Fastest Known Isotonic Regression Algorithms

Quentin F. Stout

qstout@umich.edu

University of Michigan
Ann Arbor, MI

June 2022

Abstract

This note is a status report on the fastest known isotonic regression algorithms for various L_p metrics and partial orderings. The metrics considered are unweighted and weighted L_0, L_1, L_2, and L_∞. The partial orderings considered are linear, tree, d-dimensional grids, points in d-dimensional space with component-wise ordering, and arbitrary orderings (posets). Throughout, “fastest” means for the worst case in O-notation, not in any measurements of implementations. This note will occasionally be updated as better algorithms are developed. Citations are to the first paper to give a correct algorithm with the given time bound, though in some cases two are cited if they appeared nearly contemporaneously.

Keywords: isotonic regression algorithm, shape-constrained nonparametric regression, linear order, tree, multidimensional grid, coordinate-wise ordering, dag, poset

1 Introduction

A directed acyclic graph (dag) $G(V, E)$ with n vertices $V = \{v_1, ..., v_n\}$ and m edges defines a partial order (poset) over the vertices, where $v_i \prec v_j$ if and only if there is a path from v_i to v_j. It is assumed that G is connected, and hence $m \geq n-1$. If it isn't connected then the algorithms would be applied to each component independently of the others. A real-valued function $\mathbf{z} = (z_1 \dots z_n)$ on G is *isotonic* if whenever $v_i \prec v_j$, then $z_i \leq z_j$, i.e., it is a weakly order-preserving map from G to $\Re$. In some contexts this is known as a monotonic function. By *data* $(\mathbf{y}, \mathbf{w})$ on G we mean there is a weighted value (y_i, w_i) at vertex v_i, $1 \leq i \leq n$, where y_i is an arbitrary real number and w_i, the weight, is ≥ 0. By unweighted data we mean $w_i = 1$ for all i.

For $1 \leq p \leq \infty$, or $p = 0$, given data $(\mathbf{y}, \mathbf{w})$ on dag $G(V, E)$, an L_p *isotonic regression* of the data is an isotonic function $\mathbf{z}$ over V that minimizes

$$\begin{array}{ll} \left(\sum_{i=1}^n w_i |y_i - z_i|^p\right)^{1/p} & 1 \leq p < \infty \\ \max_{i=1}^n w_i |y_i - z_i| & p = \infty \\ \sum_{i=1}^n w_i \cdot (y_i \neq z_i) & p = 0 \end{array}$$

among all isotonic functions. The L_p *regression error* is the value of this expression.

Note that if $v_i \prec v_j \prec v_k$, then for any isotonic function $\mathbf{z}$, if $z_i = z_k$ then z_j has the same value. A set $V' \subset V$ is a *level set* of $\mathbf{z}$ iff it is a maximal order-closed subset where all the values are the same. Order-closed means that if $v_i \prec v_j \prec v_k$ and $v_i, v_k \in V'$ then $v_j \in V'$. An isotonic function may have disjoint level sets with the same value. The value of the level set of an isotonic regression depends upon the metric, and is discussed in the sections below.

The orderings listed in the tables are linear (also known as total), rooted tree, points in multidimensional space with component-wise ordering, and general (i.e., an algorithm that applies to all orderings). A dag of points in multidimensional space is the isotonic version of multivariate regression. In d-dimensional space (the "dim" orderings), point $p = (p_1, \ldots, p_d)$ precedes point $q = (q_1, \ldots, q_d)$ iff $p_i \leq q_i$ for all $1 \leq i \leq d$. This is the product ordering of the linear coordinate orders. In some settings, q is said to dominate p. In the tables the multidimensional orderings are further subdivided into regular grids and points in arbitrary positions, and into dimension 2 and dimension ≥ 3. They are subdivided like this because there are different algorithms that can be used in these cases. Throughout, the analysis of time for points or grids in d-space assumes d is fixed and $n \to \infty$. The implied constants in the O-notation depend on d, but in general the papers do not explicitly determine them.

This is a compendium of the fastest known algorithms so far, not an historical review nor a survey of applications. There are many applications, a tiny random sample of which includes [6, 7, 9, 12, 14, 16, 19, 22, 23, 35]). The books [3, 25] contain numerous applications, though the books are far out of date.

I've omitted related topics such as unimodal regression, prefix isotonic regression, convex regression, river regression, isotonic regression with constraints on the number of level sets ("reduced isotonic regression") or on the differences between adjacent ones (Lipschitz), etc. No parallel algorithms are considered since regrettably there has been no interesting work in this area, even though they would be useful for large data sets.

The tables list the best times known to me, with citations to the relevant references. In the "weighted" and "unweighted" columns all algorithms are exact (to within machine error) for arbitrary real inputs, and all times are worst-case. Throughout "fastest time" is in terms of O-notation, not on any measurements of implementations, though pointers to a few implementations are included (see remark 2 in the Final Remarks 7). For all orderings except the most general one, time is given as a function of n, while for the algorithms for arbitrary dags time is given as a function of n and m. While m may be as large as $\binom{n}{2}$, for most dags of interest it is far smaller. In particular, $m = \tilde{\Theta}(n)$ for all of the other orderings considered here. However, for L_0 isotonic regression a dag with small m might be converted into a violator dag (see Section 3) where $m = \Theta(n^2)$.

Originally I did not include approximations nor algorithms with fast expected time but slow worst-case time. However, in many cases far simpler, but slower in O-notation, algorithms may be much more useful, as might algorithms with only expected case guarantees on their time, and approximations may be acceptable. Thus I've now included some such algorithms. To help make it clearer what type of algorithm is being discussed, when using Θ (or O or o): Θ_e indicates that it is expected time; Θ_δ indicates the result is accurate to within δ, where the time depends on δ; and Θ_u indicates that it is pseudo-polynomial, with the values and weights integers in $[0, U]$, where U grows at most polynomially in n and the time depends on U. These are listed in the tables in the "other" column. They are usually dependent on maximum flow algorithms, and in [34] the isotonic regression algorithms are explicitly written so that improvements in the times of flow algorithms directly give faster isotonic regression algorithms. Algorithms with time $o_u(n^{1.5})$ currently rely on using the flow algorithm in [10] which takes $\tilde{O}(m^{\frac{3}{2}-\frac{1}{328}} \log U)$ time, and others rely on the flow algorithm in [5] which takes $\tilde{O}(m + n^{\frac{3}{2}} \log U)$ time.

Throughout, ω represents the smallest value such that matrix multiplication can be done in $\Theta(n^\omega)$ time. While Strassen's algorithm (with $\omega = \log_2 7 \approx 2.81$) is practical, galactic algorithms achieving values < 2.4 have appeared. Thus one may want to interpret algorithms in terms of the smallest ω known so far, or in terms of a practical value. Many ignore Strassen's algorithm despite the fact that it is quite practical.

2 Cross-cutting Techniques

There are some approaches that have been used for all of the metrics. One is that if there is a vertex q such that $f(p) \leq f(q)$ for all $p \prec q$, and $f(q) \leq f(p)$ for all $p \succ q$, then one can always choose an isotonic regression of minimal error where the value at q is unchanged. In some cases removing q from the dag would reduce

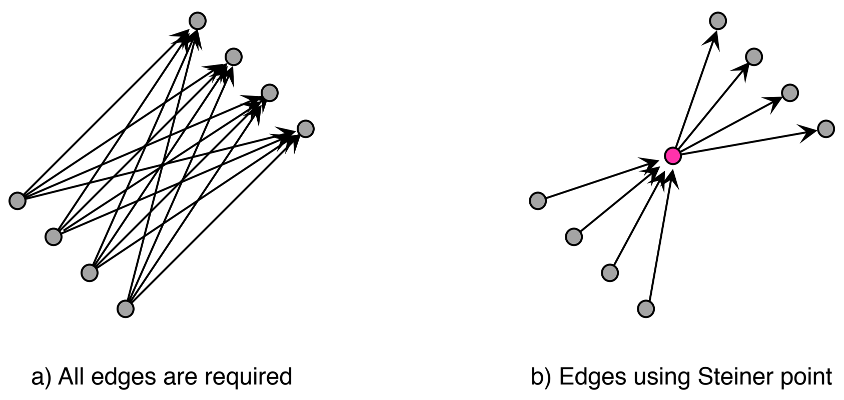

Figure 1: Dag edges for Component-wise Ordering in 2 Dimensions

the time, while in other cases it might be kept in because the number of edges may increase if it is removed because it might have to be replaced with edges from its immediate predecessors to its immediate successors.

2.1 Linear Orders

For linear orders the "pool adjacent violators", PAV, approach has been repeatedly rediscovered. To incrementally construct an isotonic regression using PAV, start with the initial data values. Whenever there are consecutive level sets A and B, where A precedes B but the regression value on A is greater than that of B (i.e., they are a violating pair), then they are joined together to form a new level set, and its regression value is determined. This continues until there are no more violating pairs. Level sets can be pooled in any order and the process will still result in an isotonic regression. In practice a simple left-right scan is used. For the L_2 metric it is trivial to implement in linear time, while for L_1 more complicated data structures are needed to achieve the fastest known time of $\Theta(n \log n)$ [1, 27].

Algorithms for L_0 do not rely on PAV [8, 24], using a longest nondecreasing sequence approach instead. For the L_∞ metric with unweighted data PAV can be used, but the generic topological sort approach mentioned in Section 6 is easier and faster. For the L_∞ metric with weighted data, previously the fastest algorithm used PAV, taking $\Theta\left(n \log n\right)$ time, but now the fastest takes $\Theta\left(n\right)$ time and is not based on PAV [31] (but is far more complex, so probably slower in practice).

Unfortunately, while PAV can also be used for trees it does not apply to more general orderings, not even 2-dimensional grids. Even for trees adjacent violating subtree level sets cannot be paired in arbitrary order. This is discussed in [20].

2.2 Points in d-dimensional Space

For points in d-dimensional space with simple component-wise ordering there is no requirement that a dimension has real coordinates, merely that it is linearly ordered (well ordered). For example, one dimension may be S, M, L, XL shirt sizes. For d-dimensional grids n points require $< nd$ edges to represent the partial order (it is strictly less than nd because of points on the boundary). Unfortunately, n points in arbitrary locations may require $\Theta(n^2)$ edges to represent the partial order, even if transitivity is taken into account. This is shown in Figure 1 a). However, sometimes adding points, called Steiner points, can reduce the number of edges required, as in Figure 1 b).

So far all of the fastest algorithms for points in d-dimensional space, $d \geq 3$, are based on order-preserving embeddings. Given set P of n d-dimensional points, they are embedded into a dag $G = (P', E)$, where $P \subset P'$, and for any $s, t \in P$, s precedes t in component-wise ordering iff s precedes t in G. G has $\Theta\left(n \log^{d-1} n\right)$ vertices and edges, and can be constructed in time linear in its size ([30]). Points in $P' \setminus P$ are given weight 0, and the isotonic regression for G is determined. This induces an isotonic regression on P.

This approach was first used in the original (2008) version of [32], but was subsequently moved to [30]. For L_1 and L_2 and $d \geq 3$, G is explicitly created and then the algorithms for general dags are applied to G. The same approach is used for L_0 and $d \geq 2$. For L_∞, the algorithm in [31] only uses G conceptually, simulating it via repeated sorting and taking only $\Theta(n)$ space. It is not based on using the L_∞ algorithm for general dags.

A symmetric version of G, where all dimensions are treated the same as opposed to having one kept as a standard linear ordering, has $\Theta\left(n \log^{d} n\right)$ vertices and edges. It too appears in [30] and is the same as the Steiner 2-transitive-closure discussed in [4].

Another use of this approach is to generate a violator graph of points. Given a function f on a dag $G = (V, E)$, a pair (u, v) of points in V is a *violating pair* if $u \prec v$ but $f(u) > f(v)$, i.e., they violate the isotonic requirement. Some algorithms are based on constructing a violator graph $G'(V, E')$ where there is a directed edge in E' from u to v iff (u, v) is a violating pair. This is quite easy to do for points in d-dimension space: for each point u just add an extra dimension with value $f(u)$, and slightly redefine the component-wise ordering so that the ordering on the last coordinate is reversed. Given the results for standard ordering of d-dimensional points, a violator graph can be constructed in $\Theta\left(n \log^{d} n\right)$ time. Apparently this was first used in [33].

3 L_0

L_0 is also known as the Hamming distance or 0-1 distance. It has only been studied much more recently than the others, appearing in [8, 24] (where it is called monotonic relabeling) and some related papers. The emphasis is on keeping values unchanged, with no consideration of how much they are changed if they need to be. Because of this, the values only need a linear ordering, with no notion of distance between them. However, sometimes the results are compared to L_1 regression with the assumption that consecutive labels are at unit distance, or that the labels are arbitrary real numbers. In the early papers the values at vertices are called labels, with the implication that there are far fewer labels than vertices. However, the same algorithms work even if there are n labels. People have noted that if there are only 2 labels then L_0 optimization is the same as L_1, if the L_1 regression is restricted to two values, typically 0 and 1. If the data on a linear order is 1, 0, then 0.5, 0.5 would be an optimal L_1 regression, but makes no sense for L_0. However, there is always an optimal L_1 regression where all of the regression values are values in the original data.

The algorithms for all but linear and tree orderings are based on *violator dags*: given data $(\mathbf{y}, \mathbf{w})$ on G, vertices v_i, v_j are a violating pair if $v_i \prec v_j$ but $y_i > y_j$. A vertex y is a *violator* if it is in some violator pair. The violator dag is $\widehat{G} = (\widehat{V}, \widehat{E})$, where $\widehat{V}$ are the violators and there is an edge from v_i to v_j iff they are a violating pair. A maximal anti-chain in this ordering corresponds to a maximum set of vertices where keeping the originally values at these vertices has no violators, i.e., minimizes the L_0 error. Once these vertices have been determined, finding suitable values for the other vertices can be done via topological sort, so previously one bottleneck was in finding this maximal set. The standard approach for finding it is via flow algorithms (see [8, 24]). Due to advances in flow algorithms, now for arbitrary dags the bottleneck is in creating the violator dag, which can be done via the transitive closure, taking $\Theta(\min\{nm, n^\omega\})$ time. However, for points in d-dimensional space it can be found far faster, so once again the bottleneck is the flow algorithm. See [34].

In general the result is not unique. E.g., for data 3, 2, 1 on a linear order, all of the vertices are violators, and any one of them can be chosen to be unchanged, forcing the other two to change. See comment 4 in Final

	time	reference
linear	$\Theta(n \log \ell)$	[8, 24]
d-dim, $d \geq 2$	$o(n^{1.5})$	[33]
arbitrary dag	$\Theta(\min\{nm, n^{\omega}\})$	[8, 24, 34]

Table 1: L_0, ℓ is the number of labels

	weighted		unweighted		other	
	time	reference	time	reference	time	reference
linear	$\Theta\left(n \log n\right)$	[1, 27]	$\Theta\left(n \log n\right)$	W		
tree	$\Theta\left(n \log n\right)$	[29]	$\Theta\left(n \log n\right)$	W		
2-dim grid	$\Theta\left(n \log n\right)$	[29]	$\Theta\left(n \log n\right)$	W		
2-dim arbitrary	$\Theta\left(n \log^2 n\right)$	[29]	$\Theta\left(n \log^2 n\right)$	W		
$d \geq 3$ grid	$\Theta\left(n^2 \log n\right)$	A	$o(n^{1.5})$	[34]		
$d \geq 3$ arbitrary	$\Theta\left(n^2 \log^d n\right)$	[29]	$o(n^{1.5})$	[34]		
arbitrary	$\Theta(nm + n^2 \log n)$	[2]	$\Theta(nm+n^2 \log n)$	W	$\Theta_{e,u}(n^{\omega})$ $\tilde{\Theta}_{e,\delta}(m^{1.5})$	[34] [17]

A: Result implied by that for arbitrary dag
W: Result implied by that for weighted data

Table 2: L_1

Remarks.

4 L_1

The L_1 metric is also known as Manhattan or taxi-cab distance, median regression, or least absolute deviation.

The L_1 regression value on a level set is a weighted median. If the data values in the set are $v_1 \ldots v_k$, with weights $w_1 \ldots w_k$, a weighted median is a value x such that $\sum\{w_i \mid v_i \leq x, 1 \leq i \leq k\} \geq W/2$, and $\sum\{w_i \mid v_i \geq x, 1 \leq i \leq k\} \geq W/2$, where $W = \sum_{1 \leq i \leq k} w_i$. In general weighted medians are not unique, e.g., for unweighted real-valued data 0, 1, 2, 5.3, any value in [1,2] is a weighted median. Weighted medians can always be chosen to be one of the data values, a fact most L_1 algorithms exploit. However, the result may not always be what is desired. For example, unweighted data 1, 0, 1 on a linear order would result in 0, 0, 1 or 1, 1, 1. These are useful if one wants to restrain regression values to the set of original values, while for some other purposes 0.5, 0.5, 1 would be considered better. See comment 4 in Final Comments.

The algorithm for L_1 isotonic regression on 2-dimensional grids given in [29] is based on recursively using dynamic programming, much like the earlier algorithm in [26] for L_2. For 2-dimensional points with arbitrary placement, [29] shows how to to use a balanced tree to simulate the 2-dimensional grid algorithms.

For a set P of arbitrary points in d-space, while it is embedded into dag G as discussed in Section 2, the time is a bit smaller than if one merely inserted the number of vertices and edges of G in the time analysis of the algorithm for arbitrary orderings. [29] shows that for L_1 regression the minimum cost flow approach in [2] uses a number of steps linear in the number of vertices with nonzero weight, which is n rather than the

	weighted		other	
	time	reference	time	reference
linear	$\Theta\left(n\right)$	PAV		
tree	$\Theta\left(n\log n\right)$	[20]		
2-dim grid	$\Theta\left(n^2\right)$	[26]	$\Theta_u(n\log n)$	[34]
2-dim arbitrary	$\Theta\left(n^2\log n\right)$	[29]	$\Theta_u(n\log^2 n)$	[34]
$d \geq 3$ grid	$\Theta\left(n^2\log n\right)$	A	$o_u(n^{1.5})$	[34]
$d \geq 3$ arbitrary	$\Theta\left(n^2\log^{2d-1} n\right)$	A	$o_u(n^{1.5})$	[34]
arbitrary	$\Theta\left(nm\log\frac{n^2}{m}\right)$	[11]	$\Theta_{e,u}(n^{\omega})$	[34]
			$\tilde{\Theta}_{e,\delta}(m^{1.5})$	[17]

A: Result implied by that for arbitrary dag

Table 3: L_2, no improvements known for unweighted data.

number of vertices in G, namely $\Theta(n\log^{d-1} n)$.

5 L_2

The L_2 metric is also known as squared error regression or Euclidean distance. Here the optimum value of a level set is just its weighted mean.

It was widely stated, by the author and others, that the fastest known algorithm for arbitrary orderings is due to Maxwell and Muckstadt [18], with a small correction by Spouge, Wan, and Wilbur [26]. However, this early work, published in 1985, gives an algorithm taking $\Theta(n^4)$ time, in contrast to the $\Theta(n^3)$ time of the later algorithm by Hochbaum and Queyranne [11]. Perhaps this oversight is due to the fact that the introduction in Hochbaum and Queyranne's paper defines the problem being solved as an integer approximation, and isotonic regression is only mentioned for the linear case (a result known for decades). However, the paper includes isotonic regression for arbitrary orderings and later they show that for L_2 one can obtain exact answers.

This illustrates an issue that has come up multiple times, namely that efficient algorithms for isotonic regression are not always discussed as such. For example, the Maxwell and Muckstadt paper does not contain the words "isotonic" nor "regression".

For L_2, the algorithms in the "other" column which require that the input is weights and values in the range [0,U] (i.e., all those where the time has a subscript u) can produce an exact result with additional logarithmic factors in the time. This is based on the fact that level sets have values that differ by at least $1/(n^2U^2)$, and hence approximating to within 1/4 of this identifies which level set each vertex will belong in. Once this is known, the exact value of the level set can be determined. See [29].

The algorithm for points on a 2-dimensional grid uses an iterative dynamic programming approach, and [29] shows how to simulate this to handle points at arbitrary positions in 2-space.

6 L_∞

The L_∞ metric is also known as minimax optimization, uniform metric, Chebyshev distance, supremum, or maximum absolute deviation.

To determine the regression value for level sets, suppose there are only two vertices v_1, v_2, with data $(\mathbf{y}, \mathbf{w})$, where $v_1 \prec v_2$ but $v_1 > v_2$. Then they need to form a level set, and the error is minimized by using

	weighted		unweighted	other	
	time	reference	time	time	reference
linear	$\Theta(n)$	[31]	$\Theta(n)$		
tree	$\Theta(n)$	[31]	$\Theta(n)$		
$d \geq 2$ grid	$\Theta(n)$	[31]	$\Theta(n)$		
$d \geq 2$ arbitrary	$\Theta\left(n \log^{d-1} n\right)$	[31]	$\Theta\left(n \log^{d-1} n\right)$		
arbitrary	$\Theta(m \log n)$	[15, 32]	$\Theta(m)$	$\Theta_e(m)$	[17]

A: Result implied by that for arbitrary dag

Table 4: L_∞, unweighted results widely known for a long time and vastly simpler

value $V(v_1, v_2) = (w_1y_1 + w_2y_2)/(w_1 + w_2)$, with regression error $e(v_1, v_2) = w_1w_2|y_1 - y_2|/(w_1 + w_2)$. These values can be obtained by the intersection of the planar line through $(y_2, 0)$ with slope w_2 and the line through $(y_1, 0)$ with slope $-w_1$. This geometric viewpoint is used by many of the algorithms for weighted L_∞ regression. For a level set with vertices $v_1, \ldots, v_k$, the regression value is $V(v_i, v_j)$, where $(v_i, v_j) = \arg\max\{e(v_i, v_j) : 1 \leq i < j \leq k\}$.

For unweighted data this simplifies significantly, with $V(v_i, v_j) = (y_i + y_j)/2$ and $e(v_i, v_j) = |v_i - v_j|/2$, and the regression value of a level set is just $(\max_{1 \leq i \leq k} y_i + \min_{1 \leq i \leq k} y_i)/2$. Using this, it is easy to show that the regression value at vertex v can be chosen to be the average of the maximum y value of all of its predecessors (including v) and the minimum y value of all of its successors (including v). This regression can easily be computed in $\Theta(m)$ time by topological sort. However, this can result in regressions that are not quite what one would want. For example, for unweighted data 1, -1, 0 on the line, the result would be 0, 0, 0.5, i.e., there is an unnecessary change in the last value. This is discussed in [28].

For arbitrary dags with weighted data, the algorithm in [32] is a modest improvement of the algorithm of Kaufman and Tamir [15], reducing the time from $\Theta(m \log n + n \log^2 n)$ to $\Theta(m \log n)$. This is faster for sparse dags where $m = o(n \log n)$, which is relevant for all of the other orderings considered, though the results in [31] make this moot as far as the tables are concerned. The approach in [15, 32] is based on parametric search, which is completely impractical, requiring a galactic algorithm. A very simple and fast algorithm, also in [32], has the same time bound, but in expected time with high probability, not worst case. A somewhat more complicated algorithm, taking $\Theta(m)$ expected time, appears in [17]. This is obviously the best possible in terms of expected time, and it is an open question if this time can be obtained in the worst case.

Many algorithms for weighted L_∞ regression use an indirect approach based on queries determining if there is an isotonic regressions with error $\leq \epsilon$, and, if so, produces one. A search is used to find the minimum such ϵ. Unfortunately the results, while optimal, are not always appealing since they result in many vertices having a large regression error. For the unweighted data 1, -1, 2, on a linear order, almost all algorithms using an indirect approach would produce 0, 0, 1, or 0, 0, 3, i.e., they behave even worse than using the approach based on topological sorts involving predecessors and successors. See the L_∞ comments in 4 in Final Remarks.

7 Final Remarks

1. Most of the entries have changed since I first posted tables in 2009. I put the tables together and posted them because it was suggested that it was too difficult to keep track of what the fastest algorithms were

at that time. I decided to work on some of the areas where improvement seemed possible or additional interest arose. Many other people did as well, and pointed out references to work I hadn't known. Results for L_0 were added in 2019.

2. Most of the algorithms in the tables are described in papers, but implementations are not provided though sometimes the algorithms are described in enough detail that they can be easily implemented. Online there are numerous implementations of the PAV algorithm for L_2 isotonic regression on a linear order. The R package listed in the entry for [27] contains ones for the L_1 metric on a linear order and for the L_1 and L_2 metrics on 2-dimensional orders. For some of the other orderings and metrics there are algorithms that are publicly available but slower than those listed above, though for practical applications they might be faster than a decent implementation of the ones in the table. Rather than go through and identify which are good and which aren't, in the bibliography I've indicated when implementations by the authors are available. I have almost certainly missed some implementations of the algorithms — feel free to contact me to improve the list.

3. For L_1 and L_0 there are always optimal regressions where regression values come from the original values, but for L_p, when $1 < p \leq \infty$, this is not always possible.

4. For L_p, when $1 < p < \infty$ there is always a unique optimal regression, but that is not true for L_0, L_1, nor L_∞.

   For L_1 the regression one might prefer is $\lim_{p \to 1} f_p$, where f_p is the L_p regression, and for L_∞ one might want $\lim_{p \to \infty} f_p$. For the former, Jackson [13] was apparently the first to determine the appropriate value of the level sets. He did this for unweighted data, but it is easily extended to the weighted case.

   The L_∞ version is introduced in [28], where it is called *strict* L_∞ regression. A $\Theta(\min\{nm, n^\omega\} + n^2 \log n)$ time algorithm appears there, and an algorithm taking $\Theta(nm)$ expected time appears in [17].

   For L_0 there doesn't appear to be a special regression defined via a limit. One possibility is to list the vertex errors of a regression in increasing order, viewing this as an n-element string. Listing these strings in lexical order, select the first one (there may be ties) as the preferred regression. In [33] this is called *strong* L_0 regression. This is similar to a property of the strict L_∞ regression, where there the strings are formed from the errors listed in decreasing error. The strict L_∞ regression is first in the lexical ordering of these strings. Strong L_0 regression maximizes the number of small errors, while strict L_∞ minimizes the number of large ones. However, no algorithm to compute strong L_0 regression has appeared, nor has anyone investigated to see if it has any special properties such as those that the strict L_∞ regression has. It would have been better if the author had chosen consistent naming and used "strict" or "strong" for both L_0 and L_∞.

5. The algorithms for weighted L_∞ regression in [31] for arbitrary points in d-dimensional space are unusual in that for fixed d the space required is $\Theta(n)$, i.e., the space does not grow with the number of edges in an explicit dag that gives the multidimensional ordering. All other algorithms referenced in this overview utilize an explicit dag no matter what the ordering.

References

[1] 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.

[2] Angelov, S, Harb, B, Kannan, S, and Wang, L-S (2006), “Weighted isotonic regression under the L_1 norm”, *Symposium on Discrete Algorithms* (SODA), pp. 783–791.

[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] Berman, P, Bhattacharyya, A, Grigorescu, E, Raskhodnikova, S, Woodruff, DP, and Yaroslavtsev, G (2014), “Steiner transitive-closure spanners of low-dimensional posets”, *J. Combinatorica* 34, pp. 255–277.

[5] 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.

[6] Caruana, R and Niculescu-Mizil, A (2006), “An empirical comparison of supervised learning algorithms”, *Proc. Int’l. Conf. Machine Learning*.

[7] Chakrabarti, D, Kumar, R and Punera, K (2007), “Page-level template detection via isotonic smoothing”, *Proc. 16th Int’l. World Wide Web Conf.*

[8] Feelders, A, Velikova, M, and Daniels, H (2006), “Two polynomial algorithms for relabeling non-monotone data”, Tech. Report UU-CS-2006-046, Dept. Info. Com. Sci., Utrecht Univ.

[9] Gamarnik, D (1998), “Efficient learning of monotone concepts via quadratic optimization”, *Proceedings of Computational Learning Theory (COLT)*, pp. 134–143.

[10] Gao, Y; Liu, Y; Peng, R (2021). “Fully dynamic electrical flows: sparse maxflow faster than Goldberg-Rao”, arXiv:2101.07233

[11] Hochbaum, DS and Queyranne, M (2003), “Minimizing a convex cost closure set”, *SIAM J. Discrete Math* 16, pp. 192–207.
The code is available at http://riot.ieor.berkeley.edu/Applications/Pseudoflow/parametric.html

[12] Kalai, A.T. and Sastry, F. (2009), “The Isotron algorithm: High-dimensional isotonic regression”, *COLT ’09*.

[13] Jackson, D (1921), “Note on the median of a set of numbers”, *Bull. Amer. Math. Soc.* 27, pp. 160–164.

[14] van de Kamp, R, Feelders, A, and Barile, N, (2009), “Isotonic classification trees”, *Advances in Intel. Data Analysis VIII*, LNCS 5772, pp. 405–416.

[15] Kaufman, Y and Tamir, A (1993), “Locating service centers with precedence constraints”, *Discrete Applied Math.* 47, pp. 251–261.

[16] Kotlowski, W and Slowinski, R (2013), “On nonparametric ordinal classification with monotonicity constraints”, *IEEE Trans. Knowledge and Data Engin.* 25, pp. 2576–2589.

[17] Kyng, R, Rao, A, and Sachdeva, S (2015), “Fast, provable algorithms for isotonic regression in all L_p-norms”, NIPS. Their algorithms are at https://github.com/danspielman/YINSlex

[18] Maxwell, WL and Muckstadt, JA (1985), “Establishing consistent and realistic reorder intervals in production-distribution systems”, *Operations Research* 33, pp. 1316–1341.

[19] Moon, T, Smola, A, Chang, Y and Zheng, Z (2010), “IntervalRank — isotonic regression with listwise and pairwise constraints”, *Proc. Web Search and Data Mining*, pp. 151–160.

[20] Pardalos, PM and Xue, G (1999), “Algorithms for a class of isotonic regression problems”, *Algorithmica* 23, pp. 211–222.

[21] Piljs, W and Potharst, R (2014), “Repairing non-monotone ordinal data sets by changing class labels”, Econometric Inst. Report EI 2014–29.

[22] Punera, K and Ghosh, J (2008), “Enhanced hierarchical classification via isotonic smoothing”, Int’l. Conf. World Wide Web.

[23] Rademaker, M, De Baets, R, and De Meyer, H (2009), “Loss optimal monotone relabeling of noisy multi-criteria data sets”, *Info. Sciences* 179, pp. 4089–4096.

[24] Rademaker, M, De Baets, B, and De Meyer, H (2012), “Optimal monotone relabelling of partially non-monotone ordinal data”, *Optimization Methods and Soft.* 27, 17–31.

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

[26] Spouge J, Wan H, and Wilbur WJ (2003), “Least squares isotonic regression in two dimensions”, *J. Optimization Theory and Applications* 117, pp. 585–605.

[27] Stout, QF (2008), “Unimodal regression via prefix isotonic regression”, *Computational Stat. and Data Analysis* 53, pp. 289–297. A preliminary version appeared in “Optimal algorithms for unimodal regression”, *Computing and Statistics* 32, 2000. Some of the algorithms are implemented in the R package UniIsoRegression.

[28] Stout, QF (2012), “Strict L_∞ isotonic regression”, *J. Optimization Theory and Applications* 152, pp. 121–135.

[29] Stout, QF (2013), “Isotonic regression via partitioning”, *Algorithmica* 66, pp. 93–112.

[30] Stout, QF (2015), “Isotonic regression for multiple independent variables”, *Algorithmica* 71, pp. 450–470.

[31] Stout, QF (2015), “L_∞ isotonic regression for linear, multidimensional, and tree orders”, arXiv 1507.02226.

[32] Stout, QF (2018), “Weighted L_∞ isotonic regression”, *J. Computer Sys. and Sci.* 91, pp. 69–81.
This is a major revision of the original version that was posted on the web in 2008. Some of the material in that paper was moved to [30].

[33] Stout, QF (2021), “L_0 isotonic regression with secondary objectives”, arXiv:2106.00279v2

[34] Stout, QF (2021), “L_p isotonic regression algorithms using an L_0 approach”, arXiv:2107.00251v2

[35] Velikova, M and Daniels, H (2008), *Monotone Prediction Models in Data Mining*, VDM Verlag.