Nonparametric Combinatorial Regression for Shape Constrained Modeling

Farinaz Koushanfar, *Member, IEEE*, Mehrdad Majzoobi, *Student Member, IEEE*, and Miodrag Potkonjak, *Member, IEEE*

***Abstract*—We introduce a unified approach for calculating nonparametric shape constrained regression. Enforcement of the shape constraint often accounts for the impact of a physical phenomenon or a specific property. It also improves the model's predicability and facilitates subsequent optimizations. The regression models are built by transforming the problem into the combinatorial domain where the shape constraints are imposed by bounding the combinatorial search space. We start by addressing isotonicity shape constraint using a dynamic programming algorithm and demonstrate how the problem can be mapped to the graph combinatorics domain. Next we show how a number of other important shape constraints including unimodality, convexity, limited level set, and limited slope can be addressed using the same framework. The flexibility of proposed framework enables solving the shape constrained regression problem with an arbitrary user-defined error metric. This flexibility is exploited to add robustness against outliers to the model. The algorithms are described in detail and their computational complexity is established. The performance and effectiveness of the shape constrained regression is evaluated on traces of temperature and humidity measurements from a deployed sensor network where a high degree of accuracy and robustness is demonstrated.**

***Index Terms*—Convex, isotonic, nonparametric, robust regression, shaped constrained regression, unimodal.**

I. Introduction

In many real world phenomena, it is often observed that a group of variables are dependant on a hidden variable and a change in the hidden variable causes a proportional change in other dependent variables. In such scenarios, addition of the isotonicity (monotonicity) constraint by integrating the directional change into the models accounts for the existence of hidden covariate(s) and improves the predictability of the learning scenario [1]. More complicated relationships could be addressed by employing more intricate models. For example, if an increase in one variable invokes an increase in another variable which is subject to saturation, a concave model may be appropriate; or, if the interdependent variable increases to some point and decreases thereafter, one should use a unimodal constraint.

Manuscript received February 25, 2009; accepted June 22, 2009. First published August 18, 2009; current version published January 13, 2010. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Biao Chen. This work was supported in part by the NSF CAREER by Grant Award 0644289.

F. Koushanfar and M. Majzoobi are with the Department of Electrical and Computer Engineering, Rice University, Houston, TX 77005 USA (e-mail: farinaz@rice.edu; mm7@rice.edu).

M. Potkonjak is with the Department of Computer Science, University of California, Los Angeles, CA 90095 USA (e-mail: miodrag@cs.ucla.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSP.2009.2028937

Modeling such relationships has been a challenging task. Statistical models are generally classified as nonparametric and parametric. Nonparametric statistical models pose very mild assumptions on the form of the functional relations. Parametric models on the other hand, assume known full form of the functional dependency, except for a few parameters. Shape constrained regression could be viewed in the context of nonparametric regression where the fit's shape and structure are constrained. Parametric modeling assumptions for addressing shape constrained regression become inefficient in practice. In most previously developed nonparametric methods, model shape is not specified a priori but is instead determined from the data.

Shape constrained regression finds many applications in diverse areas including kernel density estimation, classification, nonlinear filtering and other tasks in signal processing, statistics, and machine learning. In signal processing, for example, locally monotonic regression has been used as the optimal counterpart of iterated median filtering [2]–[4]. An important and recent application domain is sensor networks, where the sensor nodes measure variations of sources of stimuli [5]. In sensor networks, typically a multitude of sensing devices are deployed in different spatial coordinates to capture the variations of a sensed phenomenon (stimuli). Sensor readings from the nodes exposed to the same source of stimuli such as temperature often encounter the same variations; i.e., if the temperature goes higher, both sensors would measure a higher temperature value. For an observer unaware of the stimuli properties, the measured value by sensors follow a monotonic relation.

A large body of research has addressed most shape constrained models such as locally monotonic [2] piecewise monotonic [6], [4], constant runlength [3], unimodal and oligomodal [7] on some error functions. However, a unified framework for shape constrained modeling has not been addressed so far.

We introduce a *generic* combinatorial framework that facilitates creation of polynomial time algorithms for a variety of shape constrained regression models with *arbitrary user-defined* error functions. The framework can address a number of important shape constraints including isotonic, unimodal, convex (or concave), limited number of level sets, and limited slope shape constraints using dynamic programming. Limiting the number of level sets is beneficial since regular nonparametric regression methods often require a large number of parameters for model description and thus, are not suitable for optimizations and efficient storage. Furthermore, the bias-variance tradeoff often implies simpler models [8]. The method

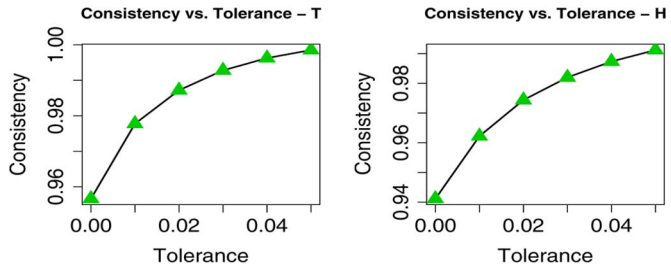

Fig. 1. Consistency (%) of monotonicity of readings at sensors 1 and 2 for temperature (left) and humidity (right).

is made robust against outliers by utilizing kernel smoothing, robust M-estimators by taking advantage of the flexibility in choosing the error function.

The remainder of the paper is organized as follows. In the next section, we motivate the need for the new unified framework by performing exploratory data analysis on sensor traces. In Section III, we survey the related literature. Section IV presents the basis of the framework by introducing and addressing univariate combinatorial isotonic regression and provides a graph combinatorial mapping of the problem. Section V further explores the flexibility of the new approach to provide solutions to several generalizations of the problem. Before we conclude the paper, we evaluate the performance and effectiveness of methods on sensor traces in Section VI.

II. Motivation

Assume that we have a sensor network whose nodes are sampling a physical phenomena in the lab (e.g., temperature). Since the network deployment is rather dense in the lab, there are spatial and temporal correlations among the readings of different nodes. If two sensors s_X and s_Y are exposed to related sources of stimuli, any changes in the sources will affect the readings at both sensors in a similar way. For example, an increase in the stimuli value will result in higher readings at both s_X and s_Y. We incorporate this observation into our prediction model by adding the isotonicity requirement such that as the measured values at sensor s_X increase, the predicted values for sensor s_Y increase and vice versa.

We have tested the monotonicity hypothesis on the node pairs in the Intel lab [5]. We computed the consistency of isotonicity as follows. If $x(t_2) > x(t_1)$ and $y(t_2) > y(t_1)$, or if $x(t_2) < x(t_1)$ and $y(t_2) < y(t_1)$, then our count of consistency increases by one (for time t_1). The final consistency is presented as the percentage of consistent measurement pairs for all considered time slots. In order to allow for small amounts of noise in the measurement process, we also consider a pair of readings to be consistent if $x(t_2) > x(t_1)$ and $y(t_2) + \delta > y(t_1)$, or if $x(t_2) < x(t_1)$ and $y(t_2) < y(t_1) + \delta$, where δ is a small noise tolerance parameter. The consistency of monotonicity between two sensors versus tolerance is shown in Fig. 1. We see that even with zero noise tolerance, the consistency of monotonicity is very high: $> 96\%$ for temperature and $> 94\%$ for humidity measurements.

III. Related Work

Calculation of the isotonic regression has a long and rich history. In this section, we briefly survey the most important contributions with respect to isotonic regression, and the related problems of unimodal, and convex (concave) regression and their robust versions.

Robertson *et al.* [1] discuss a number of applications that benefit from imposing the isotonicity constraint. More recently, applications of isotonic regression in statistical modeling of the sensor network data has been demonstrated [5]. Such models can be utilized to address different sensor network problems including density estimation for the sensed phenomena, calibration [9], [10], compression [11], [12], and query processing [13], [14].

One of the earliest and most widely used techniques in nonparametric isotonic regression is Pool Adjacent Violators algorithm (PAV) proposed by Brunk in 1955 [15]. The approach is designed for univariate regression and works by sorting the explanatory and then averaging the responses that violate the isotonicity requirements. The technique was the first to give the minimal results with respect to L_2 error norms and to have a computational complexity of $O(N)$. A drawback of the PAV algorithm is that if there are outliers or aberrant observations the PAV algorithm will produce long, flat levels and inaccurate models.

In 1964, Kruskal published a paper that in context of his multidimensional scaling solved the monotonic regression problem [16]. The theoretical complexity of Kruskal's algorithm is loosely upper bounded by $O(N^2)$. The approach is similar to the PAV algorithm and has been often referred as the up and down block (UDB) algorithm. It has been used and refined by a number of researchers and eventually, combined with algorithms for fast computation of prefix constrained structure and resulted in a fast alternative to the PAV algorithm—prefix isotonic regression proposed by Stout [7]. Prefix isotonic regression can be used for finding the optimal unimodal regression for L_1 and L_2 norms in $O(N \log N)$ and $O(N)$, respectively, where N is size of the data sequence.

Sager and Thisted [6] proposed a method for finding the best isotonic regression path by employing the dynamic programming paradigm for variables with a finite alphabet. The complexity of this algorithm is $O(NA)$, where A is the size of the alphabet. Restrepo and Bovik [2] studied locally monotonic regression as the optimal counterpart of iterated median filtering and developed a mathematical framework for studying the problem. They proposed an algorithm to the problem that was exponential in $\Re^N$.

Sidiropoulos [3] considered digital locally monotonic regression and provided a dynamic programming (Viterbi) algorithm for solving the locally monotonic regression. Local monotonicity of degree α is concerned with local subparts of the signal that are increasing (decreasing) while there might be constant levels of signal that at most have a length of $\alpha - 1$. The complexity of the proposed local monotonic regression is $O(A^2 \alpha N)$, where A is the size of the alphabet and N is the length of data sequence.

Boyarshinov *et al.* [17] take a step forward by introducing an L_1 isotonic regression that can be performed in $O(N \log A)$. They also show that if input values fall in the range of $[a, b]$ and given $\epsilon > 0$, an approximate L_1 isotonic regression with error at most the optimal plus ϵ could be solved in time $O(N \log((a-b)/(\epsilon)))$. An L_∞ isotonic regression is also solved in linear time.

Finding a convex isotonic regression for a convex cost function is a shape-constrained modeling problem that can be solved using convex optimization methods. Ahuja and Orlin [18] present a convex optimization algorithm that solves the problem in $O(N \log A)$, assuming that the inputs were sorted. Melanie and Holger also studied the problem of estimating a convex function for nonparametric regression.

Recently, there has been a great deal of interest in developing new algorithms for isotonic regression [19], [20] and in generalizing the basic shape-constrained paradigms for developing statistically more sophisticated formulations that include isotonic regression constraints [21]–[23].

Even though the method presented in this paper employs a dynamic programming approach that was also utilized in [6], [3], and [4], there are several important new aspects. First of all, the new method employs a generic framework to solve different shape constrained problems by minimizing an arbitrary objective function. It embraces isotonic, unimodal, convex (or concave) shapes while it has the flexibility to limit the slope range in the shape. Although being a nonparametric method, as we will show, it can impose a restriction on the number of parameters which simplifies the description of the regression models. This is suitable for subsequent storage and optimization tasks being conducted in sensor networks and to other environments with computational and storage constraints. Second, the algorithms perform in a time efficient manner, mostly in linear time. With a reduction in alphabet size, the time complexity can be enhanced significantly. Third, the degrading effect of outliers can be suppressed by integrating robustness into the method.

IV. Univariate Combinatorial Isotonic Regression (CIR)

In this section, we introduce the univariate CIR approach. We start by presenting the generic dynamic programming algorithm for CIR. Next, we show the mapping from the isotonic regression problem into a combinatorial shortest path problem and discuss the flexibility that can be exploited at the graph level.

A. Problem Formulation and CIR Approach

In univariate isotonic regression, the goal is to model the response $Y = \{y_t\}$ given the measured data from the predictor $X = \{x_t\}$, at time index $t = 1, 2, \ldots, T$, while the model satisfies the isotonicity constraints. Specifically, we are given T real data values (x_t, y_t, w_t), with nontrivial real-weights $w_t, t = 1, \ldots, T$, where X and Y are each drawn from a finite alphabet of size A. The T values of X can be ordered as $x_{(1)} \leq \cdots \leq x_{(t)} \leq \cdots \leq x_{(T)}$, where $T \gg A$, thus a number of consecutive values of $x_{(t)}$ can be equal. For each value of $x_{(t)}$, there is a corresponding value of $y_{(t)}$ that denotes the value of the response Y, measured at the same time epoch as $x_{(t)}$. The cost function for measuring the prediction error is denoted by ϵ_p.

The ϵ_p-isotonic regression of the data is defined as the set $\{(x_{(t)}, \hat{y}_{(t)}) : i = 1, \ldots, T\}$ that minimizes the error measure $\epsilon_p(y_{(t)}, \hat{y}_{(t)}, w_{(t)})$ subject to nondecreasing isotonic constraint:

$$\hat{y}_{(1)} \leq \hat{y}_{(2)} \cdots \leq \hat{y}_{(T)} \tag{1}$$

The CIR approach is independent of the form of the cost function ϵ_p. Commonly used forms of ϵ_p are the L_p norms of the error that are shown in (2).

$$\begin{cases} \left(\sum_{t=1}^{T} w_{(t)} |y_{(t)} - \hat{y}_{(t)}|^p\right)^{1/p} & \text{if } 1 \leq p < \infty \\ \max_{t=1}^{K} w_{(t)} |y_{(t)} - \hat{y}_{(t)}|, & \text{if } p = \infty. \end{cases} \tag{2}$$

Since $A \ll T$, the ordered set $\{x_{(t)}\}$ contains redundancies. To reduce this ordering to a strict order, we eliminate redundancies by grouping the same values together, and thus generate an order $x_{(1)} < \cdots x_{(i)} < \cdots x_{(A)}$. Thus, we have A distinct possible values for the stream X. The notation $x_t = x_{(i)}$ indicates that variable X at time t has the value $x_{(i)}$ where $x_{(i)}$ denotes the ith value in the ordered set of readings. We also produce the strict order for the data stream Y so that $y_{(1)} < \cdots y_{(j)} < \cdots y_{(A)}$, where A denotes the number of distinct values (alphabet) for stream Y. Note that the size of the alphabets for the streams X and Y is assumed to be the same for simplicity of the notation and does not have to necessarily be the same. Generalization to the cases where X and Y are drawn from two sets with different alphabet sizes is straightforward.

The generic dynamic programming method for addressing the isotonic regression consists of four steps: (i) a relative importance matrix R is created; (ii) based upon R, an error matrix E is built; (iii) a cumulative error matrix C is constructed using a dynamic programming based algorithm; and (iv) an optimum path that yields the minimum cumulative error is found on C. We explain the steps in detail below and use the example in Fig. 2 to illustrate each step.

i) We start by using the data tuples to form a relative importance matrix R where each $r_{i,j}$ indicates the number of times when stream X has value $x_{(i)}$ and stream Y has the value $y_{(j)}$ at the same time. We define $\rho_{i,j}(t)$ to be

$$\rho_{i,j}(t) = \begin{cases} 1, & \text{if } x_t = x_{(i)}, \text{and } y_t = y_{(j)} \\ 0, & \text{otherwise;} \end{cases}$$

The elements of the matrix R are calculated using equation: $r_{i,j} = \sum_{t=1}^{T} \rho_{i,j}(t)$. In the example in Fig. 2, the alphabet set is $\{1, 2, 3, 4, 5\}$ and each stream can have one of five different values $(A = 5)$. The matrix R is in fact a histogram of the entire set of $(x_{(i)}, y_{(j)})$ for $0 \leq i, j \leq A$.

ii) In this step we populate the error matrix E whose elements $e_{i,j}$ each indicate the error that would be made if the prediction of $\hat{Y}$ was selected to be $y_{(j)}$ whenever stream X had the reading $x_{(i)}$. For example in Fig. 2, whenever we have $X = x_{(1)}$, the most common observation at Y is $y_{(1)}$. Hence, the predictor $y_{(1)}$ is not always correct and error may arise for the prediction. In Fig. 2, we use the error function

$$e_{i,j} = \sum_{k} r_{ik} \left| y_{(j)} - y_{(k)} \right|$$

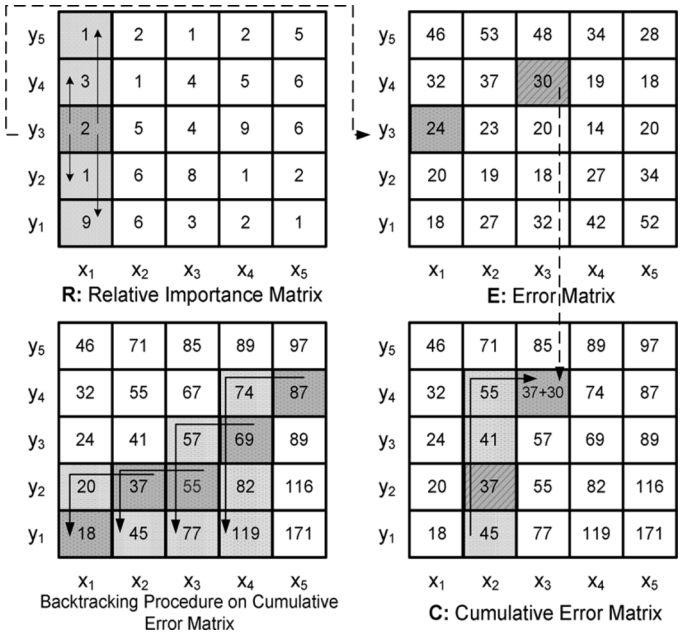

Fig. 2. Small example showing the steps for building a CIR model for a response stream Y from the data at stream X, where each stream has only 5 possible values. The bottom left matrix shows how the trace backing is accomplished.

```
Procedure CIR(E, X, Y, Ŷ)
 1. ∀ i, ∀ j, c_{i,j} = 0;
 2. ∀ j, c_{1,j} = e_{1,j};
 3. for (i = 2 to i = A)
 4.       for (j = 1 to j = A)
 5.             c_{i,j} = e_{i,j} + min_{1≤j'≤j} c_{i-1,j'};
 6.             index_{i,j} = arg min_{1≤j'≤j} c_{i-1,j'};
 7.       end;
 8. end;
 9. mini = arg min_{i=A,1≤j<A} c_{i,j};
10. for (i = A down to i = 1)
11.       ŷ_{(i)} = y_{mini};
12.       mini = index_{i,mini};
13. end;
```

Fig. 3. Pseudocode for the dynamic programming algorithm for finding the isotonic model between data streams X and Y.

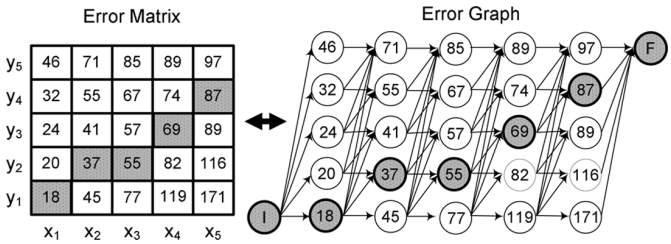

Fig. 4. Duality between the matrix computation for finding the isotonic model and the shortest path problem on a graph is shown on the small example of Fig. 2.

which corresponds to the L_1 error norm. The weight r_{ik} indicates how often the particular error $y_{(j)} - y_{(k)}$ occurs. We could define any other cost function, based on $e_{i,j} = \sum_k \epsilon_p(y_{(k)}, y_{(j)}, r_{ik})$, but for clarity of presentation we limit ourselves here to the L_1 norm. In Fig. 2, we show how the matrix E is constructed. The figure highlights $e_{1,3}$ and illustrates how it is computed using the first column of the R matrix as weights in the cost function: $e_{1,3} = 9 \times |3-1| + 1 \times |3-2| + 2 \times |3-3| + 3 \times |3-4| + 1 \times |3-5| = 24$.

iii) The goal of the isotonic regression (i.e., finding $\hat{Y} = f(X)$) is to find a nondecreasing path that traverses the matrix E while experiencing the minimum possible error along the path. The path starts at the point $x_{(1)}$ with some y-intercept, and moves in nondecreasing fashion across the space of X values to $x_{(A)}$. This path should accumulate the minimum error and visit each column exactly once. To find this path, we construct a cumulative error matrix C in which each element $c_{i,j}$ holds the minimum error incurred so far along the isotonic path to that point. We start filling the cumulative error matrix C by copying the first column of E matrix. The C matrix is then filled column by column, from left to right. Each value of $c_{i,j}$ is the sum of its corresponding element in the matrix E, namely $e_{i,j}$, and the minimum value of the elements located below and in the previous column, i.e., $\min_{k \le j} c_{i-1,k}$. In our example, we show how $c_{3,4} = 67$ is computed by adding $e_{3,4} = 30$ and the minimum of its lower or equal Y values in the previous column of C. The candidate values are highlighted in light gray; the minimum is $c_{2,2} = 37$ which lies in the darker square. For each value in the C matrix, we also keep the index that identifies the minimum value in its previous column that led to the current estimate of the cumulative error.

iv) Once the cumulative matrix C is fully constructed, we find the minimum value in the last column of C (index $j = 4$ for this example) and extract the associated index (denoted by *mini*) to the previous column (in our example $\text{mini} = 3$). Finally, the procedure backtracks over the columns updating the *mini* value at each step (lines 9–13 in the pseudocode 3). For each column, the value of $y_{(\text{mini})}$ is stored as the best predicted value for $\hat{y}_{(j)}$. The final mapping is shown by the dark highlighted boxes in Fig. 2.

The procedure used for steps 3 and 4 is a dynamic programming solution to the problem of finding an isotonic regression that minimizes the L_1 error. Our pseudocode given in Fig. 3 specifies the detailed procedure. The runtime of the procedure is dominated by the two nested loops in steps 3 and 4 which runs for $(A \times A)$ times.

B. Interpretation in Graph Combinatorics Domain

So far, we have presented a dynamic programming algorithm that takes the error matrix as its input. The error matrix can be also interpreted as a graph. In Fig. 4, we illustrate mapping of the example shown in Fig. 2 into the graph combinatorial domain. The graph G_E corresponds to the error matrix, where the vertices of the graph show the elements of the matrix E. The number assigned to each vertex is the error value for its corresponding element in the error matrix. The edges of the graph present the possible directions a path can take on the error matrix. Since the path can never be decreasing, there are only directed edges going to the right and up on the graph. The graph

also has two added vertices, initial (denoted by I) and final (denoted by F), each with an edge with the incident value of 0. The problem of finding the isotonic model for the response Y based on the readings from the explanatory variable X now corresponds to finding the shortest path in the error graph from the initial vertex (I) to the destination vertex (F). Finding an arbitrary shortest path for two points in a general graph with V edges, using the Dijkstra's algorithm [24] has the complexity of $O(V^2 \log V)$. However, since the error graph has a special sparse structure, the Dijkstra's algorithm has $O(A^2)$ runtime.

V. From Univariate CIR to a Generic Framework

In this section, we show how CIR regression method can be generalized to constitute a spectrum of other shape constraints. The first important shape constraint we discuss is limiting the number of level sets in the isotonic model. We show how unimodality and convexity, two widely utilized shape constraints, can be addressed within the same framework. We also introduce the use robust M-estimators and kernel smoothing to create a robust regression method.

A. Limiting the Number of Level Sets

Isotonic regression consists of level sets of nondecreasing values. However, the number of obtained level sets from an isotonic regression can be very large, preventing simple description for the model. To address this problem, a number of heuristic isotonic regression methods and methods based on combining isotonic regression and smoothing techniques have been proposed [25]–[29]. For improving the parsimony of the models we impose a constraint on the maximum permissible number of level sets. We find an isotonic fit that is optimal with respect to a given cost function for a fixed given number of level sets in a polynomial runtime.

The terms level set and breakpoint are used interchangeably, although the number of levels is greater than the number of breakpoints by unity. The limited level set isotonic regression method virtually follows the same steps as CIR. The algorithm calculates CIR with a restricted (prespecified) number of level sets D and a maximum local slope of S by (i) building the relative importance matrix, R, (ii) generating an error matrix E based on R (iii) constructing a structure containing lists that correspond to the error matrix elements (iv) finding the indices of best path that yields the minimum cumulative error.

Steps **(i)** and **(ii)** are the same as the CIR algorithm. **(iii)** The main difference between CIR and the procedure in this step lies on how cumulative error values are generated. In CIR there was only one optimum isotonic path that terminated at a given entry of the error matrix. However in limited level set CIR, there are multiple optimum paths each of which collects the least possible error on the way while having a certain number of breakpoints and local slopes. Therefore, the method must keep track of two other attributes in addition to the cumulative error values: local slope and the number of breakpoint so far. To attain this goal, a list is created for each entry of E. The list associated with the entry e_{ij} holds groups of three-element state vectors $[\text{ce}, s, d]$, where ce corresponds to the cumulative error resulting from the best CIR mapping that ends at $X = x_{(i)}$ and $Y = y_{(i)}$, d denotes the number of breakpoints along the path, and s refers to the slope of the final segment of the path. We choose the notation $L_{ij}^{l}[k]$ to denote the k_{th} element $(1 \leq k \leq 3)$ in the l_{th} state vector within the list associated with the entry e_{ij}. Thus, the state vector elements $L_{ij}^{l}[1], L_{ij}^{l}[2]$, and $L_{ij}^{l}[3]$ hold cumulative error, slope value, and breakpoint count, respectively. The number of state vectors in each list may differ from one list to another and is determined dynamically by the algorithm. The algorithm starts by assigning states to the first column of L, i.e., L_{ij} for $i = 1$ $j = 1, 2, \ldots, A$. We start filling the cumulative error values (ce) by copying the first column of E matrix. The prior slope for the entries of the first column are set to a default zero value. The same applies to the number of break points in the first and second columns (since it takes at least three columns to define a breakpoint). Therefore, $L_{1j} = [e_{1j}, 0, 0]$ and $L_{2j}[3] = 0$ for $j = 1, 2, \ldots, A$. At each step, the algorithm looks back at the entries in the pervious column that are located below or at the same level and also not farther than S entries away. Then, it adds up the cumulative error value (i.e., $L_{i-1,k}^{l}[1]$ for all $\max(1, j - S) \leq k \leq j$ and l) and the corresponding value in the matrix E, namely $e_{i,j}$. Meanwhile, it sets the arrival slope value to the level difference, $j - k$. At each stage, the algorithm compares the current slope value with that of the previous entries. If there is a change, it increments the number of breakpoints, d, by one and stores it in the current state; otherwise, it takes d as the current breakpoint count (lines 10–15 in Fig. 5). The cases where the breakpoint count exceeds

```
Procedure LimitedLevelsetCIR(E, X, Y, Ŷ)
1. Input S, D; %'S' for local slope and 'D' for no. of beakpoints
2. m := 1;
3. L^1_{1j} = [e_{1j}, 0, 0] for j = 1, 2, ..., A;
4. for (i = 2 to i = A)
5.    for (j = 1 to j = A)
6.       for (k = max(1, j − S) to k = j)
7.          for (l = 1 to l = the number of states in L_{i−1,k})
8.             if (i = 2)
9.                D_0 := 0;
10.            else
11.               if (L^l_{i−1,k}[2] ≠ j − k)
12.                  D_0 := L^l_{i−1,k}[3] + 1;
13.               else
14.                  D_0 := L^l_{i−1,k}[3];
15.               end
16.            end
17.            if (D_0 ≤ D)
18.               L^m_{ij} = [e_{ij} + L^l_{i−1,k}[1], j − k, D_0];
19.               m := m + 1;
20.            end
21.         end
22.      end
23.      n := 1;
24.      for (brkPoint = 0 to brkPoint = D)
25.         for (slope = 0 to slope = S)
26.            V = {ce ∈ R | [ce, slope, brkPoint] ∈ L^l_{i,j}};
27.            if (V ≠ φ)
28.               L̂^n_{ij} = [min(V), slope, brkPoint];
29.               n := n + 1;
30.            end
31.         end
32.      end
33.      replace L_{i,j} with L̂_{ij};
34.   end
35. end
```

Fig. 5. Pseudocode for the dynamic programming algorithm for finding the limited levelset isotonic model between data streams X and Y.

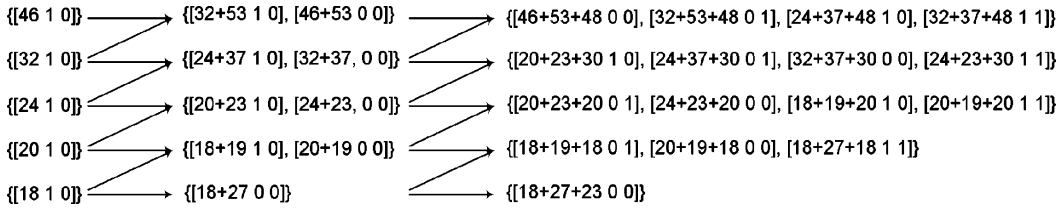

Fig. 6. Figure shows how the states are created for the first three columns of the 5 × 5 example during the limited level set isotonic regression process.

Procedure *Limited Levelset CIR, Backtracking* $(Y, \hat{Y})$
1. $[l', j']$ = arg $\min_{l,j} L^l_{Aj}[1]$;
2. $\hat{y}_{(A)} = y_{j'}$;
3. for $(i = A$ to $i = 2)$;
4. $[ce, s, d] = L^{l'}_{ij'}$;
5. $ce = ce - e_{ij'}$;
6. $j' = j' - s$;
7. $l' = \{l \in \mathbf{N} \mid L^l_{i-1,j'}[1] = ce$
$\wedge ((L^l_{i-1,j'}[2] = s \wedge L^l_{i-1,j'}[3] = d)$
$\vee (L^l_{i-1,j'}[2] \neq s \wedge L^l_{i-1,j'}[3] = d-1))\}$;
8. $\hat{y}_{(i-1)} = y_{j'}$;
9. end;

Fig. 7. Pseudocode for finding the isotonic limited breakpoint path indices.

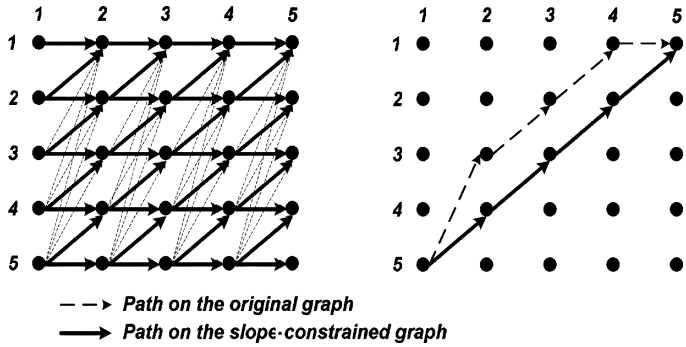

Fig. 8. An example where we constrain the slope to be either 0 or 1. The dotted edges on the left figure are removed to limit the maximum and minimum permissible slopes.

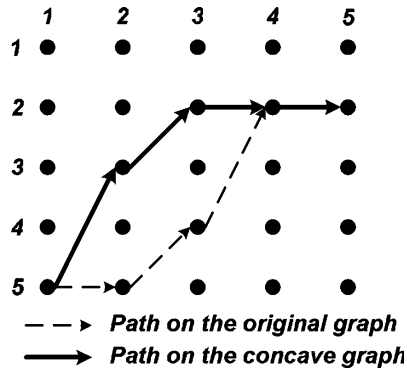

Fig. 9. An example showing the original CIR and the concave CIR fits. The concave path is a path with segments of nonincreasing slope.

D are avoided by not storing their state vectors in the list (lines 17–20). Once each list is fully generated, to ensure local optimality, we need to do a local minimum search within each list on the ce values of the vectors having the same slope and number of breakpoints, and replace them with the minimum value. Lines 23–33 in Fig. 5 show the corresponding steps. Fig. 6 illustrates the resulting lists and states for the first three columns of the example presented in the previous section.

(iv) The backtracking procedure carried out in this step is shown by the pseudocode in Fig. 7. We arrive at the final solution by finding the minimum of the cumulative error values by doing a search over the first elements of all state vectors within the lists in the last column $\text{ce} = \min_{l,j} L^l_{Aj}[1]$. Once the minimum is found, its associated arrival slope, s, and the breakpoint count d, are extracted (line 4). Meanwhile the current minimum cumulative error, ce, is subtracted from the corresponding element in the error matrix and updated to the new value (line 5). These three values (ce, s, d) specify a unique predecessor state which has a cumulative error value exactly equal to ce, a breakpoint count d, if having the same slope s or a breakpoint count $d-1$, if otherwise (line 7). This process is repeated until the algorithm reaches the first column and all of the path indices (j_s) are obtained.

1) Complexity: The complexity of the limited level set isotonic regression is dominated by: (1) the two nested loops in lines 4 and 5 that scan all of the list entries, and (2) the two nested loops in lines 6 and 7 that check the states in the previous list entries. The local search performed by loops in lines 24 and 25 has the same run time as (2). The loops in (1) iterate A^2 times. The *for* loop in line 5 repeats S times in the worst case, except for the entries close to the bottom of L. Also the list entries roughly contain $S \times D$ state vectors. Therefore, the two nested loops in line 5 and 6 run in $D \times S^2$ time steps for large A. The overall complexity is bounded by $\text{O}(A^2 \times D \times S^2)$.

B. Minimal and Maximal Slope

A straightforward modification to CIR is limiting the maximum and minimum local slope.

To constrain the slope, it is sufficient to remove the edges that correspond to high/low jumps in the dual error graph. Fig. 8 shows a small 5 × 5 example graph that limits the slope to be either 0 or 1 by removal of extra edges before solving the shortest path problem. On the right-side of Fig. 8, we show the isotonic regression on the original graph and on the slope-limited graph. As can be seen on this figure, constraining the slope can drastically alter the regression fit.

A minor modification to the CIR pseudocode in Fig. 3 is required to include the slope constraint. This can be done by bounding the search space in lines 5 and 6 for finding the minimum of cumulative error values in the previous column

$$5.c_{i,j} = e_{i,j} + \min_{j-\alpha \leq j' \leq j+\beta} c_{i-1,j'};$$
$$6.\text{index}_{i,j} = \arg \min_{j-\alpha \leq j' \leq j+\beta} c_{i-1,j'}$$

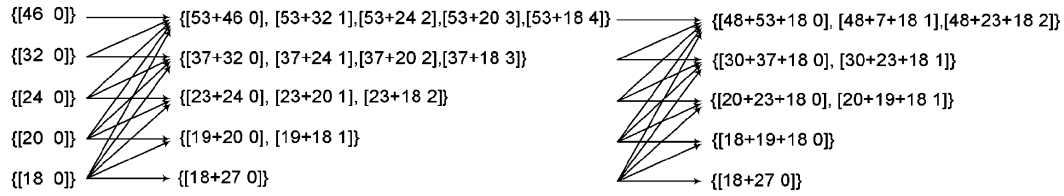

Fig. 10. Figure shows how the states are created for the first three columns of the 5×5 example during the concave isotonic regression process.

```
Procedure ConvexIsotonicRegression(E, X, Y, Ŷ)
{1.∀j L^1_1j = [e_1j 0];
 2.for (i = 2 to i = A)
 3.   for (j = 1 to j = B)
 4.       l := 1;
 5.       {V} := φ;
 6.     for (k = 1 to k = j)
 7.        for (u = 1 to u = the No. of states in L_i−1,k)
 8.          if j − k ≤ L^u_i−1,k[2]
 9.              add L^u_i−1,k[2] to {V};
10.          end
11.        end
12.     end
13.     if {V} ≠ φ
14.        L^l_ij[1] := e_ij + min{V};
15.        L^l_ij[2] := j − k;
16.        l := l + 1;
17.     end
18.  end
19.end
}
```

Fig. 11. Pseudocode for the dynamic programming algorithm for finding the concave isotonic model between data streams X and Y.

where α and β are the minimum and maximum slope, respectively.

C. Convexity

The path search space can be bounded in a way to produce a regression model that follows a concave or convex shape. Fig. 9 shows how the inclusion of concavity constraint alters the regression path. Finding the convex or concave model involves the same four essential steps as other shape constraints in the framework: (i) creating the relative importance matrix; (ii) building the error matrix; (iii) constructing lists of state vectors that correspond to the error matrix elements; and (iv) finding the indices of the best path that yields the minimum cumulative error.

Steps **(i)** and **(ii)** are the same for every shape constraint regression defined in the combinatorial framework.

(iii) we associate a list of states to each entry of the error matrix. Each state holds a cumulative error and segment slope value. A concave (convex) shape is composed of line segments with monotonically decreasing (increasing) slopes. We choose the notation L_{ij} and $[\mathrm{ce}, s]$ for the structure containing lists and state vectors, respectively. Therefore, $L^l_{ij}[1]$ and $L^l_{ij}[2]$ refer to cumulative error and slope values, respectively. The algorithm starts by initializing states for the lists associated with the first column of E. The prior slope for the entries of the first column are set to the default value of zero $L^l_{1j} = [e_{1j}, 0]$.

The pseudocode for the procedure is specified in Fig. 11. The pseudocode applies to the concave model but it could be simply modified to include convexity by reversing the equality sign in line 8. For each entry on the second column, we need to look back at the entries located below or at the same level (i.e., $L^l_{i-1,k}$ for all $1 \leq k \leq j$ and l). The states whose slope values are smaller than their previous states are only considered and the rest are discarded (lines 8 to 10). A local search is performed over the cumulative error values of the states satisfying this condition to find the minimum. The minimum is added to the corresponding value in the E matrix, i.e., $e_{i,j}$, and at the same time the slope value is set to the level difference. (lines 13 to 17). The concept of local slope could be applied by simply bounding the search space in line 6 in Fig. 11 to $(k = \max(1, j-S)$ to $k = j)$. When evaluating the complexity of the algorithm, we will show how a modification of the search space can exclude redundant search attempts in order to reduce the runtime.

(iv) The backtracking mechanism is almost identical to limited breakpoint algorithm in Fig. 7. We look for the minimum cumulative error in the last column lists, i.e., $\mathrm{ce} = \min_{l,j} L^l_{Aj}[1]$. Once the minimum is found, we look at its state vector to obtain the corresponding slope, and go back accordingly to the previous entry and retrieve the list associated to the entry. We need to do a local search within the states in the list whose slopes are greater than the forward entry and have a cumulative error value that matches the forward minimum cumulative error minus its corresponding error value. The indices of the entries (that are in fact the indices of the optimum path) are recorded throughout the process.

1) Complexity: For a sorted input, the algorithm has the worst case complexity of $O(A^3)$. The alphabet size A is often much smaller than the data sequence length T. In practice, the maximal allowable local slope S is also typically much smaller than A and the runtime of the procedure reduces to $O(SA^2)$. Since $A^2 \ll T$, the runtime of the combinatorial convex isotonic regression procedure is typically smaller than the runtime of the algorithm presented in [18]. However, the main benefit of using this model is in its inherent flexibility to solve the convex regression problem for any given cost function, regardless of the convexity of the cost function.

D. Unimodality

In unimodal regression, the goal is to find a model that has a single local maximum (minimum). The unimodality constraint can be formally expressed as

$$\hat{y}_{(1)} \leq \cdots \hat{y}_{(m)} \geq \hat{y}_{(m+1)} \geq \cdots \geq \hat{y}_{(T)}, \tag{3}$$

where $\hat{y}_{(m)}$ is the peak or trough of the model. The point $X = x_{(m)}$ is called the *crossover point*; it is where the mode of the regression function changes from an increasing trend to a decreasing trend. In general, the cross-over point for an optimal unimodal fit is not unique. The procedures followed in [30]–[34], take a point, $x_{(i)}$, as the crossover point and

fit a monotonically increasing curve to the previous values $x_{(1)},\ldots,x_{(i)}$ and a monotonically decreasing curve to the proceeding values $x_{(i+1)},\ldots,x_{(n)}$. The process is repeated for each i, in $1 \leq i \leq n$, and the point with the smallest regression error is the optimal solution. The critical observation is that each iteration involves two calls to isotonic regression algorithm which significantly increases the complexity of the algorithm. Our algorithm is able to determine all of the error values in a single iteration using a dynamic programming approach instead of resolving the problem for each new individual crossover point.

The algorithm for computing the unimodal regression is optimal with respect to any given error norm. To find the unimodel regression, we run the dynamic programming method described in Fig. 3 twice. For the first run, the model is built exactly as shown in Fig. 3, where the cumulative error matrix is built from left to right (i.e., from $i = 1$, to $i = A$). For the second run, a small modification is applied to the original algorithm such that the dynamic programming procedure builds the cumulative error matrix from right to left (i.e., from $i = A$ to $i = 1$). In other words, the second run generates the cumulative error matrix required to construct a monotonically *decreasing* model. Meanwhile, two vectors whose elements correspond to the minimum of values in each column of Cs are created. The minimum value for each column of the *right-to-left (left-to-right)* C matrix obtained from the *first* (*second*) run represents the smallest possible regression error for the best monotonically *increasing* (*decreasing*) model that *ends on (begins from)* that column.

Next, we add up the two vectors element by element and form a sum vector. The minimum value and its location in the sum vector, respectively, represent the minimum unimodal regression error and the column index on which optimal crossover point lies. In other words, the minimum value in the sum vector indicates the sum of isotonically increasing regression error before the optimal crossover point plus isotonically decreasing regression error after this point. Due to the possibility of multiple optimal crossover points, there might exist multiple minima with the same values.

1) Complexity: The presented approach for finding the unimodal regression has a runtime of $O(A^2)$. Stout [7] has proposed an algorithm for unimodal regression based on a prefix isotonic regression that can be solved in $O(T)$ for L_2 and L_∞ norms.

E. Robust Regression

1) M-Estimators: It is also possible to create a robust regression by making adjustments to the framework. The robust regression approach is used when there are large outliers. A common robust regression method proposed by [35] is to find LMS (least median of squares) defined as

$$e = \min\{\text{med}_i \quad d_i^2\}$$

subject to the regression constraint, where $d_i = y_i - \hat{y}_i$. However, implementing this method with our regression paradigm (and other proposed nonparametric algorithms) would be costly since finding the least median of square of residuals requires an exhaustive search over the error matrix to find the best isotonic path that minimizes the objective function.

Robust regression methods are differentiated by the way they assign weights to the residuals. The most common method of robust regression is *M-estimation* introduced by Huber [36]. The M-estimators minimize the sum of a symmetric, positive-definite function $\rho(\cdot)$ of the residuals d_i, with a unique minimum at zero. For the least squares method, $\rho(d) = d^2$. Several ρ functions have been proposed that reduce the influence of large residual values on the fit. Huber proposed the squared error for small residuals and the absolute error for large residuals:

$$\rho(d) = \begin{cases} k|d| - 0.5k^2 & |d| \geq k \\ 0.5d^2 & |d| < k. \end{cases}$$

Andrews [37] used a squared Sine function for small and a constant for large residuals. Beaton and Tukey's [38] biweight is another example of these ρ functions. Earlier methods to solve such regressions with ρ error functions are based on iterative application of reweighted least squares. The weights at each stage are derived based on the assumed ρ function and the data. The reliability of the initial guess is of importance and the convergence of the solution was not proven for most ρ functions. Since our regression paradigm is versatile in terms of the error function, we can readily employ the M-estimators (ρ functions) while constructing the error matrix such that

$$e_{i,j} = \sum_k r_{ik}\rho\left(y_{(j)} - y_{(k)}\right)$$

where r refers to the relative importance matrix entry.

2) Kernel Smoothing: Outliers are generally large defective measurements that appear randomly in the data. It is therefore unlikely to have repeated outliers with the same amplitude unless there is systematic fault in the measurement unit. Since outliers appear as solitary entries with low frequency in the relative importance matrix (R), we can apply kernel smoothing on R to lessen their weight in calculating the error matrix (E).

VI. Experimental Evaluation

In this section, we evaluate our CIR algorithms on traces of temperature and humidity measurements from a deployed sensor network. The network consists of 54 nodes with capability to measure temperature, humidity, and light intensity. Fig. 13 depicts the location of nodes on the measurement field. We show how inclusion of shape constraints improves the predicability of the models and also study the effect of level set limitation and quantization of the alphabet values. Robustness of the method using robust M-estimators is examined.

A. Comparing CIR to Other Regression Methods

To evaluate the quality of the new CIR approach, we compare prediction error of the combinatorial isotonic regression (CIR) to prediction errors of (a) ordinary least square (OLS) linear regression and (b) robust least-square (LOESS) nonparametric regression on sensor network data. We selected these two models for comparison because: (i) our objective is to compare the results of isotonic regression to both parametric and nonparametric family of models; (ii) OLS regression is the most widely used parametric modeling method; and (iii) LOESS regression with proper parameter selection is widely used as a robust nonparametric modeling method. Note that the major difference between CIR and LOESS is the inclusion of isotonicity

constraint. Therefore, our goal is to gain insight to what extent we can suppress the prediction error by including the isotonicity requirement.

To compare the models, we use learn and test, resampling, bootstrapping and confidence intervals as our statistical validation and evaluation methods. We first applied learn and test procedure. Assume that the goal is to model the readings of a response sensor v_Y from the readings of an explanatory sensor v_X, using N data tuples (x_i, y_i), $i = 1, \ldots, N$. We split the N data tuples into two disjoint sets: the learning set that contains N_{learn} tuples and a test set with $N_{\text{test}} = N - N_{\text{learn}}$ tuples. Each model is built using the learning set. After that, the test set is used to evaluate the accuracy of modeling. For each x_i value in the test set, the model returns a predicted value $\check{y}_i$. The residual for each prediction value is defined as $r_i = y_i - \check{y}_i$. Thus, from each model we obtain a vector of residuals $[r_1, r_2, \ldots, r_{N_{\text{test}}}]$. We can now compute summary statistics on the residuals vector for each model as an illustration of the prediction error of the model. Multiple error norms were used for obtaining the summary statistics of residuals and relative residuals. The relative residuals indicate the prediction error in terms of the ratio of the absolute residual to the absolute response. The L_p error norm for relative residuals is defined as: $L_p(\text{relative}) = (\sum_{i=1}^{\text{Ntest}} |(r_i)/(y_i)|^p)^{1/p}$, for $y_i \neq 0$ and $1 \leq p < \infty$.

We have experimented with a wide variety of error measures, including L_1 and L_2 norms of both residuals and relative residuals. The resulting prediction errors comparisons were very similar across the different error measures. Due to similarity between different measures, the results shown in this section are limited to average error (L_1) of the relative residuals. The learn and test results are shown for a case when we use the first day as the learn data and then we examine the prediction quality on the data from the second day (test set). The frequency of L_1 prediction errors over all possible node pairs in the lab are shown in Fig. 12 for all three methods: OLS, LOESS, and CIR. It is easy to see small errors are much more likely for CIR than for the two other methods and that OLS has a significantly heavier tail for large errors. Analysis indicates that for both temperature and humidity sensors the nonparametric LOESS modeling achieves on average a factor of 2 lower error when compared to OLS modeling. In our experiments, we used the $\text{span} = 0.2N_{\text{learn}}$ for the LOESS modeling method.

Isotonic modeling method achieves on average more than 4 times reduction in prediction error when compared to the LOESS modeling approach. For example, the frequency of the internode models with the relative error rate of less than 1% increases by a factor of more than 6 for isotonic fits, when compared to the LOESS models. The higher frequency is important for developing effective sleeping strategy, since the key issue is not the overall prediction quality in the network, but larger number of node pairs where one node is a good predictor of another.

One can also interpret the prediction error results in terms of absolute temperature and humidity values. During the test phase (i.e., the second day of the experiment), the average temperature in the lab was 18.76°C and the average humidity in the lab was 42.17%. A temperature model with relative error measure of 0.5% can predict the unknown value of the response sensor

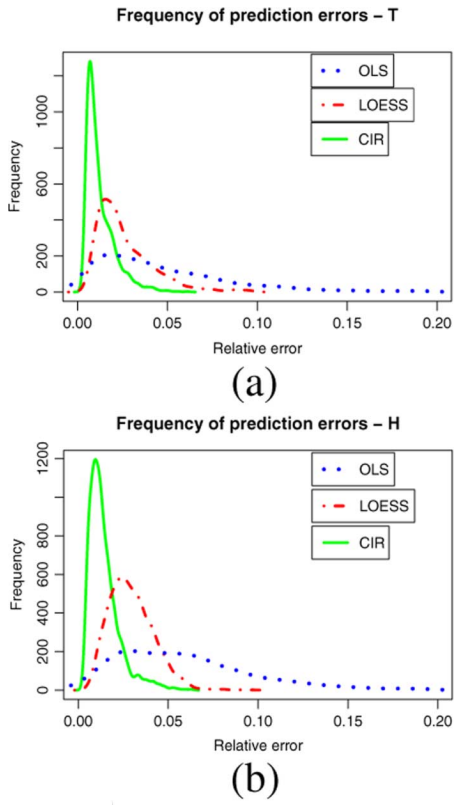

Fig. 12. Frequency for prediction error for three different prediction models on all node pairs in the network for temperature and humidity sensors: ordinary least squares linear regression, LOESS nonparametric model, and CIR isotonic regression. (a) Temperature. (b) Humidity.

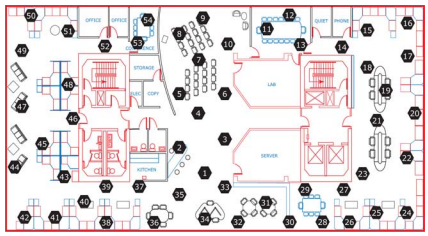

Fig. 13. The map of the sensor deployment area at Intel Berkeley Lab. The sensors nodes are shown by hexagons on the map.

with an absolute error (residual) of less than 0.1°C. Similarly, a humidity model with relative error measure of 0.5% can predict the unknown value of the response sensor with an absolute error (residual) of less than 0.2% in humidity reading.

In addition to comparing the prediction error of different models, we use other standard statistical validation techniques to assess the quality of the models. We apply resampling and bootstrapping to find the distribution and confidence intervals for the prediction errors. Specifically, we uniformly randomly select 65% of the original two-days data and use this data as a learning set to build a model. Next, we use the rest of the data as a test set and find the prediction error on test data.

We perform bootstrapping on the prediction error by repeated sampling from the original data set and estimating the sampling distribution of the prediction error. The CI% confidence interval of the modeling method is directly found from the distribution.

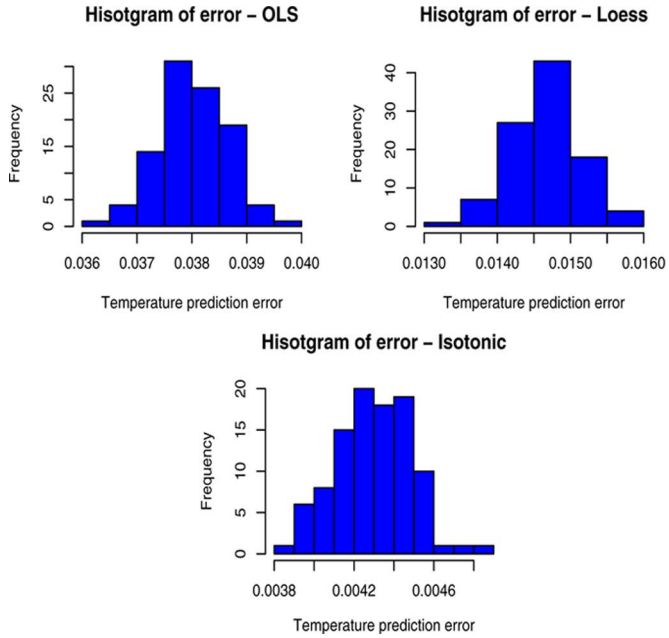

Fig. 14. The sampling distribution of temperature prediction error for predicting temperature sensor v_2 from the readings at temperature sensor v_1: OLS model (left), LOESS model (middle), and isotonic model (right).

Bootstrapping results are shown in Fig. 14, where the resampling was repeated 100 times. We show the sampling distribution of temperature prediction error for predicting sensor v_2 from the readings at sensor v_1. We present the results for the OLS model, LOESS model and the CIR model. The confidence interval of the prediction errors are readily deduced from sampling distribution. For example, we can see that the 95% confidence interval for prediction error is in range [0.0369, 0.0397] for OLS model, in range [0.0138, 0.0153] for LOESS model, and in range [0.0039, 0.0045] for CIR model. The significantly smaller range for CIR (0.0006) than for LOESS (0.0017) and OLS (0.0038) indicates that CIR does not only produces smaller errors, but also is a more consistent method.

1) Alphabet Size Reduction: Since the complexity of the algorithm depends heavily on the alphabet size (A), it is important to see how further quantization of values can affect the performance of CIR.

In our experiments, we varied the number of quanta and calculated the CIR error based on the resulting alphabet set. In Fig. 15, the vertical and horizontal axes show the regression error and alphabet size, respectively. They are normalized to the case where there is no reduction in alphabet set size. Interestingly, the reduction in alphabet size for the sensor data not only does not have an adverse effect but also improves the regression error in our application. In L_1 and L_2 cases the regression error is the smallest for alphabet sizes equal to 25% and 17% of the original alphabet size. Further size reduction leads to a faster runtime but decreases the accuracy of the model.

Fig. 17 illustrates CIR, convex CIR and limited level set CIR models for two different number of breakpoints. By looking closely at the regression models, deviations from the expected shape constraints may be found. For example, in the convex CIR model, we can find small segments where the model loses its convexity. This is due to the error introduced at the quantization step. Note that the isotonic model only guarantees that if

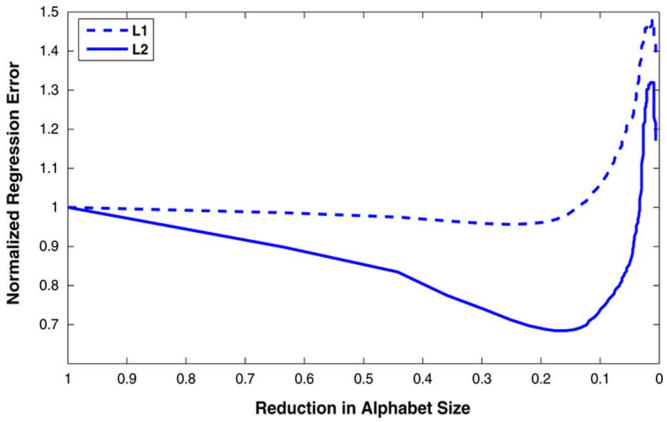

Fig. 15. The effect of alphabet size reduction on regression error and the accuracy of CIR model.

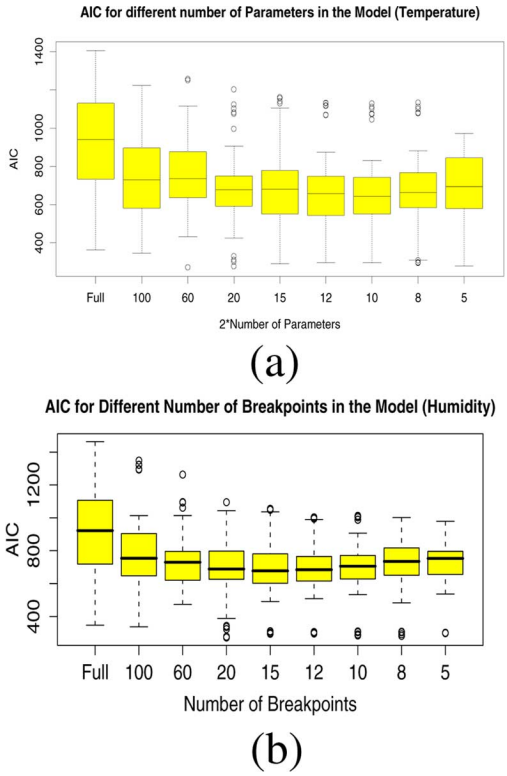

Fig. 16. AIC (y axis) for different number of breakpoints (x axis) in the model. The boxplots show the model between each node and its best predictor. (a) Temperature. (b) Humidity.

$X_i \geq X_j$, then $Y_i \geq Y_j$. However there might be instances where $X_i = X_j$, while the actual values quantized into X_i and X_j may not be equal. The coarser the quantization, the larger deviations from the expected shape and thus larger regression error will occur. Fig. 18 shows sensor node 10 predicting node 11 based on isotonic (dotted line), convex isotonic (dashed line), limited level set isotonic (dot-dashed line) models.

B. Limited Level Set CIR

We varied the number of level sets for a pair of nodes and compared the resulting regression error to that of the unlimited CIR. Table I shows the relative error value, i.e., $(e_{\text{Limited Level SetCIR}} - e_{\text{CIR}})/e_{\text{CIR}}$, for five different number of breakpoints. In the unlimited case, the CIR model has 24

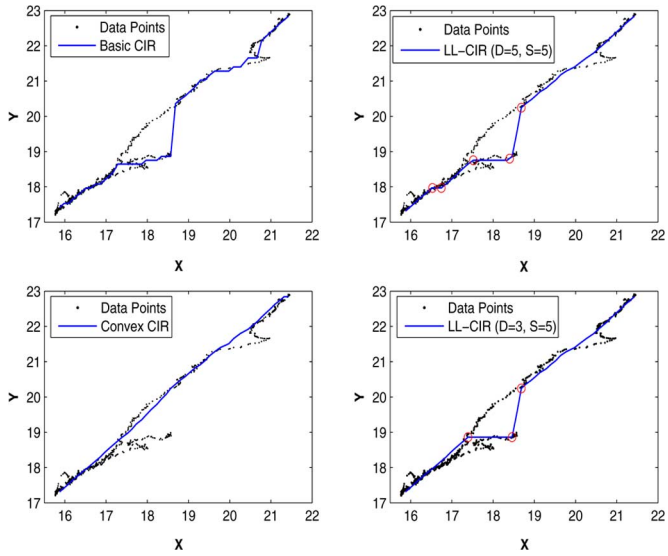

Fig. 17. Four different isotonic regression models for temperature readings; sensor-10 predicting sensor-11 values: original CIR model with 24 breakpoints (top left); convex CIR (bottom left); limited level set CIR with 5 and 3 breakpoints and a local slope of 5 (top and bottom right, respectively); level changes are indicated by circles.

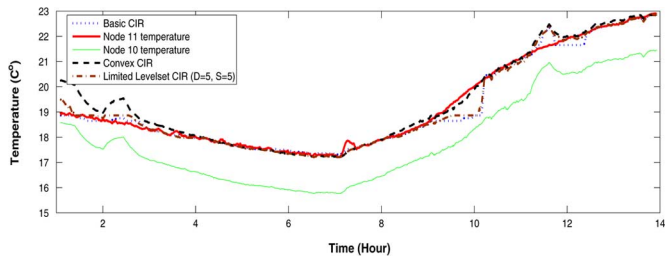

Fig. 18. Sensor node 10 and 11 measured temperature over time (solid lines). Sensor node 10 predicting node 11 based on isotonic (dotted line), convex isotonic (dashed line), limited level set isotonic (dot-dashed line) models.

TABLE I
THE REGRESSION ERROR FOR DIFFERENT NUMBER OF LEVEL SETS NORMALIZED TO THAT OF PLAIN CIR

No. of Breakpoints	1	3	5	7	9
Reg. Error	51.8%	20.7%	8%	3%	3.3%

breakpoints. Note that reducing the number of breakpoints to 7 only increases the regression error by 3% while we save 2×17 parameters (since each level set is defined at least by two parameters; one for slope and another for intercept).

The goodness of fit of a model and comparison between different models on the same data set can be quantified through the use of statistical information criteria, such as Akaike information criteria (AIC), Bayesian information criteria (BIC), or deviance information criterion (DIC). We have used AIC [39] as the goodness-of-fit measure of choice for our models. AIC uses the logarithm of the model's likelihood as the goodness-of-fit measure, while it also penalizes for the number of parameters in the model. The AIC is formally defined as, $\mathrm{AIC} = -2\ \log(\text{likelihood}) + 2D$, where D is the number of parameters in the model. Using the AIC criteria, we compare the original isotonic model to models where we restrict the number of breakpoints. The results are shown in Fig. 16(a) and (b), that shows the AIC of the best predictor of each node for temperature and humidity, respectively. Each boxplot presents the values for a defined number of breakpoints in the model. The sensors have around 1000 discrete values and the unrestricted isotonic model has on average more than 300 breakpoints. The results of these two plots shows that according to AIC criteria, limiting the number of breakpoints to 10 for the case of temperature and to 15 for the case of humidity produces the best models.

TABLE II
IMPROVEMENT IN REGRESSION ERROR USING HUBER ESTIMATOR

Outlier Density	10%	30%	50%
Huber/L_1	1.28	1.05	2.89
Huber/L_2	2.47	1.86	1.75

C. Robustness Evaluation

We now evaluate the robustness achieved by using robust M-estimators. The robust algorithms are applied to the sensor data after synthetically injecting outliers. The generated outliers have a Gaussian amplitude distribution with a mean equal to the original data sample value and variance ten times larger the original data variance. They are uniformly spread over the measurement data with different densities (i.e., number of outliers per total number of data samples) for each experiment.

1) M-Estimators: We applied the Huber ρ function with $k = 1$. Table II shows the improvement achieved by using the Huber ρ function compared to L_1 and L_2 objective functions. After the fits are determined based on the contaminated data samples, we find the residuals by comparing the fit values to the clean data sample to see the degrading effect of the outliers on the fit. The improvement factor is obtained by dividing the absolute sum of residuals when comparing L_1 and Huber and sum of square of residuals when comparing L_2 and Huber. The numbers are averaged over multiple runs. The factor indicates that the fit is more resilient to outliers when the Huber estimator is used since the deviation from the original data samples becomes smaller.

VII. CONCLUSION

We presented a general method for shape constrained regression that is based on mapping the problem to the combinatorial domain. We have illustrated the method by adding the isotonicity shape constraint and used a dynamic programming to address the problem. We showed how the presented combinatorial isotonic regression problem could be mapped to the graph combinatoric domain. Based on the same concept, we build a general framework that could be used to integrate other shape constants, including convexity, unimodality, limited slope, and limited level sets. The method is independent of the form of the error norm used. We further used the versatility of the framework to find the robust regression fit. The complexity of each model was computed and compared to previous works. We presented performance results for each shape constraint on temperature measurements from a deployed sensor network. Alphabet size reduction was performed to lower the complexity and its impact on the accuracy of the model was studied. The robustness was evaluated on data samples contaminated with synthetic outliers and a high degree of resilience was demonstrated.

REFERENCES

[1] T. Robertson, F. Wright, and R. Dykstra, *Order Restricted Statistical Inference*, ser. Probability and Mathematical Statistics. New York: Wiley, 1988.

[2] A. Restrepo and A. Bovik, "Locally monotonic regression," *IEEE Trans. Signal Process.*, vol. 41, no. 9, pp. 2796–2810, 1993.

[3] N. Sidiropoulos, "The Viterbi optimal runlength-constrained approximation nonlinear filter," *IEEE Trans. Signal Process.*, vol. 44, no. 3, pp. 586–598, 1996.

[4] N. Sidiropoulos and R. Bro, "Mathematical programming algorithms for regression-based nonlinear filtering in ir," *IEEE Trans. Signal Process.*, vol. 47, no. 3, pp. 771–782, 1999.

[5] F. Koushanfar, N. Taft, and M. Potkonjak, "Sleeping coordination for comprehensive sensing using isotonic regression and domatic partitions," in *IEEE Infocom*, 2006, pp. 1–13.

[6] T. Sager and R. Thisted, "Maximum likelihood estimation of isotonic modal regression," *Ann. Statist.*, vol. 10, no. 3, pp. 690–707, 1982.

[7] Q. Stout, "Optimal algorithms for unimodal regression," *Computing Sci. Statist.*, vol. 32, pp. 348–355.

[8] T. Hastie, R. Tibshirani, and J. H. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York: Springer, 2001.

[9] A. Ihler, J. Fisher, A. Moses, and A. Willsky, "Nonparametric belief propagation for self-calibration in sensor networks," *Inf. Process. Sens. Netw.*, pp. 225–233, 2004.

[10] V. Bychkovskiy, S. Megerian, D. Estrin, and M. Potkonjak, "Calibration: A collaborative approach to in-place sensor calibration," *Inf. Process. Sens. Netw.*, pp. 301–316, 2003.

[11] S. Pradhan, J. Kusuma, and K. Ramchandran, "Distributed compression in a dense microsensor network," *IEEE Signal Process. Mag.*, vol. 19, no. 2, pp. 51–60, 2002.

[12] S. Pattem, B. Krishnamachari, and R. Govindan, "The impact of spatial correlation on routing with compression in wireless sensor networks," *Inf. Process. Sens. Netw.*, pp. 28–35, 2004.

[13] C. Guestrin, P. Bodik, R. Thibaux, M. Paskin, and S. Madden, "Distributed regression: An efficient framework for modeling sensor network data," *Inf. Process. Sens. Netw.*, pp. 1–10, 2004.

[14] M. Paskin, C. Guestrin, and J. McFadden, "A robust architecture for distributed inference in sensor networks," *Inf. Process. Sens. Netw.*, pp. 55–62, 2005.

[15] H. Brunk, "Maximum likelihood estimates of monotone parameters," *Ann. Math. Statist.*, vol. 26, no. 4, pp. 607–616, 1955.

[16] J. Kruskal, "Nonmetric multidimensional scaling; A numerical method," *Psychometrika*, vol. 29, no. 2, pp. 115–129, 1964.

[17] V. Boyarshinov and M. Magdon-Ismail, "Linear time isotonic and unimodal regression in the l_1 and $l_{\inf}$ norms," *J. of Discrete Algorithms*, vol. 4, no. 4, pp. 676–691, 2006.

[18] R. Ahuja and J. Orlin, "A fast scaling algorithm for minimizing separable convex functions subject to chain constraints," *Operat. Res.*, vol. 49, no. 5, pp. 784–789, 2001.

[19] S. Angelov, B. Harb, S. Kannan, and L. Wang, "Weighted isotonic regression under the l_1 norm," in *Proc. ACM-SIAM Symp. Discrete Algorithm (SODA)*, 2006, pp. 783–791.

[20] X. Hu and J. Hansohm, "Merge and chop in the computation for isotonic regressions," *J. Statist. Plann. Inf.*, vol. 138, no. 10, pp. 2847–3292, 2008.

[21] I.-S. Chang, L. Chien, C. Hsiung, C. Wen, and Y. Wu, "Shape restricted regression with random Bernstein polynomials," *IMS Lecture Notes Monograph Ser.*, vol. 54, pp. 187–202, 2007.

[22] J. Wong, S. Megerian, M. Potkonjak, and S. Brook, "Symmetric monotonic regression: Techniques and applications in sensor networks," in *Sens. Appl. Symp. (SAS)*, 2007, pp. 1–6.

[23] J. Pal and M. Banerjee, "Estimation of smooth regression functions in monotone response models," *J. Statist. Plann. Inference*, vol. 138, no. 10, pp. 3125–3143, 2008.

[24] T. Corman, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms*. Boston, MA: MIT.

[25] M. Schell and B. Singh, "The reduced monotonic regression method," *Amer. Statist. Assoc.*, vol. 92, no. 437, pp. 128–135, 1997.

[26] J. Friedman and R. Tibshirani, "The monotone smoothing of scatterplots," *Technometrics*, vol. 26, pp. 243–250, 1984.

[27] K. Hildenbrand and W. Hildenbrand, Contributions to Mathematical Economics.

[28] H. Mukerjee, "Monotone nonparametric regression," *Ann. Statist.*, vol. 16, no. 2, pp. 741–750, 1988.

[29] E. Mammen, "Estimating a smooth monotone regression function," *Ann. Statist.*, vol. 19, no. 2, pp. 724–740, 1991.

[30] M. Frisén, "Unimodal regression," *The Statistician*, vol. 35, no. 4, pp. 479–485, 1986.

[31] Z. Geng and N.-Z. Shi, "Isotonic regression for umbrella orderings," *Appl. Statist.*, vol. 39, pp. 397–424, 1990.

[32] R. Mureika, T. Turner, and P. Wollan, An algorithm for unimodal isotonic regression with application to locating a maximum Statistics Tech. Rep., 1992, pp. 92–94.

[33] N.-G. Pehrsson and M. Frisén, "The uregr procedure," Gothenburg Computer Central. Göteborg, Sweden, 1983.

[34] T. Turner, "S function ufit to calculate the unimodal isotonic regression of a set of data," *Univ. New Brunswick Dep. Math. Stat.*, 1998.

[35] P. J. Rousseeuw, "Least median of squares regression," *J. Amer. Statist. Assoc.*, vol. 79, no. 388, pp. 871–880, 1984.

[36] P. Huber, *Robust Statistics*. New York: Wiley, 1981.

[37] D. Andrews, "Robust regression using repeated medians," *Technometrics*, vol. 69, no. 1, pp. 523–531, 1974.

[38] A. Beaton and J. Tukey, "The fitting of power series, meaning polynomials, illustrated on band-spectroscopic data," *Technometrics*, vol. 16, no. 2, pp. 147–185, 1974.

[39] H. Akaike, "A new look at the statistical model identification," *IEEE Trans. Autom. Control*, vol. 19, no. 6, pp. 716–723, 1974.

Farinaz Koushanfar (S'99–M'09) received the B.A.Sc. degree in electrical engineering from the Sharif University of Technology, Iran, the M.A. degree in statistics from the University of California, Berkeley, the M.Sc. degree in electrical engineering from the University of California, Los Angeles, and the Ph.D. degree in electrical engineering and computer science from the University of California at Berkeley, in 2005.

She is an Assistant Professor with the Departments of Electrical and Computer Engineering, and Computer Science, Rice University, Houston, TX, where she is also the Director of the Texas Instruments (TI) DSP Leadership University program. Prior to joining Rice University in July 2006, she held the Coordinate Science Lab (CSL) fellowship at the University of Illinois, Urbana Champaign. Her research interests include data integrity, statistical modeling and optimization, embedded systems, sensor-based and reconfigurable systems, hardware security, and hardware/software intellectual property (IP) protection.

Dr. Koushanfar is a recipient of the Office of Naval Research (ONR) Young Investigator Program Award (2009), MIT Technology Review's Top 35 Young Innovators Award (2008), the Defense Advanced Research Projects Agency (DARPA) Young Faculty Award (2007), a National Science Foundation (NSF) CAREER Award (2007), and an Intel Open Collaborative Research.

Mehrdad Majzoobi (S'05) received the B.Sc. and M.Sc. degrees in electrical and computer engineering from the University of Tehran, Iran, and Rice University, Houston, TX, in 2006 and 2009, respectively.

He is currently pursuing the Ph.D. degree in electrical and computer engineering department at Rice University. His research interests include statistical modeling and optimization, low-power embedded systems, sensor-based and reconfigurable systems, VLSI testing, hardware security, and hardware/software intellectual property (IP) protection and watermarking.

Miodrag Potkonjak (M'03) received the Ph.D. degree in electrical engineering and computer science from the University of California, Berkeley, in 1991.

After spending four years with the CCRL Lab, NEC, Princeton, NJ, he joined the Computer Science Department, University of California, Los Angeles (UCLA), where he has been Professor since 2000. He has published a book and more than 300 papers in leading CAD and VLSI design, embedded systems, real-time systems, computational sensing, and security journals and conferences. He holds ten patents. His watermarking-based intellectual property protection research formed a basis for the VSIA developing standard. His current research interests are focused on CAD and embedded systems, coordinated modeling and optimization, augmented reality, and computational sensing.

Dr. Potkonjak received the NSF CAREER award, OKAWA foundation award, UCLA TRW SEAS Excellence in Teaching Award, and a number of Best Paper Awards.