Efficient Learning of Generalized Linear and Single Index Models with Isotonic Regression

Sham M. Kakade
Microsoft Research and Wharton, U Penn
skakade@microsoft.com

Adam Tauman Kalai
Microsoft Research
adum@microsoft.com

Varun Kanade
SEAS, Harvard University
vkanade@fas.harvard.edu

Ohad Shamir
Microsoft Research
ohadsh@microsoft.com

Abstract

Generalized Linear Models (GLMs) and Single Index Models (SIMs) provide powerful generalizations of linear regression, where the target variable is assumed to be a (possibly unknown) 1-dimensional function of a linear predictor. In general, these problems entail non-convex estimation procedures, and, in practice, iterative local search heuristics are often used. Kalai and Sastry (2009) provided the first provably efficient method, the *Isotron* algorithm, for learning SIMs and GLMs, under the assumption that the data is in fact generated under a GLM and under certain monotonicity and Lipschitz (bounded slope) constraints. The Isotron algorithm interleaves steps of perceptron-like updates with isotonic regression (fitting a one-dimensional non-decreasing function). However, to obtain provable performance, the method requires a fresh sample every iteration. In this paper, we provide algorithms for learning GLMs and SIMs, which are both computationally and statistically efficient. We modify the isotonic regression step in Isotron to fit a Lipschitz monotonic function, and also provide an efficient $O(n \log(n))$ algorithm for this step, improving upon the previous $O(n^2)$ algorithm. We provide a brief empirical study, demonstrating the feasibility of our algorithms in practice.

1 Introduction

The oft used linear regression paradigm models a dependent variable Y as a linear function of a vector-valued independent variable X. Namely, for some vector w, we assume that $\mathbb{E}[Y|X] = w \cdot X$. Generalized linear models (GLMs) provide a flexible extension of linear regression, by assuming that the dependent variable Y is of the form, $\mathbb{E}[Y|X] = u(w \cdot X)$; u is referred to as the inverse link function or transfer function (see [1] for a review). Generalized linear models include commonly used regression techniques such as logistic regression, where $u(z) = 1/(1 + e^{-z})$ is the logistic function. The class of perceptrons also falls in this category, where u is a simple piecewise linear function of the form _/‾, with the slope of the middle piece being the inverse of the margin.

In the case of linear regression, the least-squares method is an highly efficient procedure for parameter estimation. Unfortunately, in the case of GLMs, even in the setting when u is known, the problem of fitting a model that minimizes squared error is typically not convex. We are not aware of any classical estimation procedure for GLMs which is both *computationally* and *statistically* efficient, and with provable guarantees. The standard procedure is iteratively reweighted least squares, based on Newton-Raphson (see [1]).

The case when *both* u and w are unknown (sometimes referred to as Single Index Models (SIMs)), involves the more challenging (and practically relevant) question of jointly estimating u and w,

where u may come from a large non-parametric family such as all monotonic functions. There are two questions here: 1) What statistical rate is achievable for simultaneous estimation of u and w? 2) Is there a *computationally efficient* algorithm for this joint estimation? With regards to the former, under mild Lipschitz-continuity restrictions on u, it is possible to characterize the effectiveness of an (appropriately constrained) joint empirical risk minimization procedure. This suggests that, from a purely statistical viewpoint, it may be worthwhile to attempt jointly optimizing u and w on empirical data.

However, the issue of *computationally efficiently* estimating both u and w (and still achieving a good statistical rate) is more delicate, and is the focus of this work. We note that this is not a trivial problem: in general, the joint estimation problem is highly non-convex, and despite a significant body of literature on the problem, existing methods are usually based on heuristics, which are not guaranteed to converge to a global optimum (see for instance [2, 3, 4, 5, 6]).

The Isotron algorithm of Kalai and Sastry [7] provides the first provably efficient method for learning GLMs and SIMs, under the common assumption that u is monotonic and Lipschitz, and assuming that the data corresponds to the model.[1] The sample and computational complexity of this algorithm is polynomial, and the sample complexity does not explicitly depend on the dimension. The algorithm is a variant of the "gradient-like" perceptron algorithm, where apart from the perceptron-like updates, an isotonic regression procedure is performed on the linear predictions using the Pool Adjacent Violators (PAV) algorithm, on every iteration.

While the Isotron algorithm is appealing due to its ease of implementation (it has no parameters other than the number of iterations to run) and theoretical guarantees (it works for any u, w), there is one principal drawback. It is a batch algorithm, but the analysis given requires the algorithm to be run on *fresh samples* each batch. In fact, as we show in experiments, this is not just an artifact of the analysis – if the algorithm loops over the same data in each update step, it really does overfit in very high dimensions (such as when the number of dimensions exceeds the number of examples).

Our Contributions: We show that the overfitting problem in Isotron stems from the fact that although it uses a slope (Lipschitz) condition as an assumption in the analysis, it does not constrain the output hypothesis to be of this form. To address this issue, we introduce the SLISOTRON algorithm (pronounced slice-o-tron, combining slope and Isotron). The algorithm replaces the isotonic regression step of the Isotron by finding the best non-decreasing function *with a bounded Lipschitz parameter* - this constraint plays here a similar role as the margin in classification algorithms. We also note SLISOTRON (like Isotron) has a significant advantage over standard regression techniques, since it does not require knowing the transfer function. Our two main contributions are:

1. We show that the new algorithm, like Isotron, has theoretical guarantees, and significant new analysis is required for this step.

2. We provide an efficient $O(n \log(n))$ time algorithm for finding the best non-decreasing function with a bounded Lipschitz parameter, improving on the previous $O(n^2)$ algorithm [10]. This makes SLISOTRON practical even on large datasets.

We begin with a simple perceptron-like algorithm for fitting GLMs, with a *known* transfer function u which is monotone and Lipschitz. Somewhat surprisingly, prior to this work (and Isotron [7]) a computationally efficient procedure that guarantees to learn GLMs was not known. Section 4 contains the more challenging SLISOTRON algorithm and also the efficient $O(n \log(n))$ algorithm for Lipschitz isotonic regression. We conclude with a brief empirical analysis.

2 Setting

We assume the data (x, y) are sampled i.i.d. from a distribution supported on $\mathbb{B}_d \times [0, 1]$, where $\mathbb{B}_d = \{x \in \mathbb{R}^d \ : \ \|x\| \leq 1\}$ is the unit ball in d-dimensional Euclidean space. Our algorithms and

[1] In the more challenging agnostic setting, the data is not required to be distributed according to a true u and w, but it is required to find the best u, w which minimize the empirical squared error. Similar to observations of Kalai et al. [8], it is straightforward to show that this problem is likely to be computationally intractable in the agnostic setting. In particular, it is at least as hard as the problem of "learning parity with noise," whose hardness has been used as the basis for designing multiple cryptographic systems. Shalev-Shwartz et al. [9] present a kernel-based algorithm for learning certain types of GLMs and SIMs in the agnostic setting. However, their worst-case guarantees are exponential in the norm of w (or equivalently the Lipschitz parameter).

Algorithm 1 GLM-TRON

Input: data $\langle (x_i, y_i) \rangle_{i=1}^m \in \mathbb{R}^d \times [0,1]$, $u : \mathbb{R} \to [0,1]$, held-out data $\langle (x_{m+j}, y_{m+j}) \rangle_{j=1}^s$

$w^1 := 0$;

for $t = 1, 2, \ldots$ **do**

$h^t(x) := u(w^t \cdot x)$;

$w^{t+1} := w^t + \frac{1}{m} \sum_{i=1}^m (y_i - u(w^t \cdot x_i)) x_i$;

end for

Output: $\arg\min_{h^t} \sum_{j=1}^s (h^t(x_{m+j}) - y_{m+j})^2$

analysis also apply to the case where $\mathbb{B}_d$ is the unit ball in some high (or infinite)-dimensional kernel feature space. We assume there is a fixed vector w, such that $\|w\| \leq W$, and a non-decreasing 1-Lipschitz function $u : \mathbb{R} \to [0,1]$, such that $\mathbb{E}[y|x] = u(w \cdot x)$ for all x. The restriction that u is 1-Lipschitz is without loss of generality, since the norm of w is arbitrary (an equivalent restriction is that $\|w\| = 1$ and that u is W-Lipschitz for an arbitrary W).

Our focus is on approximating the regression function well, as measured by the squared loss. For a real valued function $h : \mathbb{B}_d \to [0,1]$, define

$$\mathrm{err}(h) = \mathbb{E}_{(x,y)} \left[(h(x) - y)^2 \right]$$
$$\varepsilon(h) = \mathrm{err}(h) - \mathrm{err}(E[y|x]) \ = \mathbb{E}_{(x,y)} \left[(h(x) - u(w \cdot x))^2 \right]$$

$\mathrm{err}(h)$ measures the error of h, and $\varepsilon(h)$ measures the excess error of h compared to the Bayes-optimal predictor $x \mapsto u(w \cdot x)$. Our goal is to find h such that $\varepsilon(h)$ (equivalently, $\mathrm{err}(h)$) is as small as possible.

In addition, we define the empirical counterparts $\widehat{\mathrm{err}}(h)$, $\hat{\varepsilon}(h)$, based on a sample $(x_1, y_1), \ldots, (x_m, y_m)$, to be

$$\widehat{\mathrm{err}}(h) \ = \frac{1}{m} \sum_{i=1}^m (h(x_i) - y_i)^2; \qquad \hat{\varepsilon}(h) \ = \frac{1}{m} \sum_{i=1}^m (h(x_i) - u(w \cdot x_i))^2.$$

Note that $\hat{\varepsilon}$ is the standard *fixed design error* (as this error conditions on the observed x's).

Our algorithms work by iteratively constructing hypotheses h^t of the form $h^t(x) = u^t(w^t \cdot x)$, where u^t is a non-decreasing, 1-Lipschitz function, and w^t is a linear predictor. The algorithmic analysis provides conditions under which $\hat{\varepsilon}(h^t)$ is small, and using statistical arguments, one can guarantee that $\varepsilon(h^t)$ would be small as well.

3 The GLM-TRON algorithm

We begin with the simpler case, where the transfer function u is assumed to be known (e.g. a sigmoid), and the problem is estimating w properly. We present a simple, parameter-free, perceptron-like algorithm, GLM-TRON (Alg. 1), which efficiently finds a close-to-optimal predictor. We note that the algorithm works for arbitrary non-decreasing, Lipschitz functions u, and thus covers most generalized linear models. We refer the reader to the pseudo-code in Algorithm 1 for some of the notation used in this section.

To analyze the performance of the algorithm, we show that if we run the algorithm for sufficiently many iterations, one of the predictors h^t obtained must be nearly-optimal, compared to the Bayes-optimal predictor.

Theorem 1. *Suppose $(x_1, y_1), \ldots, (x_m, y_m)$ are drawn independently from a distribution supported on $\mathbb{B}_d \times [0,1]$, such that $\mathbb{E}[y|x] = u(w \cdot x)$, where $\|w\| \leq W$, and $u : \mathbb{R} \to [0,1]$ is a known non-decreasing 1-Lipschitz function. Then for any $\delta \in (0,1)$, the following holds with probability at least $1 - \delta$: there exists some iteration $t < O(W\sqrt{m/\log(1/\delta)})$ of* GLM-TRON *such that the hypothesis $h^t(x) = u(w^t \cdot x)$ satisfies*

$$\max\{\hat{\varepsilon}(h^t), \varepsilon(h^t)\} \leq O\left(\sqrt{\frac{W^2 \log(m/\delta)}{m}} \right).$$

Algorithm 2 SLISOTRON

Input: data $\langle (x_i, y_i) \rangle_{i=1}^m \in \mathbb{R}^d \times [0,1]$, held-out data $\langle (x_{m+j}, y_{m+j}) \rangle_{j=1}^s$
$w^1 := 0$;
for $t = 1, 2, \ldots$ **do**
  $u^t := \mathsf{LIR}\left((w^t \cdot x_1, y_1), \ldots, (w^t \cdot x_m, y_m)\right)$ // Fit 1-d function along w^t
  $w^{t+1} := w^t + \frac{1}{m} \sum_{i=1}^m (y_i - u^t(w^t \cdot x_i)) x_i$
end for
Output: $\arg\min_{h^t} \sum_{j=1}^s (h^t(x_{m+j}) - y_{m+j})^2$

In particular, the theorem implies that some h^t has small enough $\varepsilon(h^t)$. Since $\varepsilon(h^t)$ equals $\mathrm{err}(h^t)$ up to a constant, we can easily find an appropriate h^t by picking the one that has least $\widehat{\mathrm{err}}(h^t)$ on a held-out set.

The main idea of the proof is showing that at each iteration, if $\hat{\varepsilon}(h^t)$ is not small, then the squared distance $\left\|w^{t+1} - w\right\|^2$ is substantially smaller than $\|w^t - w\|^2$. Since the squared distance is bounded below by 0, and $\left\|w^0 - w\right\|^2 \leq W^2$, there is an iteration (arrived at within reasonable time) such that the hypothesis h^t at that iteration is highly accurate. Although the algorithm minimizes empirical squared error, we can bound the true error using a uniform convergence argument. The complete proofs are provided in the full version of the paper ([11] Appendix A).

4 The SLISOTRON algorithm

In this section, we present SLISOTRON (Alg. 2), which is applicable to the harder setting where the transfer function u is unknown, except for it being non-decreasing and 1-Lipschitz. SLISOTRON does have one parameter, the Lipschitz constant; however, in theory we show that this can simply be set to 1. The main difference between SLISOTRON and GLM-TRON is that now the transfer function must also be learned, and the algorithm keeps track of a transfer function u^t which changes from iteration to iteration. The algorithm is inspired by the Isotron algorithm [7], with the main difference being that at each iteration, instead of applying the PAV procedure to fit an arbitrary monotonic function along the direction w^t, we use a different procedure, (Lipschitz Isotonic Regression) LIR, to fit a *Lipschitz* monotonic function, u^t, along w^t. This key difference allows for an analysis that does not require a fresh sample each iteration. We also provide an efficient $O(m \log(m))$ time algorithm for LIR (see Section 4.1), making SLISOTRON an extremely efficient algorithm.

We now turn to the formal theorem about our algorithm. The formal guarantees parallel those of the GLM-TRON algorithm. However, the rates achieved are somewhat worse, due to the additional difficulty of simultaneously estimating both u and w.

Theorem 2. *Suppose $(x_1, y_1), \ldots, (x_m, y_m)$ are drawn independently from a distribution supported on $\mathbb{B}_d \times [0,1]$, such that $\mathbb{E}[y|x] = u(w \cdot x)$, where $\|w\| \leq W$, and $u : \mathbb{R} \to [0,1]$ is an unknown non-decreasing 1-Lipschitz function. Then the following two bounds hold:*

1. *(Dimension-dependent) With probability at least $1 - \delta$, there exists some iteration $t < O\left(\left(\frac{Wm}{d \log(Wm/\delta)}\right)^{1/3}\right)$ of* SLISOTRON *such that*

$$\max\{\hat{\varepsilon}(h^t), \varepsilon(h^t)\} \leq O\left(\left(\frac{dW^2 \log(Wm/\delta)}{m}\right)^{1/3}\right).$$

2. *(Dimension-independent) With probability at least $1 - \delta$, there exists some iteration $t < O\left(\left(\frac{Wm}{\log(m/\delta)}\right)^{1/4}\right)$ of* SLISOTRON *such that*

$$\max\{\hat{\varepsilon}(h^t), \varepsilon(h^t)\} \leq O\left(\left(\frac{W^2 \log(m/\delta)}{m}\right)^{1/4}\right)$$

As in the case of Thm. 1, one can easily find h^t which satisfies the theorem's conditions, by running the SLISOTRON algorithm for sufficiently many iterations, and choosing the hypothesis h^t which minimizes $\widehat{\mathrm{err}}(h^t)$ on a held-out set. The algorithm minimizes empirical error and generalization bounds are obtained using a uniform convergence argument. The proofs are somewhat involved and appear in the full paper ([11] Appendix B).

4.1 Lipschitz isotonic regression

The SLISOTRON algorithm (Alg. 2) performs Lipschitz Isotonic Regression (LIR) at each iteration. The goal is to find the best fit (least squared error) non-decreasing 1-Lipschitz function that fits the data in one dimension. Let $(z_1, y_1), \ldots (z_m, y_m)$ be such that $z_i \in \mathbb{R}$, $y_i \in [0, 1]$ and $z_1 \leq z_2 \leq \cdots \leq z_m$. The *Lipschitz Isotonic Regression* (LIR) problem is defined as the following quadratic program:

$$\text{Minimize w.r.t } \hat{y}_i \ : \frac{1}{2}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2 \tag{1}$$

subject to:

$$\hat{y}_i \leq \hat{y}_{i+1} \qquad 1 \leq i \leq m-1 \text{ (Monotonicity)} \tag{2}$$

$$\hat{y}_{i+1} - \hat{y}_i \leq (z_{i+1} - z_i) \qquad 1 \leq i \leq m-1 \text{ (Lipschitz)} \tag{3}$$

Once the values $\hat{y}_i$ are obtained at the data points, the actual function can be constructed by interpolating linearly between the data points. Prior to this work, the best known algorithm for this problem wass due to Yeganova and Wilbur [10] and required $O(m^2)$ time for m points. In this work, we present an algorithm that performs the task in $O(m \log(m))$ time. The actual algorithm is fairly complex and relies on designing a clever data structure. We provide a high-level view here; the details are provided in the full version ([11] Appendix D).

Algorithm Sketch: We define functions $G_i(\cdot)$, where $G_i(s)$ is the minimum squared loss that can be attained if $\hat{y}_i$ is fixed to be s, and $\hat{y}_{i+1}, \ldots \hat{y}_m$ are then chosen to be the best fit 1-Lipschitz non-decreasing function to the points $(z_i, y_i), \ldots, (z_m, y_m)$. Formally, for $i = 1, \ldots, m$, define the functions,

$$G_i(s) = \min_{\hat{y}_{i+1}, \ldots, \hat{y}_m} \frac{1}{2}(s - y_i)^2 + \frac{1}{2}\sum_{j=i+1}^{m}(\hat{y}_j - y_j)^2 \tag{4}$$

subject to the constraints (where $s = \hat{y}_i$),

$$\hat{y}_j \leq \hat{y}_{j+1} \qquad i \leq j \leq m-1 \text{ (Monotonic)}$$

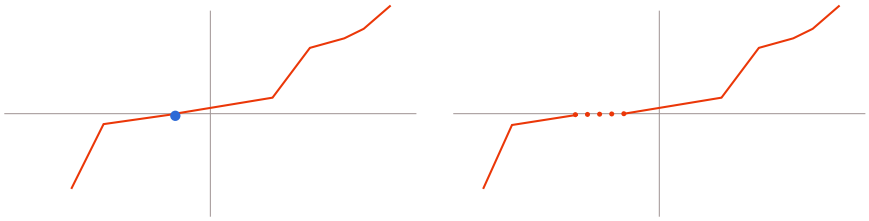
$$\hat{y}_{j+1} - \hat{y}_j \leq z_{j+1} - z_j \qquad i \leq j \leq m-1 \text{ (Lipschitz)}$$

Furthermore, define: $s_i^* = \min_s G_i(s)$. The functions G_i are piecewise quadratic, differentiable everywhere and strictly convex, a fact we prove in full paper [11]. Thus, G_i is minimized at s_i^* and it is strictly increasing on both sides of s_i^*. Note that $G_m(s) = (1/2)(s - y_m)^2$ and hence is piecewise quadratic, differentiable everywhere and strictly convex. Let $\delta_i = z_{i+1} - z_i$. The remaining G_i obey the following recursive relation.

$$G_{i-1}(s) = \frac{1}{2}(s - y_{i-1})^2 + \begin{cases} G_i(s + \delta_{i-1}) & \text{If } s \leq s_i^* - \delta_{i-1} \\ G_i(s_i^*) & \text{If } s_i^* - \delta_{i-1} < s \leq s_i^* \\ G_i(s) & \text{If } s_i^* < s \end{cases} \tag{5}$$

As intuition for the above relation, note that $G_{i-1}(s)$ is obtained fixing $\hat{y}_{i-1} = s$ and then by choosing $\hat{y}_i$ as close to s_i^* (since G_i is strictly increasing on both sides of s_i^*) as possible without violating either the monotonicity or Lipschitz constraints.

The above argument can be immediately translated into an algorithm, if the values s_i^* are known. Since s_1^* minimizes $G_1(s)$, which is the same as the objective of (1), start with $\hat{y}_1 = s_1^*$, and then successively chose values for $\hat{y}_i$ to be as close to s_i^* as possible without violating the Lipschitz or monotonicity constraints. This will produce an assignment for $\hat{y}_i$ which achieves loss equal to $G_1(s_1^*)$ and hence is optimal.



(a) (b)

Figure 1: (a) Finding the zero of G'_i. (b) Update step to transform representation of G'_i to G'_{i-1}

The harder part of the algorithm is finding the values s^*_i. Notice that G'_i are all piecewise linear, continuous and strictly increasing, and obey a similar recursive relation ($G'_m(s) = s - y_m$):

$$G'_{i-1}(s) = (s - y_{i-1}) + \begin{cases} G'_i(s + \delta_{i-1}) & \text{If } s \leq s^*_i - \delta_{i-1} \\ 0 & \text{If } s^*_i - \delta_{i-1} < s \leq s^*_i \\ G'_i(s) & \text{If } s^*_i < s \end{cases} \tag{6}$$

The algorithm then finds s^*_i by finding zeros of G'_i. Starting from m, $G'_m = s - y_m$, and $s^*_m = y_m$. We design a special data structure, called *notable red-black trees*, for representing piecewise linear, continuous, strictly increasing functions. We initialize such a tree `T` to represent $G'_m(s) = s - y_m$. Assuming that at some time it represents G'_i, we need to support two operations:

1. Find the zero of G'_i to get s^*_i. Such an operation can be done efficiently $O(\log(m))$ time using a tree-like structure (Fig. 1 (a)).

2. Update `T` to represent G'_{i-1}. This operation is more complicated, but using the relation (6), we do the following: Split the interval containing s'_i. Move the left half of the piecewise linear function G'_i by δ_{i-1} (Fig. 1(b)), adding the constant zero function in between. Finally, we add the linear function $s - y_{i-1}$ to every interval, to get G'_{i-1}, which is again piecewise linear, continuous and strictly increasing.

To perform the operations in step (2) above, we cannot naïvely apply the transformations, `shift-by`(δ_{i-1}) and `add`$(s - y_{i-1})$ to every node in the tree, as it may take $O(m)$ operations. Instead, we simply leave a note (hence the name *notable* red-black trees) that such a transformation should be applied before the function is evaluated at that node or at any of its descendants. To prevent a large number of such notes accumulating at any given node we show that these notes satisfy certain commutative and additive relations, thus requiring us to keep track of no more than 2 notes at any given node. This lazy evaluation of notes allows us to perform all of the above operations in $O(\log(m))$ time. The details of the construction are provided in the full paper ([11] Appendix D).

5 Experiments

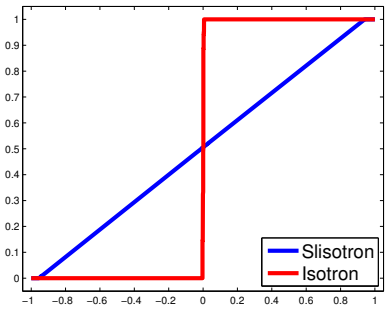
In this section, we present an empirical study of the SLISOTRON and GLM-TRON algorithms. We perform two evaluations using synthetic data. The first one compares SLISOTRON and Isotron [7] and illustrates the importance of imposing a Lipschitz constraint. The second one demonstrates the advantage of using SLISOTRON over standard regression techniques, in the sense that SLISOTRON can learn any monotonic Lipschitz function.

We also report results of an evaluation of SLISOTRON, GLM-TRON and several competing approaches on 5 UCI[12] datasets.

All errors are reported in terms of average root mean squared error (RMSE) using 10 fold cross validation along with the standard deviation.

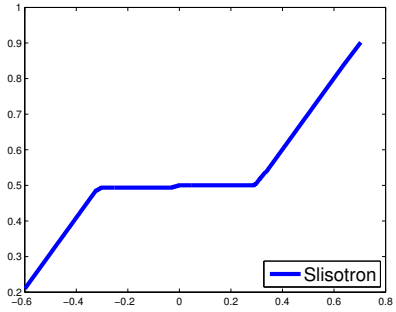
5.1 Synthetic Experiments

Although, the theoretical guarantees for Isotron are under the assumption that we get a fresh sample each round, one may still attempt to run Isotron on the same sample each iteration and evaluate the



SLISOTRON	Isotron	Δ
0.289 ± 0.014	0.334 ± 0.026	0.045 ± 0.018

(a) Synthetic Experiment 1



SLISOTRON	Logistic	Δ
0.058 ± 0.003	0.073 ± 0.006	0.015 ± 0.004

(b) Synthetic Experiment 2

Figure 2: (a) The figure shows the transfer functions as predicted by SLISOTRON and Isotron. The table shows the average RMSE using 10 fold cross validation. The Δ column shows the average difference between the RMSE values of the two algorithms across the folds. (b) The figure shows the transfer function as predicted by SLISOTRON. Table shows the average RMSE using 10 fold cross validation for SLISOTRON and Logistic Regression. The Δ column shows the average difference between the RMSE values of the two algorithms across folds.

empirical performance. Then, the main difference between SLISOTRON and Isotron is that while SLISOTRON fits the best *Lipschitz* monotonic function using LIR each iteration, Isotron merely finds the best monotonic fit using PAV. This difference is analogous to finding a large margin classifier vs. just a consistent one. We believe this difference will be particularly relevant when the data is sparse and lies in a high dimensional space.

Our first synthetic dataset is the following: The dataset is of size $m = 1500$ in $d = 500$ dimensions. The first co-ordinate of each point is chosen uniformly at random from $\{-1, 0, 1\}$. The remaining co-ordinates are all 0, except that for each data point one of the remaining co-ordinates is randomly set to 1. The true direction is $w = (1, 0, \ldots, 0)$ and the transfer function is $u(z) = (1 + z)/2$. Both SLISOTRON and Isotron put weight on the first co-ordinate (the true direction). However, Isotron overfits the data using the remaining (irrelevant) co-ordinates, which SLISOTRON is prevented from doing because of the Lipschitz constraint. Figure 2(a) shows the transfer functions as predicted by the two algorithms, and the table below the plot shows the average RMSE using 10 fold cross validation. The Δ column shows the average difference between the RMSE values of the two algorithms across the folds.

A principle advantage of SLISOTRON over standard regression techniques is that it is not necessary to know the transfer function in advance. The second synthetic experiment is designed as a sanity check to verify this claim. The dataset is of size $m = 1000$ in $d = 4$ dimensions. We chose a random direction as the "true" w and used a piecewise linear function as the "true" u. We then added random noise $(\sigma = 0.1)$ to the y values. We compared SLISOTRON to Logistic Regression on this dataset. SLISOTRON correctly recovers the true function (up to some scaling). Fig. 2(b) shows the actual transfer function as predicted by SLISOTRON, which is essentially the function we used. The table below the figure shows the performance comparison between SLISOTRON and logistic regression.

5.2 Real World Datasets

We now turn to describe the results of experiments performed on the following 5 UCI datasets: `communities`, `concrete`, `housing`, `parkinsons`, and `wine-quality`. We compared the performance of SLISOTRON (Sl-Iso) and GLM-TRON with logistic transfer function (GLM-t) against Isotron (Iso), as well as standard logistic regression (Log-R), linear regression (Lin-R) and a simple heuristic algorithm (SIM) for single index models, along the lines of standard iterative maximum-likelihood procedures for these types of problems (e.g., [13]). The SIM algorithm works by iteratively fixing the direction w and finding the best transfer function u, and then fixing u and

optimizing w via gradient descent. For each of the algorithms we performed 10-fold cross validation, using 1 fold each time as the test set, and we report averaged results across the folds.

Table 1 shows average RMSE values of all the algorithms across 10 folds. The first column shows the mean Y value (with standard deviation) of the dataset for comparison. Table 2 shows the average difference between RMSE values of SLISOTRON and the other algorithms across the folds. Negative values indicate that the algorithm performed better than SLISOTRON. The results suggest that the performance of SLISOTRON (and even Isotron) is comparable to other regression techniques and in many cases also slightly better. The performance of GLM-TRON is similar to standard implementations of logistic regression on these datasets. This suggests that these algorithms should work well in practice, while providing non-trivial theoretical guarantees.

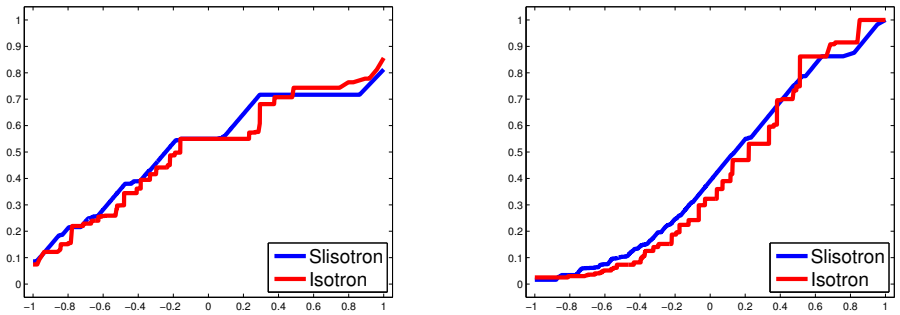
It is also illustrative to see how the transfer functions found by SLISOTRON and Isotron compare. In Figure 3, we plot the transfer functions for `concrete` and `communities`. We see that the fits found by SLISOTRON tend to be smoother because of the Lipschitz constraint. We also observe that `concrete` is the only dataset where SLISOTRON performs noticeably better than logistic regression, and the transfer function is indeed somewhat far from the logistic function.

Table 1: Average RMSE values using 10 fold cross validation. The $\bar{Y}$ column shows the mean Y value and standard deviation.

dataset	$\bar{Y}$	Sl-Iso	GLM-t	Iso	Lin-R	Log-R	SIM
communities	0.24 ± 0.23	0.13 ± 0.01	0.14 ± 0.01	0.14 ± 0.01	0.14 ± 0.01	0.14 ± 0.01	0.14 ± 0.01
concrete	35.8 ± 16.7	9.9 ± 0.9	10.5 ± 1.0	9.9 ± 0.8	10.4 ± 1.1	10.4 ± 1.0	9.9 ± 0.9
housing	22.5 ± 9.2	4.65 ± 1.00	4.85 ± 0.95	4.68 ± 0.98	4.81 ± 0.99	4.70 ± 0.98	4.63 ± 0.78
parkinsons	29 ± 10.7	10.1 ± 0.2	10.3 ± 0.2	10.1 ± 0.2	10.2 ± 0.2	10.2 ± 0.2	10.3 ± 0.2
winequality	5.9 ± 0.9	0.78 ± 0.04	0.79 ± 0.04	0.78 ± 0.04	0.75 ± 0.04	0.75 ± 0.04	0.78 ± 0.03

Table 2: Performance comparison of SLISOTRON with the other algorithms. The values reported are the average difference between RMSE values of the algorithm and SLISOTRON across the folds. Negative values indicate better performance than SLISOTRON.

dataset	GLM-t	Iso	Lin-R	Log-R	SIM
communities	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
concrete	0.56 ± 0.35	0.04 ± 0.17	0.52 ± 0.35	0.55 ± 0.32	-0.03 ± 0.26
housing	0.20 ± 0.48	0.03 ± 0.55	0.16 ± 0.49	0.05 ± 0.43	-0.02 ± 0.53
parkinsons	0.19 ± 0.09	0.01 ± 0.03	0.11 ± 0.07	0.09 ± 0.07	0.21 ± 0.20
winequality	0.01 ± 0.01	0.00 ± 0.00	-0.03 ± 0.02	-0.03 ± 0.02	0.01 ± 0.01



(a) `concrete` (b) `communities`

Figure 3: The transfer function u as predicted by SLISOTRON (blue) and Isotron (red) for the `concrete` and `communities` datasets. The domain of both functions was normalized to $[-1, 1]$.

References

[1] P. McCullagh and J. A. Nelder. *Generalized Linear Models (2nd ed.)*. Chapman and Hall, 1989.

[2] P. Hall W. Härdle and H. Ichimura. Optimal smoothing in single-index models. *Annals of Statistics*, 21(1):157–178, 1993.

[3] J. Horowitz and W. Härdle. Direct semiparametric estimation of single-index models with discrete covariates, 1994.

[4] A. Juditsky M. Hristache and V. Spokoiny. Direct estimation of the index coefficients in a single-index model. Technical Report 3433, INRIA, May 1998.

[5] P. Naik and C. Tsai. Isotonic single-index model for high-dimensional database marketing. *Computational Statistics and Data Analysis*, 47:775–790, 2004.

[6] P. Ravikumar, M. Wainwright, and B. Yu. Single index convex experts: Efficient estimation via adapted bregman losses. Snowbird Workshop, 2008.

[7] A. T. Kalai and R. Sastry. The isotron algorithm: High-dimensional isotonic regression. In *COLT '09*, 2009.

[8] A. T. Kalai, A. R. Klivans, Y. Mansour, and R. A. Servedio. Agnostically learning halfspaces. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '05, pages 11–20, Washington, DC, USA, 2005. IEEE Computer Society.

[9] S. Shalev-Shwartz, O. Shamir, and K. Sridharan. Learning kernel-based halfspaces with the zero-one loss. In *COLT*, 2010.

[10] L. Yeganova and W. J. Wilbur. Isotonic regression under lipschitz constraint. *Journal of Optimization Theory and Applications*, 141(2):429–443, 2009.

[11] S. M. Kakade, A. T. Kalai, V. Kanade, and O. Shamir. Efficient learning of generalized linear and single index models with isotonic regression. arxiv.org/abs/1104.2018.

[12] UCI. University of california, irvine: `http://archive.ics.uci.edu/ml/`.

[13] S. Cosslett. Distribution-free maximum-likelihood estimator of the binary choice model. *Econometrica*, 51(3), May 1983.