In *Computational Statistics and Data Analysis* 53 (2008), pp. 289–297

Unimodal Regression via Prefix Isotonic Regression

Quentin F. Stout

University of Michigan
Ann Arbor, MI 48109–2121

Abstract

This paper gives 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 shape-constrained nonparametric regressions, closely related to isotonic regression. For unimodal regression on n weighted points our algorithm for the L_2 metric requires only $\Theta(n)$ time, while for the L_1 metric it requires $\Theta(n \log n)$ time. For unweighted points our algorithm for the L_∞ metric requires only $\Theta(n)$ time. All of these times are optimal. Previous algorithms were for 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, determining the regression on all initial segments. The prefix approach reduces the total time required by utilizing the solution for one initial segment to solve the next.

Keywords and phrases: unimodal regression, umbrella ordering, isotonic regression, monotonic, prefix operation, scan, persistent data structure, pool adjacent violators (PAV)

1 Introduction

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

$$\begin{array}{ll} \left(\sum_{i=1}^n w_i |y_i - \widehat{y_i}|^p\right)^{1/p} & \text{if } 1 \le p < \infty \\ \max_{i=1}^n w_i |y_i - \widehat{y_i}| & \text{if } p = \infty \end{array} \tag{1}$$

subject to the increasing isotonic constraint that

$$\widehat{y_1} \le \widehat{y_2} \cdots \le \widehat{y_n}.$$

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 $\{(x_i, \widehat{y_i}) : i = 1, \ldots, n\}$ that minimizes Equation (1) subject to the unimodal constraint that there is an $m \in \{1, \ldots, n\}$ such that

$$\widehat{y_1} \le \widehat{y_2} \cdots \le \widehat{y_m} \ge \widehat{y_{m+1}} \cdots \ge \widehat{y_n},$$

i.e., such that $\{\widehat{y_i}\}$ is increasing on $1 \ldots m$ and decreasing on $m \ldots n$. The unimodal constraint is also called an umbrella ordering, and isotonic regression is often called monotonic regression, though in some application areas this term means the values decrease.

Isotonic regression does not yield a smooth curve, but rather a collection of level sets where the regression is constant. Figure 1 gives an example of an isotonic regression of a set of data with equal weights, where circles represent data points and lines represent level sets, with a filled circle representing a data point which is also a level set. Figure 2 shows a unimiodal regression.

By the *error of a regression* we mean the quantity in Equation (1). In the algorithms the value called error is actually the p^{th} power of this quantity in order to simplify calculations.

Both isotonic regression and unimodal regression are examples of nonparametric shape-constrained regression. Our interest in efficient unimodal regression was motivated by its repeated use in dose-response problems with competing failure modes [7, 10]. For such problems, as the dose increases the efficacy increases but the toxicity increases as well. The goal is to find the dose that maximizes the probability of being efficacious and non-toxic, and it is usually assumed that this probability distribution is unimodal. 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. See, for example, the references to water-level time-series data in [6] and to tree growth in [18]. The latter is another example of competing failure modes, where as trees in a newly planted grove grow, their "vigor" initially increases as they increase in size, but eventually starts decreasing as they compete for nutrients and light.

In Section 2 we examine previous work on the problem of determining unimodal regression. In Section 3 we introduce the notion of prefix isotonic regression, and in Sections 3.1 through 3.3 we develop algorithms for the L_2, L_1, and unweighted L_∞ versions of this problem, taking time $\Theta(n)$, $\Theta(n \log n)$, and $\Theta(n)$, respectively. These then yield unimodal algorithms of the same time complexity. All of these

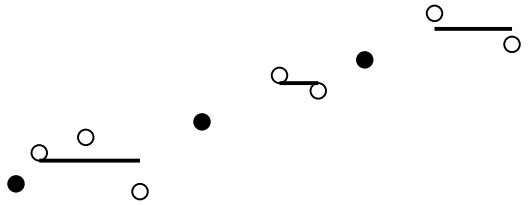


Figure 1: L_2 Increasing Isotonic Regression

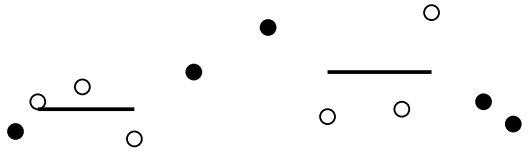


Figure 2: A Unimodal Regression

algorithms, both isotonic and unimodal, are optimal, and, except for the isotonic L_2 problem, all are the first optimal solutions to their problems. In Section 3.4 we examine the slightly different problem of determining the value at x_i of the isotonic regression on the first m values. 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 the data is given in order of increasing x_i values. If the data is not so ordered, then an initial sorting step, taking $\Theta(n \log n)$ time, is needed. Since the values of the x_i are irrelevant we simplify notation by assuming $x_i = i$.

2 Previous Work

It is well-known that the L_2 increasing isotonic regression can be determined in $\Theta(n)$ time. Apparently all published algorithms use the "pair adjacent violators" (PAV) approach [2]. 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 [15]. The PAV approach also produces the correct results for L_1 and L_∞.

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

{mode: location of mode of best unimodal fit}

do $0 = 1, n$
    errorl(i) = error_increasing_iso_regres($x_1 \ldots x_i$)
    errorr(i) = error_decreasing_iso_regres($x_i \ldots x_n$)
enddo

mode=$\arg\min$ {errorl(i)+errorr($i+1$): $1 \leq i \leq n$}

Figure 3: Best Previous Unimodal Regression Algorithm

Figure 4: Data Values with Nonunique Mode

the problem by trying each possible i as the location of the maximum, where the smallest error attained corresponds to the solution of the problem.

Testing each new value of i involved new calls to procedures to determine isotonic fits. The fastest and most straightforward approach, used in [4, 5, 9, 13, 17] and given in Figure 3, 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 takes $\Theta(n^2)$ time. A far less efficient approach, taking $\Theta(n2^n)$ time, was used in [10].

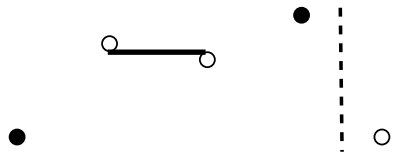
In general, the mode of the best unimodal fit is not unique. For example, if the weighted data values are as in Figure 4, then for any norm, one optimal unimodal fit has the leftmost point as mode and the mean of the other two as a level set, while another optimal fit uses a level set on the two left points and the rightmost point as mode. 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 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 $x_m \ldots x_n$, and thus the error of the unimodal regression is the sum of the errors of these two regressions. Figure 2 shows a unimodal regression where all of the data points have equal weights.

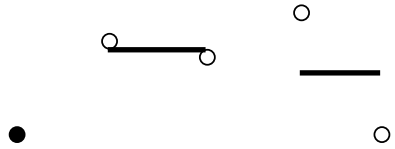
{left(i): left endpoint of level set containing x_i}
{mean(i): mean value of level set containing x_i}
{error(i): error of increasing isotonic regression on $x_1 \ldots x_i$}

mean(0) = $-\infty$
left(0) = 0
error(0) = 0
do $i = 1, n$
    initialize level set of i
    mean(i) = y_i
    left(i) = i
    while mean(i) $\leq$ mean(left(i)-1) do
        merge level set of left(i)-1 into level set of i
        left(i) = left(left(i)-1)
    endwhile
    levelerror = weighted error of mean(i) to $(y_{\text{left}(i)}, w_{\text{left}(i)}), \ldots (y_i, w_i)$
    error(i) = levelerror+error(left(i)-1)
enddo

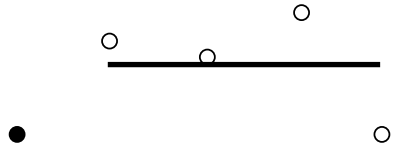
Figure 5: Prefix Isotonic Regression



Regression of first 4 points, i.e., Iso_4



Level sets of 5 points after 1st while-loop iteration



Level sets of 5 points after 2nd (final) iteration, i.e., Iso_5

Figure 6: Constructing Iso_5 from Iso_4

3 Prefix Isotonic Regression

By *determining an isotonic regression* we mean determining the error of the regression and the extents and regression values of the level sets. Given n real-valued weighted data values $\{(x_i, y_i, w_i) : 1 \leq i \leq n\}$ with nonnegative real weights w_i, and given a metric μ on the reals, let Iso_m denote the μ isotonic regression on $\{(x_i, y_i, w_i) : 1 \leq i \leq m\}$. The μ *prefix isotonic regression problem* is to determine Iso_m for all $1 \leq m \leq n$.

Note that prefix isotonic regression determines exactly the set of increasing isotonic regression problems examined by [4, 5, 9, 13, 17]. 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 a subroutine to solve each subproblem. Prefix operations, also called *scan operations*, are utilized as building blocks for a variety of efficient algorithms. 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 in increasing indexing order, adding them to the previous solution. The loop invariant is that at the start of the do-loop, Iso_{i-1} has been determined. In right to left order, it consists of:

- the level set of all points with indices in the interval [left($i-1$), $i-1$], with value mean($i-1$)
- the level set of all points with indices in the interval [left(left($i-1$) -1), left($i-1$)–1], with value mean(left($i-1$) -1)
- the level set of all points with indices [left(left(left($i-1$) -1) -1), left(left($i-1$) -1)–1] with value mean(left(left($i-1$) -1) -1)

and so on. Further, the error of this regression is error($i-1$).

If the value of the new point, y_i, is greater than the mean of the level set containing x_{i-1}, then Iso_i is Iso_{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 Iso_i follows immediately from the PAV property mentioned in Section 2. Figure 6 illustrates this process.

After the algorithm in Figure 5 has completed, for any index m, $1 \leq m \leq n$, Iso_m has error error(m) and its level sets can be recovered in $\Theta(\ell)$ time from the values stored in left and mean, where ℓ is the number of level sets. The recovery proceeds exactly as above, in right-to-left order. Note that

when the point at index i is added, only the left(i), mean(i), and error(i) entries are updated, with the earlier entries unchanged since values for other indices within the merged level set will never be referred to again. The left, mean, and error arrays form a *persistent data structure*, allowing one to rapidly recreate the intermediate regressions.

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 error of the merged level sets. As will be shown in Sections 3.1, 3.2 and 3.3, efficiently implementing these operations depends upon the metric.

> **Observation:** If the operations of determining the mean and error in the while-loop can be accomplished in $O(f(n))$ time for an increasing function f, then the algorithm requires only $O(n \cdot f(n))$ 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 be merged at most $n-1$ times. All of the other operations within the while-loop take constant time per iteration, and the operations outside the while-loop take a constant time per iteration of i.

Notice that if one determines the mean and error 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. 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 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 error of the L_2 level sets. Fortunately, it is well known that the algebraic properties of this metric make this a simple task, as is shown in Figure 7. These operations require only constant time, and hence by the Observation the algorithm takes only $\Theta(n)$ time.

3.2 L_∞ Prefix Isotonic Regression

Efficient algorithms for weighted L_∞ isotonic regression are rather complicated, see [8], so here we only consider the case

{sumwy(i): weighted sum of values in x_i's level set}
{sumwy2(i): weighted sum of squares of values in x_i's level set}
{sumw(i): sum of weights of x_i's level set}

to initialize level set of i:
    sumwy(i) = $w_i \cdot y_i$
    sumwy2(i) = $w_i \cdot y_i^2$
    sumw(i) = w_i

to merge level set of j into level set of i:
    sumwy(i) = sumwy(i)+sumwy(j)
    sumwy2(i) = sumwy2(i)+sumwy2(j)
    sumw(i) = sumw(i)+sumw(j)
    mean(i) = sumwy(i)/sumw(i)

levelerror = sumwy2(i)$-$sumwy(i)^2/sumw(i)

Figure 7: Modifications for L_2 Regression

where all of the weights are 1. The unweighted L_∞ mean of values $\{y_1, \ldots, y_k\}$ is $(y_{\min} + y_{\max})/2$, where $y_{\min} = \min\{y_1, \ldots, y_k\}$ and $y_{\max}$ is defined similarly. The error of using this mean is $(y_{\max} - y_{\min})/2$.

The simplistic nature of the L_∞ mean and error makes the isotonic regression particularly easy. We introduce functions maxy and miny, as shown in Figure 8, where maxy(i) is the maximum, and miny(i) is the minimum, of the y values in the level set containing i. These operations take only constant time, and hence by the Observation the total time is only $\Theta(n)$.

While the regression determined by Figure 8 is quite natural, it is not the only optimal L_∞ regression. For example, if the data values are (1, 4, 2, 6), then the algorithm will produce the fitted values (1, 3, 3, 6), with error 1. However, another solution with the same error is (0, 3, 3, 7), and there are infinitely many solutions with optimal error. It is easy to see that the solution found here has the property that if a level set L with value y is created on indices $i \ldots j$, then L is an optimal L_∞ isotonic regression on the values for those indices. In some applications one may prefer to specify a criterion to select among the optimal regressions, though it is usually difficult to achieve a given criterion for all prefix regressions without substantially more time and revisions from one prefix to the next.

3.3 L_1 Prefix Isotonic Regression

Weighted L_1 regression is more complex than the previous metrics. 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 smallest such value. In an

{miny(i): minimum value in x_i's level set}
{maxy(i): maximum value in x_i's level set}

to initialize level set of i:
    miny(i) = y_i
    maxy(i) = y_i

to merge level set of j into level set of i:
    miny(i) = min{miny(i), miny(j)}
    maxy(i) = max{maxy(i), maxy(j)}
    mean(i) = [miny(i)+maxy(i)]/2

levelerror = (maxy(i)−miny(i))/2
error(i) = max{error(left(i)-1), levelerror}

Figure 8: Modifications for Unweighted L_∞ Regression

{root(i): root of tree containing all y values in x_i's level set}
{p.y: the y value stored in node p}
{p.w: the weight corresponding to p.y}
{p.sumw: sum of weights in p's subtree}
{p.sumwy: sum of $w \cdot y$ in p's subtree}

to initialize level set of i:
    initialize tree(i) to have single node, root
    root.y = y_i
    root.w = w_i
    root.sumw = w_i
    root.sumwy = $w_i \cdot y_i$

to merge level set of j into level set of i:
    merge tree(j) and tree(i), updating sumw and sumv
        fields while merging
    determine mean(i) from tree(i)

determine levelerror from tree(i) and mean(i)

Figure 9: Modifications for L_1 Regression

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 that is linear in the number of values, a naive approach based on this 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. A $\Theta(n \log n)$ algorithm is presented in [1], but its use of scaling does not seem to translate into an efficient algorithm for the prefix problem. The author presented a prefix algorithm in [16], but the following is much simpler and can be applied in more general settings. This approach is outlined in Figure 9.

For a level set corresponding to (value,weight) pairs $\{(y_j, w_j), (y_{j+1}, w_{j+1}), \ldots, (y_i, w_i)\}$, create a red-black tree T containing $i-j+1$ nodes which have as keys the values, i.e., the tree is ordered by the values. Red-black trees are not specifically required, in that other balanced tree structures (AVL, weight-balanced, etc.) could be used equally well. If p is a node of the tree, then $p.\mathrm{y}$ represents the value it contains, and $p.\mathrm{w}$ the value's associated weight. Each node also has additional fields:

$$\begin{aligned} p.\mathrm{sumw} &= \sum q.\mathrm{w} \\ p.\mathrm{sumwy} &= \sum q.\mathrm{w} \cdot q.\mathrm{y} \end{aligned}$$

where the sums are over all nodes q in the subtree rooted at p. Given T, an easy top-down path traversal using $p.\mathrm{sumw}$ can determine a weighted median in time linear in the height of the tree, i.e., in $\Theta(i-j)$ time. Search trees with additional fields such as $p.\mathrm{sumw}$ and $p.\mathrm{sumwy}$ are sometimes called *augmented trees* and are often used for dynamic order statistics such as this.

To determine the error of the regression on a level set, let m be a weighted median. Let $W_<$ ($W_>$) be the sum of all weights corresponding to values less than (greater than) m, and $WY_<$ ($WY_>$) be the sum of all $w \cdot y$ products corresponding to values less than (greater than) m. The error of the regression is

$$WY_> - m \cdot W_> + m \cdot W_< - WY_<$$

Once m has been determined, another top-down path traversal involving $p.\mathrm{sumw}$ and $p.\mathrm{sumwy}$ can be used to determine $W_<$, $W_>$, $WY_<$, and $WY_>$ in time linear in the height of the tree, i.e. in $\Theta(\log n)$ time. Analyzing the time to do all tree mergers is a bit more delicate. A straightforward merger of trees of size s and t, where $s \geq t$, repeatedly inserts the elements of the smaller tree into the larger, taking $\Theta(t \log s)$ time, which would result in $\Theta(n \log^2 n)$ worst-case total time. However, the merge procedure in [3] takes $\Theta(1 + t \cdot \log(s/t))$ time, and their results show that all of the mergers can be done in $\Theta(n \log n)$ total time. Standard extensions to their procedure allow one to maintain all of the fields associated with each node without altering the time required, and thus the total time is $\Theta(n \log n)$ time. This improves upon the algorithm in [12], which takes $\Theta(n \log^2 n)$ time.

To show that $\Theta(n \log n)$ time is optimal, note that L_1 prefix isotonic regression is as hard as sorting real numbers. To see this, let $\{y_i : 1 \leq i \leq n\}$ be any set of real numbers, and let $\{y_i^* : 1 \leq i \leq n\}$ be the same set in decreasing order. Let $u = -1 + \min_i y_i$ and $v = 1 + \max_i y_i$. Then for the weighted sequence $(0, v, n+1)$, $(1, y_1, 1)$, $(2, y_2, 1)$, $\ldots (n, y_n, 1)$, $(n+1, u, 2)$, $(n+2, u, 2)$, $\ldots$, $(2n+1, u, 2)$, Iso_{n+i} is a single level set of value y_i^*, for $1 \leq i \leq n$. This is because at $n+i$ there are at least $n+i+1$ weighted values

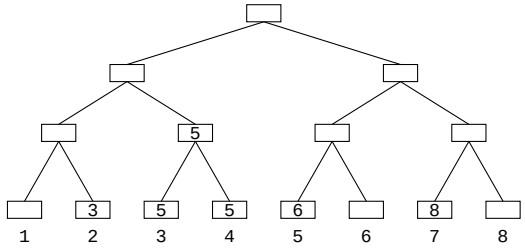


Figure 10: Coverage Tree

greater than or equal to y_i^* ($n+1$ of value v, and $y_1^* \ldots y_i^*$), and at least $n+i+1$ less than or equal to ($y_i^* \ldots y_n^*$ and $2i$ of value u). (The use of "at least" and "equal to" takes care of the possibility of ties.) Thus y_i^* is the weighted median. In Iso_{n+i} no prefix has smaller median nor does any final interval have higher median, so there is a single level set. Thus determining these regressions yields the values in decreasing sorted order, so the algorithm is optimal.

3.4 Pointwise Evaluation

There are other reasonable goals for prefix isotonic regression. For example, once the regressions have been computed, one might want to be able to determine $\text{Iso}_m(x)$ for $1 \le m \le n$ and arbitrary x. One can do this in $\Theta(\log n)$ time by creating another persistent structure in the general prefix algorithm, adding only $\Theta(n)$ time to the algorithm. Note that if x is not an abscissa of one of the data points then the value of $\text{Iso}_m(x)$ is the interval $[\text{Iso}_m(x_i), \text{Iso}_m(x_{x+1})]$ if $x_i < x < x_{i+1}$, or $(-\infty, \text{Iso}_m(x_1)]$ if $x < x_1$, or $[\text{Iso}_m(x_n), \infty)$ if $x > x_n$. Thus it suffices to be able to determine $\text{Iso}_m(x_i)$ for arbitrary index i. Note that given x one can determine the appropriate i in $\Theta(\log n)$ time.

The following is a sketch of the procedure. The data structure is illustrated in Figure 10, where the leaf nodes correspond to the indices of the values. This tree is maintained in addition to the data structures in Figure 5. Let smallest(p) and largest(p) denote the indices of the smallest and largest elements in the subtree with root p, and let cover(p) denote the smallest $r > \text{largest}(p)$ such that all indices beneath p are contained in the same level set in Iso_r, i.e., they are contained in the level set containing $\text{Iso}_r(r)$. Note that all elements of p are contained in the level set containing r in Iso_s, for $s \ge r$, and that $\text{Iso}_m(i) = \text{Iso}_s(s)$ for the largest $s \le m$ such that the level set containing i was merged with the level set containing s. Let R denote this value.

Initially all nodes have an empty cover value. Whenever a level set with indices in the interval $[a, b]$ is merged with level set $[c, d]$, $b < c$, the node corresponding to b has its cover value set to d. Let p denote this node and let

{Throughout, i was in r's level set in Iso_r}
{R is the minimal index such that $\text{Iso}_m(i)$=$\text{Iso}_R(R)$}

p=i's node
r=cover(i)
while $r >$ largest element under p do
{R is in a subtree to the right of p}
    p=parent of p
    if cover(p) $\le m$ then r=max{r,cover(p)}
    q=right child of p
    if cover(q) $\le m$ then r=max{r,cover(q)}
end while

while p not a leaf {R is in p's subtree}
    q=left child of p
    if cover(q) $\le m$ then r=max{r,cover(q)}
    p=child of p containing r
end while

$\text{Iso}_m(i)$ = $\text{Iso}_r(r)$ = mean(r)

Figure 11: Algorithm to Determine Iso_m(i)

$q = \text{parent}(p)$. If $[\text{smallest}(q), \text{largest}(q)]$ is a subset of $[a, d)$ then set $\text{cover}(q) = d$, $p = q$, $q = \text{parent}(q)$, and repeat the process. If it is not a subset then stop because no higher node can be newly covered. Note that q could not have been previously covered. The values in the nodes in Figure 10 are the cover values that would result if during the prefix construction with 8 data points, the level sets were: 1:{1}; 2:{1}{2}; 3:{1}{2,3}; 4:{1}{2,3}{4}; 5:{1}{2,3,4,5}; 6:{1}{2,3,4,5,6}; 7:{1}{2,3,4,5,6}{7}; 8:{1}{2,3,4,5,6}{7,8}

The total time to compute the cover values is $\Theta(n)$, since whenever an upward path is being followed it does not use any edge previously used and there are only $n-1$ edges.

The second loop in Figure 11 shows how this tree is used. By the end of the first loop p is the lowest node that has both i and R beneath it. To see that R is beneath p, if i is in the right subtree of p then the value of r when p was reached is greater than largest(p) and the loop would have continued. If i is in the left subtree and R is not beneath p then the level set in $\text{Iso}_R(R)$ containing R also contained all elements in the right subtree of p since they are between i and R. Hence that subtree is covered, so the loop would have continued because the value of r would have been larger than largest(p).

A similar argument can be applied to the second loop, showing that at all times R will be under p. This does not say that R is known when p is encountered, merely that it is beneath p. Since p keeps decreasing in height, eventually it is a leaf node, i.e., the node corresponding to R.

Implementing this tree is straightforward. To store the cover value of the leaf nodes use the array lcover[1:n] where

lcover(i) is the value of the node corresponding to i. For the nonleaf nodes use the array tcover[1:$n-1$]. Let $i \in [1, n-1]$ and let k be the largest power of 2 evenly dividing i. Then tcover(i) stores the value of the node over $[i-2^k+1,\ i+2^k]$. It is easy to show that this is a 1–1 correspondence between elements of tcover and nonleaf nodes in the tree.

3.5 Time Required

Combining the algorithms in the previous sections gives the following:

Theorem 1 *Given weighted data* $\{(x_i, y_i, w_i) : i = 1, \ldots, n\}$ *sorted by* x_i*, the prefix isotonic regression problem can be solved in*

- $\Theta(n)$ *time for the* L_2 *metric,*
- $\Theta(n)$ *time for the* L_∞ *metric with unweighted date,*
- $\Theta(n \log n)$ *time for the* L_1 *metric.*

Further, given this solution, for all $1 \leq m \leq n$*,*

- *In constant time one can determine the error of, and in* $\Theta(\ell)$ *time can determine the level sets of,* Iso$_m$*, where* ℓ *is the number of level sets.*
- *In* $\Theta(\log m)$ *time one can determine* Iso$_m(x)$ *for arbitrary* x*.*

□

Note that one can also use the cover information to determine $\text{Iso}_m^{-1}(y)$ in $\Theta(\log m)$ time.

4 Unimodal Regression

It is a very simple process to modify the algorithm in Figure 3 to utilize prefix isotonic regression. The errorl values are calculated via a standard prefix increasing isotonic regression, and the errorr values are calculated via a prefix increasing isotonic regression going through the data in right-to-left order. The time complexity of this algorithm is quite straightforward since its total time is dominated by the time to perform the isotonic regressions.

Theorem 2 *Given weighted data* $\{(x_i, y_i, w_i) : i = 1, \ldots, n\}$*, their unimodal regression can be determined in*

- $\Theta(n)$ *time for* L_2 *regression*
- $\Theta(n)$ *time for* L_∞ *regression on unweighted data*
- $\Theta(n \log n)$ *time for* L_1 *regression.*

□

As noted earlier, the optimum mode is not necessarily unique. The algorithm in Figure 3 merely selects an arbitrary mode among the optimal ones, but in some applications one may want to apply secondary criteria to make this selection, or to list all optimal modes.

5 Final Comments

It has been 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, 9, 13, 17], but achieves greater efficiency by organizing the regression calculations into a systematic prefix calculation. The prefix approach not only reduces the asymptotic time of unimodal regression, it does so in a manner which is noticeable even for small data sets. Prefix isotonic regression on a set of values needs only the same amount of time as regular isotonic regression, and unimodal regression needs only twice as much.

Prefix isotonic regression is of interest in its own right. For example, if the data is from an ongoing time series, then it allows one to continually update the regression using, on average, only a constant number of calculations per observation. Further, the algorithm in Section 3.4 allows one to quickly determine how later observations have changed the regression on earlier points.

One can extend unimodal regression to index structures other than the linear ordering of the index set used here. For example, for an arbitrary rooted tree of n nodes, the L_2 isotonic regression can be determined in $\Theta(n \log n)$ time [11]. An algorithm for L_1 regression on rooted trees has also been presented, but there was no analysis of its time complexity [14]. L_∞ regression on rooted trees can be determined in $\Theta(n \log^2 n)$ time by using the general digraph algorithm in [8]. If the tree structure was given as an undirected graph with no root, then a unimodal regression would be needed to locate the best root.

Acknowledgements

This work was supported in part by National Science Foundation grant DMS–0072910. A preliminary version of portions of this paper appeared in [16].

The author thanks the reviewers for their helpful comments.

References

[1] Ahuja, RK and Orlin, JB (2001), "A fast scaling algorithm for minimizing separable convex functions sub-

ject to chain constraints", *Operations Research* **49**, pp. 784–789.

[2] Ayer, M, Brunk, HD, Ewing, GM, Reid, WT and Silverman, E (1955), "An empirical distribution function for sampling with incomplete information", *Ann. Math. Statist.* **26**, pp. 641–647.

[3] Brown, MR and Tarjan, RE (1979), "A fast merging algorithm", *J. Assoc. Comp. Mach.* **26**, pp. 211–226.

[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] Haiminen, N, and Gionis, A (2004), "Unimodal segmentation of sequences", *Proc. 4*$^{\text{th}}$ *IEEE Int'l. Conf. Data Mining*, pp. 106–113.

[7] Hardwick, J and Stout, QF (2000), "Optimizing a unimodal response function for binary variables", in *Optimum Design 2000*, A. Atkinson, B. Bogacka, and A. Zhigljavsky, eds., Kluwer, 2001, pp. 195–208.

[8] Kaufman, Y and Tamir, A (1993), "Locating service centers with precedence constraints", *Discrete Applied Mathematics* **47**, pp. 251–261.

[9] Mureika, RA, Turner, TR, and Wollan, PC (1992), "An algorithm for unimodal isotonic regression, with application to locating a maximum", Univ. New Brunswick Dept. Math. and Stat. Tech. Report 92–4.

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

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

[12] Pardalos, PM, Xue, G-L and Yong, L (1995), "Efficient computation of an isotonic median regression", *Appl. Math. Lett.* **8** (2), pp. 67–70.

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

[14] Qian, S (1996), "An algorithm for tree-ordered isotonic median regression", *Stat. and Prob. Letters* **27**, pp. 195–199.

[15] Robertson, T, Wright, FT and Dykstra, RL (1988), *Order Restricted Statistical Inference*, John Wiley and Sons.

[16] Stout, QF (2000), "Optimal algorithms for unimodal regression", *Computing Science and Statistics* **32**.

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

[18] Turner, TR and Wollan, PC (1997), "Locating a maximum using isotonic regression", *Computational Statistics and Data Analysis* **25**, pp. 305–320.