This is the Pre-Published Version.
The following publication Z. Deng, K. Choi, Y. Jiang and S. Wang, "Generalized Hidden-Mapping Ridge Regression, Knowledge-Leveraged Inductive Transfer Learning for Neural Networks, Fuzzy Systems and Kernel Methods," in IEEE Transactions on Cybernetics, vol. 44, no. 12, pp. 2585-2599, Dec. 2014 is available at https://dx.doi.org/10.1109/TCYB.2014.2311014.

Generalized Hidden-mapping Ridge Regression, Knowledge-leveraged Inductive Transfer Learning for Neural Networks, Fuzzy Systems and Kernel Methods

Zhaohong Deng, *IEEE Senior Member*, Kup-Sze Choi, *IEEE Member*, Yizhang Jiang, *IEEE Member*, Shitong Wang

***Abstract*—Inductive transfer learning has attracted increasing attention for the training of effective model in the target domain by leveraging the information in the source domain. However, most transfer learning methods are developed for a specific model, such as the commonly used support vector machine (SVM), which makes the methods applicable only to the adopted models. In this regard, the generalized hidden-mapping ridge regression (GHRR) method is introduced in order to train various types of classical intelligence models, including neural networks, fuzzy logical systems and kernel methods. Furthermore, the knowledge-leverage based transfer learning mechanism is integrated with GHRR to realize the inductive transfer learning method called Transfer GHRR (TGHRR). Since the information from the induced knowledge is much clearer and more concise than that from the data in the source domain, it is more convenient to control and balance the similarity and difference of data distributions between the source and target domains. The proposed GHRR and TGHRR algorithms have been evaluated experimentally by performing regression and classification on synthetic and real world datasets. The results demonstrate that the performance of TGHRR is competitive with or even superior to existing state-of-the-art inductive transfer learning algorithms.**

***Index Terms*—Generalized hidden-mapping ridge regression, Inductive transfer learning, Knowledge-leverage, Neural networks, Kernel methods, Fuzzy systems, Regression, Classification.**

I. INTRODUCTION

Transfer learning has been studied extensively for different applications (e.g. web text classification) in recent years [1]. As illustrated in Fig. 1, it is a learning procedure to develop an effective model by using the data of the target domain and leveraging the useful information from the source domains simultaneously (definition of the domains is given in Table I). The existing work about transfer learning can be categorized generally into three main types: 1) transfer learning for classification [2-15]; 2) transfer learning for regression [16-20]; and 3) transfer learning for unsupervised learning [21-23] (e.g. clustering and dimensionality reduction [14]). Besides, based on the differences in settings, transfer learning can also be divided into *inductive transfer learning* and *transductive transfer learning*. A few data in the target domain are labeled in the former approach while all the data in the target domain are unlabeled in the latter. In this study, we focus on inductive transfer learning.

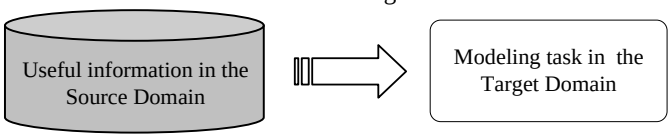


Fig.1 An illustration of transfer learning for modeling in the target domain

TABLE I DEFINITION OF DOMAINS IN TRANSFER LEANING

Terms	Explanations
Domain	A domain is a scene where a modeling task is to be accomplished. It is usually characterized by the *data* collected and the *learning task* to be performed in this domain.
Target domain	In transfer learning, it is referred to as a domain containing insufficient data or data that are difficult to be used for proper modeling, while a modeling task is required to be effectively implemented.
Source domain	It is the domain related to the target domain, with similar data distribution and/or learning task to some extent. There may be differences between the target domain and the source domain, but it is assumed that the source domain can provide some useful information for the modeling task in the target domain.

The use of inductive transfer learning in real world applications is becoming more common. One of the examples is the modeling of fermentation process [24]. In the target domain of certain microbiological fermentation process, the data collected may be insufficient for modeling. Some of the required data may be missing due to deficiency of the sensor setup. Thus, we cannot effectively model the fermentation process for this domain with the data collected. However, if the data available from other similar microbiological fermentation processes are available, they can be considered as source domains for the target domain. Transfer learning can then be exploited by making use of the information from the source domain to improve the modeling result of the target domain (see Fig. 1), thereby resulting in a model with better generalization capability. In this case, inductive transfer learning is an effective solution to the corresponding modeling task because it can enhance the model by leveraging the information available from the source domains, such as the data collected in other time frames or with other setups.

Many modeling techniques have been adopted to implement inductive transfer learning for modeling task based on different intelligence models, including support vector machine (SVM), neural networks, fuzzy logic systems and so on. For example, SVM has been extensively used to develop different inductive and transductive transfer learning methods [15, 27]. Although many inductive transfer learning algorithms have been proposed and shown to be effective in various applications, an obvious issue is

This work was supported in part by the Hong Kong Research Grants Council (PolyU 5134/12E), the National Natural Science Foundation of China (61170122).

Z.H. Deng is with the Department of Biomedicine, University of California, Davis and School of Digital Media, Jiangnan University, Wuxi, China. (e-mail: zhdeng@ucdavis.edu).

K.S. Choi is with the Centre for Smart Health, the Hong Kong Polytechnic University(e-mail: kschoi@ieee.org)

Y.Z. Jiang and S.T. Wang are with the School of Digital Media, Jiangnan University, Wuxi 214122, China (e-mail: wxwangst@aliyun.com).

that most of the algorithms are developed only for a specific model that make it difficult to use the related transfer learning mechanism for other models and restrict the applicability. Hence, there is a demand for generalized transfer learning modeling methods that can be readily used for most classical models, e.g. neural networks, fuzzy systems and kernel methods. To meet this challenge, we propose a generalized ridge regression learning method, the generalized hidden-mapping ridge regression (GHRR), as well as the associated transfer GHRR (TGHRR) established on the knowledge-leverage based transfer learning mechanism in order to realize inductive transfer learning for multiple classical models, including neural networks, fuzzy systems and kernel methods.

Compared with most existing inductive transfer learning methods, which usually implement transfer learning by using the data in the source domain directly, the proposed knowledge-leverage mechanism based TGHRR has the following advantages: (1) while the data are the original information in a source domain, the knowledge can be taken as the induced information by some learning procedure from the source domain. In practical applications, information obtained from the induced knowledge is expected to be more apparent and concise than that from the data in the source domain. (2) when the data in the source domain are used directly, the data indeed may not be always appropriate for the learning task in the target domain due to potential drifting in data distributions between the source and target domains, where controlling the balance of the similarity and difference in data distributions between these two domains in the learning procedure is an issue. However, compared with the direct use of the data in the source domain, it is expected that, by using the induced knowledge of the source domain for the target domain, the influence of the source domain can be controlled more conveniently. (3) For those scenes where inductive transfer learning is required, there are usually plenty of labeled data in the source domain but few in the target domain. In this case, if the data in the source domain is directly used for the modeling task in the target domain, the trained model will over approximate the scene in the source domain, which should be avoided in the transfer learning procedure. This is still a not well-solved problem in the existing inductive transfer learning methods. In this study, since the proposed methods do not need to directly use the data in the source domain for most situations, the problem can be effectively avoided to some extent.

In conclusion, the proposed transfer learning algorithm TGHRR has two distinctive characteristics: (1) TGHRR is not restricted to a certain model but can taken as a more general method applicable for various intelligent models, such as fuzzy systems and neural networks; and (2) the knowledge-leverage mechanism realizes transfer learning from the source domain to the target domain, which makes it more convenient to control and balance the similarity and difference of data distributions between two domains.

The rest of this paper is organized as follows. Section II describes the related work, including a review of the classical inductive transfer learning methods, the existing ridge regression learning methods and the knowledge-leverage based inductive transfer learning framework. In Section III, the GHRR method is introduced and the properties are discussed. The TGHRR method is then proposed in Section IV by integrating the knowledge-leverage based inductive transfer learning mechanism. Experimental results and analyses are given in Section V. Further discussions about the proposed methods and the potential improvements are given in section VI. Finally, the paper is concluded in section VII. For clarity, a list of acronyms used in this paper is given Table II.

TABLE II THE ACRONYMS USED IN THIS PAPER

Acronym	Description
RR, KRR, DRR, GHRR, GHRR	Ridge Regression, Kernelized RR, Dual RR, Generalized Hidden-mapping RR, Transfer GHRR
FLS, TSKFLS	Fuzzy Logic Systems, Takagi-Sugeno-Kang FLS
MHFN	Multiple Hidden-layer Feedforward Neural Networks
SVM	Support Vector Machine
RKHS	Reproducing Kernel Hilbert Space
RBF, RBF-NN	Radial Basis Function, RBF-Neural Networks
KKT	Karush–Kuhn–Tucker

II. RELATED WORK

In this section, we first review the classical inductive transfer learning methods, followed by several existing ridge regression learning methods. Finally, the knowledge-leverage based transfer learning framework is introduced.

A. Inductive Transfer Learning

Definition (Inductive Transfer Learning) [1]. *Given a source domain D_S and a learning task T_S in D_S, a target domain D_T and a learning task T_T in D_T, inductive transfer learning aims to improve the learning of the target predictive function $f_T(.)$ in D_T using the knowledge in D_S and T_S, where $T_S \neq T_T$.*

Under inductive transfer learning setting, the target task is different from the source task. In this case, some labeled data in the target domain are required to induce an objective predictive model for use in the target domain. Representative inductive transfer learning algorithms can be summarized as follows [1]. (i) *Instance-transfer approach*. Dai et al. [25] proposed a boosting algorithm called TrAdaBoos, which is typical inductive transfer learning algorithm by weighting the data in the source domain. Jiang and Zhai [26] proposed a heuristic method to remove "misleading" training examples from the source domain. Liao et al. [2] proposed a new active learning method to select the unlabeled data in a target domain, which are to be labeled with the help of the source domain data. Wu and Dietterich [27] integrated the source domain (auxiliary) data with the SVM framework to improve the classification performance; (ii) *Feature-representation-transfer approach*. This is similar to common feature learning in the field of multitask learning [28]. If no labeled data in the source domain are available, unsupervised learning methods are proposed to construct the feature representation; (iii) *Parameter-transfer approach*. Lawrence and Platt [6] proposed an efficient algorithm known as MT-IVM, which is based on Gaussian Processes (GP) to handle multitask learning. Bonilla et al. [29] also investigated multitask learning in the context of GP. Schwaighofer et al. [7] proposed to use hierarchical Bayesian framework (HB) together with GP for multitask learning. Evgeniou and Pontil [30] applied the idea of HB to SVMs for multitask learning. Gao et al. [8] proposed a locally weighted ensemble learning framework to combine multiple models for transfer learning. (iv) *Relational-knowledge-transfer approach*. Mihalkova et al. [9] proposed the TAMAR algorithm that transferred relational knowledge with Markov Logic Networks (MLNs) across relational domains. MLNs [31] is a powerful formalism for statistical relational learning, which combines the compact expressiveness of first-order logic with flexibility of probability. Mihalkova and Mooney [10] extended TAMAR to the single-entity-centered setting of transfer learning. Davis and Domingos [11] proposed an approach to transfer relational knowledge based on a form of second-order Markov logic.

Although many approaches have been proposed to realize inductive learning and they have demonstrated distinctive effectiveness in different applications, there are still difficult issues to be resolved:

(1) Many of the existing algorithms are developed for a specific model only, e.g. SVM. These algorithms are thus infeasible for other models. Even though some algorithms such as TrAdaBoost are universal for different learners, the realization of this algorithm on different models, including SVM and neural networks, is so different that the use for other models is difficult and inconvenient.

(2) Most of the algorithms assume that the data in the source domain are available and these data can be used for transfer learning directly. Since drifting in data distributions between the source and target domains can possibly exist, it is difficult to maintain a balance in similarity and difference of data distributions between these two domains in the learning procedure. Besides, due to privacy protection, it may be forbidden to disclose the data in the source domain.

In this study, we will develop a new method to deal with these two issues from the viewpoint of ridge regression learning.

B. Classical Ridge Regression

Several existing ridge regression learning methods [32-34] are briefly introduced in this section, including the basic methods, kernel methods, and dual ridge regression methods. The network structure of ridge regression is also discussed. For simplicity and clarity, all the vectors in this paper are represented with the column vectors. The notations used in the text are listed in Table I (S) of the Supplementary material.

1) Basic Ridge Regression: For a given regression task and the dataset $\{\mathbf{x}_i, y_i\}$, $\mathbf{x}_i \in R^d, y \in R,\ i = 1, \cdots, N$, the idea of basic ridge regression (BRR) [32] is to obtain a linear regression model

$$y = f(\mathbf{x}) = \mathbf{x}^T \mathbf{w} \tag{1}$$

with $\mathbf{w} \in R^d, \mathbf{x} \in R^d$, by optimizing the following objective

$$\min_{\mathbf{w}} \ \frac{1}{2}\left\|\mathbf{X}\mathbf{w} - \mathbf{y}\right\|^2 + \frac{\lambda}{2}\left\|\mathbf{w}\right\|^2, \tag{2}$$

where

$$\mathbf{X} = \left[\mathbf{x}_1, \cdots, \mathbf{x}_N\right]^T \in R^{N \times d} \tag{3}$$

$$\mathbf{y} = [y_1, \cdots, y_N]^T \in R^N. \tag{4}$$

Taking the derivatives of Eq. (2) and equating them to zero gives

$$\mathbf{w}^* = \left(\mathbf{X}^T\mathbf{X} + \lambda \mathbf{I}_N\right)^{-1}\mathbf{X}^T\mathbf{y}, \tag{5}$$

where $\mathbf{I}_N$ is an $N \times N$ identity matrix.

The output of BRR is a linear function in the original space and therefore BRR only realizes linear regression.

2) Kernel Ridge Regression: By introducing the kernel trick, the kernel ridge regression (KRR) method aims to obtain a linear regression model in the Reproducing Kernel Hilbert Space (RKHS) [33, 34], i.e.,

$$y = f(\mathbf{x}) = \varphi(\mathbf{x})^T \mathbf{w} \tag{6}$$

by optimizing the following objective

$$\min_{\mathbf{w}} \ \frac{1}{2}\left\|\mathbf{X}_\varphi\mathbf{w} - \mathbf{y}\right\|^2 + \frac{\lambda}{2}\left\|\mathbf{w}\right\|^2, \tag{7}$$

where $\varphi(\mathbf{x}) \in R^{d_\varphi}$ is the image of $\mathbf{x}$ in the RKHS and

$$\mathbf{X}_\varphi = \left[\varphi(\mathbf{x}_1), \cdots, \varphi(\mathbf{x}_N)\right]^T \in R^{N \times d_\varphi}. \tag{8}$$

Similarly, the solution can be written as

$$\mathbf{w}^* = \left(\mathbf{X}_\varphi^T\mathbf{X}_\varphi + \lambda \mathbf{I}_{d_\varphi}\right)^{-1}\mathbf{X}_\varphi^T\mathbf{y}. \tag{9}$$

Since the dimension of the feature in the RKHS is usually unknown, Eq. (9) cannot be solved directly. However, the following identity can be adopted to tackle this issue.

$$\left(\mathbf{P}^{-1} + \mathbf{B}^T\mathbf{R}^{-1}\mathbf{B}\right)^{-1}\mathbf{B}^T\mathbf{R}^{-1} = \mathbf{P}\mathbf{B}^T\left(\mathbf{B}\mathbf{P}\mathbf{B}^T + \mathbf{R}\right)^{-1}. \tag{10}$$

In Eq.(10), $\mathbf{P}$, $\mathbf{B}$ and $\mathbf{R}$ are three matrices. Let $\mathbf{P} = \frac{1}{\lambda}\mathbf{I}_{d_\varphi}$, $\mathbf{B} = \mathbf{X}_\varphi$ and $\mathbf{R} = \mathbf{I}_N$. With the identity of Eq. (10), the solution in Eq. (9) can be then given by

$$\mathbf{w}^* = \left(\mathbf{X}_\varphi^T\mathbf{X}_\varphi + \lambda \mathbf{I}_{d_\varphi}\right)^{-1}\mathbf{X}_\varphi^T\mathbf{y} = \mathbf{X}_\varphi^T\left(\mathbf{X}_\varphi\mathbf{X}_\varphi^T + \lambda \mathbf{I}_N\right)^{-1}\mathbf{y}. \tag{11}$$

Here, $\mathbf{I}_{d_\varphi}$ is a $d_\varphi \times d_\varphi$ identity matrix. The same result can also be obtained by the dual ridge regression method to be introduced in the next subsection, as shown in Eq.(20.b). With Eq. (11), the output function of KRR is

$$y = f(\mathbf{x}) = \varphi(\mathbf{x})^T\mathbf{w}^* = \varphi(\mathbf{x})^T\mathbf{X}_\varphi^T\left[\mathbf{X}_\varphi\mathbf{X}_\varphi^T + \lambda \mathbf{I}_N\right]^{-1}\mathbf{y}. \tag{12}$$

Since $\varphi(\mathbf{x})$ is usually unknown, Eq.(11) can be calculated by introducing the Mercer kernel. Define a Mercer kernel matrix for KRR as follows,

$$\mathbf{\Omega}_{KRR} = \mathbf{X}_\varphi\mathbf{X}_\varphi^T \in R^{N \times N} \tag{13}$$

with $\Omega_{KRR\,i,j} = \varphi(\mathbf{x}_i)^T\varphi(\mathbf{x}_j) = K(\mathbf{x}_i, \mathbf{x}_j)$, where $K(\mathbf{x}_i, \mathbf{x}_j)$ is a kernel function. Then, Eq.(12) can be written as

$$\begin{aligned} y = f(\mathbf{x}) &= \varphi(\mathbf{x})^T\mathbf{w}^* = \varphi(\mathbf{x})^T\mathbf{X}_\varphi^T\left[\mathbf{X}_\varphi\mathbf{X}_\varphi^T + \lambda \mathbf{I}_N\right]^{-1}\mathbf{y} \\ &= \begin{bmatrix} K(\mathbf{x}, \mathbf{x}_1) \\ \vdots \\ K(\mathbf{x}, \mathbf{x}_N) \end{bmatrix}^T \left(\mathbf{\Omega}_{KRR} + \lambda \mathbf{I}_N\right)^{-1}\mathbf{y}. \end{aligned} \tag{14}$$

When nonlinear kernel functions, such as radial basis function (RBF), are adopted, KRR can be used for nonlinear regression. Especially, if the kernel function is a linear kernel, i.e., $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T\mathbf{x}_j$, KRR is equivalent to BRR and is just linear regression in the original space. Thus, BRR can be taken as a special case of KRR.

3) Dual Ridge Regression: Instead of optimizing the primal cost functions in BRR and KRR to obtain regression models, an alternative optimization method is to solve the dual problem. Here, Eq. (7) is equivalently formulated as [34]

$$\min_{\mathbf{w},\boldsymbol{\xi}} \ \frac{1}{2}\sum_{i=1}^N \xi_i^2 + \frac{\lambda}{2}\left\|\mathbf{w}\right\|^2 \tag{15}$$

$$\text{s.t. } \varphi(\mathbf{x}_i)^T\mathbf{w} = y_i - \xi_i,\ i = 1, \ldots, N,$$

where ξ_i is the training error, i.e., the slack variable, with respect to the training sample $\mathbf{x}_i$. The Lagrangian function of Eq. (15) is

$$L_{DRR}(\mathbf{w}, \xi_i, \alpha_i) = \frac{1}{2}\sum_{i=1}^N \xi_i^2 + \frac{\lambda}{2}\left\|\mathbf{w}\right\|^2 - \sum_{i=1}^N \alpha_i\left(\varphi(\mathbf{x}_i)^T\mathbf{w} - y_i + \xi_i\right) \tag{16}$$

where $\boldsymbol{\alpha} = \left[\alpha_i, \ldots, \alpha_N\right]^T \in R^N$ is the Lagrangian multiplier vector. Using the Karush–Kuhn–Tucker (KKT) theorem, the following optimality conditions are obtained.

$$\partial L_{DRR}/\partial \mathbf{w} = 0 \ \Rightarrow \ \mathbf{w} = \frac{1}{\lambda}\sum_{i=1}^N \alpha_i\varphi(\mathbf{x}_i) \tag{17.a}$$

$$\partial L_{DRR}/\partial \xi_i = 0 \ \Rightarrow \ \alpha_i = \xi_i,\ i = 1, \ldots, N, \tag{17.b}$$

$$\partial L_{DRR}/\partial \alpha_i = 0 \ \Rightarrow \ \varphi(\mathbf{x}_i)^T\boldsymbol{\beta} - y_i + \xi_i = \mathbf{0},\ i = 1, \ldots, N. \tag{17.c}$$

Then, from Eqs. (17.a)-(17.c), we have

$$\mathbf{w} = \mathbf{X}_\varphi^T \boldsymbol{\alpha} / \lambda \,, \tag{18.a}$$

$$\boldsymbol{\xi} = \boldsymbol{\alpha} \,, \tag{18.b}$$

$$\mathbf{X}_\varphi \mathbf{w} - \mathbf{y} + \boldsymbol{\xi} = \mathbf{0} \,. \tag{18.c}$$

By substituting Eqs. (18.a) and (18.b) into Eq. (18.c), we can get the following equation.

$$\left[\mathbf{X}_\varphi \mathbf{X}_\varphi^T / \lambda + \mathbf{I}_N \right] \boldsymbol{\alpha} = \mathbf{y} \tag{19.a}$$

Then, we have

$$\boldsymbol{\alpha}^* = \lambda \left[\mathbf{X}_\varphi \mathbf{X}_\varphi^T + \lambda \mathbf{I}_N \right]^{-1} \mathbf{y} \ . \tag{20.a}$$

Furthermore, based on (17.a) , $\mathbf{w}^*$ can be obtained by

$$\mathbf{w}^* = \mathbf{X}_\varphi^T \boldsymbol{\alpha}^* / \lambda = \mathbf{X}_\varphi^T \left(\mathbf{X}_\varphi \mathbf{X}_\varphi^T + \lambda \mathbf{I}_N \right)^{-1} \mathbf{y} \,. \tag{20.b}$$

Note that Eq. (20.b) is the same as Eq. (11). Thus, if the feature in the mapping space is unknown, the kernel trick can be used as in KRR.

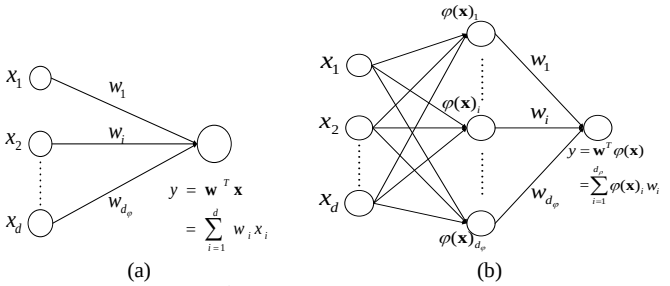


Fig.2 Network structure of regression model learned by BRR and KRR. (a) BRR; (b) KRR.

4) Network Structure of Ridge Regression

Ridge regression can be interpreted from the viewpoint of neural networks [35], as illustrated in Fig. 2. From Fig. 2(a), we can see that BRR can be used to learn a two-layer neural network, which only realize linear regression. On the other hand, as shown in Fig. 2(b), KRR can be used to learn a three-layer neural network, where the hidden layer consists of d_φ hidden nodes. The value of d_φ and the form of the activation functions in the hidden layer are usually unknown. According to the theory of neural networks, we know that the network in Fig. 2(b) usually have strong nonlinear approximation abilities with appropriate activation function. Since the form of the hidden nodes is usually unknown for neural networks associated with KRR, the kernel trick is needed for solving KRR, as described previously in section II-B-2.

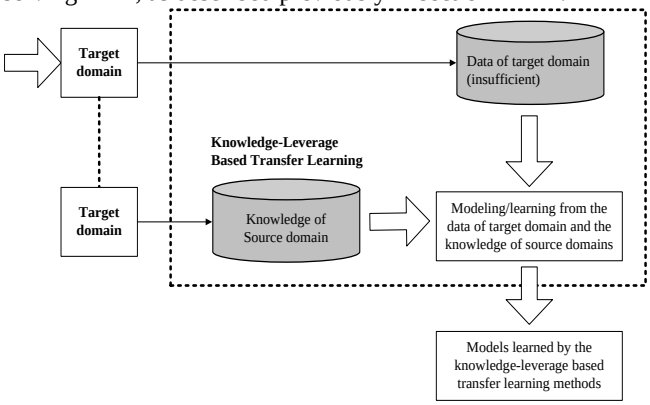


Fig. 3 A framework of knowledge-leverage based transfer learning.

C. Knowledge-leverage based Transfer Learning

Most inductive transfer learning algorithms are developed to learn directly from the *data* in the source domain with some strategies. Rather than the original data, transfer learning from the *knowledge* in the source domain is investigated recently with the knowledge-leveraged based transfer learning framework [36, 47]. As shown in Fig. 3, a generalized learning framework was proposed in [36] for knowledge-leverage based fuzzy system transfer learning. Under this framework, the model in the target domain can be learned from the data in the target domain and the knowledge in the source domain simultaneously. In this study, knowledge-leverage based inductive transfer learning for the proposed GHRR will be studied accordingly.

III. Generalized Hidden-mapping Ridge Regression

In this study, we extend BRR and KRR to propose the generalized ridge regression method GHRR, where BRR and KRR become a special case of GHRR. Especially, we will show that GHRR can be used to train a wide range of intelligence models, including forward-feed neural networks, Takagi-Sugeno-Kang fuzzy logic systems (TSKFLS) and kernel methods. The properties of GHRR are also discussed.

A. GHRR

In essence, the idea of GHRR is to obtain a linear regression model in a hidden-mapping feature space, i.e.,

$$y = f(\mathbf{x}) = \rho(\mathbf{x})^T \mathbf{w} = \sum_{i=1}^{d_\rho} \rho(\mathbf{x})_i w_i \ , \tag{21}$$

with optimizing the following objective

$$\min_{\mathbf{w}} \ \frac{1}{2} \left\| \mathbf{X}_\rho \mathbf{w} - \mathbf{y} \right\|^2 + \frac{\lambda}{2} \left\| \mathbf{w} \right\|^2 \,, \tag{22}$$

where

$$\mathbf{X}_\rho = \left[\rho(\mathbf{x}_1), \cdots, \rho(\mathbf{x}_N) \right]^T \in R^{N \times d_\rho} \,. \tag{23}$$

Here, $\rho(\mathbf{x}) \in R^{d_\rho}$ is the vector in the hidden-mapping space, which is obtained from $\mathbf{x}$ by hidden-mapping.

The following explanation is provided to clarify the difference between the proposed GHRR and the existing ridge regression methods.

(1) BRR is a special case of GHRR with $\rho(\mathbf{x}) = \mathbf{x} \in R^d$, i.e., the hidden-mapping feature space is just the original feature space.

(2) For KRR, the input variable is in the RKHS, i.e., $\varphi(\mathbf{x}) \in R^{d_\varphi}$, where the dimension number in this space is usually unknown and kernel trick is needed to solve the corresponding ridge regression. Thus, when $\rho(\mathbf{x}) = \varphi(\mathbf{x})$, KRR is reduced to a special case of GHRR.

(3) For GHRR, $\rho(\mathbf{x}) \in R^{d_\rho}$ in a hidden-mapping feature space is taken as the input variable, which can be known or unknown. When $\rho(\mathbf{x}) \in R^{d_\rho}$ is unknown, the setting is similar to that of the kernel-mapping space in KRR. However, in most situations the hidden-mapping space may be known. For example, the hidden-mapping feature space can be constructed by different classical intelligence models, such as neural networks and fuzzy logical system, which will be explained in the next subsection.

B. GHRR: A Unified Learning Method for Neural Networks, Fuzzy Systems and Kernel Methods

In this section, we will show that classical intelligent models, including neural networks, fuzzy systems and kernel methods, can be learned by using GHRR. In other words, we will show that the learning of these models can be transformed to the learning of a

linear regression model in a hidden-mapping feature space.

1) GHRR for Feedforward Neural Networks: Consider a multiple hidden-layer feedforward neural network (MHFN) with single output, which includes an input layer, M hidden layers and an output layer. This MHFN can be viewed as a generalized signal hidden-layer feedforward neural network with more complicated activation functions in the hidden layer. For a reduced single hidden-layer feedforward neural network (SHFN), the output can be formulated as

$$y = f(\mathbf{x}) = \sum_{i=1}^{N_M} g_i(\mathbf{x}, \boldsymbol{\theta}_i)^T w_i \,. \tag{24}$$

As proved in literature [37,38], if the activation functions $g_i(\mathbf{x}, \boldsymbol{\theta}_i)$ are piecewise continuous, the hidden nodes can be randomly generated independent of the training data and the corresponding network still retain the universal approximation capability. Thus, when the hidden parameters are fixed with the randomly generated hidden nodes, the MHFN training can be taken as a hidden-mapping linear regression problem. Let the hidden mapping function $\rho(\mathbf{x})$ be

$$\rho(\mathbf{x}) = [g_1(\mathbf{x}, \boldsymbol{\theta}_1), \ldots, g_{N_M}(\mathbf{x}, \boldsymbol{\theta}_{N_M})]^T \in R^{N_M} \,, \tag{25}$$

Eq. (24) can then be written as

$$y = f(\mathbf{x}) = \rho(\mathbf{x})^T \mathbf{w} \,, \tag{26}$$

and it is can be solved directly by using the GHRR method as discussed in Eqs. (21)-(23).

2) GHRR for TSKFLS: Next, we will show that classical TSKFLS [39] can be trained by using the proposed GHRR. For TSKFLS, the most commonly used fuzzy inference rules are defined as follows.

$$\begin{aligned} &\text{TSK Fuzzy Rule } R^k : \\ &\text{IF } x_1 \text{ is } A_1^k \ \wedge x_2 \text{ is } A_2^k \ \wedge \cdots \wedge x_d \text{ is } A_d^k \\ &\text{Then } f^k(\mathbf{x}) = p_0^k + p_1^k x_1 + \cdots + p_d^k x_d \ , \ k = 1, \cdots, K \,. \end{aligned} \tag{27}$$

In Eq. (27), A_i^k is a fuzzy subset subscribed by the input variable x_i for the kth rule; K is the number of fuzzy rules, and $\wedge$ is a fuzzy conjunction operator. Each rule is premised on the input vector $\mathbf{x} = [x_1, x_2, \cdots, x_d]^T \in R^d$, and maps the fuzzy sets in the input space $A^k \subset R^d$ to a varying singleton denoted by $f^k(\mathbf{x})$. When *multiplicative conjunction*, *multiplicative implication* and *additive disjunction* are employed respectively as the conjunction operator, the implication operator and the disjunction operator, the output of the TSKFLS can be formulated as

$$y = f(\mathbf{x}) = \sum_{k=1}^{K} \frac{\mu^k(\mathbf{x})}{\sum_{k'=1}^{K} \mu^{k'}(\mathbf{x})} \cdot f^k(\mathbf{x}) = \sum_{k=1}^{K} \tilde{\mu}^k(\mathbf{x}) \cdot f^k(\mathbf{x}) \tag{28}$$

where $\mu^k(\mathbf{x})$ and $\tilde{\mu}^k(\mathbf{x})$ denote the fuzzy membership and the normalized fuzzy membership associated with the fuzzy set A^k. These two memberships can be calculated with

$$\mu^k(\mathbf{x}) = \prod_{i=1}^{d} \mu_{A_i^k}(x_i) \,, \tag{29.a}$$

$$\tilde{\mu}^k(\mathbf{x}) = \mu^k(\mathbf{x}) \Big/ \sum_{k'=1}^{K} \mu^{k'}(\mathbf{x}) \,. \tag{29.b}$$

Here, clustering technique is commonly used to set the parameters of the antecedents. When the antecedents of the TSKFLS are determined, we can consider the learning of TSKFLS as the hidden-mapping linear regression below,

$$y = f(\mathbf{x}) = \rho(\mathbf{x})^T \mathbf{w} \tag{30}$$

with

$$\rho(\mathbf{x}) = \left[(\tilde{\mathbf{x}}^1)^T, (\tilde{\mathbf{x}}^2)^T, \cdots, (\tilde{\mathbf{x}}^K)^T \right]^T \in R^{K(d+1)} \tag{31.a}$$

$$\tilde{\mathbf{x}}^k = \tilde{\mu}^k(\mathbf{x}) \mathbf{x}_e \,, \tag{31.b}$$

$$\mathbf{x}_e = (1, \mathbf{x}^T)^T \,, \tag{31.c}$$

$$\mathbf{w} = \left[(\mathbf{p}^1)^T, (\mathbf{p}^2)^T, \cdots, (\mathbf{p}^K)^T \right]^T \,, \tag{31.d}$$

$$\mathbf{p}^k = (p_0^k, p_1^k, \cdots, p_d^k)^T \,. \tag{31.e}$$

Thus, the TSKFLS can be solved directly by using the GHRR method as described in Eqs. (21)-(23).

In particular, from the viewpoint of neural networks, TSKFLS can be viewed as the corresponding fuzzy neural network. As in the case of neural networks, we can transform TSKFLS into the generalized SHFN with the specified hidden nodes, where the activation functions in the hidden layer are defined as

$$\begin{aligned} &g_j(\mathbf{x}, \theta_j) = \tilde{\mu}^k(\mathbf{x}) x_{ei} \,, \\ &j = (k-1)(d+1) + i, \quad k = 1, \ldots, K, \quad i = 1, \ldots, d+1. \end{aligned} \tag{32}$$

Comparing Eq. (31.a) with Eq. (32), it is obvious that the following equation is satisfied.

$$\rho(\mathbf{x}) = \left[g_1(\mathbf{x}, \theta_1), g_2(\mathbf{x}, \theta_2), \cdots, g_{K(d+1)}(\mathbf{x}, \theta_{K(d+1)}) \right]^T \in R^{K(d+1)} \,.$$

It has been proved in literature [37, 38] that from the viewpoint of neural networks, the activation functions can also be randomly generated and the corresponding fuzzy neural networks are still universal approximators. Thus, once the hidden nodes, i.e. the antecedents of TSKFLS, are fixed by random generation or other techniques such as clustering technique, the learning of TSKFLS can be regarded as a hidden-mapping linear regression problem and solved directly by using the GHRR method proposed in Eqs. (21)-(23).

3) GHRR for Kernel Methods: Kernel methods have been studied extensively and SVM is the most classical one. SVM is used to solve a linear model in RKHS based on statistical learning theory. For the proposed GHRR method, if the hidden-mapping is unknown in the RKHS, we only need to solve the linear model in RKHS in a way similar to that achieved by KRR [33,34] (see the review of KBB in section II-B-2). In this case, the GHRR can be easily used to train the kernel methods.

C. Optimization of GHRR

The objective function of GHRR in Eq. (22) can be solved efficiently in various ways depending on the condition of the hidden-mapping $\rho(\mathbf{x})$. Here, we discuss the different cases as follows.

1) Case 1: $\rho(\mathbf{x})$ is known: In this case, we can obtain explicit values of the data $\rho(\mathbf{x})$ in the hidden-mapping space. For example, $\rho(\mathbf{x})$ can be constructed by neural networks or fuzzy logic systems. The solution for the model parameter $\mathbf{w}$ can then be obtained in a similar form as that shown in Eqs. (5) and (9), i.e.,

$$\mathbf{w}^* = \left(\mathbf{X}_\rho^T \mathbf{X}_\rho + \lambda \mathbf{I}_{d_\rho} \right)^{-1} \mathbf{X}_\rho^T \mathbf{y} \,, \tag{33.a}$$

where $\mathbf{X}_\rho^T \mathbf{X}_\rho + \lambda \mathbf{I}_{d_\rho}$ is a $d_\rho \times d_\rho$ matrix. If the d_ρ is small, the computation of the inverse of this matrix is very efficient. Otherwise, if the size of dataset is much smaller than the dimensionality d_ρ, we can instead solve GHRR more efficiently using the approach discussed in Eqs. (11) and (20), i.e.,

$$\boldsymbol{\alpha}^* = \lambda\left[\mathbf{X}_\rho\mathbf{X}_\rho^T + \lambda\mathbf{I}_N\right]^{-1}\mathbf{y}\ , \tag{33.b}$$

$$\mathbf{w}^* = \frac{1}{\lambda}\mathbf{X}_\rho^T\boldsymbol{\alpha}^* = \mathbf{X}_\rho^T\left(\mathbf{X}_\rho\mathbf{X}_\rho^T + \lambda\mathbf{I}_N\right)^{-1}\mathbf{y}\ , \tag{33.c}$$

where $\mathbf{X}_\rho^T\mathbf{X}_\rho + \lambda\mathbf{I}_N$ is an $N\times N$ matrix. If N is small, the computation of the inverse of this matrix can be very efficient.

2) Case 2: $\rho(\mathbf{x})$ is unknown: In this case, explicit formulation of the data $\rho(\mathbf{x})$ in the hidden-mapping space cannot be obtained and thus $\mathbf{w}$ cannot be specified explicitly. Kernel trick is then needed to obtain the final decision function $f(\mathbf{x})$. While the introduction of kernel trick to the solution strategy in (33.a) is difficult, it can be achieved conveniently with the solution strategy in (33.c). Following the approach described in Eqs. (12)-(14), we can obtain the decision function $f(\mathbf{x})$ as follows,

$$\boldsymbol{\alpha}^* = \lambda\left[\boldsymbol{\Omega}_{GHRR} + \lambda\mathbf{I}_N\right]^{-1}\mathbf{y}\ , \tag{34.a}$$

$$\mathbf{w}^* = \frac{1}{\lambda}\mathbf{X}_\rho^T\boldsymbol{\alpha}^* = \mathbf{X}_\rho^T\left[\boldsymbol{\Omega}_{GHRR} + \lambda\mathbf{I}_N\right]^{-1}\mathbf{y}\ , \tag{34.b}$$

$$\begin{aligned} f(\mathbf{x}) &= \rho(\mathbf{x})^T\mathbf{w}^* = \rho(\mathbf{x})^T\mathbf{X}_\rho^T\left[\boldsymbol{\Omega}_{GHRR} + \lambda\mathbf{I}_N\right]^{-1}\mathbf{y} \\ &= \begin{bmatrix} K(\mathbf{x},\mathbf{x}_1) \\ \vdots \\ K(\mathbf{x},\mathbf{x}_N) \end{bmatrix}^T \left[\boldsymbol{\Omega}_{GHRR} + \lambda\mathbf{I}_N\right]^{-1}\mathbf{y}. \end{aligned} \tag{34.c}$$

where $\boldsymbol{\Omega}_{GHRR} = \mathbf{X}_\rho\mathbf{X}_\rho^T$ with $\Omega_{GHRR\ i,j} = \rho(\mathbf{x}_i)^T\rho(\mathbf{x}_j) = K(\mathbf{x}_i,\mathbf{x}_j)$ and $K(\mathbf{x}_i,\mathbf{x}_j)$ denotes the kernel function. Hence, GHRR is equivalent to KRR in this particular case, as discussed in section II-B-2.

D. Algorithm of the GHRR

Based on the solution of GHRR described above, the details of the proposed algorithm for GHRR are given below.

Algorithm of the GHRR

Case 1: The hidden-mapping is known

Step 1: Calculate the model parameter $\mathbf{w}^*$ using Eq. (33.a) or Eq. (33.c).

Step 2: Calculate the output of the testing data using the decision function $f(\mathbf{x}) = \rho(\mathbf{x})^T\mathbf{w}^*$.

Case 2: The hidden-mapping is unknown and kernel trick is adopted

Step 1: Calculate $\boldsymbol{\alpha}$ using Eq. (34.a).

Step 2: Calculate the output of the testing data using the decision function in Eq. (34.c).

Remark 1. For the proposed GHRR algorithm, if the hidden-mapping is known, when the number of the training data is larger than the number of features in the hidden-mapping space, i.e., ($N >> d_\rho$), it is more efficient to use Eq. (33.a) rather than Eq. (33.c) considering the computational complexity of matrix inverse; otherwise, Eq. (33.c) is used.

E. Classification

GHRR is originally developed for regression. Similar to other regression methods like radial basis function neural network (RBF-NN), additional strategies are required when GHRR and the proposed TGHRR (to be discussed in section IV) are used for classification. One of the commonly used approaches is to use the regression function to approximate the class labels in the corresponding classification task. Once the model is trained, a future testing sample can be tested and the label, which is nearest to the model output, is taken as the label of the given testing sample. Here, a more effective strategy enabling the use of regression method for classification is introduced in detail as follows.

The idea of the strategy is to use a multiple output function for the classification task. For a given classification dataset including m classes, $\{\mathbf{x}_i, y_i\}$, $y_i \in \{1,\cdots,m\}$, $i = 1,\cdots,N$, we will construct a multi-output regression dataset $\{\mathbf{x}_i, \tilde{\mathbf{y}}_i\}$. If the original class label is $y_i = p$ $(1 \le p \le m)$ for the ith training sample in $\{\mathbf{x}_i, y_i\}$, the corresponding output vector containing m outputs in the constructed multi-output regression dataset $\{\mathbf{x}_i, \tilde{\mathbf{y}}_i\}$ is defined as

$$\tilde{\mathbf{y}}_i = [0,\cdots,0,\overset{p}{1},0,\cdots,0]^T \in R^m .$$

In this output vector, only the pth element of $\tilde{\mathbf{y}}_i$ is one, while the rest of the elements are set to zero.

With the corresponding multi-output regression dataset, a multi-output regression model will be trained. Once the trained model is obtained, for a given testing sample the output vector obtained by the trained model can be expressed as

$$\tilde{\mathbf{y}}_i^{\text{model}} = [y_{i,1}^{\text{model}},\cdots,y_{i,l}^{\text{model}},\cdots y_{i,m}^{\text{model}}]^T .$$

Then the predicted class label of the testing sample is the index of the element having the highest value in the output vector. For example, if $y_{i,l}^{\text{model}}$ $(1 \le l \le m)$ has the highest value among all the elements in the vector $\tilde{\mathbf{y}}_i^{\text{model}}$, i.e., $\{y_{i,j}^{\text{model}}\}$, $j = 1,\cdots,m$, the final predicted class label of the testing sample will be l.

IV. TGHRR

For the proposed GHRR, our ultimate goal is to develop the corresponding transfer learning method TGHRR for the inductive transfer learning task. In particular, the knowledge-leverage based transfer learning mechanism is introduced to the GHRR and the algorithm is presented below.

A. Objective Function for TGHRR

Based on the knowledge-leverage based transfer learning framework described in section II-C, the following generalized objective function is proposed for TGHRR.

$$\min_{\boldsymbol{\Theta}}\ J_{TGHRR} = J_t(\boldsymbol{\Theta}) + J_s(\boldsymbol{\Theta};\tilde{\boldsymbol{\Theta}})\ , \tag{35}$$

where Eq. (35) consists of two parts which are explained below.

(1) The first part, J_t, inherits from the GHRR directly, which is used to train the model by the data in the target domain.

(2) The second part, J_s, is developed for knowledge leverage from the source domain. For the design of the TGHRR, this part is varied and depends on the specified knowledge-leverage strategies adopted.

With the general objective in Eq. (35), we propose the explicit objective function below

$$\min_{\mathbf{w}}\ \frac{1}{2}\left\|\mathbf{X}_\rho\mathbf{w} - \mathbf{y}\right\|^2 + \frac{\lambda_0}{2}\left\|\mathbf{w}\right\|^2 + \frac{\lambda_1}{2}\left\|\mathbf{w} - \mathbf{w}_s\right\|^2 , \tag{36.a}$$

or equivalently,

$$\min_{\mathbf{w},\boldsymbol{\xi}}\ \frac{1}{2}\sum_{i=1}^{N}\xi_i^2 + \frac{\lambda_0}{2}\left\|\mathbf{w}\right\|^2 + \frac{\lambda_1}{2}\left\|\mathbf{w} - \mathbf{w}_s\right\|^2 , \tag{36.b}$$

$$\text{s.t. } \rho(\mathbf{x}_i)^T\mathbf{w} = y_i - \xi_i\ ,\ i = 1,\ldots,N .$$

In Eq. (36), the first two terms are inherited from the GHRR directly, which are used to learn from the data in the target domain,

while the third term is used to learn from the knowledge, i.e., the model parameters $\mathbf{w}_s$, from the source domain, where the model parameter $\mathbf{w}_s$ is assumed to be knowledge available from the source domain.

B. Optimization of TGHRR

For simplicity, we use Eq. (36.b) to derive the solutions of the two proposed objective functions in Eqs. (36.a) and (36.b). The corresponding Lagrangian function of Eq. (36.b) can be expressed as

$$L_{TGHRR}(\mathbf{w},\xi_i,\alpha_i)=\frac{1}{2}\sum_{i=1}^{N}\xi_i^2+\frac{\lambda_0}{2}\|\mathbf{w}\|^2+\frac{\lambda_1}{2}\|\mathbf{w}-\mathbf{w}_s\|^2 -\sum_{i=1}^{N}\alpha_i(\boldsymbol{\rho}(\mathbf{x}_i)^T\mathbf{w}-t_i+\xi_i) \quad (37)$$

Based on the KKT theorem, the following KKT optimality conditions are given:

$$\partial L_{TGHRR}/\partial\mathbf{w}=0 \;\Rightarrow\; \lambda_0\mathbf{w}+\lambda_1(\mathbf{w}-\mathbf{w}_s)-\sum_{i=1}^{N}\alpha_i\boldsymbol{\rho}(\mathbf{x}_i)=\mathbf{0}\,, \quad (38.a)$$

$$\partial L_{TGHRR}/\partial\xi_i=0\Rightarrow\boldsymbol{\alpha}=\boldsymbol{\xi}\,,\ i=1,\ldots,N\,, \quad (38.b)$$

$$\partial L_{TGHRR}/\partial\alpha_i=0\Rightarrow\rho(\mathbf{x}_i)^T\mathbf{w}-y_i+\xi_i=0\,,\ i=1,\ldots,N\,. \quad (38.c)$$

Then, from Eqs. (38.a)-(38.c), we have

$$\mathbf{w}=\lambda_1\mathbf{w}_s/(\lambda_0+\lambda_1)+\mathbf{X}_\rho^T\boldsymbol{\alpha}/(\lambda_0+\lambda_1)\,, \quad (39.a)$$

$$\boldsymbol{\xi}=\boldsymbol{\alpha}\,, \quad (39.b)$$

$$\mathbf{X}_\rho\mathbf{w}-\mathbf{y}+\boldsymbol{\xi}=\mathbf{0}\,,\ i=1,\ldots,N\,. \quad (39.c)$$

The solution of the proposed TGHRR can then be calculated efficiently in different ways depending on the hidden mapping $\rho(\mathbf{x})$ and the training conditions.

1) Case 1: $\rho(\mathbf{x})$ is known and the number of training data is small.

In this case, the solution of TGHRR can be calculated efficiently as follows. By substituting Eqs. (39.a) and (39.b) into Eq. (39.c), we obtain

$$\left[\frac{1}{\lambda_0+\lambda_1}\mathbf{X}_\rho\mathbf{X}_\rho^T+\mathbf{I}_N\right]\boldsymbol{\alpha}=\mathbf{y}-\frac{\lambda_1}{\lambda_0+\lambda_1}\mathbf{X}_\rho\mathbf{w}_s\,. \quad (40)$$

With Eq. (40), the solution for $\boldsymbol{\alpha}$ is given by

$$\boldsymbol{\alpha}^*=\left[\frac{1}{\lambda_0+\lambda_1}\mathbf{X}_\rho\mathbf{X}_\rho^T+\mathbf{I}_N\right]^{-1}\left(\mathbf{y}-\frac{\lambda_1}{\lambda_0+\lambda_1}\mathbf{X}_\rho\mathbf{w}_s\right). \quad (41.a)$$

Then, we can calculate $\mathbf{w}$ using (39.a) as follows.

$$\mathbf{w}^*=\frac{\lambda_1\mathbf{w}_s}{\lambda_0+\lambda_1}+\frac{\mathbf{X}_\rho^T}{\lambda_0+\lambda_1}\left[\frac{1}{\lambda_0+\lambda_1}\mathbf{X}_\rho\mathbf{X}_\rho^T+\mathbf{I}_N\right]^{-1}\left(\mathbf{y}-\frac{\lambda_1}{\lambda_0+\lambda_1}\mathbf{X}_\rho\mathbf{w}_s\right)$$
$$=\frac{\lambda_1\mathbf{w}_s}{\lambda_0+\lambda_1}+\mathbf{X}_\rho^T\left[\mathbf{X}_\rho\mathbf{X}_\rho^T+(\lambda_0+\lambda_1)\mathbf{I}_N\right]^{-1}\left(\mathbf{y}-\frac{\lambda_1}{\lambda_0+\lambda_1}\mathbf{X}_\rho\mathbf{w}_s\right). \quad (41.b)$$

The final output of the hidden-mapping linear regression model can be formulated as

$$f(\mathbf{x})=\rho(\mathbf{x})^T\mathbf{w}^*, \quad (42)$$

with $\mathbf{w}^*$ obtained in Eq. (41.b).

2) Case 2: $\rho(\mathbf{x})$ is known and the number of dimensionality of $\rho(\mathbf{x})$ is small.

In this case, the solution of TGHRR can be calculated efficiently by first obtaining $\boldsymbol{\xi}$ using Eqs. (39.a) and (39.b), i.e.,

$$\boldsymbol{\xi}=(\lambda_0+\lambda_1)\left[\mathbf{X}_\rho^T\right]^+\left[\mathbf{w}-\frac{\lambda_1\mathbf{w}_s}{\lambda_0+\lambda_1}\right], \quad (43)$$

with $\left[\mathbf{X}_\rho^T\right]^+$ as the pseudo-inverse of $\mathbf{X}_\rho^T$. Substituting Eq. (43) into Eq. (39.c), we have

$$\mathbf{X}_\rho\mathbf{w}-\mathbf{y}+(\lambda_0+\lambda_1)\left[\mathbf{X}_\rho^T\right]^+\left[\mathbf{w}-\frac{\lambda_1\mathbf{w}_s}{\lambda_0+\lambda_1}\right]=\mathbf{0}\,, \quad (44)$$

and then we can get

$$\mathbf{w}^*=\left[\mathbf{X}_\rho^T\mathbf{X}_\rho+(\lambda_0+\lambda_1)\mathbf{I}_{d_\rho}\right]^{-1}\left(\mathbf{X}_\rho^T\mathbf{y}+\lambda_1\mathbf{w}_s\right). \quad (45)$$

Furthermore, the output of the hidden-mapping linear regression model can be formulated as

$$f(\mathbf{x})=\rho(\mathbf{x})^T\mathbf{w}^*, \quad (46)$$

with $\mathbf{w}^*$ obtained in Eq. (45).

3) Case 3: $\rho(\mathbf{x})$ is unknown.

When the hidden mapping $\rho(\mathbf{x})$ is unknown, $\mathbf{w}^*$ cannot be calculated directly as in the two previous cases with Eq. (41.b) or Eq. (45). However, when the Mercer kernel is adopted, the output of the hidden-mapping linear regression model can be computed accordingly. From Eq. (41.a), we know that

$$\boldsymbol{\alpha}=\left[\frac{1}{\lambda_0+\lambda_1}\mathbf{X}_\rho\mathbf{X}_\rho^T+\mathbf{I}_N\right]^{-1}\left(\mathbf{y}-\frac{\lambda_1}{\lambda_0+\lambda_1}\mathbf{X}_\rho\mathbf{w}_s\right). \quad (47)$$

Since $\mathbf{w}_s$ is the parameter of the hidden-mapping linear model learned in the source domain by GHRR, according to Eqs. (33.a) and (33.b), it can be expressed as

$$\mathbf{w}_s=\mathbf{X}_{\rho,s}^T\boldsymbol{\alpha}_s/\lambda_s\,, \quad (48.a)$$

$$\boldsymbol{\alpha}_s=\lambda_s\left[\mathbf{X}_{\rho,s}\mathbf{X}_{\rho,s}^T+\lambda_s\mathbf{I}_{Ns}\right]^{-1}\mathbf{y}_s\,. \quad (48.b)$$

Here, $\mathbf{X}_{\rho,s}$ is the matrix constructed using the data in the source domain by Eq. (23); λ_s is the regularization parameter used in the source domain. Since $\rho(\mathbf{x})$ is unknown, we express Eq. (48.b) as

$$\boldsymbol{\alpha}_s=\lambda_s\left[\boldsymbol{\Omega}_{GHRR,s}+\lambda_s\mathbf{I}_{Ns}\right]^{-1}\mathbf{y}_s\,, \quad (48.c)$$

with $\boldsymbol{\Omega}_{GHRR,s}=\left[K(\mathbf{x}_{i,s},\mathbf{x}_{j,s})\right]_{N_s\times N_s}$. Meanwhile, substituting Eq. (48.a) into Eq. (47), we have

$$\boldsymbol{\alpha}^*=\left[\frac{1}{\lambda_0+\lambda_1}\mathbf{X}_\rho\mathbf{X}_\rho^T+\mathbf{I}_N\right]^{-1}\left(\mathbf{y}-\frac{\lambda_1}{\lambda_0+\lambda_1}\cdot\frac{1}{\lambda_s}\cdot\mathbf{X}_\rho\mathbf{X}_{\rho,s}^T\boldsymbol{\alpha}_s\right). \quad (49.a)$$

Furthermore, as $\rho(\mathbf{x})$ is unknown, Eq. (49.a) can be expressed as

$$\boldsymbol{\alpha}^*=\left[\frac{1}{\lambda_0+\lambda_1}\boldsymbol{\Omega}_{GHRR,t}+\mathbf{I}_N\right]^{-1}\left(\mathbf{y}-\frac{\lambda_1}{\lambda_0+\lambda_1}\cdot\frac{1}{\lambda_s}\cdot\boldsymbol{\Omega}_{GRRR,ts}\boldsymbol{\alpha}_s\right) \quad (49.b)$$

with $\boldsymbol{\Omega}_{GHRR,t}=\left[K(\mathbf{x}_{i,t},\mathbf{x}_{j,t})\right]_{N_t\times N_t}$ and $\boldsymbol{\Omega}_{GRRR,ts}=\left[K(\mathbf{x}_{i,t},\mathbf{x}_{j,s})\right]_{N_t\times N_s}$. Then, according to Eq. (41.b), we have

$$\mathbf{w}^*=\frac{\lambda_1\mathbf{w}_s}{\lambda_0+\lambda_1}+\frac{\mathbf{X}_\rho^T\boldsymbol{\alpha}^*}{\lambda_0+\lambda_1}=\frac{\lambda_1}{\lambda_0+\lambda_1}\frac{1}{\lambda_s}\mathbf{X}_{\rho,s}^T\boldsymbol{\alpha}_s+\frac{1}{\lambda_0+\lambda_1}\mathbf{X}_\rho^T\boldsymbol{\alpha}^*\,. \quad (50)$$

Finally, the decision function of the hidden-mapping linear model can be expressed as

$$f(\mathbf{x}) = \rho(\mathbf{x})^T \mathbf{w}^* = \frac{\lambda_1}{\lambda_0+\lambda_1}\frac{1}{\lambda_s}\rho(\mathbf{x})^T \mathbf{X}_{\rho,s}^T \boldsymbol{\alpha}_s + \frac{1}{\lambda_0+\lambda_1}\rho(\mathbf{x})^T \mathbf{X}_{\rho}^T \boldsymbol{\alpha}^*$$
$$= \frac{\lambda_1}{\lambda_0+\lambda_1}\frac{1}{\lambda_s}\left(\mathbf{X}_{\rho,s}\rho(\mathbf{x})\right)^T \boldsymbol{\alpha}_s + \frac{1}{\lambda_0+\lambda_1}\left(\mathbf{X}_{\rho}\rho(\mathbf{x})\right)^T \boldsymbol{\alpha}^*$$
$$= \frac{\lambda_1}{\lambda_0+\lambda_1}\frac{1}{\lambda_s}\begin{bmatrix} K(\mathbf{x}_{1,s},\mathbf{x}) \\ \vdots \\ K(\mathbf{x}_{Ns,s},\mathbf{x}) \end{bmatrix}^T \boldsymbol{\alpha}_s + \frac{1}{\lambda_0+\lambda_1}\begin{bmatrix} K(\mathbf{x}_{1,t},\mathbf{x}) \\ \vdots \\ K(\mathbf{x}_{Nt,t},\mathbf{x}) \end{bmatrix}^T \boldsymbol{\alpha}^*. \quad (51)$$

C. Algorithm of TGHRR

Based on the solution of TGHRR above, the details of the proposed algorithm of TGHRR are presented below.

Algorithm of TGHRR

Case 1: The hidden mapping is known

Step 1: Obtain the knowledge from the source domain, i.e., the model parameter $\mathbf{w}_s$ and the parameters of the hidden nodes in the source domain.

Step 2: Calculate the model parameter $\mathbf{w}^*$ in the target domain by using Eq. (41.b) or Eq. (45).

Step 3: Calculate the output of the testing data by using Eq. (42) or Eq. (46).

Case 2: The hidden mapping is unknown and kernel is adopted

Step 1: Obtain the knowledge and data from the source domain, i.e., $\boldsymbol{\alpha}_s, \lambda_s$, and $D_s = \{\mathbf{x}_{i,s}\}$.

Step 2: Calculate $\boldsymbol{\alpha}^*$ by using Eq. (49.b).

Step 3: Calculate the output of the testing data by using Eq. (51).

Remark2. For the proposed TGHRR algorithm, if the hidden mapping is known, when the number of the training data is larger than the number of dimensionality of the hidden-mapping features, i.e., ($N >> d_\rho$), obtaining the solution with Eq. (41) is more efficient than that with Eq. (45) considering the computational complexity of matrix inverse; otherwise, Eq. (45) is more efficient.

Remark3. When the hidden mapping is known, only the knowledge $\mathbf{w}_s$ is used for transfer learning and the data in the source domain is not required. This means that the proposed algorithm has good privacy protection ability for the data in the source domain. However, if the hidden feature mapping is unknown, the data in the source is also required, as shown in Eqs. (50) and (51), in order to effectively implement transfer learning. In this case, the proposed algorithm can no longer protect the privacy of the data in the source domain.

V. Experiments

Three sets of experiments were conducted to comprehensively evaluate the performance of the proposed GHRR and TGHRR algorithms. The first two experiments studied their performance on two regression datasets, i.e., a synthetic dataset and a real-world biomechanical process modeling dataset. The last experiment was carried out on real-world text classification datasets, i.e., email spam filtering text datasets.

A. Experimental Setup

1) Methods for Comparison: *The performance of the proposed algorithms were compared with a number of existing classical non-transfer learning and inductive transfer learning algorithms as listed in Tables III and IV, respectively.*

TABLE III NON-TRANSFER LEARNING METHODS USED FOR PERFORMANCE COMPARISON

Method	Description	Task
GHRR: SHFN(RBF)	GHRR used for training single hidden-layer neural networks with RBF-type hidden nodes	Class and Reg*
GHRR: SHFN(Sigm)	GHRR used for training single hidden-layer neural networks with Sigmoid-type hidden nodes	
GHRR: MHFN(RBF)	GHRR used for training multiple hidden-layers neural networks with RBF-type hidden nodes	
GHRR: MHFN(Sigm)	GHRR used for training multiple hidden-layers neural networks with Sigmoid -type hidden nodes	
GHRR: TSKFLS(RBF)	GHRR used for training TSK fuzzy logic systems with the RBF-type membership function	
GHRR: RR(RBF)	GHRR used for training kernel ridge regression with the RBF-type kernel function	
RBF-NN [40]	RBF neural network based on the back propagation learning algorithm	Reg
L2-TSKFLS [41]	L2-norm TSK-type fuzzy logic system learning algorithm	
LS-SVR [42]	Least square support vector regression	
KNN [43]	K near neighbors classifier	Class
C-SVM [44]	C-support vector machine	

*Class and Reg denote classification and regression respectively.

TABLE IV INDUCTIVE TRANSFER LEARNING METHODS USED FOR PERFORMANCE COMPARISON

Method	Description	Task
TGHRR: SHFN(RBF)	Transfer version of GHRR used for training different models.	Class and Reg*
TGHRR: SHFN(Sigm)		
TGHRR: MHFN(RBF)		
TGHRR: MHFN(Sigm)		
TGHRR: TSKFLS(RBF)		
TGHRR: KRR(RBF)		
TrAdaBoost (LS-SVR+RBF) [25]	Transfer AdaBoost based on the LS-SVR learner with the RBF-type kernel function for regression	Reg
HiRBF [45]	Bayesian transfer learning for nonlinear regression	
SVR-AuD (LP-SVR+RBF) [27]	Linear programming support vector regression with the RBF-type kernel function by using the auxiliary data	
TrAdaBoost (LS-SVC+RBF) [25]	Transfer AdaBoost based on the LS-SVC learner with the RBF-type kernel function for classification	Class
SVC-AuD (LP-SVC)) [27]	Linear programming support vector classification with the RBF-type kernel function by using the auxiliary data	
KNN-AuD [27]	KNN classification with the auxiliary data	

* Class and Reg denote classification and regression respectively.

2) Parameter Setting: For all the algorithms, the hyper parameters were determined with the five folds cross-validation (CV) strategy based on the training sets. To save space, the parameter sets are presented in Table II(S) of the Supplementary material.

3) Datasets: In the experiments, synthetic and real-word datasets were adopted for performance comparison. The synthetic dataset was adopted for regression task, whereas the real-world datasets were used for regression tasks in biomedical processing modeling and classification tasks in email spam filtering. For all the datasets, each of the attributes of the data inputs was

normalized into the range [-1, 1]. For the regression datasets, the attributes of the data outputs were also normalized into the range [-1, 1]. The details of these datasets are described in the following subsections respectively.

4) Evaluation Indices: In all the experiments, the performance index

$$J_{reg} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i' - y_i)^2 \Big/ \frac{1}{N}\sum_{i=1}^{N}(y_i - \bar{y})^2} \qquad (52.a)$$

is adopted for performance evaluation of the regression tasks [35], where N is the number of test data; y_i is the output for the ith test input; y_i' is the model output for the ith test input and $\bar{y} = \sum_{i=1}^{N} y_i \Big/ N$. The smaller the value of J_{reg}, the better the generalization performance.

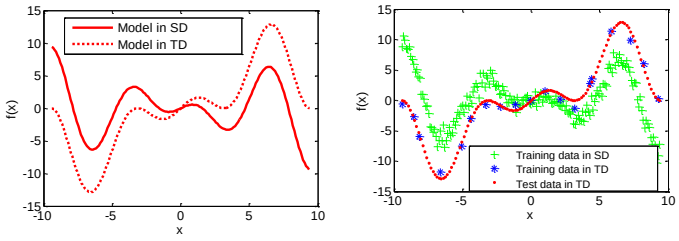
For classification tasks, the performance index below, i.e., classification accuracy, is used to evaluate the classification performance.

$$J_{clas} = \frac{\text{Number of the test samples classified correcly}}{\text{Number of the test samples}}. \qquad (52.b)$$

5) Other Settings: All the algorithms in the experiments were implemented with Matlab on a computer with a 1.66 GHz CPU and 2GB RAM.

TABLE V THE SYNTHETIC DATASETS FOR REGRESSION

Source domain	**Target domain**	
Dataset (D1)	Training set (D2)	Testing set (D2_test)
Size	Size	Size
175	19	377



(a) (b)

Fig. 4 Synthetic regression datasets: (a) two models used to generate the data in the source domain (SD) and the target domain (TD); (b) the generated data in SD and TD.

B. On Synthetic Datasets for Regression

1) Construction of Synthetic Regression Datasets: To simulate the scenarios of the inductive transfer learning tasks discussed in this study, the synthetic regression datasets should satisfy the following requirements: 1) the source domain should be related to the target domain, i.e., the source and target domains are different but related; 2) the training data of the target domain are insufficient, or part of the data are missing.

Based on these requirements, we generated the synthetic datasets by making use of the function $Y = f(x) = \cos(x) * x + N(0,\sigma)$, $x \in [-3\pi, 3\pi]$ to define the source domain. It was to generate the data in the source domain (D1), where $N(0,\sigma)$ denotes the Gaussian white noise with zero mean and standard deviation σ. On the other hand, the function $y = F(x) = \cos(x) * x + x + N(0,\sigma)$, $x \in [-3\pi, 3\pi]$ was used to define the target domain and generate the training dataset (D2) and testing dataset (D2_test) of the target domain. Fig. 4(a) shows the two functions used to simulate the related domains and Fig. 4(b) shows the data generated in the source domain and the target domain, where the standard deviations σ of Gaussian white noise in the data of source domains and the training data of target domain were both set to be 0.85, and the test data of target domain is noise-free.

2) Results and Discussion: Experiments were conducted to evaluate the regression performance of the proposed methods and the related methods on the synthetic datasets. The results are divided into four parts, namely, Parts A to D, and presented in Table VI. Parts A, B and C give the results of different non-transfer learning algorithms obtained respectively by using (i) the data in the *source domain only*, (ii) the data in the *target domain only*, and (iii) the data in *both domains*. The best values of the parameters were obtained by the CV strategy and the corresponding generalization performance indices J_{reg} on the test data in the target domain for these three cases are shown in Table IV with Part A, B and C.

On the other hand, Part D gives the results of different inductive transfer learning methods. Similarly, the corresponding generalization performance indices, with the best parameter values obtained by CV strategy, on the test data in the target domain are shown in Part D of Table VI. In this table, the adopted performance index, i.e., *Jreg*, is defined in Eq. (52.a) for the regression task. The lower the value of this index, the better the modeling effect, i.e., the better generalization abilities. From the results in Table VI, the following observations can be made.

(1) The results in Part A show that the generalization performance of different models trained by the proposed GHRR based only on the data in the source domain are comparable with the performance of the classical learning algorithms, including RBF-NN, L2-TSKFLS and LS-SVR. The results in Part B and Part C of Table VI, where the models are trained with the data in the target domain and both domains respectively, are similar to that in Part A.

(2) It can be seen by comparing the results in Part A and Part B that the models trained by only using the data in the source domain are not suitable for the regression task in the target domain. The performance of the models trained by using the data in the source domain is obviously inferior to the performance of those trained using the data in the target domain, even if the data in the target domain is insufficient.

(3) By comparing the results in Part C with that in Part A and Part B, we can see that the generalization performance of the models is not effectively improved by training the models directly by using the data in both domains. While the performance of models trained by using the data in both domains is better than that of models trained by using only data in the source domain, the performance is still inferior to that of the models trained by using only the data in the target domain. Thus, the results show that it is usually not effective to use the data in both domains for situations where the data is insufficient in the target domain due to drifting between the source domain and the target domain.

(4) From the results obtained by different inductive transfer learning methods as shown in Part D, we can see that the proposed knowledge-leveraged TGHRR algorithm has demonstrated an obvious advantage over the three classical inductive transfer learning regression algorithms, i.e., TrAdaBoost, HiRBF and SVR-AuD, which are developed to use the data of both source and target domains to realize transfer learning with some transfer learning strategies such as sample weighting.

(5) The results in Part C and Part D show that models trained by the three classical algorithms, TrAdaBoost, HiRBF and SVR-AuD, demonstrated improved generalization performance

when compared to the models trained by directly using the data of both domains. However, the models trained by these classical inductive transfer learning algorithms are not significantly advantageous over the models trained with the data in the target domain only, as shown in Part B and Part D of Table IV.

(6) It is clear from the performance of the different models trained by GHRR and TGHRR that TGHRR outperforms GHRR in the training of neural networks, fuzzy systems and kernel methods in the situations where transfer learning is required due to the lack of training data in the target domain.

To provide an intuitive illustration of the transfer learning abilities of TGHRR, the modeling effect of GHRR and TGHRR for training a TSKFLS with RBF-type membership functions, denoted by TSKFLS (RBF), is shown in Fig. 5. In the figure, the modeling effect of GHRR using the data in the source domain, target domain and both domains is presented together with that of TGHRR. Fig. 5(a) shows the effect of GHRR by only using the data in the source domain. Obviously, the model trained in the source domain is not suitable for the target domain.

Fig. 5(b) shows the effect of GHRR by only using the data in the target domain. While the performance as shown in Fig. 5(b) is better than that in Fig. 5(a), since the data in the target domain is, after all, not sufficient for training, the performance can be further improved.

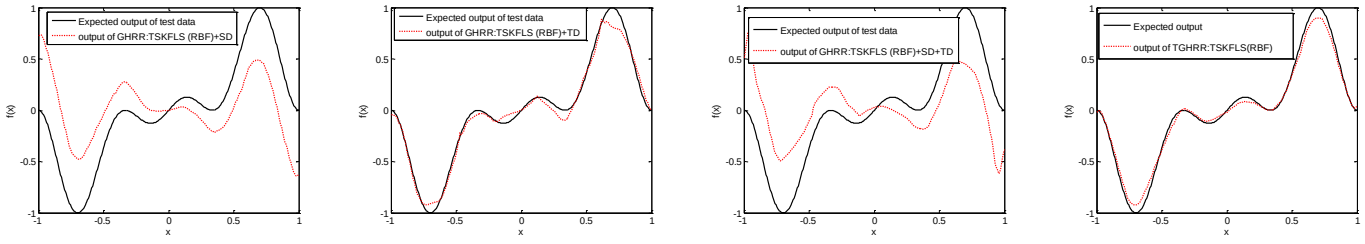
Fig. 5(c) shows the effect of GHRR by using the data in both domains directly. By comparing Fig. 5(b) with Fig. 5(c), we see that it is not effective to use the data in both domains to make up the deficiency caused by data insufficiency in the target domain. This can be explained with two reasons. First, there exists a drifting phenomenon between the source and target domain, i.e., not all data in the source domain are useful for the modeling task of the target domain and some of them may even produce negative influence. Second, the size of the source domain is larger than that of the target domain, which makes the obtained model more apt to approximate the source domain rather than the target domain.

Fig.5 (d) shows the effect of TGHRR by using the knowledge-leverage based transfer learning strategy. Compared with the other three non-transfer GHRR methods, TGHRR demonstrates the best modeling effect. The results indicate that TGHRR can effectively remedy the deficiency caused by data insufficiency in the target domain by leveraging the knowledge induced from the source domain.

TABLE VI. REGRESSION PERFORMANCE (J_{REG}) ON THE SYNTHETIC DATASETS AND THE BEST PARAMETER VALUES OF DIFFERENT ALGORITHMS.

Performance index of non-transfer algorithms (Part A, B and C)									
	RBF-NN	L2-TSKFLS	LS-SVR	GHRR					
				SHFN (RBF)	SHFN (Sigm)	MHFN (RBF)	MHFN (Sigm)	TSKFLS (RBF)	KRR (RBF)
Part A	0.8491	0.8479	0.8475	0.8489	0.8479	0.8498	0.8489	0.8497	0.8486
Part B	0.1294	0.1195	0.2269	0.1362	0.1481	0.1082	0.1478	0.1141	0.1507
Part C	0.7636	0.5892	0.8133	0.7658	0.7765	0.7719	0.7725	0.7663	0.7603
Performance index of transfer learning algorithms (Part D)									
	TrAdBoost (LS-SVR +RBF)	HiRBF	SVR-AuD (LP-SVR +RBF)	TGHRR					
				SHFN (RBF)	SHFN (Sigm)	MHFN (RBF)	MHFN (Sigm)	TSK-FLS (RBF)	KRR (RBF)
Part D	0.6909	0.6177	0.8147	0.0721	0.0758	0.0973	0.0804	0.0839	0.0692

*Part A, B and C are the results of models learned from the data in the source domain, the target domain and both domains, respectively.



(a) (b) (c) (d)

Fig. 5 Modeling effect of TSKFLS(RBF) using GHRR and TGHRR: (a) TSKFLS(RBF) trained by GHRR based on the data in the source domain; (b) TSKFLS(RBF) trained by GHRR based on the data in the target domain; (c) TSKFLS(RBF) trained by GHRR based on the data in both domains; (d) TSKFLS(RBF) trained by TGHRR.

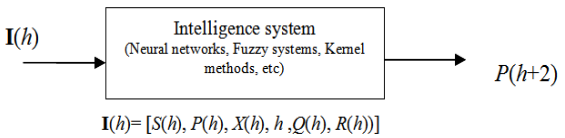


Fig. 6 Generalized intelligence system of the glutamic acid fermentation process prediction model.

C. On Real World Datasets for Biochemical Processing Modeling

1) The Glutamic Acid Fermentation Process Modeling: To further evaluate the performance of the proposed GHRR and TGHRR algorithms, an experiment was conducted to apply the proposed algorithms to model a biochemical process [36, 41, 47]. The adopted real-world regression dataset originated from a glutamic acid fermentation process. The input variables of the dataset include the fermentation time h, glucose concentration $S(h)$, thalli concentration $X(h)$, glutamic acid concentration $P(h)$, stirring speed $R(h)$, and ventilation $Q(h)$ at time h, where $h = 0, 2, \cdots, 28$. The output variable is glutamic acid concentration $P(h+2)$ at a future time $h+2$. Fig. 6 illustrates a generalized intelligence system of the biochemical process prediction model. The data in this experiment were collected from 21 batches of fermentation processes with each batch extracting 14 effective

data samples. In this experiment, in order to match the situation concerned in this study, the data were divided into two domains, i.e., the source domain and the target domain, as described in Table VII.

TABLE VII. THE FERMENTATION PROCESS MODELING DATASETS

	Data of Source Domain (D1)	Data of Target Domain	
		Training set (D2) *	Testing set (D2_test)
Batches	1-15	16-18	19-21
Size of dataset	210	21	42

*For the training set in the target domain, there are missing information at sampling time h = 2, 6, 10, … , 28.

TABLE VIII. MODELING PERFORMANCE (J_{REG}) OF DIFFERENT REGRESSION ALGORITHMS ON THE FERMENTATION PROCESS MODELING REGRESSION DATASETS.

Performance index of non-transfer algorithms (Part A, B and C)*						
	RBF-NN	L2-TSKFLS	LS-SVR	GHRR		
				SHFN (RBF)	TSKFLS (RBF)	KRR (RBF)
Part A	0.4115	0.3618	0.4211	0.4105	0.3793	0.3654
Part B	0.6297	0.4220	0.3827	0.5970	0.3444	0.4587
Part C	0.3513	0.3463	0.3687	0.3729	0.3488	0.3262
Performance index of transfer learning algorithms (Part D)						
	TrAdBoost (LS-SVR +RBF)	HiRBF	SVR-AuD (LP-SVR +RBF)	TGHRR		
				SHFN (RBF)	TSKFLS (RBF)	KRR (RBF)
Part D	0.4823	0.5910	0.3608	0.3269	0.3184	0.2905

*Part A, B and C are the results of models learned from the data in the source domain, the target domain and both domains respectively.

2) Results and Discussion: In this subsection, the performance of the glutamic acid fermentation process modeling achieved by the proposed methods and the related methods is compared. For simplicity and to save space, the proposed GHRR and TGHRR methods are only used to train the following three representative models: SHFN(RBF), TKKFLS(RBF) and KRR(RBF). The experimental results are shown in Table VIII, which is again divided into four parts in the same way as discussed previously.

As in Table VI, the adopted performance index in Table VIII is defined in Eq. (52.a), i.e., *Jreg*, for the biochemical modeling regression task. The lower the value of this index, the better the modeling effect. The following findings can be obtained from the experimental results given in Table VIII.

(1) Part A shows that the generalization performance of the different models trained by GHRR, based only on the data in the source domain, are comparable with that of the models trained by the classical algorithms, RBF-NN, L2-TSKFLS and LS-SVR.

(2) The results in Part B and Part C, where the models are trained by the data in the target domain and both domains respectively, give the similar conclusions as that in Part A.

(3) It can be seen from the results in Part A and Part B that the performance of the models, trained with the data in the source domain only, is better than those trained using the data in the target domain only. This finding reveals that the data in the target domain are severely insufficient and transfer learning is needed.

(4) By comparing Part C with Part A and B, it is found that the models trained by directly using the data in both the source and target domains could not satisfactorily improve the generalization performance of models. Although the performance is already better than the models trained by using the data in either the source domain or target domain, the improvement is indeed rather weak.

(5) The results of the different inductive transfer learning regression methods in Part D clearly indicates that the proposed TGHRR algorithm demonstrates superior performance to the three existing inductive transfer learning algorithms, TrAdBoost, HiRBF and SVR-AuD.

(6) It can be seen by comparing Part D with Part C that models trained by the classical transfer learning algorithms demonstrate improved performance when the data in both domains are used directly for the training. However, when the results in Part B and Part D are compared, it is found that the models trained by the existing inductive transfer learning algorithms do not perform better than the models trained by using only the data in the target domain.

(7) The results of the different models trained by GHRR and TGHRR indicate that it is advantageous to use TGHRR for training neural networks, fuzzy systems and kernel methods when the training data in the target domain is insufficient.

D. On Real World Datasets for Text Classification

1) Email Spam Filtering Text Dataset: The proposed approach was also applied to text classification for email spam filtering. The aim of the experiment was to design a server-based spam filter learned from public sources and apply it to individual users with the aid of transfer learning. The email spam data set, released by the ECML/PKDD Discovery Challenge 2006 [46], was adopted in the experiment. The data contains a set of publicly available messages and three sets of email messages from three individual users. 4000 samples were taken from the publicly available messages and 2500 samples of different word distributions were obtained respectively from the email messages created by the three users. In this experiment, the task was to classify spam and non-spam emails. With reference to [46], the three settings shown in Table IX were considered, namely ESF-PubvsUser1, ESF-PubvsUser2 and ESF-PubvsUser3. In each setting, the dataset included all the 4000 samples taken from the public messages, which were labeled to constitute the source domain; whereas the target domain comprised of the 2500 samples in which 2% were labeled as training data and the rest were unlabelled for testing. As in [46], each email message was characterized by using word-frequency features and 800 features with high frequency were selected in our experiment. As shown in the Table IX, in terms of the ratio of the training data and test data in the target domain, the whole dataset in the target domain were randomly partitioned and different algorithms were applied. The experiment was repeated 20 times to obtain the means and standard deviations of the classification accuracies. For classification task, the adopted performance index is defined in Eq. (52.b), i.e., *Jclas*. The higher the value of this index, the better the modeling effect.

TABLE IX. DESCRIPTION OF THE ADOPTED EMAIL SPAM FILTERING DATASETS

Datasets	**Source Domain**	**Target Domain**		
			Training set	Test set
ESF-PubvsUser1	Public (size :4000)	User1 (size : 2500)	2%	98%
ESF-PubvsUser2		User2 (size : 2500)	2%	98%
ESF-PubvsUser3		User3 (size : 2500)	2%	98%

2) Results and Discussion: The results of the experiment are reported in Tables X-XII, which are also divided into four parts in

same way as described in the previous experiments. Note that the mean and standard deviations of the classification accuracies are given with the latter put inside brackets. The following observations can be made from the results.

(1) The results in Part A of Tables X-XII show that the generalization performance of the models trained by the proposed GHRR based on the data in the source domain are comparable with the performance of the models trained by the classical algorithms.

(2) Comparing Part A and Part B of Tables X-XII respectively, we can see that the models trained by using only the data in the source domain are not suitable for the classification task in the target domain. The performance of the models trained by using the data in the source domain is obviously inferior to those trained by using the data in the target domain, even if the size of data in the target domain is small.

(3) Besides, by comparing the results in Part C with that in Part A and Part B of Tables X-XII respectively, it is found that the models trained by directly using the data in both domains cannot effectively improve the classification performance when compared with the performance of the models trained by using the data in either the source or the target domain. Furthermore, we can also see that the performance of models trained by using data in both domains is also inferior to the performance of models trained by only using data in the target domain. This reveals that the data in the source domain have negative effect on the models when the data in both domains are used simultaneously.

(4) The results of different inductive transfer learning methods in Part D of Tables X-XII show that the performance of the proposed TGHRR algorithm is better than that of the three classical inductive transfer learning regression algorithms, TrAdBoost, HiRBF and SVR-AuD.

(5) Again, the results obtained from the different models trained by GHRR and TGHRR clearly show that TGHRR is advantages over GHRR because of its ability to effectively leverage the knowledge from the source domain even when the data in the target domain is insufficient for training.

TABLE X. CLASSIFICATION PERFORMANCE OF THE EMAIL SPAM FILTERING WITH TEXT DATA ESF-PUBVSUSER1

Performance index of non-transfer algorithms (Part A, B and C)*						
	RBF-NN	KNN	C-SVC	GHRR		
				SHFN (RBF)	TSKFLS (RBF)	KRR (RBF)
Part A	0.5632 (0.0021)	0.6099 (0.0011)	0.4791 (0.0012)	0.5057 (0.0388)	0.5232 (0.0101)	0.5117 (0.0006)
Part B	0.855 (0.0352)	0.7932 (0.0567)	0.4986 (0.0006)	0.8420 (0.0960	0.8653 (0.0488)	0.8589 (0.0287)
Part C	0.7730 (0.0312)	0.7039 (0.0284)	0.5925 (0.1269)	0.7099 (0.0334	0.7714 (0.0411)	0.7937 (0.0240)
Performance index of transfer learning algorithms (Part D)						
	TrAdBoost (LS-SVC +RBF)	SVC-AuD LP-SVR	KNN-AuD	TGHRR		
				SHFN (RBF)	TSKFLS (RBF)	KRR (RBF)
Part D	0.8285 (0.0423)	0.7730 (0.0866)	0.7714 (0.0999)	0.8684 (0.0230)	0.8761 (0.0450)	0.8714 (0.0226)

*Part A, B and C are the results of models learned from the data in the source domain, the target domain and both domains, respectively.

TABLE XI. CLASSIFICATION PERFORMANCE OF THE EMAIL SPAM FILTERING WITH TEXT DATA ESF-PUBVSUSER2

Performance index of non-transfer algorithms (Part A, B and C)*						
	RBF-NN	KNN	C-SVC	GHRR		
				SHFN (RBF)	TSKFLS (RBF)	KRR (RBF)
Part A	0.5730 (0.0012)	0.5793 (0.0011)	0.5820 (0.0014)	0.5710 (0.0168)	0.5952 (0.0051)	0.5937 (0.0020)
Part B	0.8677 (0.0304)	0.8382 (0.0417)	0.8346 (0.0004)	0.8453 (0.0236)	0.8624 (0.0261)	0.8741 (0.0269)
Part C	0.8155 (0.0449)	0.7022 (0.0368)	0.5000 (0.0012)	0.7801 (0.0150)	0.8182 (0.0280)	0.8378 (0.0276)
Performance index of transfer learning algorithms (Part D)						
	TrAdBoost (LS-SVC +RBF)	SVC-AuD LP-SVR	KNN-AuD	TGHRR		
				SHFN (RBF)	TSKFLS (RBF)	KRR (RBF)
Part D	0.8212 (0.0584)	0.7357 (0.1241)	0.8532 (0.0153)	0.8678 (0.0186)	0.8802 (0.0243)	0.8910 (0.0106

*Part A, B and C are the results of models learned from the data in the source domain, the target domain and both domains, respectively.

TABLE XII. CLASSIFICATION PERFORMANCE OF THE EMAIL SPAM FILTERING WITH TEXT DATA ESF-PUBVSUSER3

Performance index of non-transfer algorithms (Part A, B and C)*						
	RBF-NN	KNN	C-SVC	GHRR		
				SHFN (RBF)	TSKFLS (RBF)	KRR (RBF)
Part A	0.3106 (0.0022)	0.6346 (0.0009)	0.6698 (0.0015)	0.6310 (0.0119)	0.2827 (0.063)	0.5000 (0.0007)
Part B	0.8653 (0.0265)	0.8763 (0.0159)	0.4987 (0.0004)	0.8262 (0.0751)	0.9020 (0.0233)	0.9130 (0.0277)
Part C	0.7510 (0.0165)	0.7722 (0.0105)	0.5000 (0.0014)	0.8046 (0.0245)	0.8282 (0.0360)	0.8635 (0.0074)
Performance index of transfer learning algorithms (Part D)						
	TrAdBoost (LS-SVC +RBF)	SVC-AuD LP-SVR	KNN-AuD	TGHRR		
				SHFN (RBF)	TSKFLS (RBF)	KRR (RBF)
Part D	0.6828 (0.0423)	0.8364 (0.0092)	0.8693 (0.0188)	0.8814 (0.0303)	0.9212 (0.0222)	0.9408 (0.0137)

*Part A, B and C are the results of models learned from the data in the source domain, the target domain and both domains, respectively.

VI. DISCUSSIONS

In this section, further discussions on the comprehensive experiments presented and potential future work are given.

From the results in section V, although it is apparent that the transfer learning algorithm TGHRR outperforms the non-transfer counterparts and other related algorithms, the degrees of improvement are different on different datasets. For example, the performance improvement on the text classification dataset ESF-PubvsUser3 is much obvious than that on the datasets ESF-PubvsUser1 and ESF-PubvsUser2. There are two possible reasons for this observation: (1) the real-world datasets could be so complicated that it may not be in accordance with the scene considered in this study. That is, if the data and knowledge in the source is really useful for the target domain, the performance improvement will be more significant; otherwise, the effect will be much weaker; (2) the transfer learning abilities of the proposed knowledge-leverage based transfer learning strategy, as shown in Eq. (36.a) or (36.b), are probably not effective enough, suggesting that there are rooms for further improvement of more advanced knowledge-leverage based transfer learning strategy.

Although TGHRR has demonstrated promising performance, there are still many issues requiring in-depth investigation in the future. Here, two potential improvements can be made for TGHRR. First, the adopted knowledge-leverage term, as shown in the objective function in Eq. (36.a) or (36.b), is relatively simple. More information, such as the statistical information [48], can indeed be introduced into the knowledge-leverage term, which is expected to further enhance the transfer learning abilities. In general, different strategies can be tried out to develop the objective functions of the modified methods. Although it may be much more difficult to directly adopt the developed objective functions in the experiments, this approach can potentially give rise to useful strategies for improving the performance of the proposed TGHRR method and thus deserve further investigation. Second, in the learning procedure of the proposed TGHRR, only the labeled data in the target domain are used while the unlabeled

data in the target domain is omitted. To deal with this issue, learning strategies in transductive transfer learning methods can be introduced to enhance the transfer learning abilities. For example, the strategy of minimizing the projected distribution distance between the source and target domains can be adopted for the design of knowledge-leverage term in the objective function of TGHRR [49].

VII. Conclusions

In this study, the GHRR method is introduced for the training of several types of classical intelligence models, including neural networks, fuzzy logical system and kernel methods. Further, the knowledge-leverage based transfer learning mechanism is introduced for the proposed GHRR to develop the TGHRR algorithm, which has been evaluated comprehensively with a number of experiments performed on synthetic and real world datasets for classification and regression tasks. The results show that the TGHRR demonstrate better performance and adaptability than the existing state-of-the-art inductive transfer learning algorithms for regression and classification.

While the proposed TGHRR is a promising machine learning algorithm, as discussed in section VI, there are still many issues to be solved and further in-depth study is required. For example, it is very important to develop new transfer GHRR methods by introducing stronger and more robust knowledge-leverage mechanism. This will be a main direction of our future research.

References

[1] S.J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowledge Data Engineering*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.

[2] X. Liao, Y. Xue, and L. Carin, "Logistic regression with an auxiliary data source," *Proc. 21st Int. Conf. Machine Learning*, pp. 505-512, Aug. 2005.

[3] J. Huang, A. Smola, A. Gretton, K.M. Borgwardt, and B. Schölkopf, "Correcting sample selection bias by unlabeled data," *Proc. 19th Ann. Conf. Neural Information Processing Systems*, 2007.

[4] S. Bickel, M. Brückner, and T. Scheffer, "Discriminative learning for differing training and test distributions," *Proc. 24th Int. Conf. Machine Learning, pp. 81-88*, 2007.

[5] M. Sugiyama, S. Nakajima, H. Kashima, P.V. Buenau, and M. Kawanabe, "Direct importance estimation with model selection and its application to covariate shift adaptation," *Proc. 20th Ann. Conf. Neural Information Processing Systems*, Dec. 2008.

[6] N.D. Lawrence and J.C. Platt, "Learning to learn with the informative vector machine," *Proc. 21st Int. Conf. Machine Learning*, July 2004.

[7] A. Schwaighofer, V. Tresp, and K. Yu, "Learning Gaussian process kernels via hierarchical Bayes," *Proc. 17th Ann. Conf. Neural Information Processing Systems*, pp. 1209-1216, 2005.

[8] J. Gao, W. Fan, J. Jiang, and J. Han, "Knowledge transfer via multiple model local structure mapping," *Proc. 14th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, pp. 283-291, Aug. 2008.

[9] L. Mihalkova, T. Huynh, and R.J. Mooney, "Mapping and revising markov logic networks for transfer learning," *Proc. 22nd Assoc. for the Advancement of Artificial Intelligence (AAAI) Conf. Artificial Intelligence*, pp. 608-614, July 2007.

[10] L. Mihalkova and R.J. Mooney, "Transfer learning by mapping with minimal target data," *Proc. Assoc. for the Advancement of Artificial Intelligence (AAAI '08) Workshop Transfer Learning for Complex Tasks*, July 2008.

[11] J. Davis and P. Domingos, "Deep transfer via second-order Markov logic," *Proc. Assoc. for the Advancement of Artificial Intelligence (AAAI '08) Workshop Transfer Learning for Complex Tasks*, July 2008.

[12] S.J. Pan, I.W. Tsang, J.T. Kwok, and Q. Yang, "Domain adaptation via transfer component analysis," *IEEE Trans. Neural Networks*, vol. 22, no.2, pp.199-210, 2011.

[13] L.X. Duan, D. Xu, I. W. Tsang, "Domain adaptation from multiple sources: a domain-dependent regularization approach," *IEEE Trans. Neural Netw. Learning Syst., vol. 23*, no. 3, pp. 504-518, 2012.

[14] L.X. Duan, I.W. Tsang, D. Xu, "Domain Transfer Multiple Kernel Learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no.3, pp. 465-479, 2012.

[15] H. Daumé III, D. Marcu, "Domain Adaptation for Statistical Classifiers," *J. Artif. Intell. Res*., 2006, vol. 26, pp. 101-126.

[16] P. Yang, Q. Tan, and Y. Ding, "Bayesian task-level transfer learning for non-linear regression," *Proc. Int. Conf. on Computer Science and Software Engineering*, pp. 62-65, 2008.

[17] L. Borzemski and G. Starczewski, "Application of transfer regression to TCP throughput prediction," *Proc. First Asian Conference on Intelligent Information and Database Systems*, pp. 28-33, 2009.

[18] W. Mao, G. Yan, J. Bai, and H. Li, "Regression transfer learning based on principal curve," *Lecture Note on Computer Science 6063*, pp. 365–372, 2010.

[19] J. Liu, Y. Chen, and Y. Zhang, "Transfer regression model for indoor 3D location estimation," *Lecture Note on Computer Science 5916*, pp. 603–613, 2010.

[20] D. Pardoe and P. Stone, "Boosting for regression transfer," *Proc. Int. Conf. Machine Learning*, pp. 863-870, 2010.

[21] W. Dai, Q. Yang, G. Xue, and Y. Yu, "Self-taught clustering," *Proc. 25th Int. Conf. Machine Learning*, pp. 200-207, July 2008.

[22] Z. Wang, Y. Song, and C. Zhang, "Transferred dimensionality reduction," *Proc. European Conf. Machine Learning and Knowledge Discovery in Databases (ECML/PKDD '08)*, pp. 550-565, Sept. 2008.

[23] W.H. Jiang and F.L. Chung, "Transfer Spectral Clustering," *In Machine Learning and Knowledge Discovery in Databases*, pp. 789-803, Springer Berlin Heidelberg, 2012.

[24] Z.H. Deng, K.S. Choi, F.L. Chung, S.T. Wang, "Scalable TSK fuzzy modeling for very large datasets using minimal-enclosing-ball approximation," *IEEE Trans. Fuzzy System*, vol. 19, no.2, pp.210-226, 2011.

[25] W. Dai, Q. Yang, G. Xue, and Y. Yu, "Boosting for Transfer Learning," *Proc. 24th Int'l Conf. Machine Learning*, pp. 193-200, June 2007.

[26] J. Jiang and C. Zhai, "Instance Weighting for Domain Adaptation in NLP," *Proc. 45th Ann. Meeting of the Assoc. Computational Linguistics*, pp. 264-271, June 2007.

[27] P. Wu and T.G. Dietterich, "Improving SVM Accuracy by Training on Auxiliary Data Sources," *Proc. 21st Int'l Conf. Machine Learning*, July 2004.

[28] A. Argyriou, T. Evgeniou, and M. Pontil, "Multi-Task Feature Learning," *Proc. 19th Ann. Conf. Neural Information Processing Systems*, pp. 41-48, Dec. 2007.

[29] N.D. Lawrence and J.C. Platt, "Learning to Learn with the Informative Vector Machine," *Proc. 21st Int'l Conf. Machine Learning*, July 2004.

[30] T. Evgeniou and M. Pontil, "Regularized Multi-Task Learning," *Proc. 10th ACM SIGKDD Int'l Conf. Knowledge iscovery and Data Mining*, pp. 109-117, Aug. 2004.

[31] M. Richardson and P. Domingos, "Markov Logic Networks," *Machine Learning*, vol. 62, nos. 1/2, pp. 107-136, 2006.

[32] A.E. Hoerl, R.W. Kennard, "Ridge regression: biased estimation for nonorthogonal problems," *Technometrics*, vol.12, no. 1, pp. 55–67, 1970.

[33] J. Li, "Linear, Ridge Regression, and Principal Component Analysis", http://www.stat.psu.edu/_jiali.

[34] C. Saunders, A. Gammerman, V. Vovk, "Ridge regression learning algorithm in dual variables," *Proc. 5th International Conference on Machine Learning*, Madison, WI, July 24–27, 1998, pp. 515–521.

[35] J.S.R. Jang, C.T. Sun, and E. Mizutani, *Neuro-fuzzy and soft-computing*. Upper Saddle River, NJ: Prentice-Hall, 1997.

[36] Z. H. Deng, Y.Z Jiang, F.L. Chung, S.T. Wang, "Knowledge-Leverage Based Fuzzy System and Its Modeling", *IEEE Trans. on Fuzzy Systems*, vol. 21, pp. 4, pp. 597-609, 2013.

[37] G.-B. Huang, L. Chen, and C.-K. Siew, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *IEEE Trans. Neural Netw.*, vol. 17, no. 4, pp. 879–892, Jul. 2006.

[38] G.-B. Huang and L. Chen, "Convex incremental extreme learning machine," *Neurocomputing*, vol. 70, pp. 3056–3062, 2007.

[39] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its application to modeling and control," *IEEE Trans. Systems Man and Cybernetics*, vol.15, no.1, pp. 116-132, 1985.

[40] S. Chen, C.F.N. Cowan, and P. M. Grant, "Orthogonal Least Squares Learning Algorithm for Radial Basis Function Networks," *IEEE Transactions on Neural Networks*, vol. 2, no. 2, March 1991, pp. 302-309

[41] Z.H. Deng, K.S. Choi, F.L. Chung, S.T. Wang, "Scalable TSK fuzzy modeling for very large datasets using minimal-enclosing-ball approximation," *IEEE Trans. Fuzzy System*, vol. 19, no.2, pp.210-226, 2011.

[42] J.A.K. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural Process. Lett*., vol. 9, no. 3, pp. 293–300, Jun. 1999.

[43] R. Duda, P.Hart, andD. Stork, *Pattern Classification*. NewYork:Wiley, 2000.

[44] C. Cortes and V. Vapnik, "Support vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[45] P. Yang, Q. Tan, and Y. Ding, "Bayesian task-level transfer learning for non-linear regression," *Proc. Int. Conf. on Computer Science and Software Engineering*, pp. 62-65, 2008.

[46] S. Bickel. ECML-PKDD Discovery Challenge 2006 Overview. *In Proc.*

ECML/PKDD Discovery Challenge Workshop, 2006.

[47] Z.H. Deng, Y.Z. Jiang, K.S. Choi, F.L. Chung and S.T. Wang, "Knowledge-Leverage based TSK fuzzy system modeling," *IEEE Trans. Neural Networks and Learning Systems, in press*,vol. 24, no. 8, pp. 1200-1212, 2013.

[48] J.W. Tao, KF.L. Chung, S.T. Wang, "On minimum distribution discrepancy support vector machine for domain adaptation." *Pattern Recognition*, vol. 45, no.11, pp. 3962-3984, 2012.

[49] B. Quanz, J. Huan, "Large margin transductive transfer learning," In *Proceedings of the 18th ACM conference on Information and knowledge management*, pp. 1327-1336, 2009.