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 regression is an increasing step function. For n data points, the number of steps in the isotonic regression may be as large as n. As a result, standard 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, $b \leq n$. However, because the previous algorithms for finding the reduced L_2 regression took $\Theta(n + bm^2)$ time, where m is the number of steps of the unconstrained isotonic regression, researchers felt that the algorithms were too slow and instead used approximations. Other researchers had results that were approximations because they used a greedy top-down approach. Here we give an algorithm to find an exact solution in $\Theta(n + bm)$ time, and a simpler algorithm taking $\Theta(n + bm \log m)$ time. These algorithms also determine optimal k-means clustering of weighted 1-dimensional data.

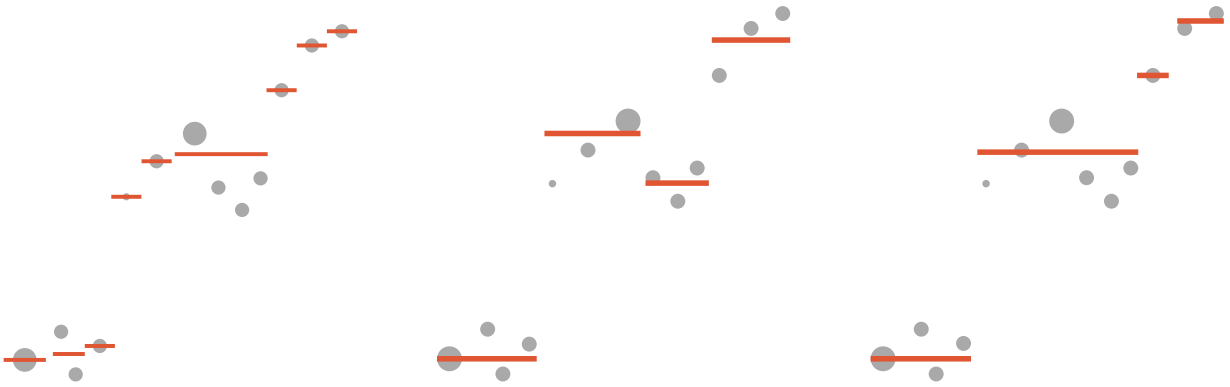
Keywords: reduced isotonic regression, step function, v-optimal histogram, piecewise constant approximation, k-means clustering, nonparametric regression

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 for all x_1, x_2 in its domain, if $x_1 < x_2$ then $f(x_1) \leq f(x_2)$. In some settings isotonic functions are called monotonic, while in others monotonic is used to indicate either nondecreasing or nonincreasing. Myriad uses of isotonic regression can be found in citations to the fundamental books of Barlow et al. [3] and Robertson et al. [14]. Nonparametric approaches are increasingly important as researchers encounter situations where parametric assumptions are dubious, and as algorithmic improvements make the calculations practical.

Isotonic regression is useful for situations in which the independent variable has an ordering but no natural metric, such as S < M < L < XL clothing 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 values (y_i, w_i), $i \in [1\!:\!n]$, where the y values are arbitrary real numbers and the w values (the weights) are nonnegative real numbers. Given weighted values $(\mathbf{y}, \mathbf{w})$ and a real-valued function f on $[1\!:\!n]$, the L_p regression or approximation error of f is

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



a) isotonic regression b) 4-step regression c) 4-step reduced isotonic

Figure 1: Stepwise regressions, size indicates weight

An L_p *isotonic regression* is an isotonic function that minimizes the L_p error among all isotonic functions. Figure 1 a) gives an example of an isotonic regression. Because researchers from varying fields often use different expressions for a single concept, we use the terms *regression* and *approximation* interchangeably. We identify approximations that are not optimal regressions as *sub-optimal* approximations.

Isotonic regressions are step functions for which the number of steps is determined by the data. In certain cases there is criticism that such functions can overfit the data [12, 15, 16] or produce a result with too many steps [5]. Consequently, some researchers utilize isotonic regressions that restrict the number of steps. Schell and Singh [16] have referred to such functions as *reduced* isotonic regressions.

Restricting the number of steps is a central issue in approximation by step functions. It arises in settings such as databases and variable width histogramming [6, 9, 13], segmentation of time series and genomic data [8, 10, 19], homogenization [4] and piecewise constant approximations [11].

A function f is an *optimal L_p b-step approximation*, $1 \leq b \leq n$, iff it minimizes the L_p error over all functions with b steps. Here we are primarily concerned with computing L_2 b-step reduced isotonic regressions, where a 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. Figure 1 gives examples of b-step regression and b-step reduced isotonic regression. Optimal b-step approximations and b-step reduced isotonic regressions are not always 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 $[2:3]$.

In 1958 Fisher [4] gave a simple algorithm for determining an optimal b-step L_2 regression in $\Theta(bn^2)$ time (this is shown in Algorithm A). His algorithm can be easily modified to determine an optimal b-step L_2 reduced isotonic regression in the same time bounds. His algorithm has been widely used and rediscovered, and often falsely attributed to Bellman. However, for many researchers the quadratic time in n makes it too slow for their applications [5, 6, 8, 10, 19]. Thus most previous work utilizing reduced isotonic regression used sub-optimal approximations, with the exception of an algorithm due to Haiminen, Gionis and Laasonen [5]. Their algorithm for the L_2 metric takes $\Theta(n + bm^2)$ time, where m is the number of pieces of the unrestricted isotonic regression. (To lessen confusion, we use "pieces" to refer to the steps of the unrestricted isotonic regression.) However, even with this reduction in time they then developed an approximation algorithm based on a greedy heuristic.

In Section 3 we decrease the time to find the optimal b-step L_2 reduced isotonic regression to $\Theta(n+bm)$, using an algorithm in Section 2.2 for the special case in which the values are themselves isotonic. A simpler algorithm, taking $\Theta(n + bm \log m)$ time, is also given. These algorithms should be fast enough to eliminate the need for approximations, even for very large data sets.

Since we are only looking for optimal approximations, we often omit "optimal".

2 Approximation by Step Functions

A real-valued function f on $[1:n]$ is a *b-step function*, $1 \leq b \leq n$, iff there are 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 $C_1 \leq C_2 \ldots \leq C_b$. 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".

Let $\mathsf{mean}_p(i,j)$ denote an L_p mean of the weighted values on $[i:j]$. For $1 \leq p < \infty$, an optimal L_p step function has the property that $C_k = \mathsf{mean}_p(j_{k-1}+1,\ j_k)$. Since we are only concerned with optimal approximations, whenever a function has a step $[i:j]$, then its value on that step is $\mathsf{mean}_p(i,j)$. Let $\mathsf{err}^p(i,j)$ denote the p^{th} power of the L_p error of the step $[i:j]$. Minimizing the sum of the err^p values is the same as minimizing the L_p approximation error and thus from now on only the err^p values will be used.

2.1 Arbitrary Data

Fisher's [4] dynamic programming approach to determining an optimal L_p 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 is an optimal $(b-1)$-step approximation of the data on $[j+1:n]$. This is obvious since if it were not optimal then replacing it with an optimal $(b-1)$-step approximation would reduce the error. Let $\mathsf{e}(i,c)$ denote the sum of the err^p values of the steps of an optimal c-step approximation on $[i:n]$, and let $\mathsf{e}'(i,j,c)$ denote the sums of the err^p values of the steps of a c-step approximation on $[i:n]$ which is optimal among c-step approximations where the first step is $[i:j]$. Fisher's observation yields the equations:

$$\mathsf{e}'(i,j,c) = \mathsf{err}^p(i,j) + \mathsf{e}(j+1, c-1) \tag{1}$$

$$\mathsf{e}(i,c) = \min\{\mathsf{e}'(i,j,c) : i \leq j \leq n-c+1\} \tag{2}$$

By storing the j that minimizes $\mathsf{e}(i,c)$ in $\mathsf{j_{min}}(i,c)$, in $\Theta(n)$ time one can generate the optimal approximation after the dynamic programming has completed. This leads to Algorithm A. The time is $\Theta(bn^2)$ plus the time to compute the $\Theta(n^2)$ err^p values. For L_∞, $\mathsf{e}'(i,j,c) = \max\{\mathsf{err}^\infty(i,j),\ \mathsf{e}(j+1, c-1)\}$.

Fisher's algorithm can be modified to determine the b-step reduced isotonic regression in the same time bounds. The lines

$$\begin{array}{l}\mathsf{for\ i = 1\ to\ n - c + 1}\\ \quad \mathsf{e(i,c) = min\{e'(i,j,c) : i \leq j \leq n - c + 1\}}\end{array}$$

should be replaced by

$$\begin{array}{l}\mathsf{for\ i = 1\ to\ n - 1}\\ \quad \mathsf{e(i,c) = min\big\{err^p(i,n),\ \ min\{e'(i,j,c) : i \leq j \leq n-1,\ mean_p(i,j) \leq mean_p(j+1,\ j_{min}(j+1, c-1))\}\big\}}\end{array}$$

Including the $\mathsf{err}^p(i,n)$ term, and changing the upper bound on i, is necessary so that, say, for unweighted data 3, 2, 1, the L_2 2-step reduced isotonic regression is correctly determined to be 2, 2, 2. Using either 3, or 3, 2, as the initial step would involve a second step that was lower, and hence the solution has only 1 step.

```
for i = 1 to n
    e(i, 1) = err^p(i, n);   j_min(i, 1) = i
for c = 2 to b
    for i = 1 to n − c + 1
        e(i, c) = min{e′(i, j, c) : i ≤ j ≤ n − c + 1}    {e′ is defined in (1)}
            {record minimizing j in j_min(i, c)}
    end for i
end for c
generate the approximation using j_min and mean_p
```

Algorithm A: Fisher's algorithm for optimal L_p b-step approximation of arbitrary data, $1 \le p \le \infty$

Throughout, the values of e and $\mathsf{j}_{\mathsf{min}}$ are stored in 2-dimensional arrays, while e' is evaluated as a function, not stored as a 3-dimensional array. To evaluate err^2, once the scan values $\sum_{j=1}^{i} w_j y_j$, $\sum_{j=1}^{i} w_j y_j^2$, and $\sum_{j=1}^{i} w_i$ have been determined for all $i \in [1\!:\!n]$, each err^2 value can then be computed in unit time.

2.2 Isotonic Data

Reducing the time of Algorithm A requires reducing the number of err^p values referenced. It is not known how to do this for arbitrary data, but isotonic data has some special properties. We give two algorithms: Algorithm B is simpler than Algorithm C, but, in O-notation, slower by a logarithmic factor. It is likely that many will prefer Algorithm B over Algorithm C. Algorithm B is given in Section 2.3, and Algorithm C is in Section 2.4.

For isotonic data, the fact that values are nondecreasing allows one to make inferences concerning the means of intervals. For example, the L_p mean of the weighted values on $[i\!:\!j]$ is no larger than that of the values on $[i{+}1\!:\!j]$. Further, for any $1 < i \le j < n$, $\mathsf{err}^p(i, j{+}1) - \mathsf{err}^p(i, j) \ge \mathsf{err}^p(i{+}1, j{+}1) - \mathsf{err}^p(i{+}1, j)$. That is, if we consider the increase in error of adding (x_{j+1}, w_{j+1}) to the step $[i\!:\!j]$, this is greater than the increase when adding it to the step $[i+1\!:\!j]$. This is true because the monotonicity insures that x_{j+1} is at least as large as the mean on $[i+1\!:\!j]$, which has a mean not more than that of $[i\!:\!j]$, and the total weight of $[i\!:\!j]$ is greater than the total weight of $[i+1\!:\!j]$. When the values are not isotonic then this inequality may not hold.

Letting $M(i, j) = \mathsf{err}^p(i, j)$, this can be rewritten as

$$M(i, j{+}1) + M(i{+}1, j) \ge M(i, j) + M(i{+}1, j{+}1) \tag{3}$$

for all $1 \le i < j < n$ and $1 \le p \le \infty$. This is known as the *Monge property*, and M is known as a Monge matrix (typically the Monge property has the inequality in the opposite order and is applied to maximization, not minimizing).

If $\mathsf{j}_{\mathsf{min}}(i)$ denotes the smallest j such that $M(i, j)$ is a minimal value in row i of M, then the Monge property implies that for any $i < i'$, $\mathsf{j}_{\mathsf{min}}(i) \le \mathsf{j}_{\mathsf{min}}(i')$, i.e., $\mathsf{j}_{\mathsf{min}}$ is isotonic. This property is typically called *monotonicity*. If we define $M(i, j) = \infty$ when $j < i$ then M satisfies (3) for all i and j. Iteratively combining this inequality over adjacent elements shows that it holds much more widely, in that for all $1 \le i_1 < i_2 \le n$ and $1 \le j_1 < j_2 \le n$,

$$M(i_1, j_2) + M(i_2, j_1) \ge M(i_1, j_1) + M(i_2, j_2) \tag{4}$$

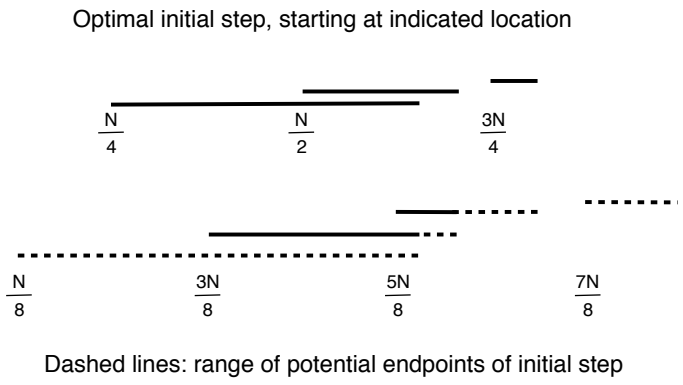


Figure 2: Possible endpoints of odd multiples of 1/8

Thus all submatricies of a Monge matrix are Monge, where a submatrix can be formed from an arbitrary set of rows and an arbitrary set of columns and the number of rows need not equal the number of columns. Since all submatricies are Monge, all are monotonic. This property is called *total monotonicity*. There are monotonic matrices that are not totally monotonic and totally monotonic matrices that aren't Monge.

The fact that M is a Monge matrix implies that M^c is a Monge matrix, for $c > 1$, where $M^c(i,j) = \mathsf{e}'(i,j,c)$. This is because

$$\begin{aligned} M^c(i,j{+}1) + M^c(i{+}1,j) &= M(i,j{+}1) + \mathsf{e}(j{+}1,c{-}1) + M(i{+}1,j) + \mathsf{e}(j{+}2,c{-}1) \\ M^c(i{+}1,j{+}1) + M^c(i,j) &= M(i{+}1,j{+}1) + \mathsf{e}(j{+}2,c{-}1) + M(i,j) + \mathsf{e}(j{+}1,c{-}1) \end{aligned}$$

Algorithm B, in Section 2.3, exploits the monotonicity of M^c and Algorithm C, in Section 2.4, exploits its total monotonicity. We will show

Theorem 2.1 *Given n isotonic weighted values $(\mathbf{y},\mathbf{w})$ and number of steps $b \le n$, Algorithm B finds an optimal L_2 b-step approximation (hence an optimal L_2 b-step reduced isotonic regression), in $\Theta(bn\log n)$ time, and Algorithm C finds one in $\Theta(bn)$ time.* □

2.3 Using Monotonicity

Let $\mathsf{j}_{\mathsf{min}}(i,b)$ denote the smallest j such that $\mathsf{e}'(i,j,b) = \mathsf{e}(i,b)$. As noted, $\mathsf{j}_{\mathsf{min}}(\cdot,b)$ is an isotonic function. This fact can be used to efficiently compute $\mathsf{e}(\cdot,b)$ and $\mathsf{j}_{\mathsf{min}}(\cdot,b)$ from the values of $\mathsf{e}(\cdot,b{-}1)$ and $\mathsf{j}_{\mathsf{min}}(\cdot,b{-}1)$. Figure 2 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.

This observation forms the basis of Algorithm B. Compared to Fisher's algorithm, for fixed c, the order in which $\mathsf{e}(i,c)$ values are determined is changed, as is the range of j values used to compute each value.

Proposition 2.2 *Given n isotonic weighted values $(\mathbf{y},\mathbf{w})$ and number of steps $b \le n$, Algorithm B finds an optimal b-step L_2 approximation in $\Theta(bn\log n)$ time.*

```
j_start . . . j_end :  range of possible endpoints

for i = 1 to n do
    e(i, 1) = err^p(i, n);   j_min(i, 1) = n
for c = 2 to b do
    for level = ⌊log2(n−c+1)⌋ downto 0 do
        for i = 2^level to n − c + 1 by 2^(level+1) do
            if i = 2^level then j_start = j
                else j_start = max{i, j_min(i − 2^k, c)}
            if i + 2^level > n − c + 1 then j_end = n − c + 1
                else j_end = j_min(i + 2^level, c)
            e(i, c) = min{e′(i, j, c) : j_start ≤ j ≤ j_end}
                {store largest minimizing j in j_min(i, c)}
        end for i
    end for level
end for c
generate the approximation using j_min and mean_p
```

Algorithm B: b-step L_p approximation of isotonic data, using monotonicity

---

Proof: Suppose that $\mathsf{e}(\cdot, c)$ and $\mathsf{j}_{\min}(\cdot, c)$ have been determined for $i_1 < i_2 \ldots < i_k$. Let $\ell_o \ldots \ell_k$ be such that $\ell_0 < i_1 < \ell_1 < i_2 \ldots < i_k < \ell_k$. To determine $\mathsf{e}(\cdot, c)$ and $\mathsf{j}_{\min}(\cdot, c)$ for the ℓ values, note that since $\mathsf{j}_{\min}(\cdot, c)$ is isotonic then $\mathsf{j}_{\min}(\ell_0, c) \in [\ell_0 : \mathsf{j}_{\min}(i_1, c)]$, $\mathsf{j}_{\min}(\ell_1, c) \in [\max\{\ell_1, \mathsf{j}_{\min}(i_1, c)\} : \mathsf{j}_{\min}(i_2, c)]$, $\ldots$, and $\mathsf{j}_{\min}(\ell_k, c) \in [\max\{\ell_k, \mathsf{j}_{\min}(i_k, c)\} : n-c+1]$. Thus, to determine $\mathsf{e}(\ell_0, c)$ and $\mathsf{j}_{\min}(\ell_0, c)$ we only need to evaluate $\mathsf{e}'(\ell_0, j, c)$ for $j \in [\ell_0 : \mathsf{j}_{\min}(i_1, c)]$; to determine $\mathsf{e}(\ell_1, c)$ and $\mathsf{j}_{\min}(\ell_1, c)$ we only need to evaluate $\mathsf{e}'(\ell_1, j, c)$ for $j \in [\max\{\ell_1, \mathsf{j}_{\min}(i_1, c)\} : \mathsf{j}_{\min}(i_2, b)]$; and so forth; i.e., we need at most $n + k$ total evaluations. In Figure 2, this corresponds to the fact that the dashed lines can overlap only at endpoints. In $1 + \lfloor \log_2 n \rfloor$ iterations all values of $\mathsf{e}(\cdot, c)$ and $\mathsf{j}_{\min}(\cdot, c)$ can be determined. This gives Algorithm B.

To complete the proof we need to show that each iteration of the "for level" loop can be completed in $\Theta(n)$ time. The j_start and j_end values that control the number of j values examined guarantee that, over all i values in in "for level" loop, a given j value is used at most twice. □

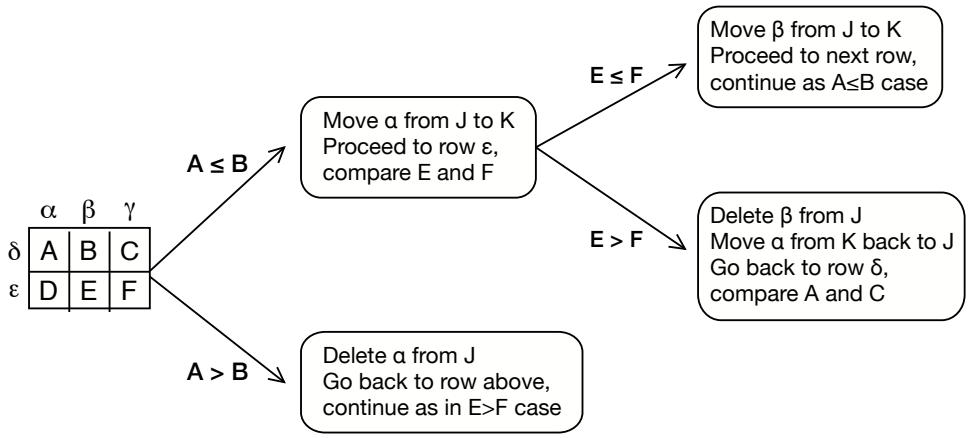
2.4 Using Total Monotonicity

The fact that M^c is totally mononotonic can be used to further reduce the total number of j values examined. Algorithm C replaces

$$\mathsf{e}(\mathsf{i}, \mathsf{c}) = \mathsf{min}\{\mathsf{e}'(\mathsf{i}, \mathsf{j}, \mathsf{c}) : \mathsf{j\_start} \le \mathsf{j} \le \mathsf{j\_end}\}$$

in Algorithm B with a while loop over a smaller set of j values, reducing the worst-case total number used at level k from $n - 2^k + 1$ to $\lfloor n/2^k \rfloor$. These j values are determined in Algorithm D. The approach used is known as the SMAWK algorithm, an anagram of the initials of the authors of [1]. It is likely that most readers are unfamiliar with SMAWK, and some might prefer to just view Algorithm D as a black box having the properties that for every c:

- for any level k and any i for which $\mathsf{j}_{\min}(i)$ is determined at level k, $\mathsf{jvalues}(k, \cdot)$ contains $\mathsf{j}_{\min}(i)$,
- the total number of j values returned over all levels is $O(n)$,



Currently at row δ, comparing A and B.
α, β, γ are the first 3 columns remaining in J; δ, ϵ are consecutive rows in the submatrix

Figure 3: An intermediate step of the SMAWK algorithm

- determine_jvalues takes $\Theta(n)$ time.

The pseudo-code given in Algorithm D is quite explicit, suitable for efficient implementation in any language. It converts the recursive list-based description in [1] to an iterative array-based one. Mention of eliminating columns, creating submatrices, etc., is merely symbolic since there aren't any real matrices: they are just conceptual representations of calculating $\epsilon'(i, j, c)$ values. The only arrays being used are to store j values.

To see how the SMAWK algorithm works, let M denote an arbitrary totally monotonic matrix. The algorithm starts with a list of columns J (jvalues), and a subset of them are moved to K and kept, with the remaining ones deleted. The final set of values in K will be the ones returned by determine_jvalues. When a column m is deleted from J and not put into K it is guaranteed that for all rows i, $m \neq \mathsf{j}_{\mathsf{min}}(i)$. The guarantees come about by exploiting two facts implied by the general Monge property (4): for the 2×2 submatrix with columns $\alpha < \beta$ and rows $\delta < \epsilon$,

a) if β is the minimal location in row δ, i.e., $M(\delta, \alpha) > M(\delta, \beta)$, then it is the minimal location in row ϵ, and hence in M α is not the minimal location in any row $\geq \delta$

b) if α is the minimal location in row ϵ, i.e., $M(\epsilon, \alpha) \leq M(\epsilon, \beta)$, then it is the minimal location in row δ, and hence in M β is not the minimal location in any row $\leq \epsilon$

At any step in the algorithm two adjacent entries of M are being compared, where they are in the same row and the first two columns (j values) remaining in J. For every row above the current row, one column has been moved into K. Suppose the algorithm is comparing A and B in Figure 3. If $A \leq B$ then it might be that $\alpha = \mathsf{j}_{\mathsf{min}}(\delta)$, and hence α is moved from J to K. Note that α might also be $\mathsf{j}_{\mathsf{min}}$ for some rows above and below δ. Relative to row δ, column β does not need to be kept. Further, for any row above δ, Monge property b) shows that β is not needed there either. However, it might be needed for lower rows,

so the algorithm proceeds to the next row, ϵ, and compares E and F. If $E \leq F$ then β is moved to K and the algorithm proceeds to the next row. However, if $E > F$ then β is not needed for row ϵ, and Monge property a) shows that it is not needed for any row below. Therefore β can be deleted from J, which in the implementation is done by merely incrementing next_j_index. Deleting β condenses the submatrix in Figure 3 to the entries A, C, D, and F. It might be that $A > C$, so the algorithm moves α from K back to J and goes back to row δ, comparing A and C. If A $\leq$ C then α is put back in K and the algorithm goes to the next row (ϵ), otherwise it is removed from J and the algorithm backs up another row, etc. If E $= \infty$, i.e., $\beta < \epsilon$, then we treat it as E $>$ F even if F $= \infty$.

If ϵ is the last row, if $E \leq F$ then γ can be deleted from J since there are no lower rows for which γ might need to be kept. Combining this with the rule that if $E > F$ then β is deleted and the algorithm goes back a row shows that if the last row is reached then all of the remaining columns are examined. Whether it occurs in the last row or earlier, eventually there is only 1 column left, which should be kept. Any row results in one column being moved to K, or is a row after the row in which the last column is reached, and hence $|K|$ is no more than the number of rows. Further, the time required is $\Theta(|J|)$.

To initialize, for level 0, which corresponds to all rows, all columns are kept, i.e., jvalues$(0, \mathsf{k}) = \mathsf{k}$ for $1 \leq \mathsf{k} \leq n$. One could apply the above reduction for level 0, but it isn't required for the time analysis nor correctness, and it slightly simplifies the implementation. At any level m above 0, the process is applied to the submatrix consisting of every second row of the submatrix used for level $m-1$, i.e., to rows that are multiples of 2^m. The initial J for level m is jvalues(m-1, 1 : num_jvalues(m-1)).

Proposition 2.3 *Given n isotonic weighted values $(\mathbf{y}, \mathbf{w})$ and number of steps $b \leq n$, Algorithm C finds an optimal b-step L_2 approximation in $\Theta(bn)$ time.*

Proof: Since each level halves the number of rows and the number of kept j values is no more than the number of rows, the total number of j values kept over all levels is $O(n)$ and the total time of determine_jvalues is $\Theta(n)$. The time for Algorithm C is linear in the total number of j values considered, so it too is $\Theta(n)$. □.

3 Reduced Isotonic Regression

For arbitrary data, isotonic regressions are somewhat easier to compute than are general approximations by step functions. One can 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 is known as "pool adjacent violators", PAV, and first appeared in 1955 in Ayer et al. [2]. For L_2 it can easily be computed in only $\Theta(n)$ time.

Isotonic regression is a very flexible nonparametric approach to many problems. However it does have its detractors due to results with impractically many steps or overfitting. Some researchers have instead used approximations with a specified number of steps [5, 19]. To reduce overfitting, Schell and Singh [16] used the approach of repeatedly merging pairs of adjacent steps whose difference had the least statistical significance. Haiminen et al. [5] used an approach that repeatedly combines the adjacent steps that cause a minimum increase in the error. These greedy (aka myopic) approaches repeatedly make the choice that seems to be the best at the moment, but may not produce an optimal reduced isotonic regression. For example, for all L_p, $1 < p \leq \infty$, given the unweighted values 0, 2, 4, 6, 8, 10, the unique optimal 3-step isotonic regression is 1, 1, 5, 5, 9, 9, and the unique optimal 2-step isotonic regression is 2, 2, 2, 8, 8, 8. Thus the 2-step isotonic regression cannot be obtained by merging steps of the 3-step isotonic regression.

The fastest previous algorithm for optimal L_2 reduced isotonic regression is due to Haiminen et al. [5], taking $\Theta(n + bm^2)$ time, where m is the number of pieces in the unconstrained isotonic regression. As

```
integer array jvalues(0:⌊log2⌋, 1:n), num_jvalues(0:⌊log2 n⌋)

for i = 1 to n do
    e(i, 1) = err^p(i, n);   j_min(i, 1) = n
for c = 2 to b do
    determine_jvalues(jvalues, num_jvalues, c)   {see Algorithm D}
    for level = ⌊log2(n−c+1)⌋ downto 0 do
        for i = 2^level to n − c + 1 by 2^(level+1) do
            if i = 2^level then j_start = i;  j_index = 1
                else j_start = max{i, j_min(i − 2^level, c)}
            if i + 2^level > n − c + 1 then j_end = n − c + 1
                else j_end = j_min(i + 2^level, c)
            e(i, c) = ∞
            while (j_index ≤ num_jvalues(level)) ∧ (jvalues(level, j_index) ≤ j_end) do
                j = jvalues(level, j_index)
                if (j ≥ j_start) ∧ (e'(i, j, c) < e(i, c)) then
                    e(i, c) = e'(i, j, c);  j_min(i, c) = j
                j_index = j_index + 1
            end while
            j_index = j_index − 1
        end for i
    end for level
end for c
generate the approximation using j_min and mean_p
```

Algorithm C: b-step L_p approximation of isotonic data, using total monotonicity for determine_jvalues

```
procedure determine_jvalues(jvalues, num_jvalues, c)

num_jvalues(0) = n
for k = 1 to n do jvalues(0, k) = k
for level = 1 to ⌊log2(n−c+1)⌋
    j = jvalues(level−1, 1);  next_j_index = 2;  k_index=0
    i = 2^level
    while next_j_index ≤ num_jvalues(level−1) do
        next_j = jvalues(level−1, next_j_index)
        if (j ≥ i) ∧ (e′(i, j, c) ≤ e′(i, next_j, c)) then
            if i + 2^level > n − c + 1 then {at last row, eliminate next_j}
                next_j_index = next_j_index + 1
            else {keep this j, increment i, j}
                k_index = k_index + 1;  jvalues(level, k_index) = j
                j = next_j;  next_j_index = next_j_index + 1
                i = i + 2^level
            end if
        else {e′(i, j, c) > e′(i, next_j, c), eliminate current j, go back to previous i, j}
            if i > 2^level then
                i = i − 2^level;  j = jvalues(level, k_index);  k_index = k_index − 1
            else {at first row}
                j = next_j;  next_j_index = next_j_index + 1
            endif
        end if
    end while
    k_index = k_index + 1;  jvalues(level, k_index) = j
    num_jvalues(level) = k_index
end for level
end determine_jvalues
```

Algorithm D: Reducing the number of relevant j values using SMAWK

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. Even though often $m \ll n$, Haiminen et al. felt that this may be too slow so they developed the greedy heuristic mentioned above. Our exact algorithms should be sufficiently fast even for very large problems.

One cannot directly find b-step reduced isotonic regression of arbitrary data by using the approaches in Algorithms B and C since it does not have the required monotonic properties. For example, for unweighted values 7, 8, 0, 6, 9, 10, the optimal L_2 2-step reduced isotonic regression has its first step on the interval $[1\!:\!4]$, while the optimal first step for the data starting at position 3 is the interval $[3\!:\!3]$, i.e., $4 = \mathsf{j}_{\mathsf{min}}(1,2) \not\leq \mathsf{j}_{\mathsf{min}}(3,2) = 3$. Howevever, a critical observation in Haiminen et al. [5] 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. Each piece becomes a weighted point, where the value of the point is the mean of the piece and the weight of the point is the total weight of the piece. In the above example, the data would be represented by the 4 weighted points (5,3), (6,1), (9,1), (10,1), and the first step of a 2-step reduced isotonic regression uses the first two pieces.

Their observation gives a simple algorithm: find the unrestricted isotonic regression, convert the pieces to weighted points, and then find a b-step approximation of these isotonic points. Haiminen et al. used Fisher's algorithm to determine the optimal b-step reduced isotonic regression in $\Theta(n + bm^2)$ time, but Algorithms B and C provide faster solutions.

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 via Algorithm B, and in $\Theta(n + bm)$ time via Algorithm C, where m is the number of pieces in the unconstrained L_2 isotonic regression.* □

Unfortunately, for $p \neq 2$ the optimal reduced isotonic regression might not be formed from pieces of the unrestricted isotonic regression. For example, for unweighted values -10, -10, -10, 0, 0, 0, -10, -1, 7, 7, 7, 7, the unique L_1 unrestricted isotonic regression has pieces $[1:3]$, $[4:8]$, and $[9:12]$, with values -10, 0, 7, respectively. The unique optimal 2-step reduced isotonic regression has steps $[1:7]$ and $[8:12]$, with values -10 and 7, which requires cleaving the middle piece. However, one can determine an approximation by constructing an optimal b-step isotonic regression among those restricted to use unbroken pieces of the unrestricted isotonic regression. By doing so, the problem is now similar to isotonic regression on isotonic data. An algorithm using this approach to approximate L_1 reduced isotonic regression appears in [7]. It is more complicated than the L_2 case since to determine medians one needs to retain the values in the original pieces, rather than combining them into a single weighted value as can be done for L_2.

For L_∞ an optimal b-step reduced isotonic regression, and an optimal b-step approximation with no isotonic restrictions, can be found in $\Theta(n + \log n \cdot b(1 + \log n/b))$ time [17]. The approaches used there are quite different, unrelated to dynamic programming.

4 Final Comments

The thousands of citations to the books by Barlow et al. [3] and Robertson et al. [14] shows a significant interest in isotonic regression. Further, this interest is growing as researchers seek to remove parametric assumptions from their modeling. Similarly, step functions with a constraint on the number of steps arise in a wide range of applications and guises [4, 6, 8, 9, 10, 11, 13, 19]. For reduced isotonic regression both aspects are important [5, 15, 16], using a reduced number of steps to simplify the regression and/or prevent overfitting.

However, researchers used approximations, rather than the optimal answer, due to the slowness of the available algorithms. The fastest previous algorithm for optimal L_2 b-step reduced isotonic regression takes $\Theta(n+bm^2)$ time [5], where m is the number of pieces in the unconstrained isotonic regression. Algorithm B reduces this to $\Theta(n+bm\log m)$ time, and the somewhat more complicated Algorithm C further reduces this to $\Theta(n+bm)$. Note that the minimal time for optimal b-step approximation, with no isotonic restrictions, is a long-standing open question.

Fisher [4] called the b-step approximations "restricted homogenization", and defined another form of approximation that he called "unrestricted homogenization": given n weighted values $(\mathbf{y}, \mathbf{w})$ and $b \in [1:n]$, partition the values 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. This is now known as *k-means clustering* of 1-dimensional data, for $k = b$. He noted it could be solved by sorting the values and then finding the optimal b-step approximation, i.e., the optimal b-step isotonic regression of the sorted data. Thus for 1-dimensional data Algorithm B solves the k-means clustering problem in $\Theta(kn\log n)$ time, and for sorted data Algorithm C reduces this to $\Theta(kn)$.

Finally, an interesting problem is that of selecting the most desirable number of steps. For reduced isotonic regression, Schell and Singh [16], Strobl et al. [18] and Haiminen et al. [5] start with an unconstrained isotonic regression and then repeatedly merge pieces until their criteria are met. However, Haiminen et al. showed that the regression error of their greedy approximation can be nearly twice that of the optimal reduced isotonic regression with the same number of steps. They believe that 2 is an upper bound on the relative error of their approximation, but that has not been proven, nor have bounds been proven for other approximation schemes. For b-step approximation, many researchers choose b *a priori* based on considerations such as storage or access time requirements. This seems to be especially true in the database community, where L_2 b-step approximations are known as "v-optimal histograms".

In contrast, the dynamic programming approach generates optimal b-step reduced isotonic regressions for each value of b as b increases. One can stop when a criterion is met and always have an optimal result. However, appropriate stopping criteria for a given application may be somewhat subtle since they would be applied repeatedly.

Acknowledgements

Research partially supported by NSF grant CDI-1027192 and DOE grant DE-FC52-08NA28616. Some of these results were announced in [7].

References

[1] Aggarwal, A, Klawe, MA, Moran, S, Shor, P and Wilber, R (1987), "Geometric applications of a matrix-searching algorithm", *Algorithmica* 2, pp. 195–208.

[2] 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.

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

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

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

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

[7] Hardwick, J and Stout, QF (2012), "Optimal reduced isotonic reduction", *Proc. Interface 2012*, May 2012.

[8] 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.

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

[10] 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.

[11] Mayster, Y and Lopez, MA (2006), "Approximating a set of points by a step function", *J. Vis. Commun. Image R.* 17, pp. 1178–1189.

[12] Niculescu-Mizil, A, and Caruana, R (2005), "Predicting good probabilities with supervised learning", *Proc. Int'l. Conf. Machine Learning* 22, pp. 625–632.

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

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

[15] 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.

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

[17] Stout, QF (2014), "An algorithm for L_∞ approximation by a step function", arXiv 1412.2379

[18] Strobl, R, Salanti, F, and Ulm, K (2003), "Extension of CART using multiple splits under order restrictions", Discussion paper, Sonderforschungsbereich 386 der Ludwig-Maximilians-Universitat Munchen, No. 364

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