World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:1, No:10, 2007

Mixtures of Monotone Networks for Prediction

Marina Velikova, Hennie Daniels, and Ad Feelders

Abstract**—** In many data mining applications, it is a priori known that the target function should satisfy certain constraints imposed by, for example, economic theory or a human-decision maker. In this paper we consider partially monotone prediction problems, where the target variable depends monotonically on some of the input variables but *not* on all. We propose a novel method to construct prediction models, where monotone dependences with respect to some of the input variables are preserved by virtue of construction. Our method belongs to the class of mixture models. The basic idea is to convolute monotone neural networks with weight (kernel) functions to make predictions. By using simulation and real case studies, we demonstrate the application of our method. To obtain sound assessment for the performance of our approach, we use standard neural networks with weight decay and partially monotone linear models as benchmark methods for comparison. The results show that our approach outperforms partially monotone linear models in terms of accuracy. Furthermore, the incorporation of partial monotonicity constraints not only leads to models that are in accordance with the decision maker's expertise, but also reduces considerably the model variance in comparison to standard neural networks with weight decay.

Keywords**—** mixture models, monotone neural networks, partially monotone models, partially monotone problems.

I. Introduction

IN many data mining applications, it is a priori known that the target function should satisfy certain constraints imposed by, for example, economic theory or a human-decision maker. In many cases, however, the final model obtained by data mining techniques alone does not meet these constraints. It is required that the algorithms have to be modified (enhanced) to obey the constraints in a strict fashion.

One type of constraint, which is common in many decision problems, is the monotonicity constraint stating that the greater an input is, the greater the output must be, all other inputs being equal. There is a wide range of applications where monotonicity properties hold. Well-known examples include credit loan approval, the dependence of labor wages as a function of age and education, investment decisions, hedonic price models, selection and evaluation tasks ([1], [2]).

Several data mining techniques have been developed, which incorporate monotonicity constraints such as neural networks ([3], [4], [5], [6]), rational cubic interpolation of one-dimensional functions ([7]), decision trees ([8], [9], [10]), etc. However, the main assumption for the implementation of most of these methods is that the function (output) being estimated should be monotone in all inputs (so-called total monotonicity). This in practice, of course, is not always the case.

In this paper we consider partially monotone problems, where we assume that the target variable depends monotonically on some of the input variables but not on all. For example, common sense suggests that the house price has monotone increasing dependence on the number of rooms and the total house area, whereas for the number of floors this dependence does not necessarily hold. Such prior knowledge about monotone relationships can be incorporated as constraints in data mining algorithms in order to improve the accuracy or interpretability of the models derived as well as to reduce their variance on new data.

It is known that non-monotone functions can often be represented as compositions of monotone functions; for example, unimodal probability distribution functions are monotone increasing on the left side of the mode point, and monotone decreasing on the right side ([6]). This implies that first we can construct a number of monotone models corresponding to the monotone regions in the non-monotone function; then we can combine the local monotone models in order to obtain the global model.

The paper is organized as follows. In the next section we introduce notations and definitions related to monotonicity, which are needed for the follow-up discussion. The main contribution of this paper is the approach for partial monotonicity presented in Section IV-A. The approach is based on the convolution of kernel functions and a special type of monotone neural networks, introduced in Section III. In Section IV-B we present the design and the results from simulation studies carried out to test the performance of the proposed approach for partial monotonicity. Section IV-C demonstrates the application of the approach on a real case study of predicting abalone age. Concluding remarks are given in Section V.

II. Notation and definitions

Let $\mathbf{x}$ denote the vector of independent variables, which takes values in a k-dimensional input space, $\mathcal{X}$, and ℓ denotes the dependent variable that takes values in a one-dimensional space, $\mathcal{L}$. We assume that a data set $D = (\mathbf{x}, \ell_{\mathbf{x}})$ of N points in $\mathcal{X} \times \mathcal{L}$ is given.

For monotone problems, we assume that the data are generated by a process with the following properties:

$$\ell_{\mathbf{x}} = f(\mathbf{x}) + \epsilon \tag{1}$$

Manuscript received April 25, 2006.

Marina Velikova is with the Center for Economic Research, Tilburg University, The Netherlands (phone: +31 13 466 8721, fax: +31 13 466 3069, e-mail: M.Velikova@uvt.nl).

Hennie Daniels is with the Center for Economic Research, Tilburg University, The Netherlands and ERIM Institute of Advanced Management Studies, Erasmus University Rotterdam, Rotterdam, The Netherlands (e-mail: daniels@uvt.nl).

Ad Feelders is with the Department of Information and Computing Sciences, Utrecht University, The Netherlands (e-mail: ad@cs.uu.nl).

where f is a function monotone in $\mathbf{x}$, and ϵ is a random variable with zero mean. Monotonicity of f on $\mathbf{x}$ is defined on all independent variables by

$$\mathbf{x}^1 \geq \mathbf{x}^2 \Longrightarrow f(\mathbf{x}^1) \geq f(\mathbf{x}^2) \tag{2}$$

where $\mathbf{x}^1 \geq \mathbf{x}^2$ is a partial ordering on $\mathcal{X}$ defined by $x_i^1 \geq x_i^2$ for $i = 1, 2, ..., k$. The pair $(\mathbf{x}^1, \mathbf{x}^2)$ is called comparable and if the relationship defined in (2) holds, it is also a monotone pair.

Note that even though f is monotone, the data generated by (1) is not necessarily monotone due to the random effect of ϵ.

For partially monotone problems, we have $1 \leq k_m < k$,

$$\begin{aligned} \mathbf{x}^m &= \{x_i \in \mathcal{X} | i = 1, \ldots, k_m\}, \\ \mathbf{x}^{nm} &= \{x_i \in \mathcal{X} | i = k_m + 1, \ldots, k\}. \end{aligned}$$

Furthermore, a data set $D = (\mathbf{x}^m, \mathbf{x}^{nm}, \ell_{\mathbf{x}})$ of N points is generated by

$$\ell_{\mathbf{x}} = f(\mathbf{x}^m, \mathbf{x}^{nm}) + \epsilon, \tag{3}$$

where f is a function monotone in $\mathbf{x}^m$ and ϵ is a random variable with zero mean. Hence, we call $\mathbf{x}^m$ the *set of monotone variables* and $\mathbf{x}^{nm}$ is the *set of non-monotone variables*.

Our objective is to find a smooth approximation $\hat{f}$ of $f(\mathbf{x}^m, \mathbf{x}^{nm})$, such that $\hat{f}$ is monotone in $\mathbf{x}^m$, i.e., $\hat{f}$ is a *partially monotone estimator*. In practice the true function $f(\mathbf{x}^m, \mathbf{x}^{nm})$ is unknown, and therefore we use $\ell_{\mathbf{x}}$ as a close proximity of $f(\mathbf{x}^m, \mathbf{x}^{nm})$ to find $\hat{f}$.

A simple solution is to consider the class of partially monotone linear functions of the form:

$$\begin{aligned} \hat{f} = a_0 + \sum_{i=1}^{m} a_i x_i^m + \sum_{j=m+1}^{k} a_j x_j^{nm} \\ \text{subject to} \quad a_i \geq 0, i = 1, \ldots, k_m. \end{aligned} \tag{4}$$

We expect that the estimate in (4) would produce good fit for simple (e.g., linear) functions; it would, however, give poor approximations for complex functions. Therefore, it is necessary to consider more flexible models for estimating an arbitrary partially monotone function.

In this study, we consider an alternative solution to partially monotone problems based on *mixture models*; these models provide a general framework for generating flexible models by combining simpler functions (components). In particular we look at mixture models of the form

$$\hat{f}(\mathbf{x}^m, \mathbf{x}^{nm}) = \sum_{c=1}^{C} \varphi_c(\mathbf{x}^{nm}) \cdot \hat{f}_c(\mathbf{x}^m) \tag{5}$$

where $\varphi_c(\mathbf{x}^{nm})$ is a positive weight (kernel) function based on $\mathbf{x}^{nm}$, and $\hat{f}_c(\mathbf{x}^m)$ is the output of a three-layer monotone neural network built on $\mathbf{x}^m$. In Section IV-A, we show that the class of functions in (5) has universal approximation capabilities.

III. Sill (monotone) networks

The network considered here is based on the three-layer (two hidden-layer) architecture introduced by Sill in [5]. The input layer is connected to the first hidden layer consisting of a set of linear units, which are combined into several groups, (the number of units in each group is not necessarily the same). Corresponding to each group is a second hidden-layer unit, which computes the maximum over all firstlayer units within the group. The final output unit computes the minimum over all groups.

In formal notation, a Sill network can be represented as follows. Let R denote the number of nodes in the second hidden layer; that is, the number of groups in the first hidden layer, with outputs $g_1, g_2, \ldots, g_R$. Let h_r denote the number of hyperplanes within group $r, r = 1, 2, \ldots, R$. The parameters (weights) of the hyperplanes in r are k-dimensional vectors denoted by $\mathbf{w}_{(r,1)}$, $\mathbf{w}_{(r,2)}$, ..., $\mathbf{w}_{(r,h_r)}$. Then, the output at group r is defined by:

$$g_r(\mathbf{x}) = \max_j \left(\mathbf{w}_{(r,j)} \cdot \mathbf{x} + \theta_{(r,j)} \right), \quad 1 \leq j \leq h_r, \tag{6}$$

where θ is a bias term.

The final output of the network is given by

$$O_{\mathbf{x}} = \min_r g_r(\mathbf{x}), \tag{7}$$

or in classification problems

$$O_{\mathbf{x}} = \min_r \sigma(g_r(\mathbf{x})), \tag{8}$$

where σ is the sigmoid function.

From (7) and (8), it follows that one group and one hyperplane within this group uniquely determine the output of the network for each input vector. Such group and hyperplane are called *active*. In case of ties in the group or network outputs (though this is unlikely, because the outputs are continuous), the choice of the active hyperplane or group is made randomly.

To guarantee that the network output is monotone, all weights for an input to the first hidden layer are constrained to be non-negative (non-positive), if increasing (decreasing) monotonicity is desired for that input. Here, we enforce the parameters in (6) to be non-negative by taking an appropriate transformation such as $w = z^2$, where z is a free parameter.

The Sill network thus described has several advantages. First, computation of the output is simple and fast due to the limited number of linear unit calculations and simple comparison operators performed. In addition, at each iteration of the training process the weights of a single linear unit (the active one) are only modified, which speeds up network's learning. Second, by constraining the coefficients of the linear units it is easy to incorporate domain knowledge in the network. Therefore, monotonicity can be easily imposed by restricting the coefficients to be positive or negative. Finally, for a particular input the output from Sill networks is easy to understand and interpret by the end user as the parameters of the linear units directly reflect the relationships in the data. In [5] it is shown in a case study on bond rating that the three-layer monotone networks perform better than a

linear model and standard neural networks for problems where monotonicity is present in the domain.

IV. Approach for partial monotonicity

A. Description

As stated earlier, our objective is to find a smooth estimation $\hat{f}$ of $f(\mathbf{x}^m, \mathbf{x}^{nm})$ given in (3) such that $\hat{f}$ is monotone in $\mathbf{x}^m$.

An intuitive solution to guarantee that the estimator $\hat{f}$ is smooth and partially monotone is to construct monotone approximations of f based on $\mathbf{x}^m$ for fixed values of $\mathbf{x}^{nm}$, and then to smooth out the resulting estimates by using weight functions (kernels) based on $\mathbf{x}^{nm}$.

We apply this approach to build a class of partially monotone functions in the form of (5). The idea is first to find a number of natural groups with respect to the set of non-monotone variables. Next, for each group by using a Sill network we construct a monotone function based on the set of monotone variables. Finally, we convolute the monotone functions and suitable weight functions (kernels) based on the set of non-monotone variables in order to obtain the overall model.

More formally, in the first step of our proposed approach, we partition the input space with respect to $\mathbf{x}^{nm}$ into a number of disjoint subsets (clusters) by using the so-called agglomerative (merging) type of hierarchical clustering with complete-linkage distance. The appropriate number of clusters is determined automatically in the following way. We first cut off the hierarchy obtained from the clustering procedure at several levels (from two to ten). Then for each of the partitioning outcomes we compute the silhouette value as a measure for the goodness of clustering (ranged from -1 for bad to +1 for good) ([11]). The outcome with the maximal silhouette value determines the final number of clusters. An additional improvement in the clustering procedure is adding weights to the variables in the standard Euclidean distance measure we use. In this way, we take into account the significance of each variable on the dissimilarities between the points and the formation of the clusters, respectively. The weights $\boldsymbol{\alpha}$ are determined a priori by taking the absolute value of the respective coefficients for each non-monotone variable obtained from the linear model fitted to the whole data set $D = (\mathbf{x}^m, \mathbf{x}^{nm}, \ell_{\mathbf{x}})$, and normalizing them to sum up to one. More formally, based on D we fit a linear model

$$\ell_{\mathbf{x}} = a_0 + \sum_{i=1}^{m} a_i x_i^m + \sum_{j=m+1}^{k} a_j x_j^{nm}.$$

Next, we compute

$$\boldsymbol{\alpha} = (\alpha_{m+1}, \ldots, \alpha_k),$$

where

$$\alpha_j = \frac{a_j}{\sum_{m+1}^{k} a_j} \quad \text{for } j = m+1, \ldots, k.$$

Hence, $\sum\boldsymbol{\alpha}= 1$, and α_j's can be considered as weights measuring the impact of each non-monotone variable on the response variable.

The outline of our clustering procedure is given below.

CLUSTER(D,$\boldsymbol{\alpha}$)
  $dist(D,\boldsymbol{\alpha})$:= $N \times N$ dissimilarity matrix containing the Euclidean distances, weighted by $\boldsymbol{\alpha}$, between the points in D
  hCl:= a hierarchical cluster tree based on $dist(D,\boldsymbol{\alpha})$ and the complete-linkage distance
  $sv_{max} := -\infty$
  C := final number of clusters
  for c:=2 to 10 **do**
    $[D_1, \ldots, D_c, \mathbf{x}_1^{nm}, \ldots, \mathbf{x}_c^{nm}]$:= disjoint clusters (subsets of D) with their centroids obtained after cutting off hCl into c clusters
    sv_c:= silhouette value obtained for c clusters
    if $sv_{max} < sv_c$ **then**
      $sv_{max} = sv_c$
      $C = c$
    end if
  end for
return $[D_1, \ldots, D_C, \mathbf{x}_1^{nm}, \ldots, \mathbf{x}_C^{nm}]$

As a result of this partitioning of the original data D, we obtain a number C of subsets $D_1, \ldots, D_C$, where the number of points in the subsets is not necessarily the same. There is no restriction on the minimal number of points in a subset. For each D_c, $c = 1, 2, \ldots, C$, which contains more than one point, the values of the non-monotone variables are fixed to the cluster mean $\mathbf{x}_c^{nm}$. Furthermore, an estimate $\hat{f}(\mathbf{x}^m)$ of f is obtained based only on the values of the monotone variables $\mathbf{x}^m$ for the points belonging to D_c. This is done by using Sill networks, which guarantees that the function approximation is monotone within each subset.

If a cluster with only one point is created (i.e., an outlier in respect to the values of the non-monotone variables is detected), then the cluster mean takes the values of the non-monotone variables for that point, and the function approximation is just the label of the point. The reasoning for not ignoring the one-point clusters is as follows. Suppose we want to predict the label $\ell_{\mathbf{z}}$ of a new point $\mathbf{z}$, which is closer to a one-point cluster than to the others (meaning that the values of the non-monotone variables are similar). Now if $\mathbf{z}$ also has values of the monotone variables that are similar to those of the point in the cluster, then the predicted label is expected to be also close to the label of the point. However if the values of the monotone variables are dissimilar between the points then $\mathbf{z}$ can be considered as a point without an analog in the data (i.e., outlier) but its label can be still predicted by using the function estimations from all the clusters as described below.

In the next step of our approach, for each subset D_c, $c = 1, 2, \ldots, C$, we define

$$\psi_c(\mathbf{x}^{nm}) = \left\| \boldsymbol{\alpha} \left(\mathbf{x}^{nm} - \mathbf{x}_c^{nm} \right) \right\|^{-1},$$

where $\|\cdot\|$ is the Euclidean distance norm weighted by $\boldsymbol{\alpha}$, $\mathbf{x}^{nm} \in D$ are the values of the non-monotone variables and $\mathbf{x}_c^{nm}$ is the mean (centroid) value based on the non-monotone

variables for the points falling in cluster c. By definition $\psi \geq 0$ and it determines the distance of a point $\mathbf{x}$ to the mean $\mathbf{x}_c^{nm}$ of cluster c. By normalizing ψ with

$$\varphi_c(\mathbf{x}^{nm}) = \frac{\psi_c(\mathbf{x}^{nm})}{\sum_{c=1}^{C} \psi_c(\mathbf{x}^{nm})}, \tag{9}$$

we obtain a function $\varphi \geq 0$, for which

$$\sum_{c=1}^{C} \varphi_c(\mathbf{x}^{nm}) = 1, \quad \text{for all } \mathbf{x}^{nm}.$$

Hence, φ can be considered as a weight function or kernel in Nadaraya-Watson form ([12], [13]).

Finally, we convolute φ_c with the corresponding monotone approximations $\hat{f}_c(\mathbf{x}^m)$ for all clusters by

$$\hat{f}(\mathbf{x}^m, \mathbf{x}^{nm}) = \sum_{c=1}^{C} \varphi_c(\mathbf{x}^{nm}) \cdot \hat{f}_c(\mathbf{x}^m)$$

to obtain the final estimate of f.

The outline of the algorithm for partial monotonicity is given in Algorithm 1.

Algorithm 1 Building partially monotone models

`Constr_set` := Construction data
`Test_set` := Test data
$\mathbf{x}^m$:= set of monotone variables from `Constr_set`
$\mathbf{x}^{nm}$:= set of non-monotone variables from `Constr_set`
$\mathbf{x}^m_{Tset}$:= set of monotone variables from `Test_set`
$\mathbf{x}^{nm}_{Tset}$:= set of non-monotone variables from `Test_set`
$[D_1, \ldots, D_C, \mathbf{x}_1^{nm}, \ldots, \mathbf{x}_C^{nm}]$=
=CLUSTER(`Constr_set`($\mathbf{x}^{nm}$),$\boldsymbol{\alpha}$)

for all $\mathbf{x}_{Tset} \in$ `Test_set` **do**
  for $c := 1$ to C **do**
    $MonNet_c(\mathbf{x}^m)$:= Sill network trained on $D_c(\mathbf{x}^m, \ell)$
    $f_c(\mathbf{x}^m_{Tset})$:= output of $MonNet_c(\mathbf{x}^m_{Tset})$
    $\psi_c(\mathbf{x}^{nm}_{Tset}) = \|\boldsymbol{\alpha}\,(\mathbf{x}^{nm}_{Tset} - \mathbf{x}^{nm}_c)\|^{-1}$
    $\varphi_c(\mathbf{x}^{nm}_{Tset}) = \psi_c(\mathbf{x}^{nm}_{Tset}) / \sum_{c=1}^{C} \psi_c(\mathbf{x}^{nm}_{Tset})$
  end for

  $\hat{f}(\mathbf{x}^m_{Tset}, \mathbf{x}^{nm}_{Tset}) = \sum_{c=1}^{C} \varphi_c(\mathbf{x}^{nm}_{Tset}) \cdot \hat{f}_c(\mathbf{x}^m_{Tset})$
end for

In the following theorem we show that our estimator has universal approximation capabilities.

Theorem 4.1: Suppose $D = (\mathbf{x}^m, \mathbf{x}^{nm}, \ell)$ is a data set of N points generated by the following process:

$$\ell_{\mathbf{x}} = f(\mathbf{x}^m, \mathbf{x}^{nm}) + \epsilon,$$

where f is a function monotone in $\mathbf{x}^m$ and ϵ is a random variable with zero mean.

As an approximation of $f(\mathbf{x}^m, \mathbf{x}^{nm})$ we take a partially monotone estimator of the form

$$\hat{f}(\mathbf{x}^m, \mathbf{x}^{nm}) = \sum_{c=1}^{C} \varphi_c(\mathbf{x}^{nm}) \cdot \hat{f}_c(\mathbf{x}^m) \tag{10}$$

where C is a number of clusters (subsets) of D, $\varphi(\mathbf{x}^{nm})$ is a positive weight function in $\mathbf{x}^{nm}$ and $\hat{f}(\mathbf{x}^m)$ is the output of a Sill network built on $\mathbf{x}^m$. Then $\hat{f}(\mathbf{x}^m, \mathbf{x}^{nm})$ is a consistent estimator of $f(\mathbf{x}^m, \mathbf{x}^{nm})$ for $N \to \infty$ and sufficiently many clusters C.

The proof is given in Appendix I.

B. Simulation studies

In this section, we present the results from the simulation studies designed to test the effectiveness of our approach for partial monotonicity. We generate an artificial data set D based on a set of independent variables and a continuous dependent variable computed by applying a function that is monotone only on a subset of the independent variables and non-monotone in the others.

Now based on D thus generated we want to build a model for predicting the dependent variable. Since the problem is partially monotone, we apply our approach for partial monotonicity as an appropriate method for estimation.

To obtain sound assessment for the performance of our approach, we use standard neural networks with weight decay and partially monotone linear models in the form of (4) as benchmark methods for comparison. The standard neural networks consist of one input layer, one hidden layer and one continuous output. In the hidden layer the activation function is sigmoid, whereas in the output it is linear. In addition, the weight decay is used as a regularization method to prevent the networks from overfitting. This is done by adding to the network's error the term $\lambda \sum_{ij} w_{ij}^2$ to penalize large weights, where λ is the weight decay parameter.

Given that our target function is continuous, the comparison between our approach and the benchmark methods is based on the mean-squared error (MSE) as a measure for the quality of estimation. In addition, as the data generating process (true function $f(\mathbf{x})$) is known, we use the bias-variance decomposition of MSE to gain more insight into the performance of the methods used in the simulation studies. As shown in [14], MSE can be decomposed into three components: squared bias, variance and irreducible error (variance of the noise term ϵ). Since the last component is independent of the model constructed and does not affect the comparative study, it is omitted from the computations of the MSE. Thus, in our simulations for each estimation $\hat{f}_{\mathbf{x}|M_D}$ based on method M_D applied on a data set D, MSE is computed by

$$\text{MSE} = \text{Bias}^2 + \text{Variance},$$

where

$$\text{Bias}^2 = (f(\mathbf{x}) - \mathcal{E}_D[\hat{f}_{\mathbf{x}|M_D}])^2,$$

and

$$\text{Variance} = \mathcal{E}_D[(\hat{f}_{\mathbf{x}|M_D} - \mathcal{E}_D[\hat{f}_{\mathbf{x}|M_D}])^2].$$

Furthermore, to obtain more complete performance analysis, the experiments with our approach for partial monotonicity and neural networks with weight decay are conducted by

TABLE I
FACTORS WITH THEIR VALUES USED IN THE SIMULATION EXPERIMENTS ON PARTIALLY MONOTONE PROBLEMS

Approach for partial monotonicity					**Neural networks with weight decay**				
Factors		**Levels (values)**			**Factors**		**Levels (values)**		
		1	**2**	**3**			**1**	**2**	**3**
1	# points in data	50	150	250	**1**	# points in data	50	150	250
2	Noise level (σ_ϵ^2)	0.01	0.5	2	**2**	Noise level (σ_ϵ^2)	0.01	0.5	2
3	# groups in Sill net	2	3	4	**3**	# hidden neurons	3	9	15
4	# planes in Sill net	2	3	4	**4**	Weight decay	0.000001	0.00001	0.0001

using several factors with different values for comparison; see Table I.

All possible combinations of four three-value factors require the experiment with each method to run 81 times (3^4). To reduce the effort and experimental cost in the simulations, we use the so-called *fractional factorial design* ([15]), where only a certain number of combinations of factor values are taken to carry out the experiments. This is done in a systematic way by combining each value of each factor only once with each level of the other factors. In our case the fractional design requires only nine runs (trials) with each method.

For each run we generate a collection of 100 data samples following the data generating procedure described in the experimental set-up. For computational convenience the values of the independent variables in each set are fixed whereas the value of the dependent variable varies across different data samples. The approach for partial monotonicity, neural networks with weight decay and partially monotone linear models are applied on the same collections of data samples.

From the experiments with each method we obtain nine estimations of the two measures–squared bias and variance of the models. Next, these results are used to compute the expected value $\mathcal{E}(\Theta_{ijkl})$ of each measure Θ for all possible combinations of factor values (i, j, k, l), where i, j, k and l range from one to three. As described in [15], this can be done by fitting the exponential model

$$\mathcal{E}(\Theta_{ijkl}) = \exp\big(\mu + (\mu_i^1 - \mu) + (\mu_j^2 - \mu) + \\ + (\mu_k^3 - \mu) + (\mu_l^4 - \mu)\big), \qquad (11)$$

where μ is the total mean computed over all nine estimations, μ_i^1, μ_j^2, μ_k^3 and μ_l^4 are the means for each factor value; the exponential fit guarantees that the estimated $\mathcal{E}(\Theta_{ijkl})$ is positive. For example, for the combination of factor values (50 data points, $\sigma_\epsilon^2 = 0.5$, four groups, three planes), i.e., $(i = 1, j = 2, k = 3, l = 2)$ the approach for partial monotonicity has not been run. To compute $\mathcal{E}(\Theta_{1232})$, we use the corresponding means μ_1^1, μ_2^2, μ_3^3 and μ_2^4 in (11).

Next, we compute MSE by summing up the corresponding estimations of the squared bias and variance for all possible combinations of factor values.

Finally, as there are two factors (number of data points and noise level), which are the same in the experiments, we want to compare the performance of the methods for all combinations of values (i, j) (in total nine) of these two factors. For this purpose, within each (i, j), out of all nine value combinations we take the minimum estimated value with corresponding variance of MSE over the other two factors.

To draw more general conclusions from our simulation study, we conduct two types of experiments, which are described below.

a) Experiment–1: First, two vectors of N values, x^m and x^{nm}, are generated independently from each other. The values of vector x^m are drawn from the uniform distribution on [0,1]. The vector x^{nm} is composition of two sub-vectors each of size $N/2$ points, which are drawn from two normal (Gaussian) distributions: $\mathcal{N}(0.02, 0.05)$ and $\mathcal{N}(0.08, 0.05)$. Finally, we compute the values of a third vector ℓ by applying a monotone function on x^m and a non-monotone function on x^{nm} plus a random perturbation $\epsilon \sim \mathcal{N}(0, \sigma^2)$:

$$\ell = 3 + \sin\left(\frac{\pi}{2} x^m\right)\big(2 + \sin(2\pi x^{nm})\big) + \epsilon. \qquad (12)$$

Hence, we can consider x^m and x^{nm} as the independent variables and ℓ as the dependent variable in a data set $D = (x^m, x^{nm}, \ell)$ of N points.

On the data thus generated, we apply the approach for partial monotonicity (PartMon), neural networks with weight decay (NNet) and partially monotone linear models (PMonLin). The results are summarized in Table II.

We can draw several conclusions based on the results obtained in this experiment:

- Given the universal approximation capabilities of standard neural networks, it is not surprising that they achieve closer approximations (i.e., lower squared bias) than our approach for partial monotonicity and partially linear monotone models.
- However, the flexible nature of neural networks and the random initialization of the network weights lead to higher variances across different runs compared to the other two approaches; this finding is especially valid for small or very noisy data sets.
- The flexibility of the mixture modeling employed by our approach for partial monotonicity allows our method to make considerably better approximations (smaller squared bias) than the partially linear monotone models across data sets with different number of points and noise levels; this leads, however, to higher variances of the models.
- In terms of minimum MSE standard neural networks considerably outperform both partially monotone models for data sets with small noise level ($\sigma_\epsilon^2 = 0.01$). For

TABLE II
MINIMUM MSE OBTAINED BY THE THREE METHODS IN EXPERIMENT–1

Method	50 points			150 points			250 points		
	$\sigma_\epsilon^2 = 0.01$	$\sigma_\epsilon^2 = 0.5$	$\sigma_\epsilon^2 = 2$	$\sigma_\epsilon^2 = 0.01$	$\sigma_\epsilon^2 = 0.5$	$\sigma_\epsilon^2 = 2$	$\sigma_\epsilon^2 = 0.01$	$\sigma_\epsilon^2 = 0.5$	$\sigma_\epsilon^2 = 2$
SQUARED BIAS									
PartMon	0.0672	0.0764	0.1090	0.0241	0.0275	0.0392	0.0352	0.0401	0.0572
NNet	0.0065	0.0102	0.0171	0.0047	0.0074	0.0124	0.0042	0.0066	0.0111
PMonLin	0.1324	0.1326	0.1331	0.0952	0.0953	0.0957	0.1170	0.1172	0.1176
VARIANCE									
PartMon	0.0035	0.0604	0.2199	0.0024	0.0403	0.1466	0.0008	0.0141	0.0513
NNet	0.0087	0.1473	0.6510	0.0024	0.0404	0.1783	0.0016	0.0278	0.1228
PMonLin	0.0006	0.0245	0.0918	0.0002	0.0102	0.0381	0.0001	0.0053	0.0197
MINIMUM MSE									
PartMon	0.0707	0.1368	0.3289	0.0265	0.0677	0.1858	0.0361	0.0542	0.1085
NNet	0.0152	0.1575	0.6680	0.0071	0.0477	0.1907	0.0059	0.0344	0.1339
PMonLin	0.1330	0.1571	0.2249	0.0954	0.1055	0.1337	0.1171	0.1224	0.1373

TABLE III
ARCHITECTURES OF SILL NETWORKS AND STANDARD NEURAL NETWORKS FOR WHICH THE MINIMUM MSE IS OBTAINED BY THE MODELS IN EXPERIMENT–1

Approach for partial monotonicity (groups × planes)				Neural networks with weight decay (hidden nodes–weight decay)			
Noise level	Number of points			Noise level	Number of points		
	50	150	250		50	150	250
$\sigma_\epsilon^2 = 0.01$	3×2	3×2	3×2	$\sigma_\epsilon^2 = 0.01$	9–0.000001	9–0.000001	9–0.000001
$\sigma_\epsilon^2 = 0.5$	3×4	3×4	3×4	$\sigma_\epsilon^2 = 0.5$	3–0.000001	3–0.000001	3–0.000001
$\sigma_\epsilon^2 = 2$	3×4	3×4	3×4	$\sigma_\epsilon^2 = 2$	3–0.000001	3–0.000001	3–0.000001

larger data sets ($N = 150$ or $N = 250$) and reasonable noise level ($\sigma_\epsilon^2 = 0.5$), neural networks produce slightly lower MSE than our approach for partial monotonicity.

- For very noisy data sets with insufficient number of points ($N < 250$), the minimum MSE is achieved by the partially monotone linear models; this is indication that the other two flexible models overfit the data set (by fitting the noise inherent in it). However, for very noisy data sets ($\sigma_\epsilon^2 = 2$) with more data points the lowest MSE is obtained by our approach for partial monotonicity, which indicates that the true relationship between the dependent and independent variables can be captured with sufficient number of points.
- Standard neural networks perform remarkably bad on small and very noisy data sets–for data sets with $N = 50$ their minimum MSE is bigger almost twice than that obtained from our method and three times than that obtained from the partially monotone linear models; this result can be explained by the fact that standard neural networks do not use in the modeling process any prior knowledge about the partial monotonicity of the true function.

In addition to the accuracy, we also checked the architectures of Sill and standard networks for which the minimum MSE is obtained; see Table III.

The results show that for all data sets our approach for partial monotonicity reaches the minimum MSE with Sill networks with three groups; for noisier data, however, it requires more hyperplanes to find a close approximation. In contrast, standard neural networks with nine hidden nodes lead to the minimum MSE for data sets with small level of noise, whereas for noisier data three hidden nodes are enough.

Finally, we checked the number of clusters found by our approach for partial monotonicity at each run. Given that the non-monotone variable has been generated by two normal distributions it is expected the approach to find them. It turned out that our approach detects two clusters in all runs with data sets with 50 and 250 points irrespective of the noise level. For data sets with 150 points, the approach detects ten clusters with respect to the non-monotone variable. A possible explanation could be that once generated the non-monotone variable is fixed for a data set with a certain number of points; so, it is very likely that the clustering procedure applied in our approach would tend to find the same number of clusters across the runs.

b) Experiment–2: For our second experiment we first generate five vectors of N points as follows:

- x^1, x^2, x^3 are drawn from the uniform distribution on [0,1],
- x^4 is a composition of two sub-vectors each of size $N/2$ points, which are drawn from two normal (Gaussian) distributions: $\mathcal{N}(0.02, 0.05)$ and $\mathcal{N}(0.08, 0.05)$, and
- x^5 is drawn from the normal distribution with mean 0.5 and variance 0.1.

TABLE IV
MINIMUM MSE OBTAINED BY THE THREE METHODS IN EXPERIMENT–2

Method	50 points			150 points			250 points		
	$\sigma_\epsilon^2 = 0.01$	$\sigma_\epsilon^2 = 0.5$	$\sigma_\epsilon^2 = 2$	$\sigma_\epsilon^2 = 0.01$	$\sigma_\epsilon^2 = 0.5$	$\sigma_\epsilon^2 = 2$	$\sigma_\epsilon^2 = 0.01$	$\sigma_\epsilon^2 = 0.5$	$\sigma_\epsilon^2 = 2$
SQUARED BIAS									
PartMon	0.0288	0.0705	0.1290	0.0164	0.0400	0.0731	0.0106	0.0260	0.0476
NNet	0.0094	0.0113	0.0236	0.0037	0.0103	0.0214	0.0036	0.0100	0.0209
PMonLin	0.1282	0.1293	0.1301	0.0841	0.0846	0.0853	0.0799	0.0804	0.0811
VARIANCE									
PartMon	0.0249	0.1369	0.4641	0.0071	0.0389	0.1320	0.0045	0.0249	0.0845
NNet	0.0093	0.2207	1.0666	0.0059	0.0858	0.4149	0.0033	0.0480	0.2320
PMonLin	0.0013	0.0572	0.2378	0.0004	0.0187	0.0771	0.0003	0.0114	0.0468
MINIMUM MSE									
PartMon	0.0537	0.2074	0.5931	0.0235	0.0789	0.2051	0.0151	0.0509	0.1321
NNet	0.0187	0.2320	1.0902	0.0096	0.0961	0.4363	0.0069	0.0580	0.2529
PMonLin	0.1295	0.1865	0.3679	0.0845	0.1033	0.1624	0.0802	0.0918	0.1279

Next, we generate a vector ℓ by

$$\ell = 3 + \sin\left(\frac{\pi}{2}x^1x^2x^3\right)\left(2 + \sin(2\pi x^4x^5)\right) + \epsilon. \qquad (13)$$

Hence, $\mathbf{x}^m = \{x^1, x^2, x^3\}$, $\mathbf{x}^{nm} = \{x^4, x^5\}$, and $D = (\mathbf{x}^m, \mathbf{x}^{nm}, \ell)$ is a data set of N points.

Analogously to Experiment–1, we apply our approach for partial monotonicity, standard neural networks with weight decay, and partially monotone linear models on 100 generated samples in the form of D to predict ℓ. We compare again the three methods by using the same factors with their levels in Table I. A summary of the results from our second experiment is given in Table IV.

The following more general conclusions can be drawn from the results obtained from our second experiment:

- Neural networks lead to more accurate models in terms of smaller MSE for data sets with a low level of noise compared to the partially monotone models. Their variance and thus MSE, however, increases considerably for nosier data sets; although the weight decay has been used as a regularization method, these results clearly indicate overfitting–a typical problem often encountered in the application of neural networks. The incorporation of monotonicity constraints in the partially monotone models helps to overcome this problem and to capture the true relationships in the target function.
- For considerably noisy data sets, especially with fewer data obseravtions, partially monotone linear models tend to have smaller MSE; this is due to the fact that despite their higher bias, they have much smaller variance across different runs compared to the more flexible counterpart methods.
- Our approach for partial monotonicity leads to models that are more accurate and with low variance for data sets with a moderate noise level and a larger number of data points.

The architectures of Sill and standard neural networks for which the minimum MSEs are obtained are reported in Table V. On the one hand, the results show that Sill networks with three groups and three hyperplanes achieve the best performance in this experiment irrespective of the noise level and the size of the data sets. On the other hand, the architectures of standard neural networks for which lowest errors are achieved vary considerably among different types of data sets. This finding indicates that for a particular data set our approach is less dependent on the network's architecture to obtain good approximation results, in contrast to the standard neural networks which are more sensitive to the noise level, the size of the data and the number of independent variables.

Finally, we check again the number of clusters detected in our approach across the runs; see Table VI. In contrast to Experiment–1, here for the different types of data sets within the 100 runs, our approach finds different number of clusters. This can be explained by the fact that in our second experiment we have two non-monotone variables drawn from different distributions. Hence, the noise level and the size of the data set would have an effect on the clustering procedure for determining the right number of data subsets. In general, given our data generating procedure, we would expect that our approach would find two clusters; their centroids would be vectors containing two values corresponding to the means used to generate the non-monotone variables. The results show that for all types of data sets, in majority of runs our approach detects indeed two clusters; the percentage of detected two clusters is above 50%, except for noisy data with 150 observations. Not surpisingly, our approach is more capable to find the right number of two clusters for less noisy data sets, irrespective of the number of observations. Of course larger data provides more information and as the results show the approach tends to detect correctly the number of clusters: in more than 80% of the runs it finds two or at most three clusters for data sets with 250 observations with different noise levels.

The results from both simulation studies indicate that there is no method that is superior in all the cases. Depending on the size of and the noise inherent in the data set, any of the three methods can achieve the best performance. In practice, of course, the data generating process is unknown, so we cannot determine beforehand which is the most appropriate method

TABLE V
ARCHITECTURES OF SILL NETWORKS AND STANDARD NEURAL NETWORKS FOR WHICH THE MINIMUM MSE IS OBTAINED BY THE MODELS IN EXPERIMENT–2

Approach for partial monotonicity (groups × planes)				Neural networks with weight decay (hidden nodes–weight decay)			
Noise level	Number of points			Noise level	Number of points		
	50	150	250		50	150	250
$\sigma_\varepsilon^2 = 0.01$	3 × 3	3 × 3	3 × 3	$\sigma_\varepsilon^2 = 0.01$	3–0.000001	9–0.000001	9–0.000001
$\sigma_\varepsilon^2 = 0.5$	3 × 3	3 × 3	3 × 3	$\sigma_\varepsilon^2 = 0.5$	15–0.00001	15–0.00001	15–0.00001
$\sigma_\varepsilon^2 = 2$	3 × 3	3 × 3	3 × 3	$\sigma_\varepsilon^2 = 2$	3–0.00001	3–0.00001	3–0.00001

TABLE VI
NUMBER OF CLUSTERS FOUND BY THE APPROACH FOR PARTIAL MONOTONICITY AT EACH RUN

	Number of points		
	50	150	250
Noise level	Number of clusters – Percentage from 100 runs		
$\sigma_\varepsilon^2 = 0.01$	2 - 68 % 7 - 1 % 8 - 3 % 9 - 9 % 10 - 19 %	2 - 89 % 4 - 1 % 5 - 1 % 8 - 5 % 10 - 4 %	2 - 74 % 3 - 18 % 5 - 1 % 7 - 5 % 9 - 1 % 10 - 1 %
$\sigma_\varepsilon^2 = 0.5$	2 - 55 % 4 - 1 % 5 - 1 % 7 - 2 % 8 - 1 % 9 - 11 % 10 - 29 %	2 - 54 % 4 - 7 % 5 - 6 % 6 - 4 % 7 - 4 % 8 - 2 % 9 - 9 % 10 - 14 %	2 - 59 % 3 - 21 % 4 - 1 % 5 - 6 % 6 - 4 % 7 - 4 % 8 - 1 % 9 - 1 % 10 - 3 %
$\sigma_\varepsilon^2 = 2$	2 - 68 % 7 - 1 % 9 - 9 % 10 - 22 %	2 - 44 % 4 - 5 % 5 - 7 % 6 - 1 % 7 - 8 % 8 - 8 % 9 - 18 % 10 - 9 %	2 - 71 % 3 - 13 % 4 - 5 % 5 - 1 % 6 - 4 % 7 - 2 % 9 - 3 % 10 - 1 %

to model the data. We expect, however, that in real cases our approach for partial monotonicity and standard neural networks would outperform partially monotone linear models as the former are more flexible. Furthermore, due to the monotonicity constraints our approach would lead to more stable models than the models derived from the unconstrained standard neural networks. This is confirmed by the case study presented in the next section.

C. Case study on abalone age prediction

In contrast to the regression problem of house pricing described in [16], here we present the results obtained from the application of the approach for partial monotonicity on a classification problem of predicting abalone age.

The abalone shellfish data set is publicly available at the UCI Repository of machine learning databases ([17]); it has been used as a benchmark to which various machine learning techniques have been applied in earlier studies ([18], [19], [20]). The data have been originally collected by an agency in the Australian state of Tasmania for ongoing research purposes ([21]). The objective is to predict the age of abalone shellfish based on eight physical measurements. Determining the age of abalone in the laboratory is time- and labor-consuming process–it requires cutting the shell through the cone, staining it and counting the number of rings through a microscope. Therefore, faster prediction can be done based on physical abalone measurements, which are easily obtained. In the current data set there are eight measurements (attributes); see Table VII. We transformed the nominal-valued sex attribute into continuous-valued by assigning the values of 0.1 for infant, 0.2 for male, and 0.3 for female. We also apply a simple transformation on the attributes WHOLE WEIGHT and SHUCKED WEIGHT to guarantee that all inputs are in the range [0, 1].

TABLE VII
DEFINITION OF THE VARIABLES FOR THE ABALONE DATA

Symbol	Definition
SEX	Male, Female, and Infant
LENGTH	Longest shell measurement (mm)
DIAMETER	Perpendicular to length (mm)
HEIGHT	With meat in shell (mm)
WHOLE WEIGHT	Whole abalone (grams)
SHUCKED WEIGHT	Weight of meat (grams)
VISCERA WEIGHT	Gut weight after bleeding (grams)
SHELL WEIGHT	After being dried (grams)

TABLE VIII
DEGREE OF MONOTONICITY OF THE ABALONE DATA

Removed variable(s)	Comparable pairs	DgrMon
SEX	33708	0.9057
SEX, WHOLE WEIGHT	33715	0.9057
SEX, SHUCKED WEIGHT	34683	0.9069
SEX, WHOLE WEIGHT, SHUCKED WEIGHT	34789	0.9070

The dependent variable needed to be predicted is the number of rings (age is easily computed by adding 1.5 to the number of rings); it has 28 values, ranging from 1 to 29 (28 is missing). Thus, these data can be treated as a regression or a classification problem. In earlier studies, the response variable has been discretized into three classes (age-groups): 1-8, 9-10, 11 or more. Here we adopt the same transformation procedure, and consider the abalone age prediction as a classification problem.

The original data consist of 4 177 observations (no missing attribute values). In our study, we take a random sample of 292 observations, which is 7% of the full data set. The sample is taken such that infants, males, and females, as well as the three classes are represented in the same proportions like in the full data.

In the data thus constructed, we expect that the age of abalone depends monotonically on some of the measurements, but not on all. For example, if we consider the sex attribute, the discrimination between males and females is not expected to be monotone with the age. Furthermore, with the abalone maturity, the abalone age may not necessarily have monotone relationships with some of the other physical measurements. To check for which attributes monotonicity with the age holds, we conduct a test. This is done by using a measure for the degree of monotonicity (DgrMon) of data, namely the fraction of monotone pairs of all comparable pairs in the data. Although the values assigned to the attribute SEX are numerical they do not imply any ordering; so does not make sense to use this attribute in the test for monotone relationships. Therefore, the measure for the degree of monotonicity is computed for the original data without the attribute SEX and for the data sets obtained after removing SEX and one or more of the other variables.

Table VIII shows that the removal of SEX, WHOLE WEIGHT and SHUCKED WEIGHT leads to a higher number of monotone pairs out of the increased number of comparable pairs; the individual removal of the other attributes, not shown in the table, leads to decrease in the degree of monotonicity compared to the original data (DgrMon ≤ 0.9052). These results indicate that we can consider the abalone data as a partially monotone classification problem where SEX, WHOLE WEIGHT and SHUCKED WEIGHT are the non-monotone variables, whereas the other attributes are the monotone variables.

Therefore, we apply our approach for partial monotonicity. Analogously to the simulation studies, we also use standard neural networks with weight decay and partially monotone linear models as benchmark methods for comparison. To obtain a sound assessment of the generalization capabilities of the model obtained, we split randomly the original data into training data of 219 observations (75%) and test data of 73 observations (25%). The former is used to build a model whereas the latter is used to test the model performance measured by the misclassification error. The random partition of the data is repeated 20 times. We use nine combinations of parameters for the Sill networks (groups - 2, 3, 4; planes - 2, 3, 4) and standard neural networks (hidden nodes - 3, 6, 9; weight decay - 0.000001, 0.00001, 0.0001) to get better insight into the performance of the models. Table IX reports the minimal, mean and maximal value as well as the variance of the estimated minimum misclassification error.

The results show that our approach tends to be more accurate on average than standard neural networks and partially monotone linear models. Furthermore, both types of partially monotone models exhibit smaller variances upon repeated sampling than their unconstrained counterpart.

To check the significance of the results we performed statistical tests. Since the test set in the experiments with the

TABLE IX

ESTIMATED PREDICTION ERRORS AND VARIANCES OF THE MODELS DERIVED FROM THE APPROACH FOR PARTIAL MONOTONICITY (PARTMON), STANDARD NEURAL NETWORKS WITH WEIGHT DECAY (NNETS), AND PARTIALLY MONOTONE LINEAR MODELS (PMONLIN) FOR ABALONE DATA

Method	Minimum error			
	Min	Mean	Max	Variance
PartMon	0.27	0.31	0.36	0.000
NNet	0.25	0.32	0.37	0.002
PMonLin	0.30	0.35	0.38	0.000

TABLE X

p-VALUES OF PAIRED T-TESTS AND ONE-SIDED CONFIDENCE INTERVALS FOR THE DIFFERENCE IN ERROR MEANS IN THE ABALONE CASE STUDY

Indicator	p-value	Confidence intervals	
		95%	90%
Minimum error (PartMon–NNet)	12.8%	(-∞, 0.005)	(-∞, 0.001)
Minimum error (PartMon–PMonLin)	0.0%	(-∞, -0.024)	(-∞, -0.027)
Minimum error (NNet–PMonLin)	0.2%	(-∞, -0.011)	(-∞, -0.014)

TABLE XI

VARIANCE ACROSS DIFFERENT NETWORK ARCHITECTURES WITHIN A RUN WITH THE ABALONE DATA

Method	Variance within a run		
	Min	Mean	Max
PartMon	0.0013	0.0032	0.0054
NNet	0.0003	0.0083	0.0496

three methods is the same, there is a natural pairing of the error rates estimated. Therefore we conducted three paired t-tests to test the null hypothesis that the models derived from one method have the same error as the models derived from the other methods against the one-sided alternatives. The p-values obtained from the tests are reported in Table X. They show that the differences in errors obtained from our approach and standard neural networks are statistically insignificant at 5% and 10% significance level. Furthermore, the flexible nature of our approach and standard neural networks leads to models that can better capture the true relationships in the data. Hence these models have significantly smaller errors than the partially monotone linear models.

Table IX suggests that the differences between the variances of the partially monotone models and standard neural networks are significant, which is also confirmed by the p-values of the two F-tests with 19 degrees of freedom, namely 0.2% and 0.1%; the difference between the variances of the models derived from our approach and from partially monotone linear models is statistically insignificant (p-value is 42.1%).

Another interesting observation is the variance of errors across different Sill and standard network architectures within a run; see Table XI.

The results show that our approach produce models with lower variance on average across various network's architectures compared to standard neural networks with weight decay. In fact in two out of the twenty runs, standard neural networks with three and six hidden nodes produced models with 100% misclassification rate. This result implies that the models derived from the neural networks have very high variability and thus higher dependence on the network's architecture.

V. CONCLUSION

In this paper we considered partially monotone prediction problems where the response variable depends monotonically on some but not on all predictor variables. An approach for building partially monotone models was presented, which is convolution of weight functions (kernels) based on non-monotone variables and Sill (monotone) networks built on the monotone variables only. Simulation and real case studies were used to test the performance of the approach and to compare it with the performance of standard neural networks and partially monotone linear models. First the models derived from our approach are generally more accurate that the partially monotone linear models on artificial and real data. Compared to standard neural networks our approach achieves comparable accuracy but significantly smaller variance upon repeated sampling. Hence, the incorporation of partial monotonicity constraints leads not only to models that are in accordance with the decision maker's expertise but also to more stable models.

APPENDIX I
PROOF OF THEOREM 4.1

Suppose we take both the number of clusters C and the number of points in each cluster equal to $\sqrt{N}$. Then for

$N \to \infty$, within the cluster c, $\text{Var}(\mathbf{x}^{nm}) \to 0$, i.e., $\mathbf{x}^{nm}$ tends to the cluster mean $\mathbf{x}_c^{nm}$. Hence, for each c, the value of $\mathbf{x}^{nm}$ is fixed and we make a local monotone estimation $\hat{f}_c(\mathbf{x}^m)$ of f by using a Sill network based on the values of $\mathbf{x}^m$ for the points belonging to that cluster. This estimation is guaranteed to be close to the local value of the true function due to the sufficiently large number $\sqrt{N}$ of points in a cluster, and the universal approximation capabilities of Sill networks (Theorem 3.1, [5]).

Hence we can consider $\left(\mathbf{x}_c^{nm}, \hat{f}_c(\mathbf{x}^m)\right)_{c=1}^{C}$ as a sample of C independent observations. Then we use these empirical data to find an approximation of f based on $\mathbf{x}^{nm}$ by

$$\hat{f}(\mathbf{x}^m, \mathbf{x}^{nm}) = \sum_{c=1}^{C} \varphi_c(\mathbf{x}^{nm}) \cdot \hat{f}_c(\mathbf{x}^m), \tag{14}$$

where $\varphi_c(\mathbf{x}^{nm})$ is the positive weighted function defined in (9).

As $\hat{f}(\mathbf{x}^m)$ is monotone in $\mathbf{x}^m$, the overall approximation is a class of partially monotone functions, which has the form of Nadaraya-Watson's estimator with kernel $\varphi(\mathbf{x}^{nm})$. As shown in [12] and [13], the approximation in (14) is a consistent estimator of $f(\mathbf{x}^m, \mathbf{x}^{nm})$.

REFERENCES

[1] O. Harrison and D. Rubinfeld. Hedonic prices and the demand for clean air. *Journal of Environmental Economics and Management*, 53:81–102, 1978.

[2] H. Mukarjee and S. Stern. Feasible nonparametric estimation of multiargument monotone functions. *Journal of the American Statistical Association*, 89(425):77–80, 1994.

[3] H.A.M. Daniels and B. Kamp. Application of MLP networks to bond rating and house pricing. *Neural Computing & Applications*, 8(3):226–234, 1999.

[4] H. Kay and L. H. Ungar. Estimating monotonic functions and their bounds. *American Institute of Chemical Engineers (AIChE) Journal*, 46(12):2426–2434, 2000.

[5] J. Sill. Monotonic networks. *Advances in Neural Information Processing Systems*, 10:661–667, 1998.

[6] S. Wang. A neural network method of density estimation for univariate unimodal data. *Neural Computing & Applications*, 2(3):160–167, 1994.

[7] M. Sarfraz, M. Al-Mulhem, and F. Ashraf. Preserving monotonic shape of the data by using piecewise rational cubic functions. *Computers and Graphics*, 21:5–14, 1997.

[8] M. Velikova and H.A.M. Daniels. Decision trees for monotone price models. *Computational Management Science*, 1(3–4):231–244, 2004.

[9] A. Ben-David. Monotonicity maintenance in information-theoretic machine learning algorithms. *Machine Learning*, 19(1):29–43, 1995.

[10] R. Potharst and A. Feelders. Classification trees for problems with monotonicity constraints. *SIGKDD Explorations Newsletter*, 4(1):1–10, 2002.

[11] P. J. Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987.

[12] E. A. Nadaraya. On estimating regression. *Theory of Probability & Its Applications*, 9(1):141–142, 1964.

[13] G. S. Watson. Smooth regression analysis. *Sankhya: The Indian Journal of Statistics, Series A*, 26(4):359–372, 1964.

[14] S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1):1–58, 1992.

[15] C. F. J. Wu and M. Hamada. *Experiments: Planning, Analysis, and Parameter Design Optimization*. John Wiley & Sons, New York, wiley series in probability and statistics edition, 2000.

[16] M. Velikova, H.A.M. Daniels, and A. Feelders. Solving partially monotone problems with neural networks. In *Proceedings of the twelfth International Conference on Computer Science, Vienna, Austria*, pages 82–87. World Enformatika Society, Turkey, 2006.

[17] D. J. Newman, S. Hettich, C. L. Blake, and C. J. Merz. UCI Repository of Machine Learning Databases, 1998. `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

[18] S. Waugh. *Extending and benchmarking cascade-correlation*. Phd dissertation, University of Tasmania, Tasmania, Australia, 1995.

[19] D. Clark, Z. Schreter, and A. Adams. A quantitative comparison of Dystal and backpropagation. In *Proceedings of the seventh Australian Conference Neural Networks, Canberra, Australia*, pages 132–137. 1996.

[20] A. M. Abdelbar. Achieving superior generalisation with a high order neural network. *Neural Computing & Applications*, 7(2):141–146, 1998.

[21] W. J. Nash, T. L. Sellers, S. R. Talbot, and W. B. Cawthorn, A. J. anf Ford. The population biology of abalone (halotis species) in tasmania. i blacklip abalone (h. rubra) from the north coast and islands of bass strait. Technical Report 48, Sea Fisheries Division, Marine Research Laboratories-Taroona, Department of Primary Industry and Fisheries, Tasmania, Australia, 1994.