

Best L_p Isotonic Regressions, $p \in \{0, 1, \infty\}$

Quentin F. Stout

Computer Science and Engineering, University of Michigan

qstout@umich.edu www.eecs.umich.edu/~qstout/

Abstract: Given a real-valued weighted function f on a finite dag, the L_p isotonic regression of f , $p \in [0, \infty]$, is unique except when $p \in [0, 1] \cup \{\infty\}$. We are interested in determining a “best” isotonic regression for $p \in \{0, 1, \infty\}$, where by best we mean a regression satisfying stronger properties than merely having minimal norm. One approach is to use strict L_p regression, which is the limit of the best L_q approximation as q approaches p , and another is lex regression, which is based on lexical ordering of regression errors. For L_∞ the strict and lex regressions are unique and the same. For L_1 , strict $q \searrow 1$ is unique, but we show that $q \nearrow 1$ may not be, and even when it is unique the two limits may not be the same. For L_0 , in general neither of the strict and lex regressions are unique, nor do they always have the same set of optimal regressions, but by expanding the objectives of L_p optimization to $p < 0$ we show $p \nearrow 0$ is the same as lex regression. We also give algorithms for computing the best L_p isotonic regression in certain situations.

Keywords: strict isotonic regression, lex regression, monotonic, Polya approach, L_0 , L_1 , L_∞ , Hamming distance

1 Introduction

This paper considers isotonic regression (also known as monotonic regression) on an arbitrary finite dag $G = (V, E)$. A real-valued function h on G is *isotonic* if for all vertices $u, v \in V$, if $u \prec v$ (i.e., if there is a path from u to v) then $h(u) \leq h(v)$, i.e., it is a weakly monotonic function from G into \mathcal{R} . We are given a weighted function (f, w) on V , where f is a real-valued function and w is the positive real-valued weights, and wish to produce a real-valued isotonic function g that is closest to f . If all the weights are the same then we say that the function is unweighted. Isotonic regression is a form of nonparametric regression, and hence very useful when parametric assumptions are unwarranted. Currently Google lists tens of thousands of results for a search on “isotonic regression”. They have become quite important in data science and machine learning, with all of the major software packages in these areas including algorithms to compute them. An extensive overview of such applications appears in [3].

Normally the distance between f and g is measured in terms of the weighted L_p distance, i.e., for $1 \leq p < \infty$ is

$$\left(\sum_{v \in V} w(v) \cdot |f(v) - g(v)|^p \right)^{1/p}$$

Finding a g that minimizes this is the same as finding one that minimizes the same sum without taking the $1/p$ root. This then extends to $0 < p < 1$, where there is no longer a true norm, but there is an F-norm. We also extend to $p = 0$ (the Hamming distance) and $p = \infty$. Thus the goal is to minimize

$$\begin{aligned} \sum_{v \in V} w(v) \cdot \mathbf{1}(f(v) \neq g(v)) & \quad p = 0 \\ \sum_{v \in V} w(v) \cdot |f(v) - g(v)|^p & \quad 0 < p < \infty \\ \max_{v \in V} w(v) \cdot |f(v) - g(v)| & \quad p = \infty \end{aligned}$$

among all isotonic functions g on G . If g is an isotonic function minimizing this quantity then we say g is an L_p isotonic regression of f .

An important special case is when G is a simple linear order and f is a non-increasing function. In this case the best isotonic regressions will be constant functions, and the value of any constant minimizing the regression error is a *weighted L_p mean*. For weighted means the location of the function values is solely determined by the function values, so we often just consider the values of f without specifically noting which vertex has which function value, i.e., it reduces to considering the weighted mean of a multiset of weighted values. Let $\text{mid}_p(S)$ denote the set of weighted L_p means of S .

For $p \in (1, \infty)$ it is well-known that an L_p regression g is always unique due to the strict convexity of the objective function, and similarly $\text{mid}_p(S)$ is unique for any S . For $p \in [0, 1]$ the objective function is not strictly convex and L_p regression, nor $\text{mid}_p(S)$, need not be unique. E.g., for all such p , for unweighted data 1, 0 on a linear order both 1, 1 and 0, 0 are optimal L_p isotonic regressions, and are the elements in mid_p for $p \in (0, 1)$. For $p \in (0, 1)$ the objective is in fact strictly concave. For $p = 1$ the objective behavior is a bit more subtle, and mid_1 is either a single point or an interval. For example, $\text{mid}_1(\{0, 1\}) = [0, 1]$. For $p = 0$ $\text{mid}_0(S)$ is the set of values with maximum total weight.

This paper is concerned with $p = 0, 1, \infty$, values for which the isotonic regression need not be unique. A natural question is whether there are “best” L_0 , L_1 and L_∞ isotonic regressions. Section 2 gives two approaches to this question. One is via limits of L_p regressions as $p \rightarrow 0, 1$, or ∞ ; the other is via lexical ordering of the regression values, useful for L_0 and L_∞ . Section 3 discusses the fact that for L_∞ the limit from below and a lexical approach give the same regression, Section 4 shows that for L_1 the two limit approaches give different results, and Section 5 shows that for L_0 the limit from above and a lexical approach give different results, but if L_p optimization is extended to $p < 0$ then the limit from below and the lexical approach are the same.

Section 6 gives algorithms for computing these regressions on linear orders using partitioning and “pool adjacent violators” (PAV), and Section 7 gives some concluding remarks.

2 Definitions

To determine best isotonic regressions, one approach is by looking at the limiting behavior of L_q regression as q approaches p . These limits are sometimes called *strict* isotonic regressions. Given a weighted function f on a dag, for L_∞ we consider the limit of L_p regressions as $p \nearrow \infty$, denoted $\text{strict}^{\uparrow\infty}(f)$ (Section 3); for L_1 we consider $\text{strict}^{\uparrow 1}(f)$ and $\text{strict}_{\downarrow 1}(f)$ (Section 4); and for L_0 consider $\text{strict}_{\downarrow 0}(f)$ (Section 5). Given the strict convexity of the objective functions being optimized, for $1 < p < \infty$ it is not difficult to show that $\text{strict}^{\uparrow p}(f)$ and $\text{strict}_{\downarrow p}(f)$ are the unique L_p isotonic regression. Many problems have been analyzed using $\text{strict}_{\downarrow 1}(f)$ and $\text{strict}^{\uparrow\infty}(f)$ [4, 5, 7, 15]. The use of $\text{strict}^{\uparrow\infty}(f)$ is sometimes called the *Polya approach*, and, less frequently, the use of $\text{strict}_{\downarrow 1}(f)$ is called the *Polya-1 approach*. While here $\text{strict}^{\uparrow\infty}(f)$ and $\text{strict}_{\downarrow 1}(f)$ are unique, for some classes of functions on infinite sets this is not true [12].

We also consider another approach that has been used to determine “best” L_∞ and L_0 regressions. For an isotonic regression g of an unweighted function f on a DAG of n vertices, take the regression errors of g at the vertices and sort them in decreasing order, giving a vector of n entries. Order all such vectors lexically, and let v be the minimal vector in this ordering. While there are infinitely many isotonic regressions of f it can be shown that v is well-defined and corresponds to a unique isotonic regression [23] h which minimizes the L_∞ error. h has been called the minimizing lex regression or similar terms. Here we denote it as $\text{lex}_\infty(f)$. For L_0 on unweighted functions use the same process, but now have the vector entries in increasing vertex error. Once again there is a well-defined minimal vector v in the lexical ordering of the

vectors, though there may be multiple isotonic regressions corresponding to v (Sec. 5), each of which is an L_0 optimal isotonic regression of f . We denote these regressions as $\text{lex}_0(f)$, though they have also been called *strong L_0 regression* [25].

For weighted values the definition of the lexical orderings have to be generalized a bit. For $\text{lex}_\infty(f, w)$, for isotonic regressions g, h define $g \prec h$ iff there is an $\alpha > 0$ such that

$$\sum \{w(v) : |g(v) - f(v)| \geq \alpha, v \in V\} < \sum \{w(v) : |h(v) - f(v)| \geq \alpha, v \in V\}$$

and for all $\beta > \alpha$,

$$\sum \{w(v) : |g(v) - f(v)| \geq \beta, v \in V\} = \sum \{w(v) : |h(v) - f(v)| \geq \beta, v \in V\}$$

For $\text{lex}_0(f, w)$, for isotonic regressions g, h define $g \prec h$ iff there is an $\alpha > 0$ such that

$$\sum \{w(v) : |g(v) - f(v)| \leq \alpha, v \in V\} > \sum \{w(v) : |h(v) - f(v)| \leq \alpha, v \in V\}$$

and for all $\beta < \alpha$,

$$\sum \{w(v) : |g(v) - f(v)| \leq \beta, v \in V\} = \sum \{w(v) : |h(v) - f(v)| \leq \beta, v \in V\}$$

As before, $\text{lex}_\infty(f, w)$ and $\text{lex}_0(f, w)$ are initial elements in their orderings.

It had previously been shown that $\text{strict}^{\uparrow\infty}(f) = \text{lex}_\infty(f)$ [23], while in Section 5 we show that $\text{strict}_{\downarrow 0}(f)$ is not always the same as $\text{lex}_0(f)$. In Section 4 we show how to approximate $\text{strict}_{\downarrow 1}$ and that in general it is different than $\text{strict}^{\uparrow 1}$. In the Appendix we show that if an L_∞ isotonic regression algorithm satisfies monotonicity and level set trimming then it is in fact lex_∞ . A slightly improved algorithm for finding the lex_∞ regression is given in Section 3.

For $p \in [0, \infty)$ let $\text{mid}_{\downarrow p}(S)$ be $\lim_{q \searrow p} \text{mid}_q(S)$, and for $p \in (0, \infty]$, $\text{mid}^{\uparrow p}(S)$ denotes $\lim_{q \nearrow p} \text{mid}_q(S)$.

If g is an isotonic function on G then a set $S \subseteq V$ is a *level set* if it is a maximal weakly connected set where all the values of g are the same. It is straightforward to show that for any $p \in (0, \infty)$, if g is an optimal L_p isotonic regression then for any level set S of g , g 's value on the level set is in $\text{mid}_p(S)$.

Given a function f on a dag $G = (V, E)$, say that $u, v \in V$ are a *violating pair* if u precedes v in G but $f(u) > f(v)$, i.e., they violate the isotonic condition. The fastest known algorithms for determining lex_∞ and lex_0 on a general dag are based on first finding all violating pairs. This can be done in time linear in the time to find the transitive closure of G , which can be done in $O(nm, n^\omega)$ time, where $n = |V|$, $m = |E|$, and ω is such that binary matrix multiplication can be done in $O(n^\omega)$ time. However, there are some orderings where the violating pairs can be found more quickly. For example, suppose the dag has vertices where the ordering relationship can be determined directly from pairwise comparisons of the vertices' labels and the edges are not explicitly given. The violating edges can therefore be determined in $O(n^2)$ time. An example of this is where vertex labels are strings and $u \prec v$ iff u is a substring of v (though a timing analysis should take the time of the comparisons into account since strings can be of varying size). Another example is planar rectangles of arbitrary orientation and size, where $p \prec q$ iff p is contained in q (and, more generally, polyhedral containment in d -dimensional space). A particularly important class where edges are given implicitly are points in d -dimensional space, where $u = (u_1, \dots, u_d) \prec v = (v_1, \dots, v_d)$ iff $u_i \preceq v_i$ for all $1 \leq i \leq d$. This is sometimes known as *domination ordering*, where v dominates u , and it is also known as *multidimensional ordering* or *coordinate-wise ordering*. For these orderings the transitive closure can be found in $\Theta(C + n \log^d n)$ time, where C is the size of the transitive closure and where the implied constants depend on d . Use of these facts to quickly find L_0 regressions appears in [25].

3 L_∞

The best L_∞ regression has been defined both as $\text{strict}^{\uparrow\infty}$ (the Polya approach) and as lex_∞ . A proof that these are the same, and the regression is unique, appears in [23]. To date apparently all efficient algorithms for determining the regression on finite dags are based on the lex_∞ definition. For an arbitrary dag of n vertices and m edges a $O(\min\{nm, n^\omega\} + n^2 \log n)$ time algorithm appears in [23] (where ω is such that boolean matrix multiplication can be performed in $O(n^\omega)$ time), and the `CompLexMin` algorithm in [9] computes it in $\Theta(nm)$ expected time. Algorithm A below is a simplification of [23] which should be the fastest in practice.

Algorithm A is essentially as follows: for each violating pair of vertices (u, v) determine the weighted average $x = (w_u f(u) + w_v f(v)) / (w_u + w_v)$, i.e., their $w\text{mean}_p$ for any $1 \leq p \leq \infty$, and let their `mean_err` be $w_u \cdot |f(u) - x|$, which is the same as $w_v \cdot |f(v) - x|$. Create a record which includes u, v , and the `mean_err` of their weighted function values. Sort these records by the `mean_err` value in decreasing order, and then process them in sequence. If the next record contains vertices $u \prec v$ and neither has had their regression value determined yet set the regression values to $x = w\text{mean}$, set the upper bound of all predecessors of v to the maximum of their previous value and x , and set the lower bound of all successors of u to the minimum of their previous value and x . If both have had their regression value determined already discard the record, and if one has had the regression value defined while the other hasn't the steps are shown in Algorithm A. A proof of correctness is essentially the same as the proof of the algorithm in [23].

The total time of the algorithm is the time to find all violating pairs, the time to sort the records, and the time to process them. For any violating pair $p \prec q$ the conceptual edge from p to q is used at most twice, once when p 's regression value is determined and once when q 's is. Since the time to process the records is linear in the number of records, the sorting time dominates the remaining steps other than the time to find the violating pairs. In terms of the original dag this is at most $\Theta(n^2 \log n)$, and thus A's worst-case time is $O(\min\{nm, n^\omega\} + n^2 \log n)$, i.e., the same as the time of the algorithm in [23] though some of the steps are slightly faster. Note that for dags such as those discussed at the end of Section 2 the time for finding the violating pairs is $\Theta(n^2)$, in which case Algorithm A takes $\Theta(n^2 \log n)$ time. Further, in terms of the number of violating pairs, m^* , the time to sort and process is $\Theta(m^* \log m^*)$ and thus the more in-order the data is the faster the later parts of the algorithm are.

4 L_1

Most published L_1 isotonic regression algorithms use a median data value as the value of a level set [1, 10, 13, 19, 20, 21, 24, 26], greatly simplifying the search for optimal regression values. Other regression values can also optimize the L_1 regression error, and have additional desirable properties. However, computing them can be more complicated. We will show that for a set S of weighted real numbers, in general $\text{mid}_{\downarrow 1}(S) \neq \text{mid}^{\uparrow 1}(S)$, and hence in general $\text{strict}_{\downarrow 1}(f) \neq \text{strict}^{\uparrow 1}(f)$. When f is unweighted data on a linear order, with values 1, 0, algorithms based on data values would result in either 0, 0 or 1, 1 as the isotonic regression. Meanwhile, $\text{strict}_{\downarrow 1}(f)$ is 0.5, 0.5 and $\text{mid}_{\downarrow 1}(\{0, 1\})$ is $\{0.5\}$, while $\text{strict}^{\uparrow 1}(f)$ is 0, 0 or 1, 1 and $\text{mid}^{\uparrow 1}(\{0, 1\})$ is $\{0, 1\}$. While some attention had been paid to $\text{mid}_{\downarrow 1}$, apparently none had been to $\text{mid}^{\uparrow 1}$.

One could also define versions of best L_1 isotonic regressions in terms of lexical properties. Take the set of L_1 optimal regressions, order their regression errors as for lex_∞ , and take a regression corresponding to the first element in this ordering. Denote this by $\text{lex}_{1, \infty}$. For a set S of weighted real numbers let $\text{mid}_{1, \infty}(S)$ be the regression of smallest L_∞ error among $\text{mid}_1(S)$. For unweighted data $S = \{1, 1, 3, 7\}$: $\text{mid}_{1, \infty}(S)$

input: weighted data (f,w) , lists of successors and predecessors for each vertex in dag G
output: $S = \text{lex}_\infty(G, f, w)$
violators: array of $(\text{mean_error}, u, v)$ for violating pairs $u \prec v, f(u) > f(v)$
 $\text{lowbd}(v), \text{upbd}(v)$: lower and upper bounds on $S(v)$

```

numviolate=0
for every vertex v
    lowbd(v) =  $-\infty$ ; upbd(v) =  $+\infty$ ; S(v) = undefined
    for every successor s of v
        if  $f(v) > f(s)$  then violators(numviolate) = (mean_err(v,s), v, s); numviolate++

sort violators by decreasing order of mean_err. For ties sort by decreasing order of weight

for i=0 to numviolate-1
    (mean_err,pred,suc)=violators(i)
    if (S(pred) defined)  $\vee$  (S(suc) defined) then cycle
    wmean = weighted mean of pred and succ
    if  $w\text{mean} \geq \text{upbd}(\text{pred})$  then {f(pred) is  $\geq \text{upbd}(\text{pred})$ , no later mean is  $< \text{upbd}(\text{pred})$ }
        S(pred) = upbd(pred)
    if  $w\text{mean} \leq \text{lowbd}(\text{suc})$  then {f(suc) is  $\leq \text{lowbd}(\text{suc})$ , no later mean is  $> \text{lowbd}(\text{suc})$ }
        S(suc) = lowbd(suc)
    if (S(pred) undefined)  $\wedge$  (S(suc) undefined) then {low(suc)  $\leq w\text{mean} \leq \text{high}(\text{pred})$ }
        S(pred) = S(suc) = wmean

    if S(pred) defined then
        for every successor s of pred
            lowbd(s) =  $\max\{\text{lowbd}(s), S(\text{pred})\}$ 
    if S(suc) defined then
        for every predecessor p of suc
            upbd(p) =  $\min\{\text{upbd}(p), S(\text{suc})\}$ 
end for i

for every vertex v
    if S(v) undefined then
        if  $f(v) \geq \text{upbd}(v)$  then S(v)=upbd(v)
        else if  $f(v) \leq \text{lowbd}(v)$  then S(v)=lowbd(v)
        else S(v)=f(v)

```

Algorithm A: Computing $S = \text{lex}_\infty(G, f, w)$ ($= \text{strict}^{\uparrow\infty}(G, f, w)$) via Transitive Closure

is 3; $\text{mid}_\infty(S)$ is 4; $\text{mid}_{\downarrow 1}(S)$ is a value in (1,3) that is closer to 3 than to 1 (see Sec. 4.1); and $\text{mid}^{\uparrow 1}(S) = 1$ (see Sec. 4.2). A similar technique can be used for $\text{lex}_{1,0}$, though this is not always unique. For example, for $S = \{1, 2, 3, 4\}$, $\text{mid}_{1,0}(S) = \{2, 3\}$ while $\text{mid}_1(S) = [2, 3]$. While there is likely some interest in $\text{lex}_{1,\infty}$ and $\text{lex}_{1,0}$, they won't be pursued any further here.

4.1 $\text{strict}_{\downarrow 1}$

Jackson [8] was apparently the first to determine $\text{mid}_{\downarrow 1}(S)$. He only considered unweighted data, but the extension to weighted data is straightforward. If S has a unique median value then that is $\text{mid}_{\downarrow 1}(S)$. Otherwise, a nonempty S must have unique values $a < b$ where both are weighted medians. In this case $\text{mid}_{\downarrow 1}(S)$ is the unique value $c \in (a, b)$ such that

$$\prod_{y_i \leq a} (c - y_i)^{w_i} = \prod_{y_i \geq b} (y_i - c)^{w_i}$$

or, equivalently,

$$\sum_{y_i \leq a} w_i \ln(c - y_i) = \sum_{y_i \geq b} w_i \ln(y_i - c)$$

Finding an L_1 isotonic regression where all level sets have data values and then using Jackson's formula on the regression values to determine the regression value of the level set does not always produce $\text{strict}_{\downarrow 1}$. For example, on a linear order with unweighted data values 0, -2, 2, 0 one optimal L_1 isotonic regression is 0, 0, 0, 0. Applying Jackson's formula to this level set regression values would of course give the same values, but $\text{strict}_{\downarrow 1}$ is -1, -1, 1, 1. Note that the level sets are not the same, let alone the regression values. Jackson's formula needs to be applied to the original data values, not the regression values.

It appears that no previous algorithm has been published which determines $\text{strict}_{\downarrow 1}$ for isotonic regression on finite sets, though $\text{strict}_{\downarrow 1}$ isotonic regression has been examined in settings where the functions are integrable [7], and has been examined in more general settings where the goal is to find a closest point on a closed convex set [11]. To find a close approximation of $\text{strict}_{\downarrow 1}$ one can determine a $p > 1$ close enough to 1 so that the L_p isotonic regression is sufficiently close to $\text{strict}_{\downarrow 1}$. Then approximate the L_p isotonic regression. To simplify, assume we want an approximation to within 2δ pointwise, $\delta > 0$, and each of these two steps has its parameters chosen to produce an approximation within δ at each vertex.

Suppose all the function values are integers in the range $[-h, h]$ and the weights are integers in the range $[0, W]$. If the values and weights aren't integers then just round and do pointwise approximation to within $\delta/2$. For a given $p > 1$, for any set S of n or fewer weighted values, the largest difference between the weighted L_1 mean of S and the L_p mean occurs if all vertices have weight W and half have value h while the other half have value $-h$. The difference in means is $nWh - (nWh^p)^{1/p}$, which is $\leq \delta$ iff $nW - (nW)^{1/p} \leq \delta/h$. Rearranging and taking the log of both sides, this will be true if $1/p \geq \ln(nW - \delta/h) / \ln(nW)$, which will hold if p is sufficiently close to 1. Note that this approach can be used for arbitrary L_1 regression, not just isotonic regression [11].

To find an approximation to the L_p isotonic regression there are several approaches. In [10] they use interior point methods, while [24, 26] iteratively use an algorithm for weighted L_1 binary-valued isotonic regression to converge to an approximation in a logarithmic number of steps. For a connected dag with n vertices and m edges, the algorithm in [10] produces an approximation within δ in $O(m^{1.5} \log^2 n \log(n/\delta))$ time, while the approach in [24] depends on the dag. If $K = nhW$ then the time is $\Theta(n \log K)$ if the dag is linear, a tree, or a 2-dimensional grid; $\Theta(n \log n \log K)$ if the dag is arbitrary points in 2-dimensional space with domination ordering; $\Theta(n^2 \log n \log K)$ for a d -dimensional grid, $d \geq 3$, and $\Theta(n^2 \log^d n \log K)$ for

arbitrary points in d -dimensional space with domination ordering (for these results the implied constants depend upon d).

The fastest known algorithms for arbitrary dags are approximations based on algorithms for L_0 [26], which are in turn based on flow algorithms [16]. Given a flow algorithm F , let $\mathcal{F}(n, m, nU)$ denote the time to solve an integer-valued flow on a graph with n vertices, m edges, and integer edge flow limits in $[nU]$ for some positive integer U . Let $G' = (V', E')$ be a violator dag of a dag $G = (V, E)$, where G' has n' vertices and m' edges. Then for $1 < p < \infty$, given a weighted function (f, w) on G with integer weights in $[0, U]$, and given G' , for $\delta > 0$ an isotonic function within δ of the L_p isotonic regression of f can be found in $O(\mathcal{F}(n', m', nU) \log K)$ time, where $K = (\max_{v \in V} f(v) - \min_{v \in V} f(v)) / \delta$ and where the implied constants in the O -notation depend on p . A recent improvement in flow algorithms [2] reduces their time to $\tilde{O}(m)$ if K grows at most polynomially in n , and thus the total time is bounded by this plus the time to construct the violator graph.

4.2 strict[↑]

Minimizing $\sum_{v \in V} w(v) \cdot |f(v) - g(v)|^p$ when $p \in (0, 1)$ is different than minimizing it when $p \geq 1$ because for $p < 1$, $|f(v) - y|^p$ is a concave function of y on $y \leq f(v)$ and on $y \geq f(v)$. This in contrast to it being convex throughout the full range of y when $p \geq 1$, and strictly convex when $p > 1$. Given the set $S = \{f(v) : v \in V\}$, let a, b two elements of S where $a < b$ and there are no elements of S in (a, b) . Then for $0 < p < 1$, the L_p error of using y as the $\text{mid}^{\uparrow 1}(S)$ is a concave function on $[a, b]$. This because each term of the error sum is concave in y and the sum of concave functions is concave. Thus the minimum on $[a, b]$ is achieved when $y = a$ or $y = b$. Since this is true for any consecutive pair of data values, the minimum, i.e. $\text{mid}^{\uparrow 1}(S)$, is achieved at one of the data values.

First we show that the value is a median. If the regression value is chosen to be \hat{y} , then, letting $p = 1 - \epsilon$, for $\epsilon > 0$ sufficiently close to 0, the Taylor expansion series shows that the regression error is

$$\sum_{v \in V} w_v |f(v) - \hat{y}| (1 + \text{l.o.t.})$$

The sum with the final factor in each term being 1 instead of $(1 + \text{l.o.t.})$ is the sum under the L_1 norm, so it is the medians which minimize it. Since the minimizer for $\text{mid}^{\uparrow 1}(S)$ must be a data value, it must be a median of S , either the unique median if S has a unique median, and otherwise there are unique values $a < b$ in S which are medians of S . Thus $\text{mid}^{\uparrow 1}(S)$ is either a or b . To determine if it is a , let $c(v) = |f(v) - a|$ and $d(v) = |f(v) - b|$. The relevant question is if

$$\sum_{v \in V, v \neq a} w(v) c(v)^{1-\epsilon} < \sum_{v \in V, v \neq b} w(v) d(v)^{1-\epsilon}$$

is true for ϵ sufficiently small. Using Taylor expansions, and the fact that $\sum_{v \in V} w(v) d(v) = \sum_{v \in V} w(v) c(v)$ (since both are medians), this will be true if

$$\sum_{v \in V, v \neq a} w(v) c(v) \ln c(v) < \sum_{v \in V, v \neq b} w(v) d(v) \ln d(v)$$

or, equivalently,

$$\prod_{v \in V, v \neq a} c(v)^{w(v)c(v)} < \prod_{v \in V, v \neq b} d(v)^{w(v)d(v)}$$

$\text{mid}^{\uparrow 1}(S)$ will be b if the inequality is reversed. If these are equal then one can go to the next order term in the Taylor expansion, etc.

Note that if there is not a unique median value then $\text{mid}_{\downarrow 1}(S)$ cannot be one of the data values, while $\text{mid}^{\uparrow 1}(S)$ is always a data value. Thus in general $\text{strict}^{\uparrow 1}$ is not the same as $\text{strict}_{\downarrow 1}$. Further, they need not have the same level sets. On a linear order, if the unweighted data values are 1, 1, -10, -11, 0, 0, -2, -3, then for the first 4 entries $\text{mid}^{\uparrow 1}$ is 1, 1, 1, 1 and for the second 4 is 0, 0, 0, 0, while $\text{mid}_{\downarrow 1}$ is negative on the first 4 and negative but somewhat larger on the second 4. For $\text{mid}^{\uparrow 1}$ the 8 elements have to be merged into a single level set with value in $\{-2, 0\}$, while for $\text{mid}_{\downarrow 1}$ the level sets are not merged.

There does not appear to be any published algorithm for determining isotonic regressions when $0 < p < 1$, so currently one cannot approximate $\text{strict}^{\uparrow 1}$ by using a published L_p approximation algorithm for $p < 1$ sufficiently close to 1.

5 L_0

The definition of the best L_0 regression as being lex_0 apparently first appeared in [25], where in the first version of that paper it was called strong L_0 isotonic regression. In general lex_0 is not the same as $\text{strict}_{\downarrow 0}$ since for 11, 8, 5, 2, 0, lex_0 is 2 and $\text{strict}_{\downarrow 0}$ is 5. However, we can consider $\text{strict}^{\uparrow 0}$ by using an extension to negative p . For $0 > p > -\infty$ let

$$\|f - g\|_p = \left(\sum_{v \in V} w(v) \cdot |f(v) - g(v)|^p \right)^{1/p}$$

This is not a norm, but we have the same goal as before, namely in minimizing it. Since $p < 0$, this requires *maximizing* $\sum_{v \in V} w(v) \cdot |f(v) - g(v)|^p$. Viewing the set of optimal g as a function of p , we define $\text{strict}^{\uparrow 0}$ as the set $\lim_{p \nearrow 0}$. We will show that this is lex_0 .

To explain our approach assume all the weights are 1. The approach works for arbitrary positive weights, but unit weights simplifies the explanation. Recall that to determine lex_0 for each L_0 regression we consider its errors at the vertices in nondecreasing order. For regression A denote this list by a_1, \dots, a_n , where $a_1 \leq a_2 \leq \dots \leq a_n$, and for regression B denote its list by b_1, \dots, b_n , where $b_1 \leq b_2 \leq \dots \leq b_n$. We will show that if A 's list precedes B 's in lexical order then A has smaller L_p error for p sufficiently close to 0, $p < 0$.

A 's precedes B 's iff there is an index i , $0 \leq i \leq n - 1$, such that $a_j = b_j$ for $1 \leq j \leq i$, and $a_{i+1} < b_{i+1}$. We can ignore the values of a_j and b_j for $j \leq i$ since they contribute the same to each sum. Let $A_p = \sum_{k=i+1}^n a_k^{1/p}$ and $B_p = \sum_{k=i+1}^n b_k^{1/p}$. Then the sum over the vertex regression errors of A to the p^{th} power, minus the vertex regression errors of B to the p^{th} power, is $A_p - B_p$. Since $p < 0$ and the a_i and b_i are nondecreasing, this is at least the value one would have if for all $j > i + 1$, $b_j = b_{i+1}$ and a_j is extremely large. In fact, we can let a_j be infinite and $1/a_j = 0$. Thus $A_p - B_p > a_{i+1}^p - (n - i)b_{i+1}^p$. As $p \rightarrow 0$, $(a_{i+1}/b_{i+1})^p \rightarrow \infty$ since $a_{i+1} < b_{i+1}$ and p is negative. Since $(n - i)$ is a constant, for small enough p , $a_{i+1}^p > (n - i)b_{i+1}^p$ (recall we were trying to maximize the sum of the errors to the p^{th} power). Thus if A precedes B in the lex_0 ordering it precedes B in the $\text{strict}^{\uparrow 0}$ ordering of regression errors. This implies that iflex_0 is $\text{strict}^{\uparrow 0}$.

Note that lex_0 , and hence also $\text{strict}^{\uparrow 0}$, are not unique since for 1, 0 on a linear order both 0, 0 and 1, 1 are optimal. Nor are they monotonic since their unique regression of 6, 6, 4, 2, 0 is all 6s, while increasing the 2 and 0 values to 4 gives a unique regression of all 4s.

6 Linear Orders

Our approach for determining $\text{strict}_{\downarrow 1}$ on a linear order is based on partitioning, using 0-1 isotonic regression to iteratively narrow down to the regression value for each point [24]. We describe the process in terms of regression values being in boxes, where the horizontal extent of each box is the interval of vertices it is representing and the vertical extent is guaranteed to include the regression value of every vertex in the box. For each iteration there are 2 stages: the first partitions each box (or the box may be unchanged), and the second pools (merges) partitions of adjacent boxes to maintain the isotonic property. At the end of an iteration the vertical extents of the boxes are consecutive intervals that are the same, followed by another sequence of vertical extents that are higher but all the same, followed by another sequence, etc. The vertical extents are intervals or single points. Figure 1 shows a sequence of steps in the pooling stage.

Initially each data point is in its own box, with the vertical extent being the range from smallest to largest data value. At an intermediate stage, the boxes with vertical extent which is the open interval (a, b) form a sequence (the lower bound a may be included if it is the smallest data value, and similarly for the upper bound b). Let $[i, j]$ be the sequence of index values, i.e., the box is $[i, j] \times (a, b)$. Denote this box by $B(i, j, a, b)$, and let $W = \sum_{k=i}^j w_k$. Let \hat{y} be any value in (a, b) , and let

$$S_{\square}(i, j, \hat{y}) = \sum \{w_k : i \leq k \leq j, y(k) \square \hat{y}\} \quad \text{for } \square \in \{<, =, >\}$$

and

$$P_{\square}(i, j, \hat{y}) = \prod \{|b - y(k)|^{w_k} : i \leq k \leq j, y(k) \square \hat{y}\} \quad \text{for } \square \in \{\leq, \geq\}$$

These values have several properties. First, for all of the blocks combined they can be computed in $\Theta(n)$ time. Further, for a given block the S and P values can be used to determine where the median interval is located and its relation to \hat{y} . Also, when two adjacent blocks with the same vertical extent are merged the S , P , and W values of the resulting block can be computed in constant time from the values of the blocks being merged. This fact is important because the 2nd step of each stage merges blocks, using *pool adjacent violators* (PAV). This is a standard operation in isotonic regression on a linear order, merging adjacent blocks with regression values in the wrong order into a larger block and then determining its regression value, which will be in the interval of the lower regression value (from the block on the right) to the larger regression value (the block on the left). In the most common implementation of PAV the blocks are examined in increasing order of the range of their independent variable, and if an out of order pair is discovered they are merged and the interval of regression values determined. This may be smaller than the preceding block, in which case they are merged, etc. Fig. 1 illustrates this process, and Fig. 3 gives an implementation used for $\text{strict}_{\downarrow 1}$. The total number of mergers throughout the entire algorithm is $\leq n - 1$.

An important aspect of Jackson's formula is that the P values are only relevant if \hat{y} is in the interval of median values and the interval has more than one point. If the interval of medians is $[c, d]$ and $\hat{y} \in [c, d]$ then when $P_{\leq}(\cdot) = P_{\geq}(\cdot)$ the optimal median is \hat{y} ; when $P_{\leq}(\cdot) > P_{\geq}(\cdot)$ then the optimal median is $< \hat{y}$; and otherwise is $> \hat{y}$. During this iteration of the partitioning process $B(i, j, a, b)$ is replaced by $B(i, j, \hat{y}, \hat{y})$ (i.e., a vertical extent of a single point) if $S_{<}(\cdot)$ and $S_{>}(\cdot)$ are both $< W/2$ or \hat{y} is one of the median values and $P_{\leq}(\cdot) = P_{\geq}(\cdot)$; else is replaced by $B(i, j, a, \hat{y})$ if \hat{y} is in the interior of the interval of medians and $P_{\leq}(\cdot) > P_{\geq}(\cdot)$; else is replaced by $B(i, \hat{y}, b)$.

The only part of the algorithm still unspecified is how to choose the \hat{y} values. There are several options depending on the objectives one wants. One way is to use data values, initially using a median data value and then at each iteration, for the sequence of blocks with vertical interval (a, b) , choosing the data value which is a median of those in that range. This would take $\Theta(n \log n)$ time, and for each level set of $\text{strict}_{\downarrow 1}(f)$

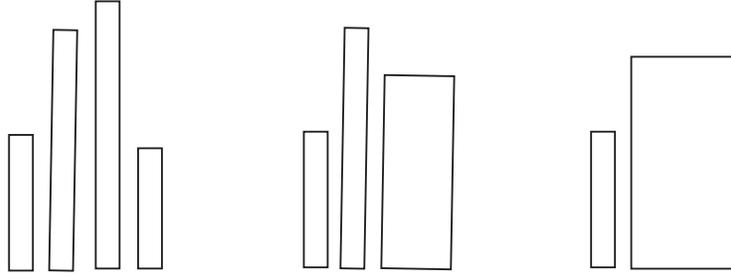


Figure 1: Steps in PAV (pool adjacent violators)

If \hat{y} is a data value in B (i.e., $S_{=} > 0$) then
 if $S_{<}$ and $S_{>}$ are $< W/2$ then \hat{y} is the unique median, shrink B to $B'(i, j, \hat{y}, \hat{y})$
 else \hat{y} cannot be the desired median
 if $S_{<} \geq W/2$ then (a, \hat{y}) contains the desired median, shrink to $B'(i, j, a, \hat{y})$
 else (\hat{y}, b) contains the median, shrink to $B'(i, j, \hat{y}, b)$
 else \hat{y} is not a data value
 if $S_{<} = S_{>}$ {both are $W/2$, \hat{y} is in the median interval, use Jackson's formula}
 if $P_{<} = P_{>}$ shrink to $B'(i, j, \hat{y}, \hat{y})$
 else if $P_{<} > P_{>}$ shrink to $B'(i, j, a, \hat{y})$
 else shrink to $B'(i, j, \hat{y}, b)$
 else \hat{y} is not in the median interval
 if $S_{<} > W/2$ shrink to $B'(i, j, a, \hat{y})$
 else shrink to $B'(i, j, \hat{y}, b)$

Figure 2: Partitioning a block for strict \downarrow_1

For the sequence of blocks all initially with vertical extent (a, b) in this stage

Determine \hat{y}
 Partition the first block {Figure 2}
 While there are still blocks in this sequence
 Let $B(i, j, a, b)$ be the next block
 Partition B , resulting in $B'(i, j, a', b')$ {note that (a', b') is (a, \hat{y}) , $[\hat{y}, \hat{y}]$, or (\hat{y}, b) }
 Repeat
 Let $B^*(h, i-1, a^*, b^*)$ be the predecessor of B {it has already been partitioned}
 If (a^*, b^*) is above (a', b') then repeat {the blocks are not isotonic}
 reset B to be the block (h, j, a, b) {i.e., merge the unpartitioned B and B^* },
 using the s, p , and W values from B and B^* to calculate those for the new B
 partition B , resulting in a new $B'(i, j, a', b')$
 until (a^*, b^*) is not above (a', b')

Go to the next sequence of blocks (their vertical extents will be above the current one)

Figure 3: PAV for a sequence of partitioned blocks with same initial vertical extent

	definition	time	reference
L_0	lex ₀ strict ^{↑0}	$\Theta(n^3)$?	[25]
L_1	strict _{↓1} strict ^{↑1}	$\Theta(n \log(\max\{n, U/\delta\}))$? See note	Section 6
L_∞	lex _∞ strict ^{↑∞}	$\Theta(n \log n)$?	[23]

Table 1: Fastest Known Isotonic Regression Algorithms for Weighted Data on a Linear Order

? : Apparently no algorithm is based on this definition

Note: The two definitions do not always produce the same regression

which is a data value the algorithm would identify this as being the level set’s value. Another option is to guarantee that at the end each interval is within δ of its true value. In this case one chooses $\hat{y} = (b - a)/2$, continuing this until $b - a \leq \delta$ for every block. This would take $\Theta(n \log(U/\delta))$ time. One could combine these, first using partitioning based on data values followed by dividing the range for blocks that do not have a data value as their correct regression value. In this case the time is $\Theta(n \log(\max\{n, U/\delta\}))$. One typically assumes that if the largest absolute value of any data value is U then U/δ grows at most polynomially with n , in which case the time is $\Theta(n \log n)$.

7 Conclusion

For real-valued weighted data on a finite dag L_p isotonic regression is typically defined purely in terms of minimizing the regression error, but here the “best” L_p isotonic regressions used additional criteria. For $1 < p < \infty$ the regression is unique and therefore imposing additional criteria is not useful. However, for $p \in [0, 1] \cup \infty$ there may be infinitely many regressions minimizing the regression error, and such criteria helps select among them.

For example, for unweighted data f on a dag G most researchers use the function $g(x) = (\max\{f(y) : y \preceq x\} + \min\{f(y) : x \preceq y\})/2$ as the optimal L_∞ regression of choice. While its simplicity and speed of computation recommend it, for data 2, 0, 1 on a linear order this results in 1, 1, 1.5. In fact, any function of the form 1, 1, r minimizes the L_∞ norm if $r \in [1, 2]$. However, likely many would prefer to use 1, 1, 1. This is both the regression obtained as the limit, as $p \rightarrow \infty$, of the L_p regression, $\text{strict}^{\uparrow\infty}(f)$, and the unique one defined via a lexical ordering of the regression errors, $\text{lex}_\infty(f)$ (see Sec. 2). The lex_∞ definition is based on explicitly minimizing large errors, not just minimizing the maximum error, and appears in [10, 23]. The $\text{strict}^{\uparrow\infty}$ definition has been used repeatedly in various analytical situations, not just isotonic regression, and is known as the Polya approach. For finite dags the definitions are equivalent but lex_∞ is more useful in terms of developing efficient algorithms.

There are numerous papers on minimizing L_1 error in various settings involving convex cones, but few are as focused as Jackson [8] in defining the best median value of a set as $\text{mid}_{\downarrow 1}$. The set of isotonic regressions forms a convex cone. Almost all papers finding L_1 regressions on finite sets result in one where all regression values are data values, but for a level set with non-unique median Jackson’s value will never be a data value. Trying to define a best L_1 regression as $\text{strict}^{\uparrow 1}$ is not as successful since L_p -optimal

regressions are not unique when $p < 1$, and the objective function is concave, not convex (Sec. 4.2).

Far less attention has been paid to L_0 , but a natural lexical definition, lex_0 , restricts the regressions minimizing the L_0 metric down to a much smaller set. lex_0 is based on maximizing the number of small errors, not just maximizing the weight of points with 0 error. A definition of best was also given in terms of limits, $\text{strict}^{\uparrow 0}$, by extending the minimization of L_p objectives to negative values of p . It was shown that $\text{strict}^{\uparrow 0} = \text{lex}_0$ (Sec. 5).

Many others have studied related problems such as the rate of convergence of $\text{strict}^{\uparrow \infty}$ and $\text{strict}_{\downarrow 1}$ [5, 15], other ways to select subsets of L_0 isotonic regressions with desired properties [17, 25], generating all L_0 isotonic regressions [22] or their core [18], and finding a minimal L_1 isotonic regression [21].

There are several open questions concerning algorithms for the problems studied here. For example, it is known how to use maximal flow algorithms to find L_0 isotonic regressions of weighted data on arbitrary dags [6, 14, 17, 25]. However, these algorithms are only guaranteed to produce L_0 isotonic regressions, not lex_0 regressions. It would be interesting to find a more efficient algorithm for general dags. For L_1 , for general dags the algorithms for finding isotonic regressions do not always produce $\text{strict}_{\downarrow 1}$, and it would be useful to find an efficient algorithm that produces $\text{strict}_{\downarrow 1}$ directly, rather than through approximation as in Sec. 4.1.

References

- [1] Angelov, S; Harb, B; Kannan, S; Wang, L-S (2006), “Weighted isotonic regression under the ℓ_1 norm”, Symp. on Discrete Algorithms (SODA), 783–791.
- [2] van den Brend, J; Lee, YT; Liu, YP; Saranurak, T; Sidford, A; Song, Z; Wang, D (2021), “Minimum cost flows, MDPs, and L_1 regression in nearly linear time for dense instances”, arXiv:2001.005719.
- [3] Cano, jR; Gutierrez, PA; Krawczyk, B; Wozniak, B; Garcia, S (2019), “Monotonic classification: an overview on algorithms, performance measures and data sets”, Neurocomputing 341, 168–182.
- [4] Darst, RB; Legg, DA; Townsend, DW (1983), “The Polya algorithm in L_∞ approximation”, J. Approx. Theory 38, 209–220.
- [5] Egger, AG; Taylor, GD (1993), “Rate of convergence of the discrete Polya-1 algorithm”, J. Approx. Theory 75, 312–324.
- [6] Feelders, A; Velikova, M; Daniels, H (2006), “Two polynomial algorithms for relabeling non-monotone data”, Tech. Report UU-CS-2006-046, Dept. Info. Com. Sci., Utrecht Univ.
- [7] Huotari, R; Legg, D; Meyerowitz, AD; Townsend, D (1988), “The natural best L_1 -approximation by nondecreasing functions”, J. Approx. Theory 52, 132–140.
- [8] Jackson, D (1921), “Note on the median of a set of numbers”, Bull. Am. Math. Soc. 27, 160–164.
- [9] Kyng, R; Rao, A; Sachdeva, S; Spielman, DA (2015), “Algorithms for Lipschitz learning on graphs”, JMLR: Workshop and Conference Proceedings 40:134.
- [10] Kyng, R; Rao, A; Sachdeva, S (2015), “Fast, provable algorithms for isotonic regression in all L_p norms”, NIPS.

- [11] Landers, D; Rogge, L. (1981), “Natural choice of L_1 approximants”, J. Approx. Theory 38, 268–280.
- [12] Marano, M; Quesada, JM (2007), “The behavior of best L_p -approximations as $p \rightarrow 1$. A counterexample of convergence”, J. Approx. Theory, 233–237.
- [13] Pardalos, PM; Xue, G-L; Yong, L (1994), “Efficient computation of an isotonic median regression”, Appl. Math Lett. 8, 67–70.
- [14] Pijls, W; Potharst, R (2014), “Repairing non-monotone ordinal data sets by changing class labels”, Econometric Inst. Report EI 2014-29.
- [15] Quesada, JM; Fernández-Ochoa, J; Martínez-Moreno, J; Bustamante, J (2005), “The Polya algorithm in sequence spaces”, J. Approx. Theory 135, 245–257.
- [16] Rademaker, M; De Baets, B; De Meyer, H (2006), “On the role of maximal independent sets in cleaning data for supervised ranking”, 2006 IEEE Int’l. Conf. on Fuzzy Systems, 1619–1624.
- [17] Rademaker, M; De Baets, B; De Meyer, H (2009), “Loss optimal monotone relabeling of noisy multi-criteria data sets”, Info. Sci. 179, 4089–4096.
- [18] Rademaker, M; De Baets, B; De Meyer, H (2012), “Optimal monotone relabeling of partially non-monotone ordinal data”, Optimization Methods and Soft. 27, 17–31.
- [19] Robertson, T; Wright (1973), “Multiple isotonic median regression,” Ann. Statist. 1, 422–432.
- [20] Rote, G (2018), “Isotonic regression by dynamic programming”, 2nd Symp. on Simplicity in Algorithms (SOSA), 1–18.
- [21] Shi, N-Z (1995), “The minimal L_1 isotonic regression”, Commun. Stat. - Theory Meth. 24, 175–189
- [22] Stegeman, L; Feelders, A, “On generating all optimal monotone classifications”, in: *Data Mining (ICDM)*, 2011 IEEE 11th Int’l. Conf. on, 685–694.
- [23] Stout, QF (2012), “Strict L_∞ isotonic regression”, J. Opt. Theory and App. 152: 121–135.
- [24] Stout, QF (2013), “Isotonic regression via partitioning”, *Algorithmica* 66:93–112.
- [25] Stout, QF (2021), “ L_0 isotonic regression with secondary objectives”, arXiv:2106.00279.
- [26] Stout, QF (2021), “ L_p isotonic regression algorithms using an L_0 approach”, arXiv:2107.00251.

Appendix

In general there are widely varying L_∞ isotonic regressions of a specific dag and data, and in [23] there are characterizations of various properties of L_∞ isotonic regression algorithms. Two properties of particular interest are monotonicity and maintaining level set trimming. For an L_∞ isotonic regression algorithm \mathcal{A} , for dag $G = (V, E)$ and weighted function (f, w) on G let $\mathcal{A}(G, f, w)$ denote the isotonic regression that \mathcal{A} produces. \mathcal{A} is an L_∞ isotonic operator on G if $\mathcal{A}(G, f, w)$ is an L_∞ isotonic regression of (f, w) for all (f, w) .

\mathcal{A} is *monotonic* on G iff for all weight functions w and functions f and g on V , if f is pointwise less than or equal to g on V then $\mathcal{A}(G, f, w)$ is pointwise less than or equal to $\mathcal{A}(G, g, w)$. Algorithm \mathcal{A} *preserves level set trimming* on G iff for any weighted function (f, w) on G , for any level set L of $\mathcal{A}(G, f, w)$, the regression values on L are $\text{mid}^{\uparrow\infty}((f, w)|L)$

In [23] it was shown that $\text{strict}^{\uparrow\infty}$ is monotonic and preserves level set trimming for all dags. The following shows that this characterizes $\text{strict}^{\uparrow\infty}$, in that any L_∞ isotonic regression algorithm which is monotonic and preserves level set trimming of weighted data functions on a dag G always produces $\text{strict}^{\uparrow\infty}$ on G . It does not appear in [23] since the author only noticed it after that paper was in the publication processes.

Theorem 1 *For any DAG $G = (V, E)$ and L_∞ isotonic regression algorithm \mathcal{A} on G , if \mathcal{A} is monotonic and preserves level set trimming then \mathcal{A} always produces $\text{strict}^{\uparrow\infty}$ (and hence lex_∞) on G .*

Proof: We use proof by contradiction. Suppose \mathcal{A} is monotonic and preserves level set trimming, and there is a weighted function (f, w) for which $\mathcal{A}(G, f, w) \neq \text{strict}^{\uparrow\infty}(G, f, w)$. Among the level sets of $\text{strict}^{\uparrow\infty}(G, f, w)$ which are not level sets of \mathcal{A} , or where the regression values differ, let L be one of maximal error. Let f_1 be f trimmed on all level sets of $\text{strict}^{\uparrow\infty}(G, f, w)$ with error greater than L 's. Then $\text{strict}^{\uparrow\infty}(G, f_1, w) = \text{strict}^{\uparrow\infty}(G, f, w)$, $\mathcal{A}(G, f_1, w) = \mathcal{A}(G, f, w)$, and the L_∞ regression error of $\text{strict}^{\uparrow\infty}(G, f_1, w)$ is its L_∞ error on L (there may be other level sets with the same error). Let c be the value of $\text{strict}^{\uparrow\infty}(G, f_1, w)$ on L . If $\mathcal{A}(G, f_1, w)$ does not equal c on all of L then it has larger regression error than $\text{strict}^{\uparrow\infty}(G, f_1, w)$, in which case it is not an isotonic regression. Otherwise, it has a level set $B \supsetneq L$ with regression value c . Let $B' = \{u : u \in B, \text{strict}^{\uparrow\infty}(G, f_1, w)(u) < c\}$ (if B' is empty then a similar proof can be applied to the set where $\text{strict}^{\uparrow\infty}(G, f_1, w) > c$). Since $\text{strict}^{\uparrow\infty}((f_1, w)) < c$ on B' , and raising the values to c would not violate the isotonic condition, it must be that $\text{mid}_\infty(B') < c$.

Let f_2 be the function formed by trimming f_1 on all level sets of $\mathcal{A}(G, f_1, w)$ except B . Then $\mathcal{A}(G, f_2, w) = \mathcal{A}(G, f_1, w)$. Define $f_3(u)$ to be $f_2(u)$ if $f_2(u) < c$ or $u \in B'$, and M otherwise, where M is the maximum value of f_2 . Since $\mathcal{A}(G, f_2, w) = c$ on B' and pointwise $f_3 \geq f_2$, by monotonicity $\mathcal{A}(G, f_3, w) \geq c$ on B' . Let h be the function where $h(u) = f_3(u)$ on $V \setminus B'$, and $h(u) = \max\{\text{mid}_\infty(B'), D\}$ on B' , where D is the largest value of f_3 less than c . Then h is isotonic, and as a regression of f_3 has no error on $V \setminus B'$ and smaller L_∞ error than $\mathcal{A}(G, f_3, w)$ on B' . Thus $\mathcal{A}(G, f_3, w)$ is not optimal and hence \mathcal{A} is not an L_∞ regression operator on G .

□