Soft Monotonic Constraint Support Vector Regression

Sapan Shah, Avadhut Sardeshmukh,
Shuaib Ahmed, Sreedhar Reddy

Tata Research Development and Design Center,
Tata Consultancy Services Limited,
Pune 411013
India
{sapan.hs, avadhut.sardeshmukh, s.ahmed2, sreedhar.reddy}@tcs.com

Abstract

This paper proposes a model for learning soft-monotonic regression functions in the presence of imperfect domain knowledge. It proposes an extension to support vector regression (SVR) wherein a new hardness parameter is introduced to configure the degree of monotonicity. The model supports multiple monotonicity constraints over multiple input dimensions simultaneously. The proposed model has been validated on synthetic datasets as well as on benchmark datasets obtained from real world problems. The results show that our model has better extrapolation capabilities than SVR. The results also demonstrate the ability of the model to generalize over multiple input dimensions.

1. Introduction

In machine learning, prior domain knowledge improves the quality of the model learnt, especially when data is scant or noisy. A common class of such prior knowledge is monotonicity constraints. For example, in credit rating, the chances of getting a loan increase with income, provided other variables are same [1]. The price of houses increases with better characteristics of houses such as the number of rooms [2]. It is reasonable to expect a machine learning algorithm to capture such knowledge in the model learnt, but that depends on the quantity and quality of the data available. Data inadequacy is a common problem in many domains. For instance, in materials engineering, process-structure-property relations play a key role in the design process. However, this data is hard to come by. In such cases, incorporating prior domain knowledge into a machine learning algorithm can improve the quality of the model learnt. For instance, in a carburization process, an increase in carbon potential increases the hardness of the material. This information can be leveraged by a learning algorithm to improve the model for a heat-treatment process.

Monotonic regression on a data set of n 1-dimensional pairs $\{(x_i, y_i)\}_{i=1}^{n}$ can be defined as follows:

$$\hat{f}(x) = \underset{f}{\operatorname{argmin}} \sum_{i=1}^{n} (y_i - f(x_i))^2$$
$$subject\ to \quad x_i \succcurlyeq x_j \Rightarrow f(x_i) \geq f(x_j); \quad \forall i, j; 1 \leq i, j \leq n \qquad (1)$$

where $f(x_i)$ is the approximation of y_i; and $\succcurlyeq$ is the preorder relation that is reflexive and transitive on input space X. Monotonically decreasing approximation can be defined by replacing the constraint as $x_i \succcurlyeq x_j \Rightarrow f(x_i) \leq f(x_j)$ in (1).

Several algorithms exist for solving the problem defined in (1). One of the earliest algorithms is the pool adjacent violator algorithm (PAVA), based on which a lot of research has been reported [3]. These are well summarized in [4,5]. However, these methods are not applicable when the input space is d dimensional i.e. $X \subseteq \mathbb{R}^d$, because they assume that the input is a sequence i.e. $x_i < x_{i+1}$, which cannot be generalized to d dimensions.

Also, domain knowledge is not always perfect. In the credit rating example, the chances of getting a loan may decrease with past record of defaulting despite higher income. In house pricing example, a flat with one room in a posh area may be costlier than a house with three rooms in a rural area. To address this, we use the notion of soft monotonic regression. Soft monotonicity balances between the best fit $(y = f(x))$ and the monotonic fit $(y = \hat{f}(x))$. This can be formally represented as,

International Conference on Management of Data
COMAD 2016, Pune, India, March 11-13, 2016

$$\hat{f}(x)_\lambda = \underset{f}{\operatorname{argmin}} \left(\begin{array}{l} \sum_{i=1}^{n} \left(y_i - f(x_i)\right)^2 + \lambda * \\ \sum_{i=1}^{n} \sum_{j=i}^{n} \mathcal{K}\left(x_i, x_j, f(x_i), f(x_j)\right) \end{array} \right) \quad (2)$$

$$where\ \mathcal{K}\left(x_i, x_j, f(x_i), f(x_j)\right) = \begin{cases} 0 & \begin{array}{l} if\ monotonically\ increasing\ and \\ \begin{cases} x_i \geq x_j\ and\ f(x_i) \geq f(x_j)\ Or \\ x_j \geq x_i\ and\ f(x_j) \geq f(x_i) \end{cases} \end{array} \\ 0 & \begin{array}{l} if\ montonically\ decreasing\ and \\ \begin{cases} x_i \geq x_j\ and\ f(x_i) \leq f(x_j)\ Or \\ x_j \geq x_i\ and\ f(x_j) \leq f(x_i) \end{cases} \end{array} \\ 1 & otherwise \end{cases}$$

Here, λ is the penalty parameter which penalizes the pair that violates the monotonicity constraint. Setting λ to 0 yields the best fit solution. As $\lambda \to \infty$, the solution approaches strictly monotonic for the problem defined in (1). This kind of problem has recently been addressed in [6] using a modified version of PAVA. However, in line with the original version, it too focuses on one dimensional input data represented as a sequence.

In this paper, we propose a solution for soft monotonic regression for multidimensional input i.e. $\mathrm{X} \subseteq \mathbb{R}^d$. More precisely, we propose an extension to SVR called monotonic constraint support vector regression (MC-SVR) and present its formulation. The proposed method has been validated with numerical experiments. The capability of MC-SVR to impose different monotonic constraints (increasing, decreasing, or none) at different input variables is studied. The effect of softness of the constraint is discussed. Results pertaining to the extrapolation capability of MC-SVR, and generalization capability over multiple input dimensions are also presented.

2. Monotonic Constraint Support Vector Regression

Support vector regression (SVR) basically takes the form [7,8]

$$\begin{array}{l} f(\mathrm{x}) = \langle \mathrm{w}, \mathrm{x} \rangle + b \\ with \quad \mathrm{w}, \mathrm{x} \in \mathbb{R}^d\ and\ b \in \mathbb{R} \end{array} \quad (3)$$

where $\langle .,. \rangle$ denotes the dot product. w is determined by solving the following convex optimization problem,

$$\min_{w,b,\xi_i,\widehat{\xi_\iota}} \left(\frac{1}{2}\|\mathrm{w}\|^2 + C \sum_{i=1}^{n} \left(\xi_i + \widehat{\xi_\iota}\right) \right)$$
$$sub.to \begin{cases} (\langle \mathrm{w}, \mathrm{x}_i \rangle + b) - y_i & \leq & \epsilon + \xi_i;\ \forall i \\ y_i - (\langle \mathrm{w}, \mathrm{x}_i \rangle + b) & \leq & \epsilon + \widehat{\xi_\iota};\ \forall i \\ \xi_i, \widehat{\xi_\iota} & \geq & 0;\ \forall i \end{cases} \quad (4)$$

where, $\xi_i, \widehat{\xi_\iota}$ are slack variables for *soft margin* to accommodate infeasible optimization that may arise due to noisy input variables.

2.1 Monotonicity Constraints

To incorporate monotonicity in SVR, additional constraints have to be formulated. The monotonicity constraint proposed in this paper is based on the following assumption,

$$\begin{array}{lll} f(\mathrm{x} + \delta\mathrm{x}) + \zeta & \geq f(\mathrm{x}); & \text{which simplifies to,} \\ \langle \mathrm{w}, \mathrm{s} \rangle - \langle \mathrm{w}, \mathrm{x} \rangle & \geq -\zeta & where\ \ \mathrm{s} = \mathrm{x} + \delta\mathrm{x} \end{array} \quad (5)$$

The above equation can be interpreted as: a fraction of addition to the input should lead to a solution that is either increasing or stays the same. It should be noted that δ is a vector $\{\delta\}_{i=1}^{d}$ where each component can be set differently for different input dimensions. If $\delta^{(i)}$ is positive, the function is monotonically increasing in i^{th} dimension; if $\delta^{(i)}$ is negative, it is monotonically decreasing; and if $\delta^{(i)} = 0$, no monotonic constraint is imposed. Since the prior domain knowledge is imperfect, some violation in the constraint is allowed. Variable ζ introduced in (5) accounts for this violation. When ζ is zero, the function is strictly monotonic. The new variable $\mathrm{s} = \mathrm{x} + \delta\mathrm{x}$ ensures that the function f is monotonic around x. s is computed once for entire input data x. It should be noted that we only need to compute the value of s and not $f(s)$.

2.2 Incorporating monotonicity constraints in SVR

The variable ζ in (5) can be incorporated in MC-SVR similar to the slack variables ξ_i in standard SVR optimization as given in (4). Considering the above formulation of constraint to implement the MC-SVR, the optimization problem of SVR given in (4) becomes,

$$\min_{w,b,\xi_i,\widehat{\xi_\iota},\zeta_i} \frac{1}{2}\|\mathrm{w}\|^2 + C \sum_{i=1}^{n} \left(\xi_i + \widehat{\xi_\iota}\right) + D \sum_{i=1}^{n} \zeta_i$$
$$\begin{array}{lll} sub.\ to & & \\ (\langle \mathrm{w}, \mathrm{x}_i \rangle + b) - y_i & \leq & \epsilon + \xi_i, \forall i; \\ y_i - (\langle \mathrm{w}, \mathrm{x}_i \rangle + b) & \leq & \epsilon + \widehat{\xi_\iota}, \forall i; \\ \langle \mathrm{w}, \mathrm{x}_i \rangle - \langle \mathrm{w}, \mathrm{s}_i \rangle & \leq & \zeta_i, \forall i: \mathrm{s}_i = \mathrm{x}_i + \delta\mathrm{x}_i \\ \xi_i, \widehat{\xi_\iota}, \zeta_i & \geq & 0 \end{array} \quad (6)$$

where D is the hardness parameter. It represents the amount of margin allowed to violate the monotonicity constraint. When D is large, the penalty for violating the constraint is high. This results in a strict monotonic solution. Conversely, when $D = 0$, monotonicity is not guaranteed.

The quadratic optimization problem in (6) can be solved by the method of Lagrangian multipliers

similar to the standard SVR [7,8]. The primal Lagrangian for (6) is as given in (7) below:

$$\begin{aligned}\mathcal{L}_p = {} & \frac{1}{2}\|\mathrm{w}\|^2 + C\sum_{i=1}^{n}\left(\xi_i + \widehat{\xi}_\iota\right) + D\sum_{i=1}^{n}\zeta_i \\ & - \sum_{i=1}^{n}\alpha_i(-(\langle \mathrm{w}, \mathrm{x}_i\rangle + b) + y_i + \epsilon + \xi_i) \\ & - \sum_{i=1}^{n}\widehat{\alpha_\iota}\left((\langle \mathrm{w}, \mathrm{x}_i\rangle + b) - y_i + \epsilon + \widehat{\xi}_\iota\right) \\ & - \sum_{i=1}^{n}\beta_i\left(\zeta_i - (\langle \mathrm{w}, \mathrm{x}_i\rangle - \langle \mathrm{w}, \mathrm{s}_i\rangle)\right) \\ & - \sum_{i=1}^{n}\mu_i\xi_i - \sum_{i=1}^{n}\widehat{\mu_\iota}\widehat{\xi}_\iota - \sum_{i=1}^{n}\lambda_i\zeta_i \\ sub.to:\ & \alpha_i, \widehat{\alpha_\iota}, \beta_i, \mu_i, \widehat{\mu_\iota}, \lambda_i \geq 0\end{aligned} \tag{7}$$

Following the saddle point condition, the partial derivatives of $\mathcal{L}_p$ with respect to primal variables $(\mathrm{w}, \mathrm{b}, \xi_i, \widehat{\xi}_\iota, \zeta_i)$ should vanish for optimality. The partial derivatives are,

$$\begin{aligned}\frac{\partial \mathcal{L}_p}{\partial w} &= w - \sum_{i=1}^{n}(\widehat{\alpha_\iota} - \alpha_i)x_i + \sum_{i=1}^{n}\beta_i(x_i - s_i) = 0 \\ \frac{\partial \mathcal{L}_p}{\partial b} &= \sum_{i=1}^{n}(\widehat{\alpha_\iota} - \alpha_i) = 0 \\ \frac{\partial \mathcal{L}_p}{\partial \xi_i} &= C - \alpha_i - \mu_i = 0 \\ \frac{\partial \mathcal{L}_p}{\partial \widehat{\xi}_\iota} &= C - \widehat{\alpha_\iota} - \widehat{\mu_\iota} = 0 \\ \frac{\partial \mathcal{L}_p}{\partial \zeta_i} &= D - \beta_i - \lambda_i = 0\end{aligned} \tag{8}$$

Substituting (8) in (7) yields the dual optimization problem as follows.

$$\begin{aligned}\mathcal{L}_D = \max_{\alpha,\widehat{\alpha},\beta} \Bigg\{ & -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(\widehat{\alpha_\iota} - \alpha_i)\left(\widehat{\alpha_\jmath} - \alpha_j\right)\langle \mathrm{x}_i, \mathrm{x}_j\rangle \\ & -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\beta_i\beta_j\langle \mathrm{x}_i - \mathrm{s}_i, \mathrm{x}_j - \mathrm{s}_j\rangle \\ & +\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(\widehat{\alpha_\iota} - \alpha_i)\beta_j\langle \mathrm{x}_i, \mathrm{x}_j - \mathrm{s}_j\rangle \\ & +\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\beta_i\left(\widehat{\alpha_\jmath} - \alpha_j\right)\langle \mathrm{x}_i - \mathrm{s}_i, \mathrm{x}_j\rangle \\ & +\sum_{i=1}^{n}(\widehat{\alpha_\iota} - \alpha_i)y_i - \epsilon\sum_{i=1}^{n}(\widehat{\alpha_\iota} + \alpha_i) \Bigg\} \\ subject\ to: & \sum_{i=1}^{n}(\widehat{\alpha_\iota} - \alpha_i) = 0; \\ & 0 \leq \widehat{\alpha_\iota}, \alpha_i \leq C\ \ \forall i; \\ & 0 \leq \beta_i \leq D\ \ \forall i\end{aligned} \tag{9}$$

The problem in (9) is convex and hence, it forms a standard quadratic optimization problem. This fact is proved in theorem 1.

In order to represent the dual in standard quadratic form, the dot products in (9) can be rewritten as,

$$\left.\begin{aligned} X_iX_j &= \langle x_i, x_j\rangle; \quad S_iS_j = \langle s_i, s_j\rangle \\ X_iS_j &= \langle x_i, s_j\rangle; \quad S_iX_j = \langle s_i, x_j\rangle = X_jS_i^T \end{aligned}\right\} \forall i, j \tag{10}$$

The quadratic term is no longer a $2n \times 2n$ matrix as in conventional linear ε-insensitive loss SVR. Instead, it is a $3n \times 3n$ matrix composed of variables $\hat{\alpha}, \alpha\ and\ \beta$ as shown below,

$$\min_{\alpha,\widehat{\alpha},\beta}\ \frac{1}{2}[\hat{\alpha}\ \ \alpha\ \ \beta]\ H\ \begin{bmatrix}\hat{\alpha}\\ \alpha\\ \beta\end{bmatrix} + C^T\begin{bmatrix}\hat{\alpha}\\ \alpha\\ \beta\end{bmatrix}$$

$$where,$$

$$H = \begin{bmatrix} XX & -XX & -(XX - XS) \\ -XX & XX & (XX - XS) \\ -(XX - XS)^T & (XX - XS)^T & (XX - SS - (XS + XS^T)) \end{bmatrix} \tag{11}$$

$$C = \left(\epsilon\begin{bmatrix}1\\1\\0\end{bmatrix} - \begin{bmatrix}y\\-y\\0\end{bmatrix}\right)$$

$$subject\ to, \begin{bmatrix}-1\\1\\0\end{bmatrix}^T\begin{bmatrix}\hat{\alpha}\\ \alpha\\ \beta\end{bmatrix} \leq 0;$$

$$0 \leq \hat{\alpha}, \alpha \leq C; 0 \leq \beta \leq D;$$

Theorem 1: *The problem of MC-SVR defined in (9) is convex.*

Proof: For the problem in (9) to be convex, the quadratic term should be necessarily positive semi-definite. The quadratic term in $\mathcal{L}_D$ while minimizing is,

$$\begin{aligned} & \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(\widehat{\alpha_\iota} - \alpha_i)\left(\widehat{\alpha_\jmath} - \alpha_j\right)\langle \mathrm{x}_i, \mathrm{x}_j\rangle \\ & \quad + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\beta_i\beta_j\langle \mathrm{x}_i - \mathrm{s}_i, \mathrm{x}_j - \mathrm{s}_j\rangle \\ & \quad - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(\widehat{\alpha_\iota} - \alpha_i)\beta_j\langle \mathrm{x}_i, \mathrm{x}_j - \mathrm{s}_j\rangle \\ & \quad - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\beta_i\left(\widehat{\alpha_\jmath} - \alpha_j\right)\langle \mathrm{x}_i - \mathrm{s}_i, \mathrm{x}_j\rangle \\ & = \frac{1}{2}\Big\langle \sum_i(\widehat{\alpha_\iota} - \alpha_i)\mathrm{x}_i - \sum_i\beta_i(\mathrm{x}_i - \mathrm{s}_i)\ , \\ & \qquad \sum_j\left(\widehat{\alpha_\jmath} - \alpha_j\right)\mathrm{x}_j - \sum_j\beta_j\left(\mathrm{x}_j - \mathrm{s}_j\right)\Big\rangle \\ & = \frac{1}{2}\left\|\sum_i(\widehat{\alpha_\iota} - \alpha_i)\mathrm{x}_i - \sum_j\beta_j\left(\mathrm{x}_j - \mathrm{s}_j\right)\right\|^2 \\ & \geq 0 \end{aligned}$$

Thus, the quadratic term is positive semi-definite. Hence, (9) is convex.

In order to solve the MC-SVR problem, it is important to find the parameters of interest i.e. w and b. Using the partial derivatives of the primal Lagrangian, variable w can be written as,

$$\mathrm{w} = \sum_{i=1}^{n} (\hat{\alpha_i} - \alpha_i)\mathrm{x}_i - \sum_{i=1}^{n} \beta_i(\mathrm{x}_i - \mathrm{s}_i) \quad (12)$$

It is possible to describe w completely in terms of training data x_i (since, s_i is derived from x_i) even for MC-SVR. Substituting (12) in (3) yields,

$$f(x) = \sum_{i=1}^{n} (\hat{\alpha_i} - \alpha_i)\langle x_i, x\rangle - \sum_{i=1}^{n} \beta_i(\langle x_i, x\rangle - \langle s_i, x\rangle) - b \quad (13)$$

The variable b can be found following the Karush-Kuhn-Tucker (KKT) conditions [8] which state that at optimality, the product of dual variable and the constraint should vanish. Thus, b can be identified as,

$$b = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \left(\sum_{j=1}^{n}(\hat{\alpha_j} - \alpha_j)\langle \mathrm{x}_j, \mathrm{x}_i\rangle - \sum_{j=1}^{n}\beta_j(\langle \mathrm{x}_j, \mathrm{x}_i\rangle - \langle \mathrm{s}_j, \mathrm{x}_i\rangle)\right) - \epsilon\right) \quad (14)$$

It is important to note that in (9) to (14), instead of dot product, a kernel expansion can also be applied, which will lead to non-linear MC-SVR. Standard kernels for SVR (such as, Linear, Polynomial, Gaussian, and so on) can be effectively applied.

The hardness parameter, D, bounds the extra variable β in (9). Hence, when D is sufficiently small, then $\beta \ll \alpha$, and there will not be any significant effect of monotonicity constraint. Conversely when D is sufficiently large, then $\beta \gg \alpha$, and the effect of monotonicity will be dominant. For the sake of brevity, the MC-SVR formulation in (6) uses the same parameter D for all monotonicity constraints. Similarly, it uses the same slack variable ζ_i for all monotonicity constraints. The formulation in (6) can be easily extended to support separate D and ζ_i for each monotonicity constraint. In this case, the parameter D will be a vector. The components of D will specify hardness for each constraint. Similarly, there will be a separate set of slack variables $\zeta_i{}^C$ for each constraint.

2.3 Degree of Monotonicity

To assess the degree of monotonicity, we use Kendall correlation metric [9]. Let $(x_1, y_1),\ (x_2, y_2), \ldots, (x_n, y_n)$ be a set of observations of variables X and Y respectively. Any pair of observations (x_i, y_i) and (x_j, y_j) are said to be *concordant* if the ranks for both elements agree (i.e. $x_i > x_j\ and\ y_i > y_j\ OR\ x_i < x_j\ and\ y_i < y_j$). They are *discordant* if $x_i > x_j\ and\ y_i < y_j\ OR\ x_i < x_j\ and\ y_i > y_j$. The Kendall correlation metric is then defined as,

$$Kendall(X,Y) = \frac{\#concordant\ pairs - \#discordant\ pairs}{\frac{1}{2}n(n-1)} \quad (15)$$

If the target variable is a monotonically increasing function of input then Kendall correlation is 1; while for a monotonically decreasing function, it is -1.

We define *degree of monotonicity* as follows,

$$degree\ of\ monotonicity(X,Y) = \frac{(Kendall(X,Y) * sign + 1)}{2}$$
$$where,\quad sign = \begin{cases} 1 & ;\ Y\ is\ a\ monotonically\ increasing\ \ function\ of\ X \\ -1 & ;\ Y\ is\ a\ monotonically\ decreasing\ \ function\ of\ X \end{cases} \quad (16)$$

This measure is in the range of 0 to 1 and it normalizes Kendall correlation such that for a perfect monotonic function (increasing or decreasing), it reaches the value of 1. We define the *degree of monotonicity of a model* with respect to a pair of variables as the degree of monotonicity that would be expected to be observed between the variables when data is generated randomly from the model. It is related to the hardness parameter D in the following sense:

$D_1 > D_2$ => degree of monotonicity of a model learnt with D_1 is greater than the degree of monotonicity of a model learnt with D_2.

Thus we can increase or decrease the degree of monotonicity of a model by increasing or decreasing D.

Degree of monotonicity is a more intuitive parameter for a domain expert to specify. A value of 1 specifies that the model is completely monotonic; a value of 0.8 specifies that 80% of the data is expected to be monotonic, while 20% may be contrarian, and so on. It is possible to convert degree of monotonicity into an appropriate value of D by writing a wrapper function that internally searches for a value of D that gives the desired degree of monotonicity.

3. Experimental Results

We have carried out two sets of experiments to validate MC-SVR: one set with synthetic models and the other with real-life datasets.

3.1 Experiments with synthetic models

Our objective in using synthetic models is to study how the MC-SVR behaves under various controlled experimental conditions, such as dimensionality of input data, degree of the true function, effect of noise in data, etc.

These experiments use the following scheme: A true model of a chosen degree and dimensionality is constructed. Data is generated from this model. To simulate the effect of noise in observed data, we add random sinusoidal noise (with amplitude drawn from Gaussian distribution) to the generated data. To study the effect of degree of monotonicity, we add an additional term to the true model to introduce regions where monotonicity is violated. We then compare the learnt models with the true model. The output of MC-SVR for constraint violation region is analysed. Softness in MC-SVR is controlled by the *degree* parameter. The effect of softness on the learnt model is also analysed.

In all experiments we first learn the SVR model using a Gaussian kernel and optimize the hyper parameters γ and C using 10-fold cross validation. We then use the same hyper parameters for MC-SVR and set the monotonicity parameters (i.e. monotonicity constraint and degree of monotonicity) as required. Both SVR and MC-SVR are implemented in octave 3.8 using PR_LOQO [10] as the QP solver.

3.1.1 Comparison between SVR and hard MC-SVR (degree of monotonicity =1)

In this experiment we choose a simple linear function with one input. The true model and the data generation model with noise term are as given in (17) below,

$$\begin{aligned} &\text{True model: } y = \text{x} \\ &\text{Data model: } y = x + \mathcal{N}(0.08, 0.04) * \sin(25 * x) \end{aligned} \quad (17)$$

The amplitude of the noise is drawn from a Gaussian distribution with an arbitrary mean 0.08 and standard deviation 0.04. We generated 200 data points in the range 0 to 1. Figure 1 shows the true model as well as the observed data points. As mentioned in section 2.1, the sign $(\delta^{(i)} > 0 \; or \; \delta^{(i)} < 0)$ determines whether the constraint is monotonically increasing or decreasing. As the true function is monotonically increasing in x, we set the monotonicity constraint as $\delta = \{0.1\}$. This corresponds to 10% of the normalized data range, specifying that the monotonicity constraint should hold with data points that are within this range of an input data point.

An SVR with Gaussian kernel is optimized for minimizing the root mean squared error (RMSE). The best parameters found are $\gamma = 2^{-4} \; and \; C = 10$. MC-SVR is learnt using the same hyper-parameters. As the true model is monotonic in the full input range, we set the degree of monotonicity to 1 (hard MC-SVR). The results are shown in Figure 1. It can be observed from the figure that MC-SVR learns a better fit to the true model than SVR. The RMSE of MC-SVR and SVR are 0.0430 and 0.0545 respectively, showing that MC-SVR performs significantly better than SVR (approximately 20% improvement).

Next we study the behaviour of hard MC-SVR for a non-linear function. The true model in this case is a simple quadratic function *viz.* $y = x^2$. Similar to the linear case, the data generation model adds noise terms with amplitude drawn from a Gaussian distribution (with mean = 0.08 and standard deviation = 0.04). 200 data points are generated in the range 0 to 1. Figure 2 shows the true model and the observed data points. As the function is monotonically increasing in x, the monotonicity constraint is set as $\delta = \{0.1\}$. An SVR with Gaussian kernel is optimized to find the best parameter setting ($\gamma = 2^{-3} \; and \; C = 10$). MC-SVR is learnt with the same hyper-parameters. The degree of monotonicity is set to 1 as the true model is monotonic in the full input range. Again, as can be observed from Figure 2, MC-SVR learns a better model compared to SVR. The RMSE of MC-SVR and SVR are 0.0387 and 0.0523 (approximately 26% improvement over SVR).

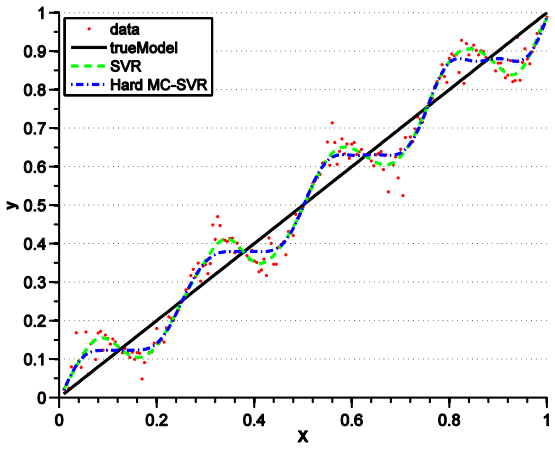

Figure 1: Comparison of SVR and Hard MC-SVR (degree = 1) for 1 dimensional linear function

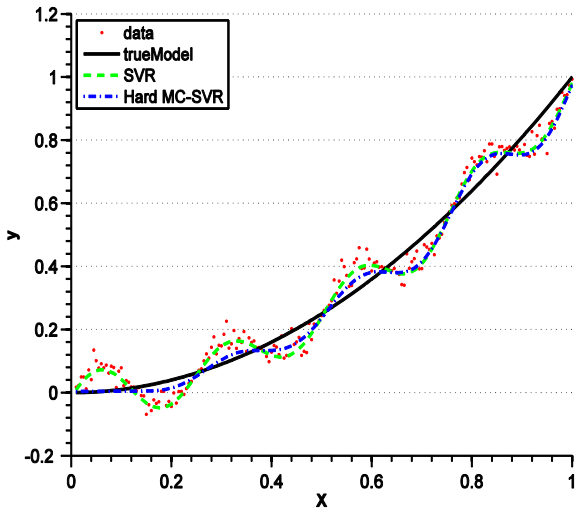

Figure 2: Comparison of SVR and Hard MC-SVR (degree = 1) for 1 dimensional quadratic function

We have also studied the behaviour of hard MC-SVR for multidimensional input. Similar to the one dimensional case, we set up a two dimensional experiment using an arbitrary linear function with two inputs. The true model and the data generation model with noise are as given in (18) below,

$$\begin{aligned} &\text{True model: } z = 0.7 * x - 0.5 * y \\ &\text{Data model: } z = 0.7 * x - 0.5 * y \\ &\qquad + \mathcal{N}(0.05,0.025) * \sin(25 * x) \\ &\qquad + \mathcal{N}(0.05,0.025) * \sin(25 * y) \end{aligned} \tag{18}$$

The data model adds noise to both the input dimensions. We choose 300 data points randomly from a set of 1156 generated data points (a two dimensional grid with spacing 0.03) in the range 0 to 1. The true model in (18) is monotonically increasing in x and decreasing in y. Hence we set the constraint as $\delta = \{0.1, -0.1\}$.

Similar to the one-dimensional case, an SVR with Gaussian kernel is used to minimize the root mean squared error. The best parameters found are $\gamma = 2^{-2}$ and $C = 32768$. MC-SVR is learnt using the same hyper-parameters. The degree of monotonicity is set to 1 as the true model has perfect monotonic behaviour in both dimensions (increasing in the first and decreasing in the second). Once again, it has been observed that MC-SVR produces a better fit to the true model than SVR. MC-SVR achieves a significant improvement in RMSE over SVR – from 0.0607 for SVR to 0.0428 for MC-SVR, an improvement of around 30%. This experiment shows that MC-SVR can learn different monotonicity relations in different dimensions, and the model incorporating these relations performs better than SVR.

The experiments in this section suggests that the difference between MC-SVR and SVR becomes more significant as the complexity (degree or dimensionality) of the problem increases.

3.1.2 Softness capability of MC-SVR

In order to evaluate the softness capability of MC-SVR, we modify the true model of (17) by imposing monotonically decreasing behaviour in input x for approximately 10% of the data. If we think of this data as coming from a real-life process, the monotonically decreasing part represents exceptions to the generally expected behaviour. We show that MC-SVR with an appropriately defined degree of monotonicity (which can be set by a domain expert) has the capacity to model this behaviour and is capable of achieving better accuracy than both SVR and hard MC-SVR.

The true model and the data generation model for this experiment are as follows,

$$\begin{aligned} &\text{True model: } y = \mathrm{x} - 0.4 * e^{-\left(\frac{x-0.5}{0.07}\right)^2} \\ &\text{Data model: } y = \mathrm{x} - 0.4 * e^{-\left(\frac{x-0.5}{0.07}\right)^2} \\ &\qquad + \mathcal{N}(0.08,0.04) * \sin(25 * x) \end{aligned} \tag{19}$$

The exponential term adds monotonically decreasing behaviour around 0.5 to an otherwise monotonically increasing true model. As before, the data model adds noise to the data points before they are observed. We generated 200 data points in the range 0 to 1. Figure 3 shows the true model as well as the observed data points. This function is studied with monotonically increasing constraint i.e. $\delta = \{0.1\}$.

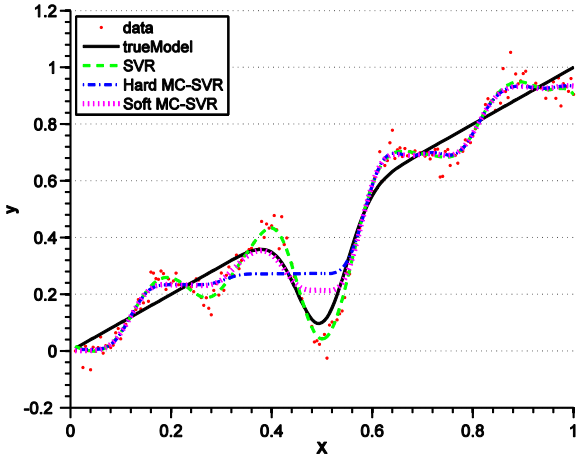

Figure 3: Comparison of Soft MC-SVR with SVR and Hard MC-SVR on linear soft monotonic function

Table 1: RMSE for synthetic 1-dimensional linear data – monotonically increasing function with opposite behavior in approximately 10% data

Function	RMSE
SVR	0.0522
$soft\ MC-SVR\ (degree = 0.9)$	0.0441
$hard\ MC-SVR\ (degree = 1)$	0.0595

Again, an SVR with Gaussian kernel is optimized ($\gamma = 2^{-4}$ and $C = 10$) for minimizing the root mean squared error. MC-SVR is learnt using the same hyper-parameters. The target variable in (19) is a monotonically increasing function of input. However, approximately 10% data in the true model has opposite behaviour. To account for this, we incorporate softness in MC-SVR by specifying the degree of monotonicity as 0.9. We also learn hard MC-SVR (i.e. degree of monotonicity = 1). As observed from Figure 3, soft MC-SVR produces a better fit compared to hard MC-SVR. This is due to the fact that hard MC-SVR does not use the information that 10% of data has opposite behaviour and builds a model that is monotonic in the full input range. The model leant using SVR captures this behaviour. However, it also learns the noise present in the observed data. The model learnt using soft MC-SVR provides a balance between SVR and hard MC-SVR. Table 1 shows the RMSE values obtained for these three models. Soft MC-SVR achieves significantly better RMSE compared to both SVR and hard MC-SVR.

Next we study the softness capability of MC-SVR on a non-linear function. The true model and the data generation model with noise are as given below,

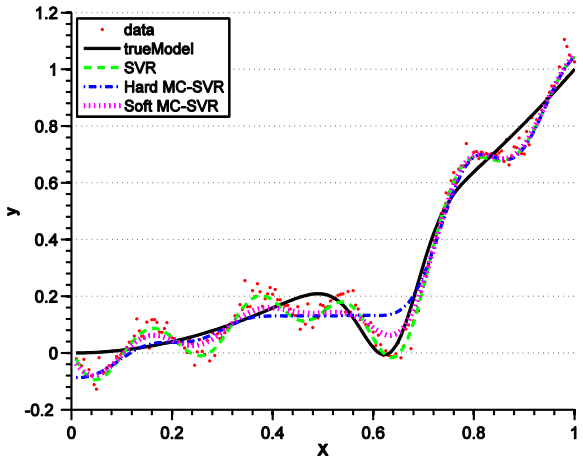

Figure 4: Comparison of Soft MC-SVR with SVR and Hard MC-SVR on quadratic soft monotonic function

Table 2: RMSE for synthetic 1-dimensional quadratic data – monotonically increasing function with opposite behavior in approximately 10% data

Function	RMSE
SVR	0.0535
$soft\ MC-SVR\ (degree = 0.9)$	0.0445
$hard\ MC-SVR\ (degree = 1)$	0.0543

Table 3: RMSE for synthetic 2-dimensional data – monotonically increasing function with opposite behavior in approximately 10% data

Function	RMSE
SVR	0.1746
$soft\ MC-SVR\ (degree = 0.9)$	0.0486
$hard\ MC-SVR\ (degree = 1)$	0.0810

$$\text{True model: } y = \mathrm{x}^2 - 0.4 * e^{-\left(\frac{x^2-0.4}{0.1}\right)^2}$$
$$\text{Data model: } y = \mathrm{x}^2 - 0.4 * e^{-\left(\frac{x^2-0.4}{0.1}\right)^2} + \mathcal{N}(0.08,0.04) * \sin(30 * x) \quad (20)$$

The model adds contrarian behaviour around 0.4 to an otherwise monotonically increasing function. 200 data points are generated in the range 0 to 1. Figure 4 shows the true model and the observed data points. As the function is monotonically increasing in x for the most part, the constraint is set as $\delta = \{0.1\}$.

As before, an SVR with Gaussian kernel is optimized to find the best parameter setting ($\gamma = 2^{-3}\ and\ C = 250$). MC-SVR is also learnt with the same hyper-parameters. The degree of monotonicity is set to 0.9 for soft MC-SVR as approximately 10% of data exhibits the opposite behaviour. Hard MC-SVR is also learnt to compare the effect of softness. Figure 4 shows the results. As can be seen, here also soft MC-SVR produces a better fit than SVR and hard MC-SVR. Table 2 reports the RMSE for the three models. As expected, soft MC-SVR performs significantly better than both SVR and hard MC-SVR.

We also studied the effect of softness on the learnt models for multidimensional input. The true model and the data generation model are as given below,

$$\text{True model: } \mathrm{z} = 0.7 * \mathrm{x} - 0.5 * \mathrm{y} - 0.5 * e^{-\left(\frac{x-0.5}{0.09}\right)^2}$$
$$\text{Data Model: } z = 0.7 * x - 0.5 * y - 0.5 * e^{-\left(\frac{x-0.5}{0.09}\right)^2} + \mathcal{N}(0.05,0.025) * \sin(25 * x) + \mathcal{N}(0.05,0.025) * \sin(25 * y) \quad (21)$$

The model is the same as that used in (18), except that, here, the true model has a monotonically decreasing behaviour in x for approximately 10% of the data.

Here again, an SVR with Gaussian kernel is optimized for minimizing root mean squared error ($\gamma = 2^{-3}\ and\ C = 10$). The same hyper-parameters are then used for MC-SVR. Since approximately 10% of the data exhibits opposite behaviour, we set the degree of monotonicity to 0.9. We also learn hard MC-SVR (degree of monotonicity = 1). Once again, as expected, soft MC-SVR produces a better fit to the true model than SVR and hard MC-SVR. A comparison of RMSE obtained with all these three approaches is shown in Table 3. It can clearly be observed that MC-SVR, with the additional knowledge about degree of monotonicity (as opposed to just the knowledge about monotonicity) outperforms both SVR and hard MC-SVR.

We have also studied the effect of softness on multidimensional non-linear functions using $x^2 - y^2$ as the true model with 10% contrarian behaviour. Once again soft MC-SVR performed significantly better than both SVR and hard MC-SVR.

3.2 Experiments on Real world Datasets

3.2.1 Extrapolation capability of MC-SVR

In this section, global warming dataset has been considered which is first studied in [11]. Recently, Tibshirani *et. al.* developed a nearly monotonic regression using modified PAVA for this dataset [6]. The dataset contains annual temperature anomalies from 1856 to 1999, relative to the 1961-1990 mean. It has 150 data points. It can be observed from Figure 5 that the actual data is monotonically increasing with respect to year, with possible decrease around 1900.

An SVR with Gaussian kernel is optimized ($\gamma = 2^{-2}\ and\ C = 250$) for minimizing the root mean squared error. Figure 5 shows the models produced by SVR, hard MC-SVR (degree of monotonicity = 1) and soft MC-SVR (degree of monotonicity = 0.83). It can be observed that SVR misses the monotonically increasing characteristic and learns a model that has decreasing behaviour at multiple places. On the other hand, hard MC-SVR learns a model that is increasing in the full input range. Soft MC-SVR produces a fit

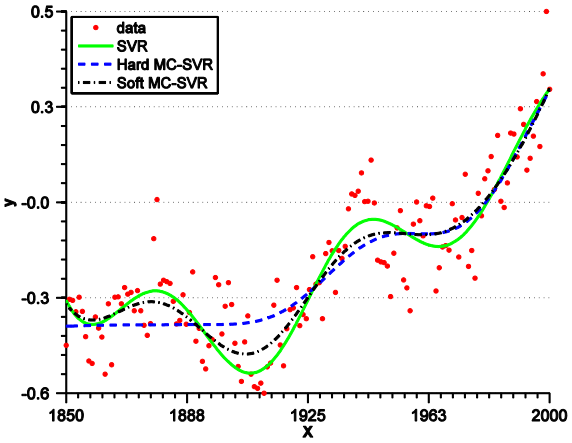

Figure 5: Results of SVR, Soft MC-SVR and Hard MC-SVR on Global Warming dataset

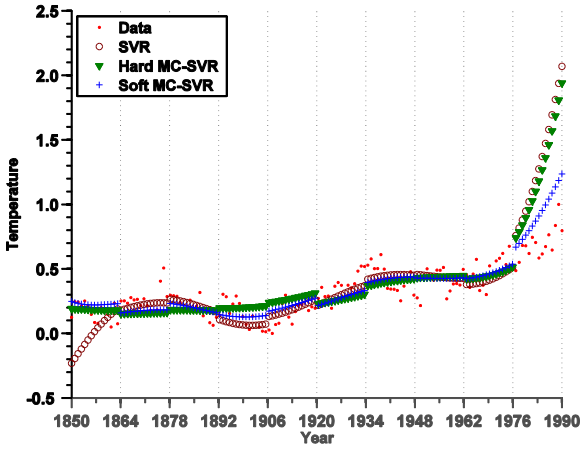

Figure 6: Results on extrapolation capability of SVR, Soft MC-SVR and Hard MC-SVR on Global Warming dataset

that captures the decreasing behaviour around year 1900 but is monotonically increasing otherwise.

In order to study the extrapolation capability of MC-SVR, the dataset is partitioned into 10 equally spaced bins. At a time, 9 bins have been used for training and the remaining for evaluation. This process has been continued for all the 10 bins. It has been observed that soft MC-SVR has produced a significantly better RMSE than standard SVR and full monotonic approximation as given in Table 4. Considering that the RMSE is reported on normalized data, soft MC-SVR improves SVR and hard MC-SVR by 17% and 10% respectively. The results are also portrayed in Figure 6. It can be observed that the extrapolation at the beginning as well as at the end is better for soft MC-SVR than for SVR and hard MC-SVR. Similar results have been observed by changing the number of bins to 5 and 20.

Table 4: RMSE for Global Warming dataset

Function	RMSE
SVR	0.1193
$Soft\ MC-SVR\ (degree = 0.83)$	0.0988
$Hard\ MC-SVR\ (degree = 1)$	0.1101

3.2.2 Generalization capability of MC-SVR over multiple input dimensions

In order to study monotonicity for multidimensional input, Cars dataset has been considered. This dataset has been studied in [12,13]. It contains 4 input attributes of cars viz. displacement, engine output in horsepower, weight and time to accelerate from 0 to 60 mph (acceleration time); the output is the prediction of fuel efficiency in miles per gallon. The first 3 input attributes have monotonically decreasing relation with the output attribute while the last attribute, acceleration time, has monotonically increasing relation. The dataset contains 392 instances.

The quantitative performance on RMSE has been assessed by performing 10-fold cross validation on normalized data. The constraint has been set as monotonically decreasing for *displacement*, *engine output* and *weight* while monotonically increasing for *acceleration time*. Grid search is performed to find the hyper- parameters (γ *and* C) that minimize root mean squared error for SVR. The same hyper-parameters are then used for MC-SVR. The degree of monotonicity is varied to incorporate softness in MC-SVR.

Table 5 reports RMSE for various experiments. The first experiment uses all four input attributes and accordingly the constraint is set as $\delta = \{-0.1, -0.1, -0.1, 0.1\}$. As can be noticed from Table 5, MC-SVR (soft as well as hard constraints) gives better results than SVR (approximately 10% improvement). The second experiment uses a 2-dimensional input – displacement and acceleration time. Once again,

Table 5: Cars dataset: RMSE for SVR and Soft MC-SVR and Hard MC-SVR for multiple input dimensions

Input Variables	SVR	Hard MC-SVR	Soft MC-SVR
Displacement, Engine Output, Weight, Acceleration Time	0.1037	0.0930	0.0929
Displacement, Acceleration time	0.1290	0.1229	0.1228

MC-SVR performs better than SVR (approximately 5% improvement).

The results show that MC-SVR leverages the monotonicity information present in multiple input dimensions. It can also be noticed that the difference between MC-SVR and SVR seems to become more significant as the dimensionality of the problem increases.

5. Related Work

Monotonic function learning has been extensively studied in literature for both classification as well as regression problems. Classification of ordered classes is often assumed to be monotonic with respect to input features. Hence, incorporating monotonicity for ordinal classification was investigated by many researchers. Neural networks is one of the well-studied algorithms in this direction. [14,15] approached monotonicity in neural networks by enforcing constraints on the weights and architecture of the network. An additional error term called monotonicity error, was implemented in neural network to ensure monotonicity in [16]. Makino *et. al.* [17] proposed classification trees with monotonicity constraint for binary classification. This was extended for multiclass decision trees in [2] as quasi-monotone decision trees. An instance based method for ordinal classification called ordinal stochastic dominance learner was proposed in [18]. Decision rules and ensemble of decision rules were also proposed for ordinal classification using monotonicity constraint [19]. Apart from modifying the classical learning algorithms, a method of altering the training data such that they are consistent with monotonicity definition was also proposed in [20]. Many real world applications were investigated using ordinal classification. These include liver disorder diagnosis [16], house pricing [2], internet content filtering [21] and breast cancer diagnosis [22]. The methods discussed above have been reported to perform well for ordinal classification. However, they cannot be directly extended to problems with continuous variables.

In order to implement monotonic regression, a smoothing method based on constraints was proposed in [12]. The capability of smoothing method was demonstrated on a few real world datasets such as cars and onion. In [23], monotonic regression was modelled using Gaussian process. Monotonicity was incorporated by virtual training examples that are generated from derivatives of actual data. This method was shown to improve performance on a synthetic dataset. Incorporating prior knowledge in the form of equality and inequality constraints was extensively discussed in [24,25], where linear programming formulation was used for support vector regression. Quadratic programming based learning with monotonicity of sequential data was studied in [26]. In this method, the input data are one dimensional and are assumed to be in sequence and cannot be used for multidimensional dataset. All the methods discussed above assume perfect prior domain knowledge about monotonicity. They also do not support monotonicity across multiple input dimensions. The solution proposed in this paper addresses these short comings. It provides a way of specifying partial a priori knowledge in the form of degree of monotonicity. The solution can be applied to multiple input dimensions where the monotonicity constraints can be increasing or decreasing in each individual dimension.

6. Conclusion

This paper addresses the problem of learning soft-monotonic regression functions in the presence of imperfect domain knowledge. A novel monotonicity constraint based support vector regression has been proposed. A new hardness parameter, D, is introduced in order to configure the degree of monotonicity required. The working of MC-SVR has been validated using datasets obtained from linear and non-linear synthetic functions. Our experiments on global warming dataset show that MC-SVR with soft constraint has better extrapolation capability than standard SVR. The experiment on Cars dataset shows the generalization capability of MC-SVR over multiple input dimensions. The formulation presented can also be extended to support soft convex constraints [6] and soft positive constraint SVR [13]. Though the results seem encouraging, more theoretical investigations are required to study its generalization properties.

References

1 Chen, Chih-Chuan and Li, Sheng-Tun. Credit Rating with a Monotonicity-Constrained Support Vector Machine Model. *Expert Syst. Appl.*, 41, 16 (2014), 7235-7247.

2 Potharst, Rob and Feelders, Adrianus Johannes. Classification Trees for Problems with Monotonicity Constraints. *ACM SIGKDD Explorations Newsletter*, 4, 1 (2002), 1-10.

3 Brunk, H. D. Maximum Likelihood Estimates of Monotone Parameters. *Ann. Math. Statist.*, 26, 4 (1955), 607-616.

4 Barlow, R. E.and Bartholomew, D., Bremner, J. M., and Brunk, H. D. *Statistical Inference under Order Restrictions; The Theory and Application of Isotonic Regression*. Wiley, New York, 1972.

5 Jan, de Leeuw, Mair, Patrick, and Hornik, Kurt. Isotone Optimization in R: Pool-Adjacent-Violators Algorithm (PAVA) and Active Set Methods. *J stat soft*, 32, 5 (2009), 1-24.

6 Tibshirani, J., Ryan, Hoefling, Holger, and Tibshirani, Robert. Nearly--Isotonic Regression. *Technometrics*, 53, 1 (2011), 54-61.

7 Smola, Alex J and Sch. A Tutorial on Support Vector Regression. *Stat. comput.*, 14, 3 (2004), 199-222.

8 Vapnik, Vladimir Naumovich and Vapnik, Vlamimir. *Statistical Learning Theory*. Wiley New York, 1998.

9 Kendall, Maurice G. A New Measure of Rank Correlation. *Biometrika* (1938), 81-93.

10 Vanderbai, Robert J. LOQO: an interior point code for quadratic programming. *Optimization Methods and Software*, 11, 1 (1999), 451-484.

11 Wu, Wei Biao, Woodroofe, Michael, and Mentz, Graciela. Isotonic Regression: Another Look at the Changepoint Problem. *Biometrika*, 88, 3 (2001), 793-804.

12 Mammen, E., Marron, J. S., Turlach, B. A., and Wand, M. P. A General Projection Framework for Constrained Smoothing. *Statistical Science: a Review Journal*, 16, 3 (2001), 232-248.

13 Ichiro, Takeuchi, Quoc, V. Le, Timothy, D. Sears, and Alexander, J. Smola. Nonparametric Quantile Estimation. *JMLR*, 1231-1264.

14 Wang, S. Neural Network Techniques for Monotonic Nonlinear Models. *Computers & OR*, 21 (1994), 143-154.

15 Daniels, H. and Kamp, B. Application of MLP Networks to Bond Rating and House Pricing. *Neural Comput. Appl.*, 8 (1999), 226-234.

16 Sill, Joseph and Abu-Mostafa, Yaser S. Monotonicity Hints. *Adv. Neural Inf. Process. Syst.* (1997), 634-640.

17 K. Makino, T. Suda, H. Ono and Ibaraki, T. Data Analysis by Positive Decision Trees. *IEICE Transations on Information and Systems*, E82-D (1999), 76-88.

18 Cao-Van, Kim and De Baets, Bernard. Growing decision trees in an ordinal setting. *Int. J..Intell. Syst.*, 18, 7 (2003), 733-750.

19 Dembczy, Kot, and S. Ensemble of Decision Rules for Ordinal Classification with Monotonicity Constraints. In *Rough Sets and Knowledge Technology*. Springer, 2008.

20 Horv, Eckhardt, Alan, Buza, Kriszti, Vojtas, P, and Schmidt-Thieme, Lars. Value-Transformation for Monotone Prediction by Approximating Fuzzy Membership Functions. In *IEEE CINTI 2011* (2011), 367-372.

21 Jacob, Varghese S, Krishnan, Ramayya, and Ryu, Young U. Internet Content Filtering using Isotonic Separation on Content Category Ratings. *ACM TOIT*, 7, 1 (2007), 1.

22 Ryu, Young U, Chandrasekaran, Ramaswamy, and Jacob, Varghese S. Breast Cancer Prediction using the Isotonic Separation Technique. *Eur. J. Oper. Res.*, 181, 2 (2007), 842-854.

23 Riihim and Vehtari, Aki. Gaussian Processes with Monotonicity Information. In *International Conference on Artificial Intelligence and Statistics* (2010), 645-652.

24 Fabien, Lauer and Gerard, Bloch. Incorporating Prior Knowledge in Support Vector Regression. *Mach. Learn.*, 70, 1 (2008), 89-118.

25 Olvi, L. Mangasarian, Jude, W. Shavlik, and Edward, W. Wild. Knowledge--Based Kernel Approximation. *JMLR*, 5 (2004), 1127-1141.

26 Gamarnik, David. Efficient Learning of Monotone Concepts via Quadratic Optimization. *Proceedings of the Eleventh Annual Conference on Computational Learning Theory* (1998.), 134-143.