In *Proceedings Interface 2012: Future of Statistical Computing*

Optimal Reduced Isotonic Regression

Janis Hardwick and Quentin F. Stout

jphard@umich.edu qstout@umich.edu

University of Michigan
Ann Arbor, MI

Abstract

Isotonic regression is a shape-constrained nonparametric regression in which the ordinate is a non-decreasing function of the abscissa. The regression outcome is an increasing step function. For an initial set of n points, the number of steps in the isotonic regression, m, may be as large as n. As a result, the full isotonic regression has been criticized as overfitting the data or making the representation too complicated. So-called "reduced" isotonic regression constrains the outcome to be a specified number of steps, b. The fastest previous algorithm for determining an optimal reduced isotonic regression takes $\Theta(n + bm^2)$ time for the L_2 metric. However, researchers have found this to be too slow and have instead used approximations. In this paper, we reduce the time for the exact solution to $\Theta(n+bm \log m)$. Our approach is based on a new algorithm for finding an optimal b-step approximation of isotonic data. This algorithm takes $\Theta(n \log n)$ time for the L_1 and L_2 metrics.

Keywords: step function, histogram, piecewise constant approximation, reduced isotonic regression

1 Introduction

Isotonic regression is an important form of nonparametric regression that allows researchers to relax parametric assumptions and replace them with a weaker shape constraint. A real-valued function f is *isotonic* iff it is nondecreasing; i.e., if for all x_1, x_2 in its domain, if $x_1 < x_2$ then $f(x_1) \leq f(x_2)$. Similarly, a function is *antitonic* iff it is nonincreasing. Such functions are also called monotonically ordered, where sometimes the term is used to mean isotonic and other times it means either isotonic or antionic. Thousands of references to isotonic regression cite the fundamental books of Barlow et al. [4] and Robertson et al. [15]. Further, there are tens of thousands of references to the search terms "isotonic" and "monotonic" regression.

Fitting isotonic functions to data is useful in situations in which the independent variable has an ordering but no natural metric, such as S, M, L, XL sizes. Since the only important property of the domain is its ordering, we assume that it is the integers $1 \ldots n$ for some n, and use $[i : j]$, $1 \leq i \leq j \leq n$ to denote the range $i \ldots j$. By *weighted values* $(\mathbf{y}, \mathbf{w})$ on $[1 : n]$, we mean weighted values (y_i, w_i), $i \in [1 : n]$, where the y values are arbitrary real numbers and the w values (the weights) are positive real numbers. Given weighted values $(\mathbf{y}, \mathbf{w})$ and a real-valued function f on $[1 : n]$, the L_p approximation error of f is

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

An L_p *isotonic regression* is an isotonic function that minimizes the L_p error among all isotonic functions and hence is optimal in that regard. Figure 1 gives an example of an isotonic regression.

Isotonic regressions are step functions where the number of steps, $m \leq n$, is determined by the data. In certain cases, there is criticism that such functions can overfit the data [16, 17] or produce a result with too many steps [7]. Consequently, some researchers utilize isotonic regressions that restrict the number of steps. Schell and Singh [17] have referred to such functions as *reduced isotonic regressions*.

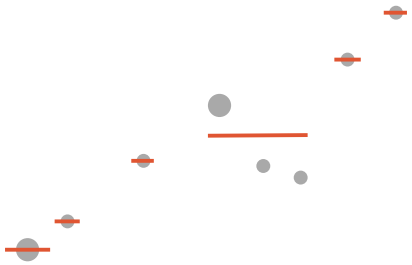

Figure 1: Isotonic Regression, Where Size Indicates Weight

Restricting the number of steps is also a central issue in the general area of approximation by step functions. This concern arises in a wide variety of settings, such as segmentation of time series and genomic data [9, 11, 21], homogenization [6], histogramming [8, 10, 12, 14], x-monotone functions [22] and piecewise constant approximations [5]. A survey of a variety of types and applications of histograms to database queries appears in Poosala et al. [14].

In general, a function f is an *optimal L_p b-step approximation*, $b = 1, \ldots, n$, iff it minimizes the L_p error over all functions with b steps. Figure 2 is an example of an optimal approximation by a step function. Since the specific value of p will always be clear, we refer to such functions merely as optimal b-step approximations. Generally, optimal b-step approximations are not unique. For example, with unweighted values 1, 2, 3 on $[1:3]$ and $b = 2$, for any p the function which is 1.5 on $[1:2]$ and 3 at 3 is optimal, as is the function which is 1 at 1 and 2.5 on $[2:3]$.

Here we are concerned with computing reduced isotonic regressions. An isotonic function f is an *optimal L_p b-step reduced isotonic regression*, $b = 1, \ldots, m \leq n$, iff it minimizes the L_p error over all isotonic functions having b steps. Again we omit the "L_p" when citing such functions. Our main results are faster algorithms for generating optimal b-step isotonic regressions. In doing this, we draw upon approaches used for generating optimal step approximations. In 1958 Fisher [6] gave a dynamic programming algorithm for determining an optimal b-step approximation in $\Theta(bn^2)$ time. His algorithm was for L_2 and can easily be extended to the L_1 metric, taking $\Theta((b+\log n)n^2)$ time. It remains the fastest known algorithm for general data and has been widely used and repeatedly rediscovered. However, a problem cited by many researchers is that the quadratic time (in n) of Fisher's algorithm makes it too slow for many applications [7, 8, 9, 11, 12, 21].

In Section 2.2, we examine the special case in which the values are themselves isotonic. In this case, optimal b-step approximations and optimal b-step reduced isotonic regressions are the same. The "unrestricted maximum homogeneity" approximations of Fisher and the optimal variable-width "serial histograms" of Ioannidis [10] are instances of optimal b-step approximations of isotonic data. Here we show that for L_1 and L_2, optimal b-step isotonic regressions can be found in $\Theta(bn \log n)$ time by a dynamic programming approach that exploits isotonic properties.

Most previous work for reduced isotonic regression with general data has only produced suboptimal approximations. It appears that the only published algorithm that produces optimal reduced isotonic regressions is due to Haiminen, Gionis and Laasonen [7]. Their algorithm is for the L_2 metric, taking $\Theta(n + bm^2)$ time, where m is the number of pieces of the unrestricted isotonic regression and b is the number of steps in the reduced isotonic regression. (To lessen confusion, we use "pieces" to refer to the steps of the unrestricted isotonic regression.) The algorithm is based on first finding an unrestricted isotonic regression and then applying Fisher's algorithm to its pieces. In Section 3, we reduce this time to $\Theta(n + bm \log m)$ by applying our algorithm for isotonic data. Section 4 we give an algorithm for L_1 reduced isotonic regression based on the same approach. However, it produces an approximation which in practice is very good but not necessarily optimal.

Section 5 contains some final comments.

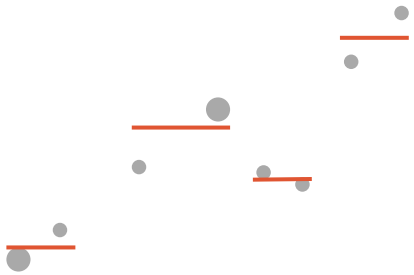

Figure 2: Optimal 4-Step Approximation of the Weighted Values in Figure 1

2 Approximation by Step Functions

As discussed, the isotonic condition implies that the regression function is a sequence of steps or pieces. A real-valued function f on $[1:n]$ is a *b-step function*, $1 \leq b \leq n$, iff there is a set of indices $j_0 = 0 < j_1 \ldots < j_b = n$ and real values C_k, $k \in [1:b]$, such that $f(x_i) = C_k$ for $i \in [j_{k-1}+1 : j_k]$. If f is isotonic then the steps are nondecreasing with $C_1 \leq C_2 \ldots \leq C_b$. An approximation with fewer than b steps can be converted to a b-step approximation by merely subdividing steps, and thus we do not differentiate between "b steps" and "no more than b steps".

For $1 \leq p < \infty$, an optimal L_p step function has the additional property that C_k is an L_p mean of the weighted values on $[j_{k-1}+1 : j_k]$. Since we are only concerned with optimal approximations; from now on, whenever a function has a step $[i:j]$, then its value on that step is an L_p weighted mean of the data. For a given $1 \leq p < \infty$, let $\mathsf{err}^p(i,j)$ denote the p^{th} power of the L_p error when using an L_p mean as the approximation of the weighted values on $[i:j]$. Minimizing the sum of these values is the same as minimizing the L_p approximation error, i.e., determining steps $j_0 = 0 < j_1 \ldots < j_b = n$ that minimize $\sum_{k=1}^{b} \mathsf{err}^p(j_{k-1}+1, j_k)$ is the same as determining steps $j_0 = 0 < j_1 \ldots < j_b = n$ and values $C_1 \ldots C_b$ that minimize $\left(\sum_{k=1}^{b} \sum_{i=j_{k-1}+1}^{j_k} w_i |y_i - C_k|^p\right)^{1/p}$. Thus from now on only the $\mathsf{err}^p(\cdot)$ values will be used.

Given a set of weighted values, their L_p mean is unique for $1 < p \leq \infty$, while for L_1 it is a weighted median value, where the weighted medians might form an interval. Thus L_1 isotonic regressions are not unique. For L_∞, the situation is even less constrained. For example, for unweighted values 5, 1, 4, an L_∞ isotonic regression must have values 3, 3, x, where $x \in [3, 6]$; i.e., a regression value on a piece need not be the mean of the piece, although it must be on at least one piece with maximum error.

2.1 Arbitrary Data

Fisher's [6] dynamic programming approach to determining an optimal b-step approximation for $1 \leq p < \infty$ is based on the observation that if f is an optimal b-step approximation of the data, with a first step of $[1:j]$, then f restricted to $[j+1:n]$ is an optimal $(b-1)$-step approximation of the data on $[j+1:n]$. This is obvious since, if it were not optimal on $[j+1:n]$, then replacing it with an optimal $(b-1)$-step approximation on that interval would reduce the error.

Let $\mathsf{e}(i,c)$ denote the sum of the $\mathsf{err}^p(\cdot)$ values of the pieces of an optimal c-piece approximation on $[i:n]$. Fisher's observation yields the equation:

$$\mathsf{e}(i,c) = \min\{\mathsf{err}^p(i,j) + \mathsf{e}(j+1, c-1) : i \leq j \leq n - c + 1\}$$

By recording the j that minimizes this one can generate the optimal approximation. This leads to Algorithm A. The time required is $\Theta(bn^2)$ plus the time to compute the $\Theta(n^2)$ $\mathsf{err}^p()$ values.

```
for i = 1 to n
    e(i, 1) = err^p(i, n),   e_end(i, 1) = n
for c = 2 to b
    for i = 1 to n − c + 1
        e(i, c) = min{err^p(i, j) + e(j+1, c−1) : i ≤ j ≤ n − c + 1}
            {record minimizing j in e_end(i, c)}
    end for i
end for c
```

Algorithm A: Optimal b-Step Approximation Of Arbitrary Data

2.2 Isotonic Data

Reducing the time of Algorithm A requires reducing the number of $\mathsf{err}^p()$ values referenced. It is not known how to do this for arbitrary data, but for isotonic data there is a special property that can be exploited. This is given in Lemma 2.2 and will be used to prove:

Theorem 2.1 *Given n isotonic weighted values $(\mathbf{y}, \mathbf{w})$ and number of steps $b \leq n$, for the L_1 and L_2 metrics one can find an optimal b-step approximation, and hence find an optimal b-step isotonic regression, in $\Theta(bn \log n)$ time.*

For isotonic data, the fact that values are nondecreasing allows one to make inferences concerning the means of intervals. For example, the weighted values on $[i:j]$ have an L_p mean no larger than that of the values on $[i:j+1]$. This forms the basis of the following lemma:

Lemma 2.2 *For any $1 \leq p < \infty$, given isotonic weighted values $(\mathbf{y}, \mathbf{w})$ on $[1:n]$ and $1 \leq b \leq n-1$, let f be an optimal b-step approximation on $[1:n]$, with first step ending at j_f. Then there is an optimal b-step approximation g on $[2:n]$ with first step ending at j_g, where $j_f \leq j_g$.*

Proof: Let h be an optimal b-step L_p approximation on $[2:n]$ with first step ending at j_h. If $j_h \geq j_f$ then we are done, so assume otherwise. Let I_h denote the interval $[2:j_h]$ and I_f denote the interval $[2:j_f]$. Let f^- be the b-step function on $[2:n]$ with first step being I_f and the others, on $[j_f+1:n]$, being f. Since h is optimal, the error of f^- must be at least as large as that of h. If they are equal then choosing $g = f^-$ satisfies the lemma, so assume the error of f^- is strictly greater than that of h.

The mean of the values on I_f is at least as large as that on I_h, and it has strictly greater weight. Further, the value at 1 is no larger than either mean. Therefore $\mathsf{err}^p(1 \cup I_f) - \mathsf{err}^p(I_f) \geq \mathsf{err}^p(1 \cup I_h) - \mathsf{err}^p(I_h)$, with equality iff all values on $[1:j_f]$ are the same and the errors are all 0. This implies that the error of f^- equals the error of h. Since this case has already been eliminated we assume the inequality is strict. Let h^+ be the function with first step $1 \cup I_h$, and h on the remainder. Since f has a first step of $1 \cup I_f$, and which on the remaining values is f^-, the error f is $\mathsf{err}^p(1 \cup I_f) + \mathrm{error}(f^-) - \mathsf{err}^p(I_f)$. Similarly the error on h^+ is $\mathsf{err}^p(1 \cup I_h) + \mathrm{error}(h) - \mathsf{err}^p(I_h)$. Thus the error on f is greater than the error on h^+, contradicting the optimality of f. Therefore the error of f^- cannot be larger than that of h. □

The above does not always hold if the values are not isotonic. For example, for L_2 with $b = 2$ and unweighted values 4, 0, 4, 7 on $[1:4]$, the optimal first step for the values on $[1:4]$ is $[1:3]$, with value 8/3. However, the optimal first step for the values on $[2:4]$ is $[2:2]$, with value 0. Further, it need not hold for L_∞ even if the values are isotonic: e.g., for values -1, 1, 10, 20 and weights 10, 10, 1, 1 on $[1:4]$, one optimal 2-step approximation on $[1:4]$ has $[1:3]$ as its first step, with value 0, and value 20 on $[4:4]$. However, any optimal 2-step approximation on $[2:4]$ must have a first step of $[2:2]$ with a value in $[0.5, 1.5]$, and a second step on $[3:4]$ with value 15. The above can be made to hold for L_∞ and isotonic values if one uses only strict L_∞ regression, i.e., regression which is the

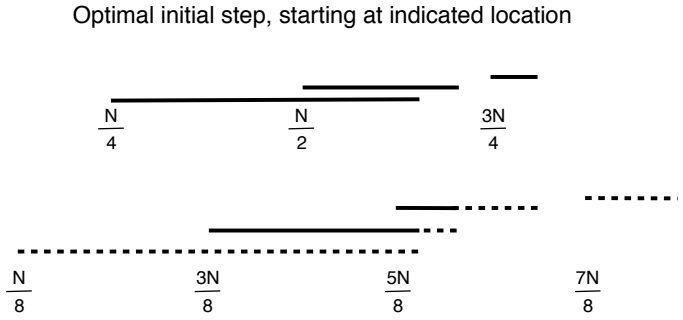

Figure 3: Possible Endpoints of Odd Multiples of 1/8

limit, as $p \to \infty$, of L_p isotonic regression. In many senses, strict L_∞ regression behaves like L_p regression for $1 < p < \infty$ [20]. However, as is mentioned in Section 5, there are faster algorithms for L_∞ which are not based on dynamic programming.

Among all optimal b-step regressions on $[i:n]$ let $\mathsf{R}(i,b)$ denote the function having the largest endpoint of the first step. The lemma shows that for isotonic data, $\mathsf{R}(\cdot,b)$ is an isotonic function. This fact can be used to efficiently compute $\mathsf{e}(\cdot,b)$ and $\mathsf{R}(\cdot,b)$ from the values of $\mathsf{e}(\cdot,b-1)$ and $\mathsf{R}(\cdot,b-1)$. Figure 3 shows an intermediate stage of the calculations for a single stage. The optimal first step for each multiple of 1/4 has been computed and now the first step for each odd multiple of 1/8 needs to be determined. For each of these, the possible values of the endpoint of the optimal first step are the range indicated by the dashed lines with the solid line indicating the part that any optimal first step must include.

Suppose that $\mathsf{e}(\cdot,c)$ and $\mathsf{R}(\cdot,c)$ have been determined for $i_1 < i_2 \ldots < i_k$. Let $j_o \ldots j_k$ be such that $j_0 < i_1 < j_1 < i_2 \ldots < i_k < j_k$. To determine $\mathsf{e}(\cdot,c)$ and $\mathsf{R}(\cdot,c)$ for the j values, note that since $\mathsf{R}(\cdot,c)$ is isotonic then $\mathsf{R}(j_0,c) \in [j_0 : \mathsf{R}(i_1,c)]$, $\mathsf{R}(j_1,c) \in [\max\{j_1, \mathsf{R}(i_1,c)\} : \mathsf{R}(i_2,c)]$, ..., and $\mathsf{R}(j_k,c) \in [\max\{j_k, \mathsf{R}(i_k,c)\} : n-c+1]$. Thus, to determine $\mathsf{e}(j_0,c)$ and $\mathsf{R}(j_0,c)$ we only need to evaluate $\mathsf{err}^p(j_0,\ell) + \mathsf{e}(\ell+1,c-1)$ for $\ell \in [j_0 : \mathsf{R}(i_1,c)]$, to determine $\mathsf{e}(j_1,c)$ and $\mathsf{R}(j_1,c)$, we only need to evaluate $\mathsf{err}^p(j_1,\ell) + \mathsf{e}(\ell+1,c-1)$ for $\ell \in [\max\{j_1, \mathsf{R}(i_1,c)\} : \mathsf{R}(i_2,b)]$, and so forth; i.e., we need at most $n+k$ total evaluations. In Figure 3, this corresponds to the fact that the dashed lines can overlap only at endpoints. In $\lceil \log_2 n \rceil$ iterations all values of $\mathsf{e}(\cdot,c)$ and $\mathsf{R}(\cdot,c)$ can be determined. This gives Algorithm B.

To complete the proof of Theorem 2.1, we need to show that each iteration of the "for k" loop can be completed in $\Theta(n)$ time. For L_2 this can be done quite easily using a standard technique. By first computing the scan values $\sum_{j=1}^{i} w_j y_j$, $\sum_{j=1}^{i} w_j y_j^2$, and $\sum_{j=1}^{i} w_i$ for all $i \in [1:n]$, each $\mathsf{err}^2()$ value can then be computed in unit time.

For L_1 there is no such algebraic simplification and an approach is needed that exploits the fact that the values are isotonic. The following explains the code in Algorithm C which shows how to compute each iteration of the "for k" loop in Algorithm B. Algorithm C utilizes some simple routines that are given in Figure 4.

For an interval $[i:j]$, let m denote the location of a median. Let $\mathsf{w\_lt} = \sum_{k=i}^{m-1} w_k$, $\mathsf{w\_gt} = \sum_{k=m+1}^{j} w_k$, $\mathsf{yw\_lt} = \sum_{k=i}^{m-1} y_k w_k$ and $\mathsf{yw\_gt} = \sum_{k=m+1}^{j} y_k w_k$. Then, the L_1 error of using y_m as the regression value on $[i:j]$, i.e., $\mathsf{err}^1(i,j)$, is $y_m \mathsf{w\_lt} - \mathsf{yw\_lt} + \mathsf{yw\_gt} - y_m \mathsf{w\_gt}$. To determine the corresponding m, $\mathsf{w\_lt}$, $\mathsf{w\_gt}$, $\mathsf{yw\_lt}$ and $\mathsf{yw\_gt}$ values (and hence $\mathsf{err}^1(\cdot)$) for interval $[i:j+1]$, note that the location of the new median is $\geq m$. Let $\mathsf{w\_gt} = \mathsf{w\_gt} + w_{j+1}$ and $\mathsf{yw\_gt} = \mathsf{yw\_gt} + y_{j+1} w_{j+1}$. If $\mathsf{w\_gt} > \mathsf{w\_lt} + w_m$ then the location of the median for $[i:j+1]$ is

```
i_start ... i_end :  range of possible endpoints

for i = 1 to n do
    e(i, 1) = err^p(i, n),   R(i, 1) = n
for c = 2 to b do
    for k = ⌈log_2 n⌉ − 1 downto 0
        for j = 2^k to n − c + 1 by 2^{k+1} do
            if j = 2^k then i_start = j
                else i_start = max{j, R(j − 2^k, c)}
            if j + 2^k > n − c + 1 then i_end = n − c + 1
                else i_end = R(j + 2^k, c)
            e(j, c) = min{err^p(j, i) + e(i+1, c−1) : i_start ≤ i ≤ i_end}
                {store largest minimizing i in R(j, c)}
        end for j
    end for k
end for c
```

Algorithm B: Stepwise Constant Approximation of Isotonic Data

---

```
med_loc : location of median
w_lt, w_gt : Σ_{h=j}^{med_loc−1} w(h),   Σ_{h=med_loc+1}^{i} w(h)
yw_lt, yw_gt : Σ_{h=j}^{med_loc−1} v(h) ∗ w(h),   Σ_{h=med_loc+1}^{i} v(h) ∗ w(h)
i_start ... i_end : range of feasible ends of first step
j ... i_start − 1 : any first step must include these positions

med_loc = 0
w_lt = w_gt = yw_lt = yw_gt = 0
for j = 2^k to n − c + 1 by 2^{k+1} do
    if j = 2^k then i_start = j
    else
        i_start = max{j, R(j − 2^k, c)}
        for i = j − 2^{k+1} to min{R(j − 2^k, c), j} − 1 do   {used in previous j but not this one}
            remove_position(i)
        end for i
    if j + 2^k > n − c + 1 then i_end = n − c + 1
                           else i_end = R(j + 2^k, c)
    e(j, c) = ∞
    for i  = i_start to i_end do
        if not ((j > 2^k) and (i_start = R(j − 2^k, c)))  then   {only omitted if used in previous j}
            insert_position(i)
        err_end_at_i = (w_lt ∗ median − yw_lt) + (yw_gt − w_gt ∗ median) + e(i+1, c−1)
        If err_end_at_i ≤ e(j, b) then
            e(j, c) = err_end_at_i,   R(j, c) = i
    end for i
end for j
```

Algorithm C: Interior of "for k" Loop of Algorithm B for L_1 Isotonic Regression

---

```
insert_position(i) :   {i now rightmost active position}
    if med_loc = 0 then
        med_loc = i
        median = v(i)
    else
        w_gt = w_gt + w(i),   yw_gt = yw_gt + v(i) * w(i)
        update_median

remove_position(i) :   {i was leftmost position}
    if (w_lt = 0) and (w_gt = 0) then   {i was last remaining position}
        med_loc = 0
    else
        w_lt = w_lt - w(i),   yw_lt = yw_lt - v(i) * w(i)
        update_median

update_median :
    while w_gt > w_lt + w(med_loc) do   {median is larger}
        w_lt = w_lt + w(med_loc)
        yw_lt = yw_lt + v(med_loc) * w(med_loc)
        med_loc = med_loc + 1
        w_gt = w_gt - w(med_loc)
        yw_gt = yw_gt - v(med_loc) * w(med_loc)
    end while
    median = v(med_loc)
```

Figure 4: Routines Used in Algorithm C

$> m$ (if there are duplicate values then it is possible that y_m is a median value and m is increased, but only if y_{m+1} is the same value). Set $\mathsf{w\_lt} = \mathsf{w\_lt} + w_m$, $\mathsf{yw\_lt} = \mathsf{yw\_lt} + y_m w_m$, $\mathsf{w\_gt} = \mathsf{w\_gt} - w_{m+1}$, $\mathsf{yw\_gt} = \mathsf{yw\_gt} - y_{m+1} w_{m+1}$, and $m = m + 1$. Continue this incremental process until $\mathsf{w\_gt} \leq \mathsf{w\_lt} + w_m$, at which point y_m is a median. Now $\mathsf{err}^1(i, j+1)$ and the error of using $[i\!:\!j+1]$ as the first step can be determined.

The remaining concern is to calculate $\mathsf{err}^1(j, \ell)$ for $\ell \in [\mathsf{R}(j', c)\!:\!\mathsf{R}(j'', c)]$, where $j' < j < j''$ are the largest preceding and smallest succeeding values computed in the previous k iteration. To do this using the incremental procedure above, one first needs to start with the values in $[j\!:\!\mathsf{R}(j', c)-1]$. Fortunately these were already included in the final calculation for $\tilde{j} = j - 2^{k+1}$, which was the previous j value in the "for j" loop. The last $\mathsf{err}^1(\tilde{j}, \cdot)$ calculated was for the interval $[\tilde{j}\!:\!\mathsf{R}(j', c)]$. Thus, removing the values in $[\tilde{j} : j-1]$ will give the required start. This involves a sequence of operations for determining the location of the median and related w and yw values for $[r+1 : s]$ given their values for $[r\!:\!s]$. This is done similarly to the case of incrementing s where here too the location of the median cannot decrease.

For a fixed k, since all of the calculations are linear in the number of times the start or end of an interval is incremented or when m is increased, the total time to compute the $\mathsf{err}^1()$ values is $\Theta(n)$. This completes the proof of Theorem 2.1. □

Fisher defined another form of binning which he called *unrestricted homogenization*: give n weighted values $(\mathbf{y}, \mathbf{w})$ and $b \in [1\!:\!n]$, partition them into b subsets $P_i, i \in [1\!:\!b]$ and assign a value C_i to each P_i so as to minimize

$$\sum_{i=1}^{b} \sum_{j \in P_i} w_j |y_j - C_i|^2$$

among all such partitions. He noted that this could be solved by sorting the values and then finding the optimal b-step approximation. This fact easily extends to all L_p metrics, $1 \leq p \leq \infty$.

Ioannidis [10] defined a problem that is essentially the same. His, called *variable width serial histograms*, was used to estimate the most efficient way to answer certain database join queries. The values in his case corresponded to the number of entries in the database having each key. He mistakenly believed that finding these values requires time exponential in b. Jagadish et al. [12] later showed that Fisher's dynamic programming algorithm can be used to find this in $\Theta(bn^2)$ time. Theorem 2.1 shows that this can be reduced yet further.

Corollary 2.3 *Given n weighted values and number of steps $b \leq n$, for the L_1 and L_2 metrics, one can determine an optimal b-step unrestricted homogenization and an optimal b-step variable width serial histogram in $\Theta(bn \log n)$ time.* □

Lemma 2.2 applies to all $1 \leq p < \infty$, but Theorem 2.1 only refers to $p = 1, 2$. For the remaining values of p, the bottleneck is in the computation of the L_p weighted mean. The only other p's for which one can get a closed-form solution for the mean are $p = 3, 4$ since finding the mean requires solving a polynomial equation. For $p = 4$, the approach for $p = 2$ can be used to give the same time bounds. However, for $p = 3$, there is the problem that $|y_i - f(i)|^3$ is either $(y_i - f(i))^3$ or its negative. One can first use binary search to determine the i for which $y_i \leq C \leq y_{i+1}$, where C is the mean, and then compute the sums with the appropriate signs and determine the mean exactly. Note that the mean is not known as i is being determined. Rather, if y_i is used as the mean, one can determine whether the sum of the regression errors for $y_1 \ldots y_{i-1}$ is greater than the sum for $y_{i+1} \ldots y_n$. In this case, i should be increased. If it is smaller, then i should be decreased. For arbitrary p, one can find the bracketing index and then use approximations to find the mean to a desired accuracy.

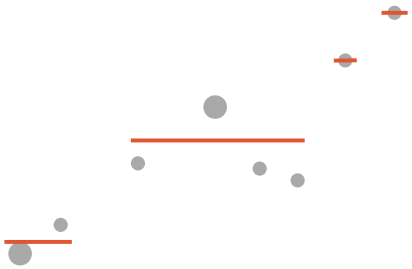

Figure 5: Optimal 4-step Reduced Isotonic Regression of Values in Figure 1

3 Reduced Isotonic Regression

Isotonic regressions are somewhat easier to compute than are general approximations by step functions. For L_2 the isotonic regression can be determined in $\Theta(n)$ time, a fact apparently first proven in 1955 by Ayer et al. [3] and then often rediscovered. For L_1 the isotonic regression can be determined in $\Theta(n \log n)$ time [2, 19]. For L_∞ the time is also $\Theta(n \log n)$ [13, 18] and for unweighted data it is $\Theta(n)$ [19]. These algorithms use a simple left-right scan where each location is initially a step and then adjacent steps are merged whenever they violate the isotonic condition. This approach is known as "pair adjacent violators", PAV.

Isotonic regression is a very flexible nonparametric approach to many problems. However, as noted, it does have its detractors due to results with impractically many steps or due to potential overfitting. Some researchers have instead used approximations with a specified number of steps [7, 21]. To reduce overfitting, Schell and Singh [17] used the approach of repeatedly merging pairs of adjacent steps whose difference had the least statistical significance. Haiminen et al. [7] also used an approach that repeatedly combines the adjacent steps which cause a minimum increase in the error. The latter two approaches are known as greedy or myopic since they repeatedly make the choice that seems to be the best at the moment. In contrast, dynamic programming considers the interaction of the current choice with all future choices.

Greedy approaches do not always produce an optimal reduced isotonic regression. For example, given the unweighted values 1, 2, 3, 4, 5, 6, the optimal 3-step regression is 1.5 on the first two points, 3.5 on the second two and 5.5 on the third two. Meanwhile, the optimal 2-step regression is 2 on the first three points and 5 on the last three. Thus a greedy merging approach which produced the optimal 3-step regression could not then also produce the optimal 2-step regression.

We address the problem of finding the optimal isotonic regression with a specified number of steps; i.e., an optimal b-step reduced regression. As a reminder, we use "pieces" to refer to the steps of an unrestricted isotonic regression and "steps" to refer to the steps of a reduced isotonic regression. In Haiminen et al. the authors provide an exact algorithm for an optimal solution, but this takes $\Theta(n + bm^2)$ time for the L_2 metric, where m is the number of pieces in the unconstrained isotonic regression. Even though typically $m \ll n$, they felt that this was too slow for their application and as a result turned to the greedy heuristic just mentioned. Our exact algorithm, which takes $\Theta(n + bm \log m)$ time, should be sufficiently fast to be practical even for large problems. Figure 5 is an example of a reduced isotonic regression of the weighted values in Figure 1. While the values of the unconstrained L_2 isotonic regression are uniquely determined, the step boundaries are not since a sequence of identical values might be represented as multiple steps. However, if we require that adjacent steps with the same mean be merged, then for L_2 the step boundaries, and hence m, are also uniquely determined.

A critical observation in the Haiminen et al. paper is that, given the pieces of an L_2 unrestricted isotonic regression, the steps of an optimal L_2 reduced isotonic regression can be formed by merging

the pieces. Their observation immediately gives a simple algorithm: find the unrestricted isotonic regression and represent each piece as a weighted point with a value that is the mean and a weight that is the sum of the weights. Then a b-step isotonic regression of these points gives a b-step isotonic regression of the original data. Haiminen et al. used Fisher's algorithm to determine the optimal b-step isotonic regression, but Theorem 2.1 provides a faster solution.

Theorem 3.1 *Given n weighted values $(\mathbf{y}, \mathbf{w})$ and number of steps b, an optimal L_2 b-step reduced isotonic regression can be found in $\Theta(n + bm \log m)$ time, where m is the number of pieces in the unconstrained L_2 isotonic regression.* □

4 Quasi-Optimal L_1 Reduced Regression

Unfortunately, the approach of Haiminen et al. does not extend to any p other than 2. For example, for L_1 with values -3, 1, 0, -3, -0.1, 2, and weights 10, 1, 1, 1, 2, 10, the unrestricted isotonic regression has pieces $[1 : 1]$, $[2 : 5]$, and $[6 : 6]$, with values -3, -0.1, and 2, respectively. The unique optimal 2-step reduced isotonic regression has steps $[1 : 4]$ and $[5 : 6]$, with values -3 and 2, which requires cleaving the middle piece. In practice, however, a very good L_1 reduced isotonic function can be generated using Algorithm B without splitting the pieces of an unrestricted isotonic regression. This is especially true if the unrestricted isotonic regression is carefully constructed. Given weighted values $(\mathbf{y}, \mathbf{w})$, let $\mathcal{S} = \{S_k : k = 1, m\}$ be the unique set of steps such that

- there is an L_1 isotonic regression g with steps $\mathcal{S}$ where $g(S_1) < g(S_2) \ldots < g(S_m)$, and
- for any $k \in [1 : m]$, if h is an L_1 isotonic regression, then h is constant on S_k

It can be shown that $\mathcal{S}$ is the intersection of the steps of all L_1 isotonic regressions of the data, with the requirement that if an isotonic regression has adjacent steps with the same value then they are merged. Further, $\mathcal{S}$ can be found in $\Theta(n \log n)$ time through a PAV approach where adjacent steps are not merged unless the isotonic condition forces it. We say that $\mathcal{S}$ is the set of steps of a *fully refined* isotonic regression. For example, for values -2, 1, -2, 2, 1, 3 with weights 10, 1, 1, 1, 1, 10, one unrestricted isotonic regression is -2, 1, 1, 1, 1, 3, while a fully refined one is -2, -0.5, -0.5, 1.5, 1.5, 3. In fact, this is the unique L_p isotonic regression for all $1 < p < \infty$. Further, the optimal 2-step reduced isotonic regression has values -2, -2, -2, 3, 3, 3, which requires cleaving the middle piece of the former but which can be formed from pieces of the latter.

A b-step isotonic function is a *quasi-optimal* L_1 reduced isotonic regression iff it minimizes the L_1 regression error among all b-step isotonic functions which have steps that are unions of steps in $\mathcal{S}$. Our approach for determining quasi-optimal regressions is similar to that for L_2, first finding an unrestricted isotonic regression and then using it to find a reduced isotonic regressions. In certain cases, however, the pieces of the unrestricted regression cannot be collapsed into a single value. For example, for unweighted values 1, 0, 0, 2, 2, 1, 3, 3, 1, $\mathcal{S} = \{[1 : 3], [4 : 6], [7 : 9]\}$, and the unique isotonic regression is 0, 2, and 3 on these steps. The unique 1-piece optimal reduced isotonic regression has value 1. If instead the last value were 3, then the unconstrained isotonic regression would remain the same but the 1-piece reduced isotonic regression would have value 2.

Theorem 4.1 *Given n weighted values $(\mathbf{y}, \mathbf{w})$ and number of steps $b \leq m$, a quasi-optimal L_1 reduced isotonic regression can be found in $\Theta(n \log n + bn \log m)$ time, where m is the number of pieces in a fully refined L_1 isotonic regression.*

Proof: The approach of Algorithm C will be used. Let the pieces of the fully refined L_1 isotonic regression be I_k, $k = 1 \ldots m$, where $I_k = [a(k) : b(k)]$. For the remainder of this proof the indexing will be on the pieces not the original data, i.e., it is subsets of $[1 : m]$, not $[1 : n]$. As before, if the median of pieces $[i : j]$ is known, then determining the median of pieces $[i : j + 1]$

```
piece i is [a(i):b(i)] of original data
y_sort : v in sorted order
π(h) : location of v(h) in y_sort
w_sort(π(h)) : w(h) if location h in an active piece, else 0.  Initially 0
med_loc : initially 1

insert_piece(i) :   {i is now rightmost piece}
    for h = a(i) to b(i) do
        w_sort(π(h)) = w(h)
        if π(h) < med_loc then
            w_lt = w_lt + w(h),   yw_lt = yw_lt + v(h) * w(h)
        else if π(h) > med_loc then
            w_gt = w_gt + w(h),   yw_gt = yw_gt + v(h) * w(h)
    end for h
    update_median

remove_piece(i) :   {i was leftmost piece}
    for h = a(i) to b(i) do
        w_temp = w_sort(π(h))
        w_sort(π(h)) = 0
        if π(h) < med_loc then
            w_lt = w_lt − w_temp,   yw_lt = yw_lt − v(h) * w_temp
        else if π(h) > med_loc then
            w_gt = w_gt − w_temp,   yw_gt = yw_gt − v(h) * w_temp
    end for h
    update_median

update_median :
    while w_gt > w_lt + w_sort(med_loc) do   {median is larger}
        w_lt = w_lt + w_sort(med_loc)
        yw_lt = yw_lt + y_sort(med_loc) * w_sort(med_loc)
        med_loc = med_loc + 1
        w_gt = w_gt − w_sort(med_loc)
        yw_gt = yw_gt − y_sort(med_loc) * w_sort(med_loc)
    end while
    median = y_sort(med_loc)
```

Figure 6: Routines Used in Algorithm C for Quasi-Optimal L_1 Reduced Isotonic Regression

merely requires adding the weighted values in I_{j+1} and then locating the new median, which is not less than the previous one. Similarly, determining the median of $[i-1:j]$ requires removing the values corresponding to I_i. Balanced trees could be used to keep track of the information, where each data value can be inserted or removed in $\Theta(\log n)$ time, and the weighted median can be determined in the same time. Since insertion and deletion happens at most once per value, the total time for an iteration of the "for k" loop in Algorithm B would be $\Theta(n \log n)$, giving a total time of $\Theta(n \log n + bn \log n \log m)$.

The time per iteration can be reduced to $\Theta(n)$, as for isotonic data, although the operations are more complicated. First the values are sorted and stored in an array y_sort, and a mapping π is created so that $\pi(i)$ is the location of y_i in y_sort. A left-right scan of y_sort mimics the left-right scan of y for isotonic data. An array w_sort will be used, where w_sort(i) corresponds to y_sort(i). It will be used as a flag, in that if pieces $[i:j]$ are currently represented, then w_sort(k) is 0 if $\pi^{-1}(k) \notin I_i \cup \ldots \cup I_j$, while otherwise it is the correct weight. In this setting we say that pieces $[i:j]$ are "active". Initially w_sort is zero.

Moving from the calculation of the median of pieces $[i:j]$ to that of $[i:j+1]$ merely means that for each ℓ in piece I_{j+1}, w_sort$(\pi(\ell)) =$ w$_\ell$ and the w_ and yw_ sums are updated. Then, the incremental procedure to locate the new median begins. However, an update during the insertion may involve adding to w_lt and yw_lt since some values in I_{j+1} may be smaller than the median. Removing a piece I_i occurs similarly, setting w_sort$(\pi(\ell)) = 0$ for each $\ell \in I_i$ and then updating the median.

Further, med_loc is modified so that it is initially 1 and is never reset to 0. It may be that w_sort(med_loc) $= 0$ at some point in the algorithm (such as the start), but when median is determined in update_median then either v_sort(med_loc) is a median of the active pieces (though not necessarily a value in the active pieces), or there are no active pieces but when the next piece is inserted its median will be at least this value since it is a median of the last piece removed. The detailed operations needed for Algorithm C are given in Figure 6. As before, the time per iteration is $\Theta(n)$, yielding the time claimed in the theorem. □

5 Final Comments

We have shown, in Theorem 2.1, that for n isotonic values, for the L_1 and L_2 metrics, an optimal b-step isotonic regression can be computed in $\Theta(bn \log n)$ time. For the more general problem with arbitrary data, we have shown in Theorem 3.1 that an optimal L_2 b-step reduced isotonic regression can be found in $\Theta(n + bm \log m)$ time, where m is the number of pieces in the unconstrained isotonic regression. Prior to these results, for L_2, the best known times to solve these problems were $\Theta(bn^2)$ for isotonic data and $\Theta(n + bm^2)$ for general data [7]. For L_1 we did not provide an algorithm guaranteed to find the optimal reduced regression, but, in Section 4 we did provide one that generates a good approximation in $\Theta(n \log n + bn \log m)$ time. It is an open question as to whether there is an $O(n \log n + bn \log m)$ time algorithm to produce L_1 optimal reduced isotonic regressions.

In an extended version of this paper, results for L_∞ will be included. Using a quite different approach, we show that the reduced isotonic regression can be found in $\Theta(n \log n)$ time. Note that b does not appear in the time. This is because in linear time one can decide, for a given ϵ, if there is a b-step isotonic function with L_∞ error $\leq \epsilon$. (This does not seem possible for the L_p error when $p < \infty$.) Combining this with a technique known as parametric search gives the time claimed. An algorithm using the same approach and time bound is also given for finding an L_∞ optimal b-step approximation. For this problem the fastest previously known algorithm, due to Chen and Wang [5], takes $\Theta(\min\{n \log^2 n,\ n \log n + b^2 \log^4 n\})$ time. For unweighted data the time can be further reduced to $\Theta(n + m \log m)$.

Finally, an interesting problem that we have not addressed is that of selecting the most desirable number of steps for applications calling for fewer than m steps. Schell and Singh and Haiminen et al. start with an unconstrained isotonic regression and then repeatedly merge pieces together until their criteria are met. As was shown, the resulting reduced regressions may be sub-optimal among reduced isotonic regressions with the same number of steps. Other researchers choose b *a priori* based on considerations such as storage or access time requirements. In contrast, dynamic programming approaches offer the possibility of creating optimal b-step isotonic regressions for each value of b as b increases. One can then stop when a criterion is met and always have an optimal result. We are unaware of any research, even giving sub-optimal results, that has taken this last approach. Equally interesting questions arise regarding what a good criterion would be for any particular application.

Acknowledgements

Research partially supported by NSF grant CDI-1027192 and DOE grant DE-FC52-08NA28616

References

[1] Agarwal, PK and Sharir, M (1998), "Efficient algorithms for geometric optimization", *ACM Computing Surveys*, pp. 412-458.

[2] Ahuja, RK and Orlin, JB (2001), "A fast scaling algorithm for minimizing separable convex functions subject to chain constraints", *Operations Research* 49, pp. 784–789.

[3] Ayer, M, Brunk, HD, Ewing, GM, Reid, WT, and Silverman, E (1955), "An empirical distribution function for sampling with incomplete information", *Annals of Math. Stat.* 5, pp. 641–647.

[4] Barlow, RE, Bartholomew, DJ, Bremner, JM, and Brunk, HD (1972), *Statistical Inference Under Order Restrictions: The Theory and Application of Isotonic Regression*, John Wiley.

[5] Chen, DZ and Wang, H (2012), "Approximating points by a piecewise linear function", *Algorithmica*, to appear.

[6] Fisher, WD (1958), "On grouping for maximum homogeneity", *J. Amer. Stat. Assoc.* 53, pp. 789–798.

[7] Haiminen, N, Gionis, A, and Laasonen, K (2008), "Algorithms for unimodal segmentation with applications to unimodality detection", *Knowl. Info. Sys.* 14, pp. 39–57.

[8] Halim, F, Karras, P, and Yap, RHC (2009), "Fast and effective histogram construction", *Proc. Conf. Info. and Knowl. Manag.*, pp. 1167–1176.

[9] Himberg, J, Korpiaho, K, Mannila, H, Tikanmaki, J and Toivonen, H (2001), "Time series segmentation for context recognition in mobile devices", *Int'l. Conf. Data Mining*, pp. 203–210.

[10] Ioannidis, YE (1993), "Universality of serial histograms", *Proc. 19th VLDB Conf.*, pp. 256–267.

[11] Jacob, E, Nair, KNR, and Sasikumar, R (2009), "A fuzzy-driven genetic algorithm for sequence segmentation applied to genomic sequences", *Applied Soft Computing* 9, pp. 488–496.

[12] Jagadish, HV, Koudas, N, Muthukrishnan, S, Poosala, V, Sevcik, K and Suel, T. (1998), "Optimal histograms with quality guarantees", *Proc. 24th VLDG Conf.*, pp. 275–286.

[13] Lin, T-X, Kuo, C-C, Hsieh, Y-H, and Wang, B-F (2009), "Efficient algorithms for the inverse sorting problem with bound constraints under the L_∞-norm and the Hamming distance", *J. Comp. and Sys. Sci.* 75, pp. 451-464.

[14] Poosala, V, Ioannidis, Y, Haas, P, and Shekita, E (1996), "Improved histograms for selectivity estimation of range predicates", *Proc. SIGMOD*, pp. 294–305.

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

[16] Salanti, G and Ulm, K (2003), "A nonparametric changepoint model for stratifying continuous variables under order restrictions and binary outcome", *Stat. Methods Med. Res.* 12, pp. 351–367.

[17] Schell, MJ and Singh, B (1997), "The reduced monotonic regression method", *J. Amer. Stat. Assoc.* 92, pp. 128–135.

[18] Stout, QF (2008) "Weighted L_∞ isotonic regression", submitted.

[19] Stout, QF (2008), "Unimodal regression via prefix isotonic regression", *Computational Statistics and Data Analysis* 53, pp. 289–297. A preliminary version appeared in "Optimal algorithms for unimodal regression", *Computing and Statistics* 32, 2000.

[20] Stout, QF (2012), "Strict L_∞ isotonic regression", *J. Opt. Theory and Appl.* 152, pp. 121–135.

[21] Terzi, E and Tsaparas, P (2006), "Efficient algorithms for sequence segmentation", *Proc. 6th SIAM Conf. Data Mining*.

[22] Wang, DP (2002), "A new algorithm for fitting a rectilinear x-monotone curve to a set of points in the plane", *Pattern Recogn. Let.* 23, pp. 329–334.