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 $L_\infty$. The partial orderings considered are linear, tree, d-dimensional grids, points in d-dimensional space with component-wise ordering, and arbitrary orderings (posets). We consider only exact solutions (to within machine error), not approximations. Throughout, “fastest” means for the worst case in $O$-notation, not in any measurements of implementations nor in expected case. 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 $z = (z_1 \ldots z_n)$ 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 $\mathbb{R}$. In some contexts this is known as a monotonic function. By data $(y, w)$ 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 $\geq 0$. By unweighted data we mean $w_i = 1$ for all $i$.

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

$$\left(\sum_{i=1}^n w_i |y_i - z_i|^p\right)^{1/p} \quad 1 \leq p < \infty$$

$$\max_{i=1}^n w_i |y_i - z_i| \quad p = \infty$$

$$\sum_{i=1}^n w_i \cdot (y_i \neq z_i) \quad p = 0$$

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 $z$, if $z_i = z_k$ then $z_j$ has the same value. A set $V' \subset V$ is a level set of $z$ 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), tree, points in multidimensional space, 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 \( \geq 3 \). 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 small sample of which includes [6, 7, 9, 11, 13, 15, 18, 21, 22, 33]. The books [4, 24] contain numerous applications, though they are far out of date.

Approximation algorithms are not included in the tables. I’ve also omitted related topics such as unimodal regression, integer-valued regression, prefix isotonic regression, convex 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, and citations to the relevant references. All times are worst-case, and “best times” is in terms of O-notation, not any measurements of implementations. In many cases 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. 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.

2 Cross-cutting Techniques

There are some approaches that have been used for all of the metrics.

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 \( L_1 \) and \( L_2 \), where \( L_1 \) precedes \( L_2 \) but the regression value on \( L_1 \) is greater than that of \( L_2 \) (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.

Apparently the first paper that used PAV is by Ayer et al. [3]. For the \( L_2 \) metric it is trivial to implement in linear time, and similarly for the \( L_\infty \) metric with unweighted data, while for the others appropriate data structures are needed. Previously the fastest algorithm for weighted \( L_\infty \) used this approach, taking \( \Theta(n \log n) \) time, but now the fastest takes \( \Theta(n) \) time and is not based on PAV [30].

Unfortunately, while PAV can also be used for trees it does not apply to more general orderings. However, for trees adjacent violating subtree level sets cannot be paired in arbitrary order. This is discussed in [19].
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. Unfortunately, \(n\) points may require \(\Theta(n^2)\) edges to represent this ordering, 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 \(\Theta(n \log^{d-1} n)\) vertices and edges, and can be constructed in time linear in its size (\cite{29}). 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 \cite{31}, but was subsequently moved to \cite{29}. 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 \(L_{\infty}\), the algorithm in \cite{30} only uses \(G\) conceptually, simulating it via repeated sorting and taking only \(\Theta(n)\) space. It is not based on using the \(L_{\infty}\) algorithm for general dags.

A more symmetric version of \(G\), where all dimensions are treated the same as opposed to having one kept as a standard linear ordering, has \(\Theta(n \log^d n)\) vertices and edges. It too appears in \cite{29} and provides a Steiner 2-transitive-closure of the points \cite{5}.

3 \(L_0\)

\(L_0\) is also known as the Hamming distance. It has only been studied much more recently than the others, appearing in \cite{8, 23} (where it is called monotonic relabeling). 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, and \cite{32} considers extensions when 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. The papers note 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 the two values. E.g., 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 the regression values are values in the original data.

The algorithms for all but linear and tree orderings are based on violator pairs: given data $(y, w)$ on $G$, vertices $v_i, v_j$ are a violating pair if $v_i < v_j$ but $y_i > y_j$. A vertex $y$ is a violator if it is in some violator pair. The violator graph is $\hat{G} = (\hat{V}, \hat{E})$, where $\hat{V}$ are the violators and there is an edge from $v_i$ to $v_j$ if they are a violating pair. Vertices that are not violators will have their original value in any optimal regression, so the goal is to identify a maximal subset of violators $V' \subseteq \hat{V}$ where there is an optimal isotonic regression $g$ with $g(v_i) = y_i$ for all $v_i \in V'$. Determining which vertices are violators can be done via topological sort and reverse topological sort in $\Theta(m)$ time. Explicit construction using violating pairs is also used in the fastest algorithm for $L_1$ on general dags [2].

Let $\hat{n} = |\hat{V}|$ and $\hat{m} = |\hat{E}|$. The violator graph may have many more edges than the original graph, e.g., on a linear order, suppose the values are 4, 5, 6, 1, 2, 3. Then every vertex is a violator, and there is an edge from every vertex in the first half to every one in the second half, i.e., $\hat{n} = n$ and $\hat{m} = \Theta(m^2)$. In other circumstances $\hat{n}$ and $\hat{m}$ may be much smaller than $n$ and $m$. In the initial applications the number of different labels is quite small, so I've included $\hat{\ell}$, the number of labels in the violator graph, as one of the parameters in the analysis. The table shows the time to build the violator graph, and the time to find the regression using it. Explicit construction using violating pairs is also used in the fastest algorithm for $L_1$ on general dags [2].

Let $\hat{n} = |\hat{V}|$ and $\hat{m} = |\hat{E}|$. The violator graph may have many more edges than the original graph, e.g., on a linear order, suppose the values are 4, 5, 6, 1, 2, 3. Then every vertex is a violator, and there is an edge from every vertex in the first half to every one in the second half, i.e., $\hat{n} = n$ and $\hat{m} = \Theta(m^2)$. In other circumstances $\hat{n}$ and $\hat{m}$ may be much smaller than $n$ and $m$. In the initial applications the number of different labels is quite small, so I've included $\hat{\ell}$, the number of labels in the violator graph, as one of the parameters in the analysis. The table shows the time to build the violator graph, and the time to find the regression using it. Explicit construction using violating pairs is also used in the fastest algorithm for $L_1$ on general dags [2].

Let $\hat{n} = |\hat{V}|$ and $\hat{m} = |\hat{E}|$. The violator graph may have many more edges than the original graph, e.g., on a linear order, suppose the values are 4, 5, 6, 1, 2, 3. Then every vertex is a violator, and there is an edge from every vertex in the first half to every one in the second half, i.e., $\hat{n} = n$ and $\hat{m} = \Theta(m^2)$. In other circumstances $\hat{n}$ and $\hat{m}$ may be much smaller than $n$ and $m$. In the initial applications the number of different labels is quite small, so I've included $\hat{\ell}$, the number of labels in the violator graph, as one of the parameters in the analysis. The table shows the time to build the violator graph, and the time to find the regression using it. Explicit construction using violating pairs is also used in the fastest algorithm for $L_1$ on general dags [2].

### Table 1: Time to find an $L_0$-optimal relabeling of $G$ given a violator graph $\hat{G}$, and time to construct $\hat{G}$

<table>
<thead>
<tr>
<th>Order</th>
<th>Time to construct $\hat{G}$</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear</td>
<td>$\Theta(n + \hat{n} \log \hat{\ell})$</td>
<td>[8, 23]</td>
</tr>
<tr>
<td>Tree</td>
<td>$\Theta(n + \hat{n} \hat{\ell})$</td>
<td>[32]</td>
</tr>
<tr>
<td>$d$-dim, $d \geq 2$</td>
<td>$\Theta\left(n^2 \log^{d-1}(\hat{n}) \log \hat{\ell}\right)$</td>
<td>[32]</td>
</tr>
<tr>
<td></td>
<td>$\Theta\left(n^2 \log^{2(d-1)}(\hat{n}) \log^2 \hat{\ell}\right)$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\Theta\left(n \log^{d-1} n + \hat{n} \log^{d-1}(\hat{n}) \log \hat{\ell}\right)$</td>
<td></td>
</tr>
<tr>
<td>Arbitrary dag</td>
<td>$\Theta(\hat{m})$</td>
<td>[8, 23]</td>
</tr>
<tr>
<td></td>
<td>$\Theta(m + \hat{m})$</td>
<td></td>
</tr>
</tbody>
</table>

### A: Result implied by that for arbitrary dag

Unweighted and weighted data same time except for $d$-dim. See the comments below.
<table>
<thead>
<tr>
<th></th>
<th>weighted</th>
<th>unweighted</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>time</td>
<td>reference</td>
</tr>
<tr>
<td>linear</td>
<td>(\Theta(n \log n))</td>
<td>[1][26]</td>
</tr>
<tr>
<td>tree</td>
<td>(\Theta(n \log n))</td>
<td>[28]</td>
</tr>
<tr>
<td>2-dim grid</td>
<td>(\Theta(n \log n))</td>
<td>[28]</td>
</tr>
<tr>
<td>2-dim arbitrary</td>
<td>(\Theta(n \log^2 n))</td>
<td>[28]</td>
</tr>
<tr>
<td>(d \geq 3) grid</td>
<td>(\Theta(n^2 \log n))</td>
<td>A</td>
</tr>
<tr>
<td>(d \geq 3) arbitrary</td>
<td>(\Theta(n^2 \log^d n))</td>
<td>A</td>
</tr>
<tr>
<td>arbitrary</td>
<td>(\Theta(nm + n^2 \log n))</td>
<td>[2]</td>
</tr>
</tbody>
</table>

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

Table 2: \(L_1\)

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 \(\sum\{w_i | v_i \leq x, 1 \leq i \leq k\} \geq W/2\), and \(\sum\{w_i | v_i \geq x, 1 \leq i \leq k\} \geq W/2\), where \(W = \sum_{1 \leq i \leq k} w_i\). In general weighted medians are not unique, e.g., for unweighted data \(0, 1, 2, 5\), 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 [3] in Final Comments.

The algorithm for \(L_1\) isotonic regression on 2-dimensional grids given in [28] is based on recursively using dynamic programming, much like the earlier algorithm in [25] for \(L_2\). For 2-dimensional points with arbitrary placement, [28] 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. [28] 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 number of vertices in \(G\), namely \(\Theta(n \log^{d-1} n)\).
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.

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 [17], with a small correction by Spouge, Wan, and Wilbur [25]. 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 [10]. Perhaps this oversight is due to the fact that the introduction in their 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”.

The algorithm for points on a 2-dimensional grid uses an iterative dynamic programming approach, and [28] shows how to simulate this to handle points at arbitrary positions in 2-space. It seems highly likely that there are faster algorithms for 2-dimensional grids.

6 $L_\infty$

The $L_\infty$ 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 $(y, w)$, where $v_1 < v_2$ but $v_1 > v_2$. Then they need to form a level set, and the error is minimized by using value $V(v_1, v_2) = (w_1 y_1 + w_2 y_2) / (w_1 + w_2)$, with regression error $e(v_1, v_2) = w_1 w_2 |y_1 - y_2| / (w_1 + w_2)$. 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 $L_\infty$ 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 at any predecessor

<table>
<thead>
<tr>
<th>time</th>
<th>reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>linear</td>
<td>$\Theta(n)$</td>
</tr>
<tr>
<td>tree</td>
<td>$\Theta(n \log n)$</td>
</tr>
<tr>
<td>2-dim grid</td>
<td>$\Theta(n^2)$</td>
</tr>
<tr>
<td>2-dim arbitrary</td>
<td>$\Theta(n^2 \log n)$</td>
</tr>
<tr>
<td>$d \geq 3$ grid</td>
<td>$\Theta(n^2 \log n)$</td>
</tr>
<tr>
<td>$d \geq 3$ arbitrary</td>
<td>$\Theta(n^2 \log^{2d-1} n)$</td>
</tr>
<tr>
<td>arbitrary</td>
<td>$\Theta(n m \log \frac{n^2}{m})$</td>
</tr>
</tbody>
</table>

A: Result implied by that for arbitrary dag

Table 3: $L_2$, no improvements known for unweighted data.
(including \(v\)) and the minimum \(y\) value at any successor (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.

For arbitrary dags, the algorithm in \([31]\) is a modest improvement of the algorithm of Kaufman and Tamir \([14]\), reducing the time from \(\Theta(m \log n + n \log^2 n)\) to \(\Theta(m \log n)\). This is faster for sparse graphs where \(m = o(n \log n)\), which is relevant for all of the other orderings considered, though the results in \([30]\) make this moot as far as the tables are concerned. Unfortunately, the approach is based on parametric search, which is completely impractical. A very simple algorithm, also in \([31]\), 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 \([16]\).

Many algorithms for weighted \(L_\infty\) 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 \(\epsilon\). 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, 0 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 comment 3 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 algorithms were available. 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. 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.

3. 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 \(L_\infty\). For \(L_1\) one might want \(\lim_{p \to 1} f_p\), where \(f_p\) is the \(L_p\) regression, and for \(L_\infty\) might want \(\lim_{p \to \infty} f_p\). For the former, Jackson \([12]\) was apparently the first to determine the appropriate value.

<table>
<thead>
<tr>
<th></th>
<th>weighted</th>
<th>unweighted</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td>(\Theta(n))</td>
<td>(\Theta(n))</td>
</tr>
<tr>
<td>tree</td>
<td>(\Theta(n))</td>
<td>(\Theta(n))</td>
</tr>
<tr>
<td>2-dim grid</td>
<td>(\Theta(n \log n))</td>
<td>(\Theta(n \log n))</td>
</tr>
<tr>
<td>2-dim arbitrary</td>
<td>(\Theta(n \log^d n))</td>
<td>(\Theta(n \log^d n))</td>
</tr>
<tr>
<td>d (\geq 3) grid</td>
<td>(\Theta(n))</td>
<td>(\Theta(n))</td>
</tr>
<tr>
<td>d (\geq 3) arbitrary</td>
<td>(\Theta(n \log^d n))</td>
<td>(\Theta(n \log^d n))</td>
</tr>
<tr>
<td>arbitrary</td>
<td>(\Theta(m \log n))</td>
<td>(\Theta(m))</td>
</tr>
</tbody>
</table>

Table 4: \(L_\infty\), unweighted results widely known for a long time
of the level sets. He did this for unweighted data, but it is easy to extended to the weighted case. \( L_\infty \) is discussed in [27], where a \( \Theta(\min\{nm, n^{\omega}\} + n^2 \log n) \) time algorithm appears. \( \omega \) represents the smallest value so that matrix multiplication can be done in \( \Theta(n^{\omega}) \) time. While Strassen’s algorithm (with \( \omega = \log_2 7 \approx 2.81 \)) and some of its successors are practical, now algorithms achieving smaller values of \( \omega \) have extremely large constants and complexity and would never be used. An algorithm taking \( \Theta(nm) \) expected time appears in [16] requiring the reasonable assumption that the weights grow at most as a polynomial function of \( n \). \( L_0 \) is discussed in [32].

4. The differences between algorithms for weighted vs. unweighted vary widely across the metrics. For \( L_\infty \) the weighted case is significantly more complicated than the unweighted case for all of the orderings, even though the times are the same except for general dags. For \( L_2 \) there is essentially no difference, and for \( L_1 \) the differences don’t affect the algorithms for linear, tree, and 2-dimensional orderings but do for the others. For \( L_0 \), for arbitrary dags the simple Ford-Fulkerson flow algorithm can be used for the unweighted case, but a far more complex flow algorithm is needed for the general case.

5. The algorithms for weighted \( L_\infty \) regression in [30] 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.

6. While the tables do not include expected time algorithms, algorithms with fast expected times for a large collection of isotonic regression problems appear in [16]. Their results are exact for \( L_1 \) and \( L_\infty \), but not for \( L_2 \), and \( L_0 \) is not discussed.

References


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 [29].
