Generalized Constraint Neural Network Regression Model Subject to Linear Priors

Ya-Jun Qu and Bao-Gang Hu, *Senior Member, IEEE*

***Abstract*— This paper is reports an extension of our previous investigations on adding transparency to neural networks. We focus on a class of linear priors (LPs), such as symmetry, ranking list, boundary, monotonicity, etc., which represent either linear-equality or linear-inequality priors. A generalized constraint neural network-LPs (GCNN-LPs) model is studied. Unlike other existing modeling approaches, the GCNN-LP model exhibits its advantages. First, any LP is embedded by an explicitly structural mode, which may add a higher degree of transparency than using a pure algorithm mode. Second, a direct elimination and least squares approach is adopted to study the model, which produces better performances in both accuracy and computational cost over the Lagrange multiplier techniques in experiments. Specific attention is paid to both "*hard (strictly satisfied)*" and "*soft (weakly satisfied)*" constraints for regression problems. Numerical investigations are made on synthetic examples as well as on the real-world datasets. Simulation results demonstrate the effectiveness of the proposed modeling approach in comparison with other existing approaches.**

***Index Terms*— Linear constraints, linear priors, nonlinear regression, radial basis function networks, transparency.**

I. Introduction

BECAUSE of the ability to provide excellent universal function approximation for multivariate input/output spaces on a training set, artificial neural networks (ANNs or NNs for short) are widely used in various applications such as system identification and control, decision making, pattern recognition, sequence recognition, and data mining [1], [2]. However, ANNs aim at learning an unknown function based on a training dataset as a "*black box*" approach, which is usually unable to be interpreted through intuition. The nontransparency feature will result in "*trial and error*" design procedures without "*comprehensibility*," which greatly restrain further advances of the NN technique in applications. Therefore, we believe that "*adding transparency to ANNs will provide a direct and straightforward solution to the important issues, such as structure design, training, and generalization capabilities*" [3].

In fact, adding transparency and comprehensibility to nonlinear prediction models has been a long-standing problem in this paper of machine learning. In recent years, this problem has received much attention, particularly for learning machines of ANNs [3], [4] and support vector machines (SVMs) [5]–[7]. In the work on ANNs, Hu *et al.* [3], [8] described a hierarchical diagram (Fig. 1) of classifying a study in adding transparency to nonlinear models by two main strategies, that is:

1) incorporating prior information into the models [2], [9]–[11];
2) extracting knowledge or rules from the models [12]–[16].

For attaining the ideal goal of being "*white-box*" tools to the nonlinear models, both strategies present the theoretical challenges from their two different technical solutions. In this paper, we will focus mainly on the first strategy. For simplifying discussions, we take the notion of "*prior information*" as a general term for representing "*any known information about or related to the targets investigated, such as data, knowledge, specifications, etc*." [8], which may be called "*domain knowledge*," "*prior knowledge*," "*prior facts*," "*hints*," "*biases*," or "*side information*."

For the first strategy, significant investigations have been reported for diverse forms of prior information. Mitchell [17] stressed on the notion of "*bias*" in inductive learning, which could be "*any basis for choosing one generalization over another, other than strict consistency with the observed training instances*." One of the most successful "*biases*" is the "*smoothness*" imposed on nonlinear functions for forming different learning machines [2], [18], [19]. Abu-Mostafa [20] summarized five classes of prior information in terms of "*hints*," such as, "*invariance, monotonicity, approximation, binary, and examples*." Mitchell [9] described three means for embedding prior information into models, that is, "*to initiate the hypothesis, to alter the search objective, and to augment search operators*." Based on this paper, Yu [10] extended the approaches and studied their three related issues on "*consistency*," "*generalization*," and "*convergence*." Lauer and Bloch [7] defined two main groups of prior information in classifications, that is, "*class-invariance*" and "*knowledge on data*." Then, they presented a hierarchy of embedding methods with three groups, namely, "*samples*," "*kernel*," and "*optimization problem*." All these studies provide an important implication of implementing the first strategy. "*Classifications of prior information will be helpful for embedding the given priors into models effectively and efficiently in a systematic way*." The remarkable examples are knowledge-based and fuzzy NNs [21], [22], which apply linguistic knowledge. Bayesian inferences are also considered as an elegant frame-

Manuscript received May 26, 2010; revised December 10, 2010, May 18, 2011, and August 24, 2011; accepted August 25, 2011. Date of publication September 26, 2011; date of current version December 13, 2011. This work was supported in part by the Natural Science of Foundation of China under Grant 61075051.

The authors are with the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China (email: yjqu@nlpr.ia.ac.cn; hubg@nlpr.ia.ac.cn).

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

Digital Object Identifier 10.1109/TNN.2011.2167348

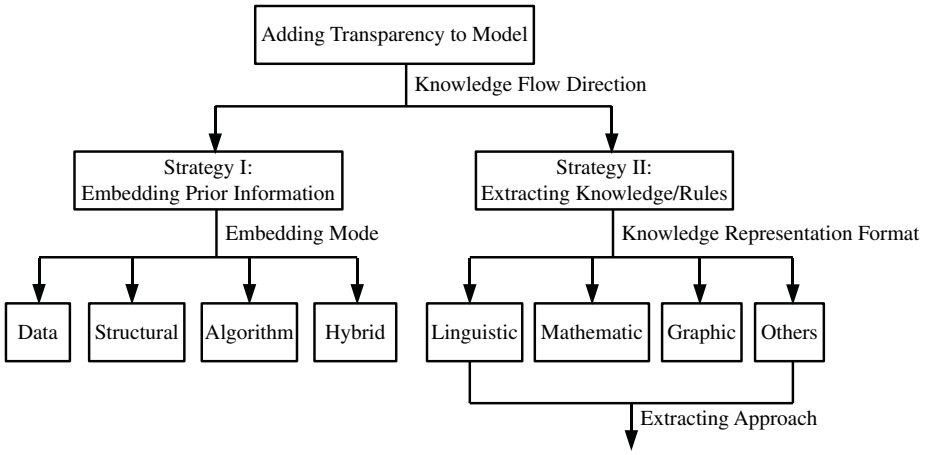


Fig. 1. Hierarchical diagram of classifying studies in adding transparency to nonlinear models [8].

work for utilizing prior distribution information in statistical learning, such as for NNs [23].

To better understand the numerous approaches from the perspective of their associated explicitness to the models, Hu *et al.* [8] suggested three basic "*information embedding modes*" for examining each individual approach, namely, "*structural*," "*algorithm*," and "*data*" modes, respectively (Fig. 1). With regard to transparency, the three modes they considered were generally ranked in a descending order. This ranking supposes that the given prior information is better inserted into models through a structural way, rather than by the other two means even if they might be possible. The "*hybrid model*" [24], [25] is a typical example for a structural method of embedding prior information, which integrates two submodels from both empirical functions and NNs. Kernel selections [26] and constraints on searching the space/step [2], [18] fall into an algorithm mode. Virtual samples [27] and initial data for searching [9] present a data mode. This classification is roughly correct since some approaches may share the different embedding modes at the same time. Sometimes, the three embedding modes may be mutually applied for better understanding and implementations. Probabilistic graphical models [28] present an excellent example on this aspect.

This work described in this paper is an extension on the previous investigations [3], [29], [30] of the generalized constraint NN (GCNN). In [3] and [29], they only considered the "*hard*" constraints, which required strict agreements with the constraints. In [30], "*soft*," or weakly satisfied, constraints were investigated, but the specific technique was adopted in the process. This paper will concentrate on developing a more generic approach for handling a class of linear priors (LPs). For this reason, we call our approach GCNN-LPs. "*LPs*" are defined to be a class of prior information that exhibits a linear relation to the attributes of interests, such as variables, free parameters, or their functions of the models. This type of LPs may be either expressed by the conventional constraints to the model, or embedded by other forms, such as being inserted directly within the objective function. For example, in Section V-A, the monotonicity prior can be described by the constraints of derivatives or ranking list to the objective function. It can also be realized by using the monotonous basis function to the model. In this paper, we adopt the term "LPs" for stressing that there exist various forms for inserting priors and that they are more problem dependent. This paper will focus mainly on the conventional form while comparing them with other forms if applicable. The LPs are explicitly formed using a similar network structure, instead of using a pure algorithm format. Although the nonlinear priors are more general in scope, we consider LPs to be more important in applications. We further limit our investigations on regression problems. Both "*hard*" and "*soft*" constraints will be systematically studied by the same technique of GCNN-LP.

The organization of this paper is as follows. In Section II, the existing methods that incorporates a class of LPs into regression models are summarized. The algorithm of the conventional radial basis function (RBF) network is reviewed briefly in the next section. Section IV is dedicated to the descriptions about the structure and learning method of the GCNN-LP models. In order to adapt the prior with noise, "*soft*" and "*hard*" constraints are investigated in this paper. In addition, modification of GCNN-LP is presented to handle continuous constraints. Section V presents the simulation studies of GCNN-LP on synthetic examples and real-world regression problems. Finally, we summarize the GCNN-LP models in Section VI.

II. Related Work

In this section, we will discuss the first strategy within a class of LPs. Table I lists the commonly used LPs in regression models together with their associated terms of prior information. Some priors may be equivalent to or, be represented by, the others. We list them individually for highlighting different semantic terms of the given prior information. Significant progress has been made on the subjects. We will present the related progresses on two parts below, namely, linear equality priors, and linear inequality priors.

A. Linear Equality Priors

1) Interpolation Points: In regression problems, some type of prior information, such as interpolation points, extreme points, etc., can be available. This prior information is usually

TABLE I

GENERAL TYPES OF LPS IN REGRESSIONS. (EXAMPLE ON 1-D VARIABLE PROBLEM, $y = f(x, \theta)$, WHERE θ IS A PARAMETER VECTOR. THE SUBSCRIPTS d AND u REPRESENT FOR "KNOWN" AND "UNKNOWN," RESPECTIVELY)

LPs	General types of prior information	
	Prior terms	Examples
Equality	Interpolation points	$f(x_d) = c_d,\ f'(x_d) = c_d$
	Fixed points	$f(x_d) = x_d,\ f(x_u) = x_u$
	Equality points	$f(x_{d1}) = f(x_{d2})$
	Inflection point	$f''(x_d) = 0$
	Extrema	$f'(x_d) = 0,\ \min f(x) = c_d$
	Integration	$\int f(x)dx = c_d$
	Equilibrium	$f(x \to \infty) = c_u,\ f'(x \to \infty) = 0$
	Singularity	$f(x_d) = -\infty,\ f(x_u) = \infty$
	Symmetry	$f(x + c_d) = f(-x + c_d),\ f(x + c_u) = -f(-x + c_u)$
	Identity	$f(x) = x$
	Periodicity	$f(x) = f(x + kc_u),\ k = \pm 1, \pm 2, \ldots$
	Proportionality	$f(x) = c_u/x,\ f(x) = c_u 2^x$
	Continuity	$f'(x \to x_u^-) = f'(x \to x_u^+)$
Inequality	Boundary	$c_{d1} \leqslant f(x) \leqslant c_{d2},\ \theta_{d1} \leqslant \theta_i \leqslant \theta_{d2}$
	Max/Min	$\min f(x) \geqslant c_d,\ \max f'(x) \leqslant c_d$
	Nonnegativity	$f(x) \geqslant 0,\ \theta_i \geqslant 0$
	Monotonicity	$f'(x) \geqslant 0,\ f'(x) \leqslant 0$ when $x \geqslant 0$
	Convexity	$f(\alpha x_1 + (1-\alpha)x_2) \leqslant \alpha f(x_1) + (1-\alpha) f(x_2),\ \alpha \in [0, 1],\ x_1 \neq x_2$
	Ranking	$f(x_{d1}) \geqslant f(x_{d2}),\ f'(x_u) \leqslant f'(x_d)$
	Smoothness	$\int \lvert f'(x)\rvert < \infty$
	Sparsity	$\sum \lvert\theta_i\rvert_1 \leqslant c_d$
	Finiteness	$\int \lvert f(x)\rvert < \infty$
	Nonequality points	$f(c_{d1}) \neq f(c_{d2})$
	Nonsingularity	$1 - c_u x^2 \geqslant 0,\ 1 - x^2 \neq 0,\ \lvert f(x)\rvert < \infty$
	Discontinuity	$f(x \to x_d^-) \neq f(x \to x_d^+)$

expressed through linear equality priors. However, methods that incorporate this type of prior information into the regression models are different. Lauer *et al.* [11] and Qu *et al.* [29] explored the incorporation of prior information by the addition of constraints. The difference between them is that the former is a linear programming support vector regression (LP-SVR) while the latter involves elimination and least squares regression in a NN.

2) Invariance Transformation: In regression problems, invariance transformation mainly includes symmetry and periodicity. Three types of methods, namely, the virtual support vector method, jittered SV method, and invariance hyperplane method, for incorporating invariance into kernel have been summarized in [31]. The basic idea involves adding invariance as penalty terms into the objective function. Chen *et al.* [32] utilized symmetric RBF functions to incorporate a global symmetric property into the NNs. However, a general approach may be needed to explore the symmetric constraints within either a local range or a subset of input variables.

B. Linear Inequality Priors

1) Ranking: This type of prior information is a relationship between a set of categories. The ranking only contains the order information of the objects but not the real distance between objects. Therefore, ranking is known as the weaker knowledge. However, this weaker knowledge can improve the models. Zhu and Andrew [33] proposed a semisupervised learning method that consists of enforcing the order preferences as regularization in a risk minimization framework. In [30], a method that can incorporate ranking information as prior information into the NNs is presented. Compared with other methods, which treat the ranking information as hard constraints, this method handles ranking reasonably by maximization of normalized discount cumulative gain (NDCG) as an information retrieval evaluation measure.

2) Boundary: This type of prior information, which may be classified as a functional boundary (boundary, sparsity) or functional derivatives boundary such as monotonicity, convexity, or concavity, is used to recover the overall shapes of the regression functions. If these properties of function are available, they can be incorporated into the regression model. In [34] and [35], sparse SVMs, which combine a prediction error term with a term associated to the sparsity in the number of selected variables, are employed for feature selection. Niyogi *et al.* [27] used the virtual example method to incorporate monotonicity hint into machine learning. However, some drawbacks from virtual example methods are inevitable, e.g., how many virtual examples should be generated, and virtual examples are highly correlated with the original data. In [36]–[39], monotonicity as prior information is incorporated into the architecture of NNs that can satisfy the monotonous constraints via the sum of a series of monotonous functions. These structural methods can allow the target function to satisfy the monotonous constraints strictly, but they can only be applied to the global monotonous function. That is to say,

TABLE II
COMPARISON OF VARIOUS METHODS INCORPORATING PRIOR INFORMATION INTO REGRESSION MODEL

Priors	Prior information	Methods
Equality	Interpolation points	LP-SVR [11]
	Long-term memory	Elimination + gradient descent algorithm [44]
	Equilibrium points	Elimination + LS [29]
	Invariance transformation	Regularization [31]
		Kernel function method [32]
Inequality	Smoothness	Regularization [2], [18]
	Order preferences	Regularization [33]
	Ranking list	NDCG + regularization [30]
	Sparsity	Regularization [34], [35]
	Boundary (Monotonicity)	virtual example [27]
		Structural method/kernel function method [36]–[39]
		Discretization-based method [11], [29], [40], [41]
		CPM/R-CPM [42]
	Multiple outputs regression	Kernel method [43]
		Discretization-based method [11]

they cannot deal with the monotonous function in a local input space. In [11], [29], [40], and [41], the discretization-based methods are used to reduce the infinite continuous constraints to finite subset constraints. Although these methods can ensure that the prior information is satisfied in these discrete points, they cannot guarantee that the nonlinear bound on the function holds in the whole nonlinear region. The CPM (CPM for low dimension) and relaxed cutting plane method (R-CPM for high dimension) [42] can tackle the continuous constrained kernel-based regression problems, which can overcome the drawbacks of discretization-based methods.

3) Multiple Outputs Regression: This problem of multiple regression has been briefly reviewed in [11]. Weston *et al.* [43] presents a method that incorporates prior information into multiple outputs. Prior information and/or invariance in the loss function can be embedded into the output kernels. Another method, proposed in [11], allows for a simple formulation of the dependencies among multiple outputs, and the discretization-based method is used to generate a finite set of constraints, which leaves the nature of the learning problem unchanged.

All the above approaches, which can be used to incorporate prior information into the NN or SVR, are summarized in Table II. At this stage, to the authors' knowledge, there still exist two open issues in this paper of embedding priors into nonlinear regression models.

1) To utilize any type of prior information maximally and explicitly for enhancing generalization and transparency of the models.
2) To process both "*hard*" and "*soft*" constraints for the same structure and the learning technique of the models.

For being a standard technique in very-large-scale integration implementation of ANNs, the two issues above are better to be solved, which therefore is the motivation of this paper presented here.

III. CONVENTIONAL RBF NETWORKS

Suppose a multiple input and single output regression problem with training set $T = \{\mathbf{x}_l, y_l\}_{l=1,2,\ldots,N}$, where $\mathbf{x}_l \in \mathbf{R}^{1\times n}$ is an input vector, $y_l \in \mathbf{R}$ denotes the value of desired network output for the input $\mathbf{x}_l$, and N is the number of training sets. The output of a RBF network is calculated according to

$$\hat{y} = \mathbf{\Phi}(\mathbf{x})\mathbf{W} \tag{1}$$

where $\hat{y}$ is the output of the network, $\mathbf{W} = [\omega_0, \omega_1, \ldots, \omega_h]^T$ are the weights of the $\hat{y}$, and $\mathbf{\Phi}(\mathbf{x}) = [1, \phi_1(\mathbf{x}), \ldots, \phi_h(\mathbf{x})]$, h is the number of neurons in the hidden layer, and $\phi_\iota(\cdot)$ is a basis function of the RBF network. In this paper, the Gaussian RBF $\phi_\iota(\mathbf{x}) = \exp(-\|\mathbf{x} - \mathbf{c}_\iota\|^2/\sigma_\iota^2)$, $\iota = 1, 2, \ldots, h$ is discussed, where $\mathbf{c}_\iota \in \mathbf{R}^{1\times n}$ are the RBF centers and the parameter σ_ι controls the "width" of the RBF. The most common approach is to determine the centers and widths of the RBFs first, and then the weights are estimated subsequently. In this paper, k-center and simple heuristic relationship [45] are used to determine the centers and the widths, respectively.

Once the centers and widths are chosen, the network training task is changed to determine the appropriate settings of the weights between the neurons. A common optimization criterion is the mean square error between the actual and desired network outputs. Therefore, the optimal set of weights minimizes the performance measure [2]

$$\min_{\mathbf{W}\in\mathbf{R}^{h+1}} : Q(\mathbf{W}) = \frac{1}{2}(\mathbf{y} - \mathbf{\Phi}\mathbf{W})^T(\mathbf{y} - \mathbf{\Phi}\mathbf{W}) \tag{2}$$

where $\mathbf{y} = [y_1, \ldots, y_N]^T \in \mathbf{R}^{N\times 1}$ denotes the vector of desired network outputs and $\mathbf{\Phi} = [\mathbf{\Phi}^T(\mathbf{x}_1), \mathbf{\Phi}^T(\mathbf{x}_2), \ldots, \mathbf{\Phi}^T(\mathbf{x}_N)]^T \in \mathbf{R}^{N\times(h+1)}$ (the first row is 1). Gradient descent method or least squares strategies can be used to determine the weights of (2). Due to the speed and accuracy of the training, the least squares algorithm is used in this paper. Minimizing the performance measure $Q(\mathbf{W})$, we have [2]

$$\mathbf{W} = \left[\mathbf{\Phi}^T\mathbf{\Phi}\right]^{-1}\mathbf{\Phi}^T\mathbf{y} = \mathbf{\Phi}^+\mathbf{y} \tag{3}$$

where $\mathbf{\Phi}^+$ denotes the pseudo-inverse of the nonlinear mapping matrix $\mathbf{\Phi}$. From (3), the weights of RBF network are determined and the problem of network training can have a closed-form solution.

IV. GCNN-LPs

A. General Description

Generally, supervised training is formulated as an unconstrained optimization problem consisting of the network weights $\mathbf{W}$. Assume the given LP $C = (\mathbf{x}_n, \xi_n)_{n=1,2,\ldots,p}$, where ξ_n is the value or the partial derivatives value of the desired function for the input $\mathbf{x}_n$, and p is the number of LPs. Fig. 2 shows the structure of GCNN-LP model. This structure is a combination of two submodels, the first is a standard RBF network and the other one is dependent network that preserves the given prior information, represented by thin and thick lines, respectively, in Fig. 2. Let $\Phi_U(\mathbf{x}) = [1, \varphi_{U1}(\mathbf{x}), \ldots, \varphi_{Uh}(\mathbf{x})]$ and $\Phi_R(\mathbf{x}) = [\varphi_{R1}(\mathbf{x}), \ldots, \varphi_{Rp}(\mathbf{x})]$, where $\varphi_{U\imath}(\mathbf{x}), \imath = 1, \ldots, h$ and $\varphi_{RJ}(\mathbf{x}), J = 1, \ldots, p$ are the original RBF $\phi(\mathbf{x})$ or its partial derivatives with respect to variable $\mathbf{x}$. Then, we define the following GCNN-LP:

$$\begin{aligned} \min_{\mathbf{W}} \quad & \frac{1}{2}||\mathbf{y} - \mathbf{\Phi W}||^2 + \sum_{k \in E \cup I} \gamma_k \varepsilon_k \\ \text{s.t.} \quad & -\varepsilon_j \leqslant \Phi(\mathbf{x}_j)\mathbf{W} - \xi_j \leqslant \varepsilon_j, \quad j \in E, \\ & \Phi(\mathbf{x}_i)\mathbf{W} - \xi_i \geqslant -\varepsilon_i, \quad i \in I, \\ & \varepsilon_k \geqslant 0, \quad k \in E \cup I \end{aligned} \tag{4}$$

where $\Phi(\mathbf{x}_j) = [\Phi_U(\mathbf{x}_j), \Phi_R(\mathbf{x}_j)]$, E and I are, respectively, the collections of indices of equality and inequality constraints, ε_k is the positive auxiliary variable to soft constraints, and γ_k is the weight of the variable ε_k determining the tradeoff between the prior information and the training data. In other words, a large value of γ_k indicates a high degree of confidence of prior information, whereas a small value suggests a low degree of reliability of prior information. We refer to the set of linear equality and inequality constraints as LPs.

When the prior information is not perfect or has noise, soft constraints, expressed as $\exists k \in E \cup I, \varepsilon_k \neq 0$ in (4), are suitable for handling this type of prior.

For the purpose of optimizing (4) conveniently, according to the Karush–Kuhn–Tucke (KKT) condition, the dual problem can be obtained and written in the form of vectors as follows:

$$\begin{aligned} \max_{\Theta} \quad & \mathbf{g}^T\Theta - \frac{1}{2}\Theta^T\left(\mathbf{\Phi}_{EI}\mathbf{H}^{-1}\mathbf{\Phi}_{EI}^T\right)\Theta \\ \text{s.t.} \quad & \gamma_j \geqslant \theta_j + \theta_{j+n_E}, \quad j = 1, \ldots, n_E \\ & \theta_j \geqslant 0, \quad j = 1, \ldots, 2n_E \\ & \gamma_i \geqslant \theta_i \geqslant 0, \quad i = 2n_E + 1, \ldots, 2n_E + n_I \end{aligned} \tag{5}$$

where $\mathbf{H}^{-1} = (\mathbf{\Phi^T\Phi})^{-1}$, $\Theta = [\theta_1, \ldots, \theta_{2n_E+n_I}]^T$ is the vector of Lagrange multipliers, and n_E and n_I are the numbers of equality and inequality constraints, respectively. The other remaining terms are defined by (6)

$$\begin{cases} \mathbf{g} = \Xi_{EI} + \mathbf{\Phi}_{EI}\mathbf{H}^{-1}\mathbf{\Phi}^T\mathbf{y}, \\ \Xi_{EI} = \left[\Xi_E^T, -\Xi_E^T, \Xi_I^T\right]^T, \\ \Xi_E = \left[\xi_1, \ldots, \xi_{n_E}\right]^T, \\ \Xi_I = \left[\xi_1, \ldots, \xi_{n_I}\right]^T, \\ \mathbf{\Phi}_{EI} = \left[\mathbf{\Phi}_E^T, -\mathbf{\Phi}_E^T, \mathbf{\Phi}_I^T\right]^T, \\ \mathbf{\Phi}_E = \left[\mathbf{\Phi}^T(\mathbf{x}_1), \ldots, \mathbf{\Phi}^T(\mathbf{x}_{n_E})\right]^T, \\ \mathbf{\Phi}_I = \left[\mathbf{\Phi}^T(\mathbf{x}_{2n_E+1}), \ldots, \mathbf{\Phi}^T(\mathbf{x}_{2n_E+n_I})\right]^T. \end{cases} \tag{6}$$

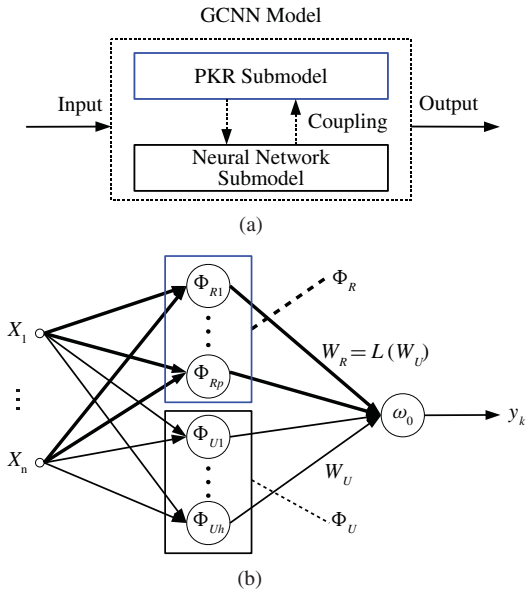


Fig. 2. (a) GCNN model [3]. (b) GCNN-LP network architecture, which includes two submodels one a standard RBF network and the other a constrained network that can impose prior information.

Direct elimination and taking the least squares, which are introduced in the next subsection, are used to optimize (5). Once the dual problem (5) has been solved, the original (4) can be obtained by the relationship, which is defined as follows:

$$\mathbf{W} = \mathbf{H}^{-1}\left(\mathbf{\Phi}^T\mathbf{y} + \mathbf{\Phi}_{EI}^T\Theta\right). \tag{7}$$

Finally, the GCNN-LP problem can be solved and the regression model can be obtained. We next present two modifications of GCNN-LP depending on the type of constraints.

B. Modifications of GCNN-LP

1) GCNN-LP with Hard Constraints (GCNN-hLP): When $\varepsilon_k = 0, \forall k \in E \cup I$, we refer to the constraints in (4) as *hard constraints*.

GCNN with linear equality constraints: Prior information, such as extreme points, equilibrium points, etc., can be expressed by

$$\Phi_U\mathbf{W}_U + \Phi_R\mathbf{W}_R - \Xi = 0 \tag{8}$$

where $\Phi_U = [\Phi_U^T(\mathbf{x}_1), \ldots, \Phi_U^T(\mathbf{x}_p)]^T \in \mathbf{R}^{p\times(h+1)}$, $\Phi_R = [\Phi_R^T(\mathbf{x}_1), \ldots, \Phi_R^T(\mathbf{x}_p)]^T \in \mathbf{R}^{p\times p}$, $\mathbf{W} = [\mathbf{W}_U^T, \mathbf{W}_R^T]^T \in \mathbf{R}^{(h+1+p)\times 1}$, and $\Xi = [\xi_1, \ldots, \xi_p]^T \in \mathbf{R}^{p\times 1}$. The rank of matrix Φ_R is p, which means Φ_R is invertible. $\mathbf{W}_U \in \mathbf{R}^{(h+1)\times 1}$ is the vector of independent free parameters, which is estimated by network learning. $\mathbf{W}_R \in \mathbf{R}^{p\times 1}$ is the vector of dependent parameters, which is determined by prior information and $\mathbf{W}_U$. If (8) satisfies the implicit function theorem in [44], we can obtain an invertible matrix Φ_R. Therefore (8) can be reformulated uniquely as equality constraints and of the form

$$\mathbf{W}_R = \mathbf{L}(\mathbf{W}_U) = \Phi_R^{-1}\left(\Xi - \Phi_U\mathbf{W}_U\right). \tag{9}$$

Generally, the solution of a quadratic programming problem under linear equality constraints (9) can be provided by the

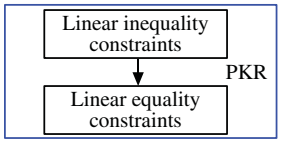


Fig. 3. Structure of the partially known relationship (PKR) submodel in the GCNN-LP model.

algorithm of Lagrange multipliers or by elimination [46], [47]. In this paper, we adopt the latter approach. Then, due to the equality constraints (9), the elimination can be applied by writing the objective function as

$$V(\mathbf{W}_U) = Q(\mathbf{W}_U, \mathbf{W}_R) = Q(\mathbf{W}_U, \mathbf{L}(\mathbf{W}_U)) \tag{10}$$

such that the value of $\mathbf{W}_U$ can be determined independently of $\mathbf{W}_R$. Thus, constrained optimization problems can be transformed into unconstrained optimization problems. Finally, we can get the desired network.

We prefer the solver of elimination plus the least squares method since it is more efficient than the Lagrangian method for GCNN-LP models in a general sense. If the Lagrangian method is used and transformed to the form of least squares, then the number of free parameters will be $(h+1+2p)$. Thus the complexity of inverse or pseudo-inverse in Lagrangian method is $O((h+1+2p)^3)$. However, the elimination method can be used to incorporate constraints into the model that can reduce the number of free parameters. In addition, the complexity of inverse or pseudo-inverse is also reduced to $O((h+1)^3)$.

2) GCNN with Linear Inequality Constraints: Prior information, such as boundary, monotonicity, etc., is required to use inequality constraints.

A general approach to solve the general constrained quadratic programming problem is the active set algorithm [48]. A key to the efficient implementation of the active set algorithm is the recycling of information from solving the equality-constrained subproblem, which is given in the next iteration. That is to say, the general constrained quadratic programming problem can be regarded as the equality-constrained quadratic programming problem. In each iteration, some of the inequality constraints are regarded as equality constraints, and form the working set S_k (the index collection of equality constraints and active inequality constraints at the kth iteration), while the remaining inequality constraints are disregarded. The working set at the optimum is known as the "active set" of the solution. The active set is particularly important for optimization theory, as it determines which constraints will influence the final result of optimization. In quadratic programming, the solution is not mandatory for one of the edges of the bounding polygon. An estimation of the active set gives us a subset of inequalities to monitor while searching the solution, which reduces the complexity of the search. In this paper, a structure of the PKR submodel, in which the active linear inequality constraints are selected and transformed into linear equality constraints, is used in the GCNN-LP model in Fig. 3.

Considering the kth iteration in the active set algorithm [48], we have the feasible point $\mathbf{W}_k$ (a point is feasible only if it satisfies the constraints of GCNN-LP) and the working set $S_k \subset E \cup I$). We obtain $\mathbf{d}_k$ from solving

$$\begin{aligned} \min_{\mathbf{d}_k \in \mathbf{R}^{h+1+p}} \quad & Q(\mathbf{W}_k + \mathbf{d}_k) \\ \text{s.t.} \quad & \Phi(\mathbf{x}_j)\mathbf{d}_k = 0, \quad j \in S_k. \end{aligned} \tag{11}$$

If $\mathbf{d}_k \neq \mathbf{0}$, it means that $\mathbf{W}_k + \mathbf{d}_k$ may not be a feasible point of GCNN-LP, and another feasible point should be found for the next iteration. Otherwise, the $\mathbf{W}_k$ is the KKT point of the problem as follows:

$$\begin{aligned} \min_{\mathbf{W}_k \in \mathbf{R}^{h+1+p}} \quad & Q(\mathbf{W}_k) = \frac{1}{2}(\mathbf{y} - \mathbf{\Phi}\mathbf{W}_k)^T(\mathbf{y} - \mathbf{\Phi}\mathbf{W}_k) \\ \text{s.t.} \quad & \Phi(\mathbf{x}_j)\mathbf{W}_k = \xi_j, \qquad j \in S_k. \end{aligned} \tag{12}$$

Consider (11) to be solved by the active set algorithm. If its solution $\mathbf{d}_k = \mathbf{0}$, then $\mathbf{W}_k$ is the KKT point of (12). However, we can directly solve (12) and obtain $\mathbf{W}_k$ by using the methods of elimination. Thus, we can use the least squares method by substituting the gradient algorithm to solve (12), which makes implementation of the algorithm simple and convenient.

Therefore, the training of GCNN-LP model can be summarized by Algorithm 1, where $j_{\max}$ can be calculated by

$$j_{\max} = \arg \max_{j \in I \setminus S_k, \Phi(\mathbf{x}_j)\mathbf{W}_k < \xi_j} \{Q(\mathbf{W}_k), S_k \leftarrow j \cup S_k\} \tag{13}$$

and $Q(\mathbf{W}_k)$ is calculated from (12).

Corollary 1: If Algorithm 1 is terminated within finite steps, then the point of convergence must be a KKT point of GCNN-LP.

Corollary 2: If GCNN-LP is bounded and $\Phi(\mathbf{x}_j)(j \in S_k)$ is linearly independent, Algorithm 1 is terminated at finite steps.

The proofs of these corollaries are given in the Appendix. Therefore, Algorithm 1 converges to the optimal solution in a finite number of steps. However, compared with the active set algorithm, Algorithm 1 has the following advantages.

1) The starting point does not need to be a feasible point. And the suggested algorithm can directly use the equality constraints to the starting working set, which can avoid the trouble of searching for a feasible starting point.

Algorithm 1 Training of GCNN-LP model

Step 1: Assume $S_1 \leftarrow E, k \leftarrow 1$.

Step 2: Solve (12) with elimination method and get $\mathbf{W}_k$.

Case 1: if $\exists j \in I \setminus S_k$ (which means the index j is in collection I but not in S_k) and $\Phi(\mathbf{x}_j)\mathbf{W}_k < \xi_j$, then go to Step 3.

Case 2: else if Lagrange multipliers $\lambda_j^{(k)} \geqslant 0, \forall j \in S_k \cap I$, then the algorithm stops.

Case 3: else calculate j_k by solving

$$\lambda_{j_k}^{(k)} = \min_{j \in S_k \cap I} \lambda_j^{(k)} < 0$$

$S_k \leftarrow S_k \setminus \{j_k\}$, go to Step 4.

Step 3: Find $j \in I \setminus S_k$ and $\Phi(\mathbf{x}_j)\mathbf{W}_k < \xi_j$, then $S_k \leftarrow S_k \cup \{j_{\max}\}$, go to Step 4.

Step 4: $k \leftarrow k+1$; go to Step 2.

Algorithm 2 GCNN-LP with Continuous Constraints (GCNN-*c*LP)

Step 1: Determine the tolerance $\epsilon > 0$, $P_D^{(1)} = P_D = \{P_i^{(1)}\}$, $1 \leqslant i \leqslant (N_s + 1)$ based on sampling points D, $y_m^{(1)} = 0$, and $k \leftarrow 1$.

Step 2: For each subregion $P_i^{(k)}$, evaluate a point $\mathbf{x}^{(k)} \in P_i^{(k)}$ such that

$$F(\hat{y}_m^{(k)}, \mathbf{x}^{(k)}) = \max_{\mathbf{x} \in P_i^{(k)}} F(\hat{y}_m^{(k)}, \mathbf{x}).$$

If $F(\hat{y}_m^{(k)}, \mathbf{x}^{(k)}) \leqslant \epsilon$, then $P_D^{(k)} \leftarrow P_D^{(k)} \backslash P_i^{(k)}$ and go to Step 4; otherwise, go to Step 3.

Step 3: $y_m^{(k)} \leftarrow y_m^{(k)} + \phi(\mathbf{x})\omega$, where the basis function $\phi(\mathbf{x})$ can be a Gaussian or a sigmoid function. Center $\mathbf{c} = \mathbf{x}^{(k)}$, width σ is determined by the width of subregion $P_i^{(k)}$. Parameter ω is updated by

$$\omega = -\frac{F(\hat{y}_m^{(k)}, \mathbf{x}^{(k)})}{F(\phi(\mathbf{x}^{(k)}), \mathbf{x}^{(k)})}.$$

The subregion $P_i^{(k)}$ is divided into two subregions $P_{i1}^{(k)}$ and $P_{i2}^{(k)}$ by the point $\mathbf{x}^{(k)}$ and replaced by them as $P_i^{(k)} \leftarrow \{P_{i1}^{(k)}, P_{i2}^{(k)}\}$.

Step 4: If $P_D^{(k)} = \emptyset$, stop and return $y_m^{(k)}$, otherwise go to Step 5.

Step 5: $k \leftarrow k + 1$; go to Step 2.

2) The least squares method is used instead of gradient descent. In each kth iteration, the optimal solution $\mathbf{W}_k$ of the (12) can be directly solved by the least squares method. Therefore, the suggested algorithm is simpler.

The GCNN-LP model is given by (4), but we need to specify which constraint is defined to be hard or soft one. The selection of constraint types is dependent on the correctness of the prior information. In other words, if the prior information is completely correct or reliable, such as the symmetric or monotonic properties of the target system to be exactly given, we can impose the hard constraint on the GCNN model. However, if the prior information is not perfect or with noise, we can set the soft constraint. For example, the prior of ranking information can be useful but not perfectly reliable. In the later section of numerical experiments, we demonstrate such soft constraints on the prediction model on the real-world datasets.

3) GCNN-LP with Continuous Constraints (GCNN-cLP): Continuous constraints, such as boundary, monotonicity, and symmetry, may cause computational difficulties due to the fact that a continuous feature implies an infinite number of constraints over the input variables. The discretization-based methods [11], [40], [41] reduce the infinite set of constraints to a finite subset of constraints. Although the methods can ensure that the prior information is satisfied in these discrete points, it cannot guarantee that the constrained bound on the function holds in the whole input regions. Therefore, we want to modify the unsampled region that does not satisfy the continuous constraints. Because the RBF NN is a local approximating network, the RBF network can approximate the nonlinear function of a local area without affecting the value of other regions. Therefore, we want to use a submodel of RBF network to modify the violated region.

Assume that linear continuous constraint is defined as $F(\hat{y}, \mathbf{x}) \leqslant 0, \forall \mathbf{x} \in P$, where $\hat{y}$ is defined by (1). Function $F(\hat{y}, \mathbf{x})$ is linear with respect to the free parameters $\mathbf{W}$, which can be expressed as $F(\hat{y}, \mathbf{x}) = F(\sum_{l=0}^{h} \phi_l(\mathbf{x})\omega_l, \mathbf{x}) = \sum_{l=0}^{h} F(\phi_l(\mathbf{x}), \mathbf{x})\omega_l$, where $\phi_0(\mathbf{x}) = 1$. Discrete sampling points $D = \{\mathbf{x}_1, \ldots, \mathbf{x}_{N_s}\}$ can be obtained by a discretization-based method [11] from the continuous region P. Therefore, the whole region P can be divided into $(N_s + 1)$ subregions, defined as $P_D = \{P_1, \ldots, P_{N_s+1}\}$, and subregions satisfy $P = P_1 \cup P_2 \cup \cdots \cup P_{N_s+1}$ and $P_i \cap P_j = \emptyset, \forall i \neq j, 1 \leqslant i, j \leqslant (N_s + 1)$. According to GCNN-LP, we can obtain $F(\hat{y}, \mathbf{x}_i) \leqslant 0, \forall \mathbf{x}_i \in D$. Therefore, we want to find a submodel y_m to satisfy

$$F(\hat{y}_m, \mathbf{x}) \leqslant 0, \qquad \forall \mathbf{x} \in P \tag{14}$$

where $\hat{y}_m = \hat{y} + y_m$ and $\hat{y}$ is replaced by $\hat{y}_m$ as output of network. Considering the convergence, we restrict (14) to $F(\hat{y}_m, \mathbf{x}) \leqslant \epsilon$, where $\epsilon > 0$ is a slack variable, and then algorithm GCNN-*c*LP can be used to determine the submodel y_m.

Note that the choice of the basis function in submodel y_m should be changed with different continuous linear constraints F. For example, if F is a boundary constraint, a Gaussian function is chosen as the basis function. But if F is a monotonous constraint, a sigmoid function may be a good choice for the basis function, because a sigmoid function is monotonic. In addition, continuous equality constraints, such as symmetry, $\hat{y}(x) = \hat{y}(-x)$ can be relaxed to $|\hat{y}(x) - \hat{y}(-x)| \leqslant \epsilon$.

V. Numerical Experiments

In this section, several experiments are performed to validate the proposed method. For each experiment, we used 10-fold cross validation to find the optimal size of RBF network h, and RBF centers and widths were selected by k-center and a simple heuristic relationship [45], respectively.

A. Prior Information on Properties of Regression Model in Synthetic Examples

In this example, we use the $\sin c = \sin(x)/x$ function to generate data. Twenty-one training data and 200 testing data are generated randomly from the sinc function added with Gaussian noise $N(0, 0.2^2)$.

The specific types of prior information are given as follows.

1) Prior on extreme points

$$\begin{cases} \hat{y}(0) = 1 \\ \hat{y}'(0) = 0. \end{cases} \tag{15}$$

2) Prior on the symmetry of function

$$\hat{y}(-x_j) = \hat{y}(x_j), x_j \in [-10, 10], j = 1, 2, \ldots, N_s \tag{16}$$

where $\{x_1, \ldots, x_{N_s}\}$ are from discretization method [11].

3) Prior on the ranking list

$$\hat{y}(3) \geqslant \hat{y}(9). \tag{17}$$

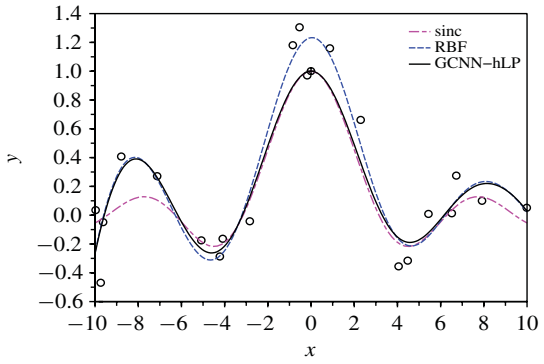


Fig. 4. Example A: 1) Prior information on extreme point (15). $\oplus$ is point (0, 1).

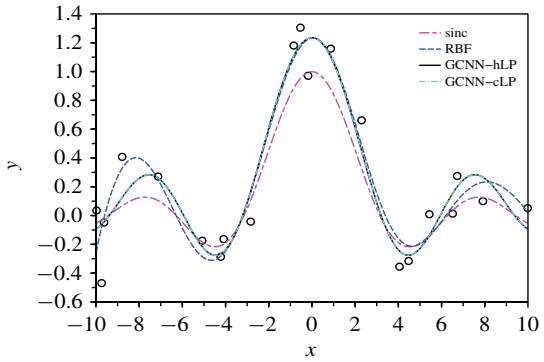


Fig. 5. Example A: 2). Prior information on symmetry of (16). The value of parameter is $\epsilon = 0.0001$.

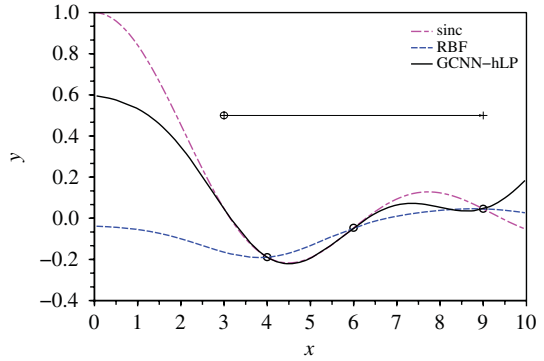


Fig. 6. Example A: 3). Prior information on ranking list (17). The value of point $\oplus$ is no less than $+$'s.

4) Prior on the upper bound function

$$\hat{y}(x_j) \leqslant \begin{cases} 1, & |x_j| \leqslant 1 \\ \frac{1}{|x_j|}, & |x_j| > 1 \\ x_j \in [-10, 10], & j = 1, 2, \ldots, N_s \end{cases} \tag{18}$$

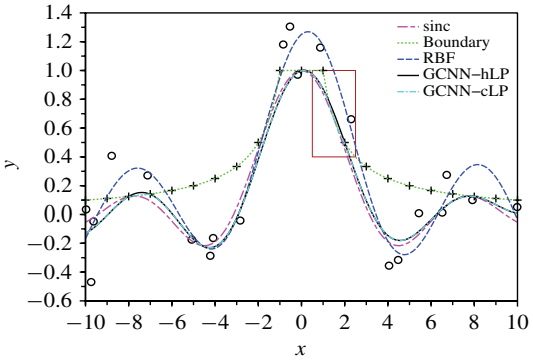
where $\{x_1, \ldots, x_{N_s}\}$ are from discretization method [11].

5) Prior on monotonicity: Method of derivatives

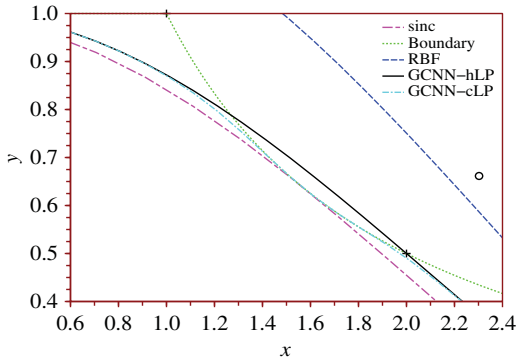
$$\hat{y}'(x_j) \geqslant 0, x_j \in [-4, 0], \quad j = 1, 2, \ldots, N_s. \tag{19}$$

Method of ranking list

$$\begin{cases} \hat{y}(x_1) \leqslant \hat{y}(x_2) \leqslant \cdots \leqslant \hat{y}(x_{N_s}) \\ -4 \leqslant x_1 \leqslant x_2 \leqslant \cdots \leqslant x_{N_s} \leqslant 0 \end{cases} \tag{20}$$



(a)



(b)

Fig. 7. Example A: 4). Prior information on upper bound function (18). The value of parameter is $\epsilon = 0.0001$. (a) Output of RBF and GCNN-*h*LP. $+$ means the sampling points. (b) Magnification of the square box (dark red) in (a).

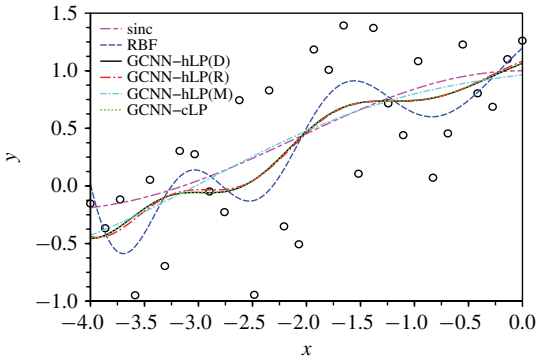


Fig. 8. Example A: 5) Prior information on monotonicity. The value of parameter is $\epsilon = 0.0001$. GCNN-*h*LP(D) means (19) is incorporated into the proposed model, while GCNN-*h*LP(R) and GCNN-*h*LP(M) mean (20) and (21), respectively.

where $\{x_1, \ldots, x_{N_s}\}$ are from the discretization method [11].

Method of monotonous function [36]

$$\begin{aligned} \min_{\mathbf{W}} : \quad & \frac{1}{2}||\mathbf{y} - \mathbf{\Phi W}||^2 \\ \text{s.t.} \quad & \omega_i \geqslant 0, \quad i = 1, \ldots, h \end{aligned} \tag{21}$$

where basis function $\phi_t(\cdot)$ is the sigmoid function. Thirty training data from sinc function at interval $x \in [-4, 0]$ with noise $N(0, 0.5^2)$ are used.

TABLE III

RESULTS OF RBF WITHOUT PRIOR AND GCNN-LP WITH DIFFERENT PRIOR INFORMATION IN sinc FUNCTION IN SECTION V-A. N_{RBF} MEANS THE SIZE OF RBF. N_G AND N_M ARE THE SIZE OF GCNN-LP AND GCNN-cLP, RESPECTIVELY. ($\epsilon = 0.0001$)

Linear Constraints	Prior information (Eq)	sinc Examples						
		RBF			GCNN-LP			
		N_{RBF}	MSE_{train}	MSE_{test}	$(N_G + N_M)$	MSE_{train}	MSE_{test}	$MSE_{GCNN-cLP}$
Hard equality	Extreme points (15)	7	0.0258	0.0629	7	0.0359	**0.0517**	-
	Symmetry (16)	7	0.0258	0.0629	(10 + 0)	0.0267	0.0552	**0.0552**
Hard inequality	Ranking list (17)	3	1.25E-16	0.351	3	4.65E-16	**0.214**	-
	Boundary (18)	7	0.0258	0.0629	(9 + 12)	0.0405	0.0418	**0.0414**
	Monotonicity (Derivative) (19)	6	0.224	0.284	(6 + 6)	0.250	0.251	0.249
	(Ranking) (20)	6	0.224	0.284	6	0.248	0.251	-
	(Monotonous function) (21)	6	0.224	0.284	6	0.269	**0.247**	-
	Multioutput $y_1(x)$ (22)	7	0.0258	0.0629	(10 + 0)	0.0312	0.0535	**0.0535**
	Multioutput $y_2(x)$ (22)	8	0.0272	0.0539	(10 + 0)	0.0387	0.0467	**0.0467**
	Noisy ranking list (23)	7	0.0258	0.0629	10	0.100	0.0987	-
Soft	Noisy ranking list (23)	7	0.0258	0.0629	10	0.0322	**0.0530**	-

6) Prior on dependent multioutput models

$$\begin{aligned} \min_{\mathbf{W}^k} \quad & \sum_{k=1}^{2} \beta^k ||\mathbf{y}^k - \mathbf{\Phi}^k \mathbf{W}^k||^2 \\ \text{s.t.} \quad & \hat{y}^{1\prime}(x_j) = \hat{y}^2(x_j), \\ & x_j \in [-10, 10], \quad j = 1, 2, \ldots, N_s \end{aligned} \tag{22}$$

where $\hat{y}^k(\mathbf{x}) = \mathbf{\Phi}^k(\mathbf{x})\mathbf{W}^k, k = 1, 2$ are the multi-output regression models. One set of training data is from $y^1 = \text{sinc} + \delta^1_{noise}$ and the other is from $y^2 = d(\text{sinc})/dx + \delta^2_{noise}$, where δ^1_{noise} and δ^2_{noise} are from noise $N(0, 0.2^2)$. $\{x_1, \ldots, x_{N_s}\}$ are from the discretization method [11].

7) Prior on a noisy ranking list

$$\begin{cases} \hat{y}\left(x_i^p\right) \geqslant \hat{y}\left(x_j^p\right), & \text{if} \quad y_i^p \geqslant y_j^p \\ \hat{y}\left(x_i^p\right) < \hat{y}\left(x_j^p\right), & \text{otherwise} \end{cases} \tag{23}$$

where $(x_i^p, y_i^p)|_{i=1,2,\ldots,10}$ are from sinc with additive noise $N(0, 0.2^2)$.

Figs. 4–10 show the outputs of RBF with and without the prior constraints (15)–(23), respectively. In addition, three different forms, (19)–(21), are used to incorporate the monotonicity priors into the regression model in Fig. 8. This example demonstrates that LPs may have several forms for embedding into the model. While the GCNN-LP approach proposed in this paper provides a structural and uniform means for embedding priors, the other forms are better examined if possible. The final selection of embedding approaches is generally problem dependent. A monotonous function seems to be the best approach for its guaranteed monotonicity (and the smallest testing error in this example), but this approach cannot be applicable for sinc function within a local range. On the contrary, the other two methods are suitable to imposing the prior locally. Table III shows the results of GCNN-LP and RBF models with and without using prior information, respectively. Generally, GCNN-LP models show a better performance over RBF ones because they apply the prior information for modeling. However, one can observe that

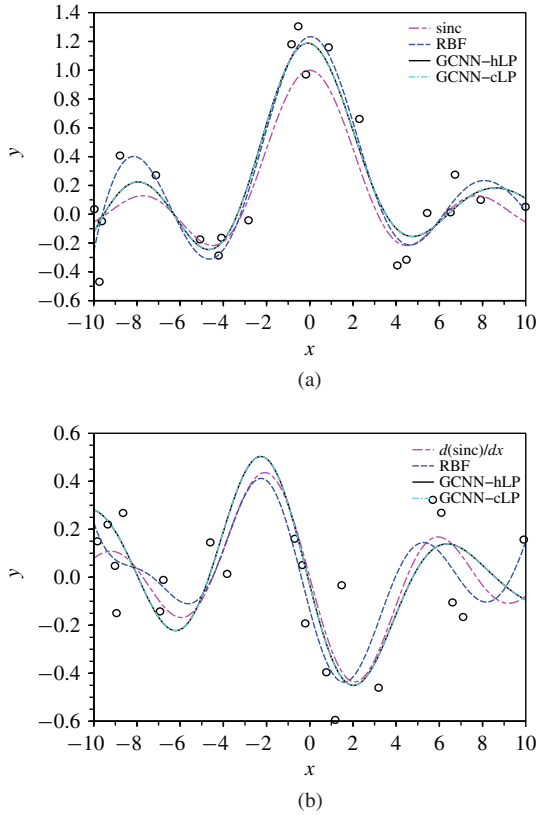


Fig. 9. Example A: 6). Prior information on multioutput (22). The values of parameters are $\beta^1 = \beta^2 = 0.5, \epsilon = 0.0001$. (a) Approximation of $y^1(x) = \text{sinc}$. (b) $y^2(x) = d(\text{sinc})/dx$.

the GCNN-hLP model shows a poorer prediction than that of the RBF (0.0987 > 0.0629 on MSE_{test}) when the ranking list prior is given. We cannot guarantee the outperformance of models by applying the conventional, or hard, constraints from prior information. For comparisons, by using the soft constraints on the prior of noisy ranking list, we obtain an improved performance from GCNN-LP model ($MSE_{test} = 0.0530$). This numerical example demonstrates a challenge

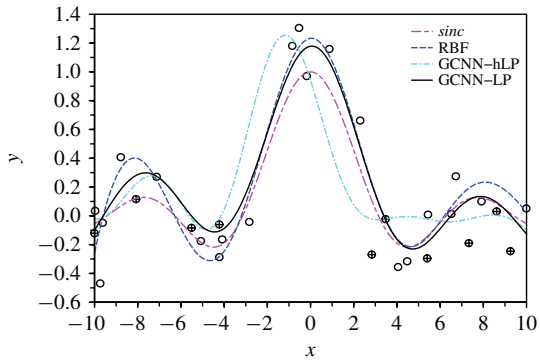


Fig. 10. Example A: 7). Prior information on noisy ranking list (23). ⊕ means the ranked points as prior information.

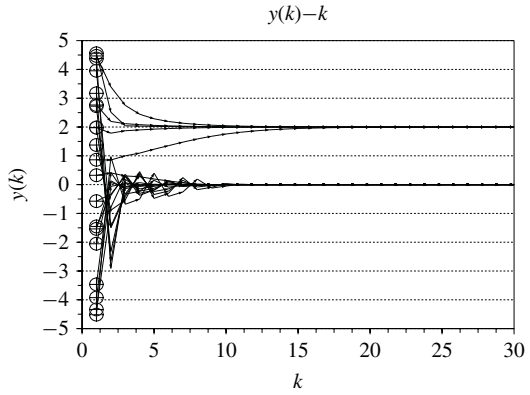


Fig. 11. Example B: Prior information on equilibrium states in nonlinear dynamic process. Twenty initial states are given randomly, and the figure shows that, as times changed, almost all states are attracted into two states: 0 and 2. ⊕ means the initial point.

of embedding prior information into models properly and positively.

B. Prior Information on Equilibrium States in Nonlinear Dynamic Process

In this example, we consider the discrete-time nonlinear dynamic system [11], [49] described by the second-order difference equation

$$y(k+1) = \frac{y(k)y(k-1)[y(k)+2.5]}{1+y^2(k)+y^2(k-1)} + u(k) \qquad (24)$$

where $u(k)$ and $y(k)$ are the input and output of the system at time k, respectively. In order to train the model, some analyses of the (24) are required. Assuming no input in the system ($u(k) = 0$), we assign the the initial states ($y(0), y(1)$) randomly and observe the output of the system after time 30 ($k = 30$). Fig. 11 shows that almost all states are attracted into one of two states (0 or 2) over time. In other words, the output $y(k)$ is either the sequence $\{0\}$ or the sequence $\{2\}$ for the input $u(k) = 0$. And it is in accordance with the equilibrium states described in [49]. Therefore, according to analysis in above, two states ($y(k) = 0, y(k-1) = 0$ and $y(k) = 2, y(k-1) = 2$) in state space of the enforced system (for $u(k) = 0$) are used for prior information in this example. The model $y(k+1) = f([y(k), y(k-1), u(k)]^T)$ is used for predicting the new state of the nonlinear dynamic system, and the prior information can be reformulated as

$$\begin{cases} f\left([0,0,0]^T\right) = 0 \\ f\left([2,2,0]^T\right) = 2. \end{cases} \qquad (25)$$

We used 100 of the input/output data pairs $[y(k), y(k-1), u(k), y(k+1)]$ ($k = 1, \ldots, 100$) generated from the (24) as training data, where the initial condition is $y(1) = 1, y(0) = 1$ and the input sequence $u(k)$ is from a uniformly distributed random in the interval $[-2, 2]$ added with Gaussian noise $N(0, 0.1^2)$. And the testing data are also 100 of the input/output data pairs from the same system, but the initial condition and the input $u(k)$ are replaced by $y(1) = 0.5, y(0) = 0.5$, and $u(k) = \sin(2\pi k/25)$, respectively. Standard RBF and SVR are applied on these data without prior information, while our model with prior information on the equilibrium state (25).

The program runs 100 times for getting 100 groups of test MSEs. Table IV shows the average and standard deviation of the 100 test MSEs. From the table, we can see that the proposed model not only can decrease the average of test MSEs, which means the accuracy has been improved, but also can decrease the standard deviation of the test MSEs, which means that the generalization capability has been improved. Therefore, the suggested method can achieve higher performance when prior information on equilibrium states is used for learning.

TABLE IV
COMPARISON OF MODELING PERFORMANCE FOR THE STANDARD RBF, SVR, AND GCNN-*h*LP IN SECTION V-B. THE ERROR MEANS THE AVERAGE AND STANDARD DEVIATION OF THE TEST MSES ON THE 100 GROUPS OF TEST DATA

Method	Total number of free parameters	MSE (Mean/Std.)
Standard RBF	50	0.695/0.814
SVR [11]	-	0.445/0.457
SVR + Priors [11]	-	0.354/0.356
GCNN-*h*LP	50	**0.273/0.272**

TABLE V
CHARACTERISTICS OF DATASETS

Dataset name	N	d	Target feature
Pollution	60	16	Mortality rate
Boston Housing	506	14	MEDV
California Housing	20640	9	House price

C. Prior Information on Ranking List in Real-World Data

In this section, a comparative study of GCNN-LP, conventional RBF, and SVR algorithms using real-world datasets is conducted. Table V gives the characteristics of the datasets, where N is the number of available observations and d denotes the number of attributes, which means the dimensions of the observations. In the last column, the names of target features are presented.

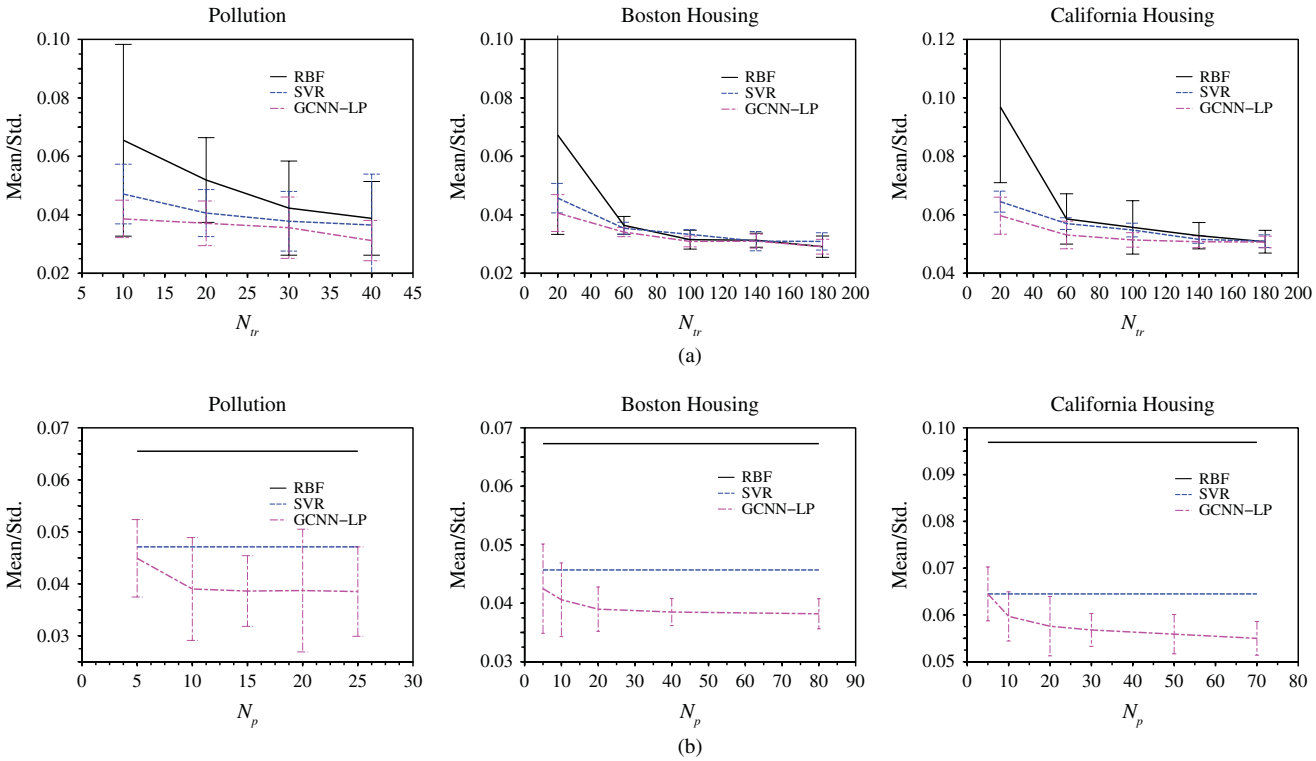


Fig. 12. Example C: Benchmark example. The influences from different N_{tr} and N_p on the performance of proposed model are shown. The values of N_{tr} and N_p are from Table VI. When one is fixed, the benefit of the ranking varies from changing the other one. (a) Results of changing N_{tr}. (b) Results of changing N_p.

TABLE VI

COMPARISON OF MODELING PERFORMANCE FOR RBF, SVR, AND GCNN-LP IN SECTION V-C. THE ERROR MEANS AND STANDARD DEVIATION OF THE MSE ON THE 20 GROUPS OF TEST DATA. N_p IS THE NUMBER OF PRIOR OBSERVATIONS, CI IS THE SHORT OF CONFIDENCE INTERVALS

Dataset	N_{tr}	N_{te}	MSE (Mean/Std.)			p-value, CI	
			RBF	SVR	GCNN-LP (N_p)	$\delta = \mu_r - \mu_g$	$\delta = \mu_s - \mu_g$
Pollution	10	40	0.0655/0.0328	0.0471/0.0102	**0.0386/0.0064**(10)	0.00088, [0.0118, 0.0421]	0.0030, [0.0031, 0.014]
Boston Housing	20	466	0.0673/0.034	0.0457/0.0050	**0.0406/0.0063**(20)	0.0017, [0.0107, 0.0426]	0.0072, [0.0015, 0.0088]
California Housing	20	20600	0.0969/0.0259	0.0645/0.0035	**0.0597/0.0070**(20)	0.00013, [0.025, 0.0493]	0.0096, [0.0012, 0.0084]

In fact, in these real-world problems, much ranking information can be created and available easily. As an example, we consider the task of predicting real estate prices. The users or experts always rate the housing prices by giving the overall characteristics of houses. They can independently give the scoring or ranking of these housing prices roughly according to the features of these houses, such as, area, location, number of bedrooms, etc. For example, if everything else is roughly equal, then the scoring of a three-bedroom house is higher than a two-bedroom one. However, in this paper, in order to demonstrate the effectiveness of the suggested method, N_p observations, which are one part of dataset and are not used in training data or testing data, will be used to generate the simulated ranking information. Thus the whole dataset will be divided into three parts randomly and independently: N_p prior observations, N_{tr} training data, and N_{te} testing data. Note that the true values of N_p prior observations are never given out.

The divisions in different data sets are shown in Table VI. Standard RBF and SVR are trained on these data without any prior, while our model is trained with the ranking information. In contrast, the target feature of each dataset was normalized in [0, 1] [50]. We used 10-fold cross validation to find the optimal weights $\gamma = \gamma_k, \forall k \in E \cup I$. The parameters were tuned on a 10×10 logarithmic grid in $10^{-3} \leqslant \gamma \leqslant 10^3$. The program ran 20 times for getting 20 groups of the test MSEs. Table VI shows the average and standard deviation of the 20 test MSEs. For a comparison, GCNN-hLP is also employed but shows a large MSE, which is not included in this example. However, in order to illustrate statistical significance of GCNN-LP, we add the p-value and CI measures. Assume that 20 groups of test MSEs (RBF, SVR, and GCNN-LP) are from populations with means μ_r, μ_s, and μ_g, respectively. Let us test the null hypothesis that $H_0 : \delta = 0$ versus $H_1 : \delta \neq 0$. A paired t-test is applied to obtain p-value and CI. Table VI shows that the results are statistically significant (significant level is 0.05).

In addition, because the ranking information just contains the order of these points, the ranking as prior information is weaker, and the improvement by this prior is limited. But one wants to know what influence it has on the regression

model if one changes the number of training data, and what if one changes the number of ranking data. Fig. 12 shows the experimental results for each dataset. One can observe that increasing the training data and ranking data will generally improve the performance of the proposed approach . However, the benefit of incorporating priors becomes more significant when training data are scarce.

VI. Conclusion

In this paper, a new structure of GCNN-LP has been proposed for RBF networks with LPs by means of linear equality and inequality constraints. Hard constraints and soft constraints were discussed to apply the priors as the properties of the desired output and the priors with noise, respectively. Direct elimination and least squares have been used to train the new structure of the network. Compared with the conventional RBF networks, the suggested modeling approach may add transparency through a structural mode and can improve the approximation and generalization capability especially in difficult cases, such as an insufficient size of the training set or high noise in the training data. Because discretization-based methods cannot guarantee that the approximate function satisfies the continuous constraints, a modification of GCNN-LP was proposed to handle this problem. Finally, numerical results from synthetic examples and real-world regression problems demonstrated the effectiveness of the proposed modeling approach.

The GCNN-LP proposed in this paper demonstrates a methodology of incorporating and processing LPs for nonlinear regression models. The principle behind the methodology is neither limited to RBF networks as the model structures, nor to regression problems. The challenges, varying from different model structures, such as SVMs or deep-learning machines, will be the future work.

Appendix

Proof of Corollary 1: Because Algorithm 1 is terminated within finite steps, the only entrance to the termination of Algorithm 1 is Case 2. If the Case 2 is implemented, then the iteration point must be the feasible point of generalized constraint neural network-linear prior (GCNN-LP). And if the condition of Case 2 ($\lambda_j^{(k)} \geqslant 0, \forall j \in S_k \cap I$) is satisfied, according to the Karush-Kuhn-Tucke (KKT) lemma, then we can conclude that the point of termination must be a KKT point of GCNN-LP. ■

Proof of Corollary 2: Assume Algorithm 1 is not terminated at finite steps. Due to the finite number of constraints, neither the infinite decrease of the number of the working set S_k without increasing, nor infinite increase without decreasing will happen. Therefore, there must exist a k_0, for any $k \geqslant k_0$, such that exactly one of the following two statements is true.

S1: $\mathbf{W_k}$ is not a feasible point.

S2: $\mathbf{W_k}$ is a feasible point, and $\exists j \in S_k \cap I$, $\lambda_j^{(k)} < 0$.

That is to say, we only prove that if we view the condition above as a pseudo-proposition, then the proposed algorithm is terminated within finite steps. According to the view above, there must exist $k_2 > k_1 > k_0$, such that $\mathbf{W_{k_1}}$ satisfies the statement S2 and $\mathbf{W_{k_2}}$ satisfies the statement S1. And assume the optimal working set of original problem is S^*. Because $\mathbf{W_{k_1}}$ satisfies the statement S2, we have $S^* \subseteq S_{k_1}$, $j \in S_{k_1}$, and $j \notin S_{k_1+1}$.

1) If $\mathbf{W_{k_1+1}}$ satisfies the statement S1, then we can know that S_{k_1+1} is not a feasible set and $S^* \nsubseteq S_{k_1+1}$. Therefore, we can know that j is the active constraint, and according to the KKT theory we have $\lambda_j^{(k)} \geqslant 0$. Thus, $\mathbf{W_{k_1}}$ does not satisfy the statement S2, which is contradictory to the condition.
2) Else, if $\mathbf{W_{k_1+1}}$ satisfies the statement S2, then the problem is changed to $\mathbf{W_{k_1+1}}$ satisfies the statement S2 and $\mathbf{W_{k_2}}$ satisfies the statement S1. Because S_{k_2} is finite, there must exist a minimum t and $t < \infty$, such that $k_1 + t = k_2$. Thus, it is similar to the proof in above, and there exists the same contradictory counterpoint to the condition.

Therefore, Algorithm 1 is terminated at finite steps similar to the active set described in [48]. ■

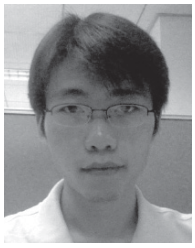
Acknowledgment

The authors would like to thank C. Ocier for editorial proofreading. They would also like to thank the anonymous reviewers and the editors for their valuable comments and suggestions.

References

[1] R. O. Duda, P. E. Hart, and D. Stork, *Pattern Classification*, 2nd ed. New York: Wiley, 2001.
[2] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 1999.
[3] B. G. Hu, H. B. Qu, Y. Wang, and S. H. Yang, "A generalized-constraint neural network model: Associating partially known relationships for nonlinear regressions," *Inf. Sci.*, vol. 179, no. 12, pp. 1929–1943, May 2009.
[4] W. Pedrycz and R. A. Aliev, "Logic-oriented neural networks for fuzzy neurocomputing," *Neurocomputing*, vol. 73, nos. 1–3, pp. 10–23, Dec. 2009.
[5] E. Krupka and N. Tishby, "Incorporating prior knowledge on features into learning," in *Proc. Int. Conf. Articial Intell. Stat.*, 2007, pp. 1–8.
[6] V. Vapnik and A. Vashist, "A new learning paradigm: Learning using privileged information," *Neural Netw.*, vol. 22, nos. 5–6, pp. 544–557, Jul. 2009.
[7] F. Lauer and G. Bloch, "Incorporating prior knowledge in support vector machines for classification: A review," *Neurocomputing*, vol. 71, nos. 7–9, pp. 1578–1594, Mar. 2008.
[8] B. G. Hu, Y. Wang, S. H. Yang, and H. B. Qu, "How to add transparency to artificial neural networks," *Pattern Recognit. Artif. Intell.*, vol. 20, no. 1, pp. 72–84, 2007.
[9] T. M. Mitchell, *Machine Learning*. New York: McGraw-Hill, 1997.
[10] T. Yu, "Incorporating prior domain knowledge into inductive machine learning: Its implementation in contemporary capital markets," Ph.D. thesis, Faculty Inf. Technol., Univ. Technology, Sydney, Australia, 2007.
[11] F. Lauer and G. Bloch, "Incorporating prior knowledge in support vector regression," *Mach. Learn.*, vol. 70, no. 1, pp. 89–118, 2008.
[12] L. M. Fu, "Rule generation from neural networks," *IEEE Trans. Syst., Man, Cybern.*, vol. 24, no. 8, pp. 1114–1124, Aug. 1994.
[13] A. Tickle, R. Andrews, M. Golea, and J. Diederich, "The truth will come to light: Directions and challenges in extracting the knowledge embedded within trained artificial neural networks," *IEEE Trans. Neural Netw.*, vol. 9, no. 6, pp. 1057–1068, Nov. 1998.
[14] R. Setiono, W. K. Leow, and J. M. Zurada, "Extraction of rules from artificial neural networks for nonlinear regression," *IEEE Trans. Neural Netw.*, vol. 13, no. 3, pp. 564–577, May 2002.
[15] D. Martens, B. Baesens, and T. V. Gestel, "Decompositional rule extraction from support vector machines by active learning," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 2, pp. 178–191, Feb. 2009.

[16] T. Q. Huynh and J. A. Reggia, "Guiding hidden layer representations for improved rule extraction from neural networks," *IEEE Trans. Neural Netw.*, vol. 22, no. 2, pp. 264–275, Feb. 2011.
[17] T. Mitchell, "The need for biases in learning generalizations," Dept. Comput. Sci., Rutgers Univ., Cream Ridge, NJ, Tech. Rep. CBM-TR-117, 1980.
[18] T. Poggio and F. Girosi, "Networks for approximation and learning," *Proc. IEEE*, vol. 78, no. 9, pp. 1481–1497, Sep. 1990.
[19] V. Vapnik, *Statistical Learning Theory*. New York: Wiley, 1998.
[20] Y. Abu-Mostafa, "A method for learning from hint," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 5. 1992, pp. 73–80.
[21] G. Towell and J. Shavlik, "Knowledge-based artificial neural networks," *Artif. Intell.*, vol. 70, nos. 1–2, pp. 119–165, Oct. 1994.
[22] J.-S. R. Jang, "ANFIS: Adaptive-network-based fuzzy inference system," *IEEE Trans. Syst., Man, Cybern.*, vol. 23, no. 3, pp. 665–685, May–Jun. 1993.
[23] D. J. C. MacKay, "A practical Bayesian framework for backpropagation networks," *Neural Comput.*, vol. 4, no. 3, pp. 448–472, 1992.
[24] D. Psichogios and L. H. Ungar, "A hybrid neural network-first principles approach to process modeling," *AIChE J.*, vol. 38, no. 10, pp. 1499–1511, Oct. 1992.
[25] M. L. Thompson and M. A. Kramer, "Modeling chemical processes using prior knowledge and neural networks," *AIChE J.*, vol. 40, no. 8, pp. 1328–1340, Aug. 1994.
[26] B. Scholkopf, P. Simard, A. Smola, and V. Vapnik, "Prior knowledge in support vector kernels," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 10. 1998, pp. 640–646.
[27] P. Niyogi, F. Girosi, and T. Poggio, "Incorporating prior information in machine learning by creating virtual examples," *Proc. IEEE*, vol. 86, no. 11, pp. 2196–2209, Nov. 1998.
[28] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul, "An introduction to variational methods for graphical models," in *Learning in Graphical Models*, M. I. Jordan, Ed. Cambridge, MA: MIT Press, 1999.
[29] Y. J. Qu and B. G. Hu, "RBF networks for nonlinear models subject to linear constraints," in *Proc. IEEE Int. Conf. Granular Comput.*, Nanchang, China, Aug. 2009, pp. 482–487.
[30] Y. J. Qu, B. Dai, and B. G. Hu, "Neural-network based regression model with prior from ranking information," in *Proc. Int. Joint Conf. Neural Netw.*, Barcelona, Spain, Jul. 2010, pp. 3005–3012.
[31] B. Scholkopf and A. Smola, *Learning with Kernel, Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA: MIT Press, 2002.
[32] S. Chen, C. J. Harris, and X. Hong. (2009). *Grey-Box Radial Basis Function Modelling: The Art of Incorporating Prior Knowledge* [Online]. Available: http://eprints.ecs.soton.ac.uk/17774/1/gbrbfSSP09.pdf
[33] X. J. Zhu and B. G. Andrew, "Kernel regression with order preferences," in *Proc. Assoc. Advance. Artif. Intell.*, 2007, pp. 681–686.
[34] J. Bi, K. Bennett, M. Embrechts, C. Breneman, and M. Song, "Dimensionality reduction via sparse support vector machines," *J. Mach. Learn. Res.*, vol. 3, pp. 1229–1243, Mar. 2003.
[35] V. Gomez-Verdejo, M. Martinez-Ramon, J. Arenas-Garcia, M. Lazaro-Gredilla, and H. Molina-Bulla, "Support vector machines with constraints for sparsity in the primal parameters," *IEEE Trans. Neural Netw.*, vol. 22, no. 8, pp. 1269–1283, Aug. 2011.
[36] A. Minin, M. Velikova, B. Lang, and H. Daniels, "Comparison of universal approximators incorporating partial monotonicity by structure," *Neural Netw.*, vol. 23, no. 4, pp. 471–475, May 2010.
[37] M. Velikova, H. Daniels, and A. Feelders, "Mixtures of monotone networks for prediction," *Int. J. Comput. Intell.*, vol. 3, no. 3, pp. 204–214, 2006.
[38] C. Dugas, Y. Bengio, F. Bélisle, C. Nadeau, and R. Garcia, "Incorporating functional knowledge in neural networks," *J. Mach. Learn. Res.*, vol. 10, pp. 1239–1262, Jun. 2009.
[39] H. Daniels and M. Velikova, "Monotone and partially monotone neural networks," *IEEE Trans. Neural Netw.*, vol. 21, no. 6, pp. 906–917, Jun. 2010.
[40] O. L. Mangasarian, J. W. Shavlik, and E. W. Wild, "Knowledge-based kernel approximation," *J. Mach. Learn. Res.*, vol. 5, pp. 1127–1141, Sep. 2004.
[41] O. L. Mangasarian and E. W. Wild, "Nonlinear knowledge in kernel approximation," *IEEE Trans. Neural Netw.*, vol. 18, no. 1, pp. 300–306, Jan. 2007.
[42] Z. Sun, Z. K. Zhang, H. G. Wang, and M. Jiang, "Cutting plane method for continuously constrained kernel-based regression," *IEEE Trans. Neural Netw.*, vol. 21, no. 2, pp. 238–247, Feb. 2010.
[43] J. Weston, O. Chapelle, A. Elisseeff, B. Scholkopf, and V. Vapnik, "Kernel dependency estimation," in *Proc. Adv. Neural Inf. Syst.*, vol. 15. 2003, pp. 873–880.
[44] S. Ferrari and M. Jensenius, "A constrained optimization approach to preserving prior knowledge during incremental training," *IEEE Trans. Neural Netw.*, vol. 19, no. 6, pp. 996–1009, Jun. 2008.
[45] M. X. Zhu and D. L. Zhang, "Study on the algorithms of selecting the radial basis function center," *J. Anhui Univ.*, vol. 24, no. 1, pp. 72–78, 2000.
[46] R. F. Stengel, *Optimal Control and Estimation*. New York: Dover, 1994, pp. 36–41.
[47] D. P. Bertsekas, *Constrained Optimization and Lagrange Multiplier Methods*. Belmont, MA: Athena Scientific, 1996.
[48] R. Fletcher, *Practical Methods of Optimization*, 2nd ed. Chichester, U.K.: Wiley, 1987.
[49] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neural Netw.*, vol. 1, no. 1, pp. 4–27, Mar. 1990.
[50] M. Bortman and M. Aladjem, "A growing and pruning method for radial basis function networks," *IEEE Trans. Neural Netw.*, vol. 20, no. 6, pp. 1039–1045, Jun. 2009.



Ya-Jun Qu received the B.S. degree in electronic information engineering from Soochow University, Suzhou, China, in 2007. He studied computer sciences with the National Laboratory of Pattern Recognition (NLPR), Institute of Automation, Chinese Academy of Sciences, Beijing, China. He is currently pursuing the Ph.D. degree in computer sciences at NLPR.

His current research interests include machine learning theory and neural networks.

Bao-Gang Hu (M'94–SM'99) received the M.Sc. degree from the University of Science and Technology, Beijing, China, and the Ph.D. degree from McMaster University, Hamilton, ON, Canada, both in mechanical engineering, in 1983 and 1993, respectively.

He was a Research Engineer and Senior Research Engineer at C-CORE, Memorial University of Newfoundland, St. John's, NF, Canada, from 1994 to 1997. From 2000 to 2005, he was the Chinese Director of computer science, control, and applied mathematics with the Chinese–French Joint Laboratory, National Laboratory of Pattern Recognition (NLPR), Institute of Automation, Chinese Academy of Sciences, Beijing, China. He is currently a Professor at NLPR. His current research interests include pattern recognition and plant growth modeling.