteration of i.

Notice that if one determines the mean and distance functions for a level set by just calling a function to compute them, given all the elements, then it will take $\Omega(m)$ time for a set of size m, and it is easy to see that this would require the algorithm to take $\Omega(n^2)$ total time in the worst case, i.e., it would be as bad as the algorithm in Figure 2. To achieve better results, one needs to utilize previous calculations for the level sets to aid in the calculations for the newly merged sets. Techniques to do this will depend upon the metric.

3.1 L_2 Prefix Isotonic Regression

To apply the prefix isotonic regression algorithm to the L_2 metric, one needs procedures for determining the mean and distance of the L_2 level sets. Fortunately, it is well known that the algebraic properties of this metric make this a simple task. We introduce additional functions sum(), sumsq() and weight(), as shown in Figure 6.

```
{sum(i): weighted sum of values in x_i's level set}
{sumsq(i): weighted sum of squares of values in x_i's
 level set}
{weight(i): sum of weights of x_i's level set}

do i = 1, n
    sum(i) = w_i · y_i
    sumsq(i) = w_i · y_i^2
    weight(i) = w_i
    while (ŷ ≤ mean(ℓ-1)) do
        {merge with level set to left}
        sum(i) = sum(i)+sum(ℓ-1)
        sumsq(i) = sumsq(i)+sumsq(ℓ-1)
        weight(i) = weight(i)+weight(ℓ-1)
        ℓ = left(ℓ-1)
        ŷ = sum(i)/weight(i)
    endwhile
    newdist = sumsq(i)−sum(i)^2/weight(i)
enddo
```

Figure 6: Additions to Determine L_2 Regression

With the simple additions of Figure 6 to Figure 5, the following result concerning running time follows immediately from Observation 2 since the operations in Figure 6 take constant time per iteration. The time for recovering solutions follows from Observation 1.

Theorem 1 *Given weighted data* $\{(x_i, y_i, w_i) : i = 1, \ldots, n\}$*, in* $\Theta(n)$ *total time the algorithm of Figure 5, with the additions in Figure 6, will determine, for every* m*, the norm of the* L_2 *increasing isotonic regression of* $\{(x_i, y_i, w_i) : i = 1, \ldots, m\}$*. Further, after the algorithm has been run, for any* m*, in* $\Theta(m)$ *time the* L_2 *increasing isotonic regression of* $\{(x_i, y_i, w_i) : i = 1, \ldots, m\}$ *can be recovered.* □

3.2 L_1 Prefix Isotonic Regression

For L_1 regression the situation is much more complex than it was for L_2 regression. Given a weighted set of values, their L_1 mean is the weighted median. Weighted medians are not always unique, so for simplicity we utilize the largest such value. In a specific application one might wish to add secondary criteria to determine which weighted median to use. While it is well-known that one can determine a weighted median in time linear in the number of values, such an approach

would only yield an algorithm taking $\Theta(n^2)$ time. Unfortunately there are no algebraic identities which easily allow one to reuse calculations when merging level sets, so a more complicated approach is needed.

For each level set with right endpoint i we maintain two additional data structures, top(i) and bottom(i). In top(i) we maintain information about all of the elements of the level set greater than the median (which is kept in mean(i)), while bottom(i) maintains information about all of the elements less than or equal to the median. The operations for top(i) are:

- top(i).sum: weighted sum of elements in top(i).
- top(i).weight: sum of weights of elements in top(i).
- add(top$(i), y, w$): add value y with weight w to top(i).
- merge(top(i), top(j)): merge tops of two level sets.
- init(top(i)): initialize top(i).sum and top(i).weight to 0.

These operations are quite trivial, so the data structure is implemented as a simple record containing merely the sum and weight values, updated as shown in Figure 7. All of the operations can be implemented in constant time.

The information in bottom(i) is more complex, providing the operations:

- bottom(i).sum: weighted sum of elements in bottom(i).
- bottom(i).weight: sum of weights of elements in bottom(i).
- bottom(i).max: largest value in bottom(i).
- pop(bottom(i)): remove largest value from bottom(i).
- merge(bottom(i), bottom(j)): merge bottoms of two level sets.
- init(bottom$(i), y, w$): initialize bottom(i) to contain value y with weight w.

Note that there is an asymmetry between top() and bottom(). When two level sets are being merged, the new median will be less than the median of the left level set, and hence one can think of moving items from bottom() to top(). Once an item is in a top set it stays in top sets. This would not quite be true if one did a straightforward implementation of the algorithm in Figure 5, in that if one builds a new level set and then determines that it is below the level set to the left, then from the viewpoint of the right level set the median moves up, which might require items to move from top() to bottom(). To prevent this, we instead do a "look ahead" to see if the mean of the level set being worked on will be below the level set to the left, and if so then we immediately merge them rather than completing the determination of the mean of the current level set. Figure 7 shows how this is accomplished.

Efficiently implementing the bottom() operations is a crucial part of the L_1 implementation. Implementing pop() and max in logarithmic time is quite easy via a standard abstract data structure known as a *heap*, appearing in any undergraduate data structures text, but adding the merge() operation while retaining logarithmic time is a bit more complicated. Fortunately, all of these operations are supplied by the abstract data structure known as *mergeable heaps*, which appears in many standard texts on advanced data structures or algorithms, such as [3]. By implementing the mergeable heaps via binomial heaps or Fibonacci heaps (see [3]), each of these three operations can be completed in $O(\log m)$ time for a level set of m items. (For binomial heaps this is the worst-case time, while Fibonacci heaps have better amortized time per operation.) Providing the additional operations in constant time is straightforward.

Theorem 2 *Given weighted data* $\{(x_i, y_i, w_i) : i = 1, \ldots, n\}$*, the algorithm in Figure 7 takes* $\Theta(n \log n)$ *total time and will determine, for each* m*, the norm of the* L_1 *increasing isotonic regression of* $\{(x_i, y_i, w_i) : i = 1, \ldots, m\}$*. Further, after the algorithm has been run, for any* m*, in* $\Theta(m)$ *time the* L_1 *increasing isotonic regression of* $\{(x_i, y_i, w_i) : i = 1, \ldots, m\}$ *can be recovered.*

Proof: For the timing analysis of the algorithm, in the innermost while loop, each iteration corresponds to moving an item from bottom() to top(). Since this can happen at most $n - 1$ times, and all operations inside the loop take at most $\Theta(\log n)$ time, the time is no worse than claimed. The time to recover regressions follows immediately from Observation 1. □

To help understand why $\Theta(n \log n)$ time is required, rather than $\Theta(n)$, we note that L_1 prefix isotonic regression is as hard as sorting real values. To see this, let $\{y_i : 1 \le i \le n\}$ be any set of real values, and let $\{\tilde{y}_i : 1 \le i \le n\}$ represent the same set in decreasing order. Let $u = -1 + \min_i y_i$ and $v = 1 + \max_i y_i$. Then for the weighted data sequence $(0, v, n+1)$, $(1, y_1, 1)$, $(2, y_2, 1)$, $\ldots (n, y_n, 1)$, $(n+1, u, 2)$, $(n+2, u, 2)$, $\ldots$, $(2n+1, u, 2)$, the level set value at $n + i$ for the regression on indices $0 \ldots n + i$ is $\tilde{y}_i$, for $1 \le i \le n$, and thus determining these regressions will yield the values in sorted order.

It was an open question as to whether L_1 isotonic regression could be computed in $\Theta(n \log n)$ worst-case time. This was pointed out in [10], where they give a $\Theta(n \log^2 n)$ algorithm for this problem. They also point out that a claim of a worst-case $\Theta(n \log n)$ algorithm in [2] is incorrect. The algorithm in Figure 7 resolves this question in the affirmative.

```
mean(0) = -∞
left(0) = 0
dist(0) = 0
do i = 1, n
    ŷ = y_i
    ℓ = i
    init(bottom(i)) = (y_i,w_i)
    top(i).sum = 0  {init(top(i))}
    top(i).weight = 0
    while (ŷ ≤ mean(ℓ-1)) do  {merge with level set to left}
        top(i).sum = top(i).sum + top(ℓ-1).sum  {merge(top(i),top(ℓ-1))}
        top(i).weight = top(i).weight + top(ℓ-1).weight
        merge(bottom(i),bottom(ℓ-1))
        ℓ = left(ℓ-1)
        (ŷ,w) = bottom(i).max
        while ((ŷ > mean(ℓ-1)) ∧ (top(i).weight+w < bottom(i).weight−w)) do
            {have not yet found median of level set, nor gone below left level set, move bottom(i).max
             to top(i)}
            top(i).sum = top(i).sum+w · ŷ  {add(top(i),ŷ,w}
            top(i).weight = top(i).weight+w
            pop(bottom(i))
            (ŷ,w) = bottom(i).max
        endwhile
    endwhile
    mean(i) = ŷ
    left(i) = ℓ
    newdist = top(i).sum−ŷ·top(i).weight+ŷ·bottom(i).weight−bottom(i).sum
    dist(i) = newdist+dist(ℓ-1)
enddo
```

Figure 7: L_1 Prefix Isotonic Regression

4 Unimodal Regression

It is a very simple process to modify the algorithm in Figure 2 to utilize prefix isotonic regression. This is given in Figure 8, producing an optimal algorithm for unimodal regression.

The time complexity of this algorithm is quite straightforward, since its total time is dominated by the time to perform the isotonic regressions.

Theorem 3 *Given weighted data* $\{(x_i, y_i, w_i) : i = 1, \ldots, n\}$*, the algorithm in Figure 8 will determine the unimodal regression in*

- $\Theta(n)$ *time for* L_2 *regression, and*
- $\Theta(n \log n)$ *time for* L_1 *regression.*

□

As noted earlier, the optimum mode is not necessarily unique. The algorithm in Figure 8 merely selects an arbitrary mode among the optimal ones, but in some applications one may want to apply secondary criteria to make this selection.

5 Final Comments

We have shown that the problem of determining the unimodal regression of a set of data can be optimally solved by using an approach based on prefix isotonic regression. This approach is quite similar to that in [4, 5, 7, 11, 13], but achieves greater efficiency by organizing the regression calculations into a systematic prefix calculation. It is also dramatically more efficient than the approach used in [8]. We had developed the prefix approach for the L_2 unimodal problem a few years ago to aid in creating new adaptive sampling designs for dose-response problems with competing failure modes (see [6]), and then encountered these articles which alerted us to the fact that the optimal algorithm was not previously known.

The prefix approach not only reduces the asymptotic time of unimodal regression, it does so in a manner which is not-

{distl(i): norm of increasing isotonic regression on indices 1 ... i}
{distr(i): norm of decreasing isotonic regression on indices i ... n}
{meanl(i): mean of level set containing x_i in increasing isotonic regression on 1 ... i}
{meanr(i): mean of level set containing x_i in decreasing isotonic regression on i ... n}
{left(i): left endpoint of level set containing x_i in increasing isotonic regression on 1 ... i}
{right(i): right endpoint of level set containing x_i in decreasing isotonic regression on i ... n}
{mode: mode of unimodal regression}

determine increasing_prefix_isotonic_regression, storing results in distl, meanl, left
determine decreasing_prefix_isotonic_regression, storing results in distr, meanr, right
mode = $\arg\min$ {distl(i)+distr(i): $1 \le i \le n$}
unimodal_reg = increasing_isotonic_reg{1 ... mode} $\cup$
decreasing_isotonic_reg{mode ... n}

Figure 8: Unimodal Regression

icable even for small data sets. Prefix isotonic regression on a set of values needs only the same amount of time as the fastest published algorithms for a single isotonic regression on the same values.

Most work on isotonic or unimodal regression has concentrated on L_2 regression, which is analytically and computationally simpler than L_1 regression. Even for regular L_1 isotonic regression, it was an open question whether it could be accomplished in $\Theta(n \log n)$ worst-case time (see the concluding remarks in [10]). This paper shows that this can indeed be achieved, and our approach can also be adapted to solve the L_1 isotonic regression problem when there are multiple weighted data values with the same independent coordinate, again taking $\Theta(n \log n)$ time for n data values. (This problem was examined in [10])

In a later paper, we will also give an algorithm for L_∞ prefix isotonic regression and unimodal regression, which also appears to supply the first algorithm for standard L_∞ isotonic regression. Isotonic or unimodal regression for other L_p norms appears to be of miniscule interest, and is computationally more challenging since it may not even be possible to exactly determine the mean of a level set. However, the basic approach used for L_1 prefix regression in Figure 7 should be of use for general metrics, moving single points from bottom() to top() until a mean is reached (though the mean may now be between data values).

One might also consider extending unimodal regression to index structures other than the linear ordering of the index set used here. For example, L_2 isotonic regression on rooted trees has been studied, and it is known that it can be determined in $\Theta(n \log n)$ time for an arbitrary tree of n nodes [9]. If the basic tree structure were known, but the root unknown, then a unimodal regression would be needed to locate the best root, and it is an open question as to whether this can be accomplished in $\Theta(n \log n)$ time. Further, the question of general L_p isotonic or unimodal regression on trees has apparently never been studied, perhaps because no applications have yet required this.

Acknowledgements

This work was supported in part by National Science Foundation grant DMS-9504980. I thank Janis Hardwick for her help in preparing this paper.

References

[1] Ayer, M., Brunk, H.D., Ewing, G.M., Reid, W.T. and Silverman, E. (1955), "An empirical distribution function for sampling with incomplete information", *Ann. Math. Statist.* **26**, pp. 641–647.

[2] Chakravarti, N. (1989), "Isotonic median regression: a linear programming approach", *Math. of Oper. Research* **14** (2), pp. 303–308.

[3] Corman, T.H., Leiserson, C.E., and Rivest, R.L. (1989), *Introduction to Algorithms*, MIT Press.

[4] Frisén, M. (1980), "Unimodal regression", *The Statistician* **35**, pp. 304–307.

[5] Geng, Z. and Shi, N.-Z. (1990), "Isotonic regression for umbrella orderings", *Appl. Statist.* **39**, pp. 397–424.

[6] Hardwick, J. and Stout, Q.F. (2000), "Optimizing a unimodal response function for binary variables", in *Optimum Design 2000*, A. Zhigljavsky, ed., Kluwer, in press.

[7] Mureika, R.A., Turner, T.R., and Wollan, P.C. (1992), "An algorithm for unimodal isotonic regression, with application to locating a maximum", Univ. New Brunswick Dept. Math. and Stat. Tech. Report 92–4.

[8] Pan, G. (1996), "Subset selection with additional order information", *Biometrics* **52**, pp. 1363–1374.

[9] Pardalos, P.M. and Xue, G.-L. (1999), "Algorithms for a class of isotonic regression problems", *Algorithmica* **23**, pp. 211–222.

[10] Pardalos, P.M., Xue, G.-L. and Yong, L., "Efficient computation of an isotonic median regression", *Appl. Math. Lett.* **8** (2), pp. 67–70.

[11] Pehrsson, N-G. and Frisén, M. (1983), "The UREGR procedure", Gothenburg Computer Central, Göteborg, Sweden.

[12] Robertson, T., Wright, F.T. and Dykstra, R.L. (1988), *Order Restricted Statistical Inference*, John Wiley and Sons.

[13] Turner, T.R. (1998), "S Function ufit to calculate the unimodal isotonic regression of a set of data", Univ. New Brunswick Dept. Math. and Stat.

[14] Turner, T.R. and Wollan, P.C. (1997), "Locating a maximum using isotonic regression", *Computational Statistics and Data Analysis* **25**, pp. 305–320.

A Exponential Time

To see that the approach used in [8] examines exactly 2^{n-1} orderings of n items, a 1-1 map between binary sequences of length $n-1$ and orderings examined will be created. Given an ordering, we create a binary sequence as follows: for each item after the first, if its index is less than the first, then put a 0 in the sequence, else put a 1 in the sequence. Thus, for example, the sequence $(x_3, x_2, x_4, x_5, x_1)$ would become the sequence 0110.

To determine the reverse map, given a sequence, we know that the index of the first item (i.e., the index of the maximum) is 1 more than the number of 0's in the sequence. The first 0 in the sequence corresponds to the index one less than the index of the maximum, the next 0 is 2 less than the maximum, etc. Similarly, the first 1 is one more than the maximum, the next 1 is 2 more, etc.

Thus it is easy to see that the mapping is 1-1 and onto, and hence the number of orderings equals the number of binary strings.