Deep Additive Least Squares Support Vector Machines for Classification With Model Transfer

Guanjin Wang, Guangquan Zhang, Kup-Sze Choi, *Member, IEEE*, and Jie Lu, *Senior Member, IEEE*

***Abstract*—The additive kernel least squares support vector machine (AK-LS-SVM) has been well used in classification tasks due to its inherent advantages. For example, additive kernels work extremely well for some specific tasks, such as computer vision classification, medical research, and some specialized scenarios. Moreover, the analytical solution using AK-LS-SVM can formulate leave-one-out cross-validation error estimates in a closed form for parameter tuning, which drastically reduces the computational cost and guarantee the generalization performance especially on small and medium datasets. However, AK-LS-SVM still faces two main challenges: 1) improving the classification performance of AK-LS-SVM and 2) saving time when performing a grid search for model selection. Inspired by the stacked generalization principle and the transfer learning mechanism, a layer-by-layer combination of AK-LS-SVM classifiers embedded with transfer learning is proposed in this paper. This new classifier is called deep transfer additive kernel least square support vector machine (DTA-LS-SVM) which overcomes these two challenges. Also, considering that imbalanced datasets are involved in many real-world scenarios, especially for medical data analysis, the deep-transfer element is extended to compensate for this imbalance, thus leading to the development of another new classifier iDTA-LS-SVM. In the hierarchical structure of both DTA-LS-SVM and iDTA-LS-SVM, each layer has an AK-LS-SVM and the predictions from the previous layer act as an additional input feature for the current layer. Importantly, transfer learning is also embedded to guarantee generalization consistency between the adjacent layers. Moreover, both iDTA-LS-SVM and DTA-LS-SVM can ensure the minimal leave-one-out error by using the proposed fast leave-one-out cross validation strategy on the training set in each layer. We compared the proposed classifiers DTA-LS-SVM and iDTA-LS-SVM with the traditional LS-SVM and SVM using additive kernels on seven public UCI datasets and one real world dataset. The experimental results show that both DTA-LS-SVM and iDTA-LS-SVM exhibit better generalization performance and faster learning speed.**

***Index Terms*—Classification, deep architectures, support vector machine (SVM), transfer learning.**

I. Introduction

THE ADDITIVE kernel least squares support vector machine (AK-LS-SVM) has been well used in many classification tasks due to its distinct advantages. For example, the generalization performances of kernel-based learning methods depend highly on the parameter selection, such as the model selection of good values for regularization and kernel parameters. To make an unbiased selection, parameter selection strategies based on the minimization of the leave-one-out cross validation estimate is usually preferred but with unavoidably expensive computation. However, since the analytical solution of AK-LS-SVM can formulate a simple yet effective leave-one-out cross validation estimate in a closed form, it only has a negligible additional computational cost [1]. This particularly benefits real world applications with small or medium size datasets. For example, many medical applications involve small or medium datasets due to the complex and/or expensive data collection processes and the concerns of the patients' privacy. Therefore, it becomes necessary for us to minimize the leave-one-out error on these datasets, which is an almost unbiased estimator of the generalization error [1]. In these scenarios LS-SVM can improve the generalization performances by using the fast leave-one-out cross validation estimate. Also, in terms of additive kernels in AK-LS-SVM, a kernel is additive if it can be broken down into a sum of kernel functions of each dimension [2], that is

$$k(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{d} k(x_i, y_i). \tag{1}$$

Additive kernels are widely used in different classification and regression tasks [3]–[5], such as the histogram intersection kernel in (2), the χ^2 kernel in (3) and the additive Gaussian kernel used in this paper [see (6)]

$$k_{\text{hist}}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{d} \min(x_i, y_i) \tag{2}$$

Manuscript received May 8, 2017; accepted September 17, 2017. This work was supported in part by the Australian Research Council under Grant DP140101366, in part by the Hong Kong Research Grants Council under Grant PolyU152040/16E, and in part by the YC Yu Scholarship for Centre for Smart Health. This paper was recommended by Associate Editor H. Ying. *(Corresponding author: Jie Lu.)*

G. Wang is with the Centre for Artificial Intelligence, School of Software, Faculty of Engineering and Information Technology, University of Technology Sydney, Broadway, NSW 2007, Australia, and also with the Centre for Smart Health, School of Nursing, Hong Kong Polytechnic University, Hong Kong (e-mail: guanjin.wang@student.uts.edu.au).

G. Zhang and J. Lu are with the Centre for Artificial Intelligence, School of Software, Faculty of Engineering and Information Technology, University of Technology Sydney, Broadway, NSW 2007, Australia (e-mail: guangquan.zhang@uts.edu.au; jie.lu@uts.edu.au).

K.-S. Choi is with the Centre for Smart Health, School of Nursing, Hong Kong Polytechnic University, Hong Kong (e-mail: thomasks.choi@polyu.edu.hk).

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/TSMC.2017.2759090

$$k_{\chi^2}(\boldsymbol{x}, \boldsymbol{y}) = \frac{2x_i y_i}{x_i + y_i}. \quad (3)$$

In addition, additive kernels exhibit their advantages when tackling special scenarios. For example, they have been used in handling missing data in community health studies [6], where only the kernel functions of each dimension without missing values are summed together. In [7], additive kernels are expected to well represent the distance/similarity measures for medical data with mixed types of features, such as features that change over time or features that do not change.

Based on the above summary, we found that the application of AK-LS-SVM has huge potential in the real world. However, it still faces two main challenges: 1) the classification performance of AK-LS-SVM is comparatively low and 2) the grid search for good values for model selection is still time consuming. The aim of this current work is to develop a stacked hierarchical AK-LS-SVM classifier, i.e., deep transfer additive least square support vector machine (DTA-LS-SVM), by integrating the stacked generalization principle and the transfer learning mechanism, to overcome these two challenges. Furthermore, many real-world scenarios involve imbalanced datasets. For example, in medicine, there may be only a few examples of patients diagnosed as having contracted diseases compared to those who are healthy. We consider these scenarios and extend DTA-LS-SVM to iDTA-LS-SVM for handling imbalanced datasets. In the proposed hierarchical structure, each layer has an AK-LS-SVM. Following the stacked generalization principle, the original data inputs and the predictions from the previous layer become the new data input of the next layer to enhance the performance of the whole stacked architecture. Moreover, transfer learning is used to guarantee a consistency between the adjacent layers such that the previously learned model knowledge from the previous layer can be leveraged for model construction in the adjacent higher layer to further improve the generalization capability. The contributions of this paper can be summarized below.

1) The novel classifiers DTA-LS-SVM and iDTA-LS-SVM proposed can significantly enhance the generalization performance of AK-LS-SVM, respectively, on balanced and imbalanced datasets. Following the stacked generalization principle, the proposed classifiers organize multiple modules in a layer-by-layer structure in which every module contains an AK-LS-SVM. From the second layer, the data input space for each layer comprises the original data features and the predictions from the previous layer as an appended feature. Through this deep stacking architecture, the appended feature space helps open the manifold structure in the original data space in a stacked way so as to achieve the classification performance improvement.
2) Transfer learning is embedded from the second layer in the proposed classifiers DTA-LS-SVM and iDTA-LS-SVM to ensure that the consistency across adjacent layers is guaranteed and the classification capability of the higher layer can be further enhanced. Moreover, the kernel and regularization parameters are randomly selected in DTA-LS-SVM and iDTA-LS-SVM, and AK-LS-SVM with transfer learning in each layer to ensure the minimal leave-one-out error on the training set of each layer is achieved by using the proposed fast leave-one-out cross validation strategy.
3) The experimental results show that both DTA-LS-SVM and iDTA-LS-SVM give improved generalization performance and fast learning speed on seven balanced and imbalanced UCI datasets and one real community healthcare dataset, compared with the traditional least square support vector machine (LS-SVM) and support vector machine (SVM) using the same additive kernels.

The remainder of this paper is organized as follows. The related work is introduced in Section II. In Section III the proposed classifier DTA-LS-SVM is presented. In particular, a fast leave-one-out cross validation strategy for parameter tuning is introduced. In Section IV, the experimental results are provided on the seven UCI datasets and one real community healthcare dataset. Finally, the conclusions and future work are given in Section V.

II. Related Work

In this section some background knowledge of LS-SVM and AK-LS-SVM is described because this paper involves both proposed classifiers integrated with transfer learning in a deep structure.

A. Shallow and Deep Architecture

Classifiers in shallow architecture traditionally consist of an input layer, a single processing layer, and an output layer. For example, LS-SVM and AK-LS-SVM used in this paper have a layer of kernel function which is applied to the input, followed by a linear combination of the kernel outputs. Such shallow machines have been widely used in classification and regression with good performances [8]–[10]. However, shallow architectures have problems in providing an efficient representation for certain types of functions. Deep structures are known to comprise several layers of nonlinear operations, such as in the neural network with many hidden layers, which aim to learn hierarchical representations. Benefiting from the multiple levels of representation and abstraction, deep architectures learn the complicated functions mapping the input to the output of the data.

Using shallow classifiers like SVM in combination with deep architectures has been proposed in the literature. For example, Abdullah *et al.* [11] presented a deep SVM (D-SVM), which used the nonlinear combinations of kernel activations of support vectors as inputs to train an SVM on the upper layer. Moreover, the D-SVMs were combined with the product rule ensemble for combining multiple image descriptors. Tang [12] used linear SVMs rather than a softmax activation function in convolutional neural networks (CNNs) to learn to minimize a margin-based loss instead of the cross-entropy loss. In this paper, we focus on the least squares version of SVM in deep architecture.

B. Deep Stacked Architecture

There are different types of deep architectures, such as CNNs [13], deep belief networks (DBNs) [14], deep

Boltzmann machines [15], and deep auto encoders [16], etc. In this paper, the proposed classifier DTA-LS-SVM is built in a deep stacked architecture [17], which is trained in a supervised and module-wise fashion. Unlike other deep architectures, such as DBNs, it does not employ back propagation over all modules and does not aim to find the transformed feature representation. To learn complex functions effectively, it stacks multiple modules in a chain, and the outputs from the previous layer are fed into the inputs of the module of the next layer. Due to this nature of building a hierarchy of simplified modules trained by itself, deep stacked architecture is relatively simple and easy to implement. Through the deep stacked architecture, the appended feature space opens the original data manifolds. This architecture follows the philosophy of "stacked generalization [18]" which indeed enhances classification performances and improves generalization in learning complex functions. In order to learn complex functions or classifiers effectively, it stacks multiple modules in a chain, and the outputs from the module(s) from the previous layer are fed into the inputs of the module of the next layer. Deep stacked architecture was first introduced in 2011 [17]. After that, a kernel version DSNs was proposed by Deng *et al.* [19], which integrates deep learning and kernel learning. By using the kernel trick, the hidden neurons in each DSN layer becomes infinity. Another novel DSNs, which is called tensor-DSNs was presented by Hutchinson *et al.* [20]. In this method, each module has a bilinear mapping from two hidden layers to the output layer by incorporating higher order statistics of the hidden binary features through a weight tensor. Vinyals *et al.* [21] proposed a recursive perceptual representation using layers of linear SVMs and incorporating with random projections of weak predictions from each layer.

Through the deep stacked architecture, the appended feature space using the predictions from the module(s) of the previous layers open the original data manifolds such that the generalization performances may be improved. There are two ways to append the feature space with the increase of the depth in deep stacking architecture. As depicted in Fig. 1(a), the new feature space for the module in the higher layer comes from the concatenation of the predicted outputs from all the modules of the previous layers and the original input features. The mentioned work above belongs to this category. Fig. 1(b) illustrates a different type to append feature space on which this paper concentrates. In this paper, the new feature space for the module of the higher layer comes from the concatenation of the predicted outputs, that is, from the module of the original input features and the predicted outputs only for the module of the previous layer.

C. Transfer Learning

Transfer learning has been another important research topic in machine learning. Given a source domain D_S and its learning task T_S, a target domain D_T and its learning task T_T, The goal of transfer learning is to help improve the learning process in D_T by leveraging the knowledge in D_S, when $D_S \neq D_T$, or $T_S \neq T_T$. In our scenario, from the bottom to top in the deep stacked architecture, the module of the previous layer is used as the source domain, and the adjacent module of next layer is used as the target domain.

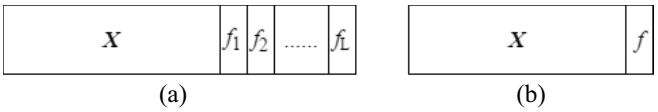

Fig. 1. Two ways to expand the feature space in deep stacked architecture. (a) f_l is the prediction from the previous module l. (b) f changes with the current module layer. For example, in module 2, $f = f_1$, in module 3, $f = f_2, \ldots,$ in module L, $f = f_{L-1}$.

In the proposed deep stacking architecture, the adjacent models always have the same original feature space, and an additional feature is taken as the appended feature for the predictions of the adjacent layers to reflect the discriminative information about the classes in the original feature space. Therefore, we can postulate that there is a certain consistency between D_S and D_T such that transfer learning between the adjacent layers can be introduced. In other words, it is worthwhile embedding transfer learning into the deep stacking architecture, which uses previously learned knowledge from D_S to help the learning process in D_T.

To deploy transfer learning, the main question is "what to transfer." It asks which part of knowledge is used to transfer across domains or tasks. From the literature, the leveraged knowledge can be categorized into instances-, feature-, or model/parameters-based transfer learning [22]. The instance-transfer approach reweights certain data points in the source domain for use in the target domain [23], [24]. The feature-representation-transfer approach encodes the transferred knowledge across domains into a shared representative feature structure, and the target model is guided in the new feature space [25], [26]. The model/parameter-transfer approach assumes that the source and target domains share parameters to some extent or prior distributions of the classifier [27], [28]. For readers who are interested, the following articles are suggested for further investigation: [22], [29], [30]. In this paper, the transfer learning used belongs to the model/parameters-based transfer category.

III. Proposed Deep Transfer Least Squares Support Vector Machine

We use $\boldsymbol{S} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$ to represent the dataset adopted in this paper. The input set is denoted as $\boldsymbol{X}$ and the corresponding output set is denoted as $\boldsymbol{Y}$. $\boldsymbol{x}_i = (x_1^i, x_2^i, \ldots, x_d^i) \in \boldsymbol{X} \subset \boldsymbol{R}^d$, $y_i \in \{+1, -1\}$, and each sample $\boldsymbol{x}_i$ $(i = 1, 2, \ldots, N)$ contains d features. The general framework of DTA-LS-SVM is illustrated in Fig. 2. It can be seen that the proposed approach is built with multiple layers of AK-LS-SVM-based modules in a stacked structure. The original data are given as data inputs to layer 1. For each module from layer 2, a transfer-based AK-LS-SVM is trained using the transformed data with the original features and the appended feature, where values are the previous layers' predictions. Moreover, model transfer is embedded to leverage relevant model knowledge from the adjacent module of the previous layer.

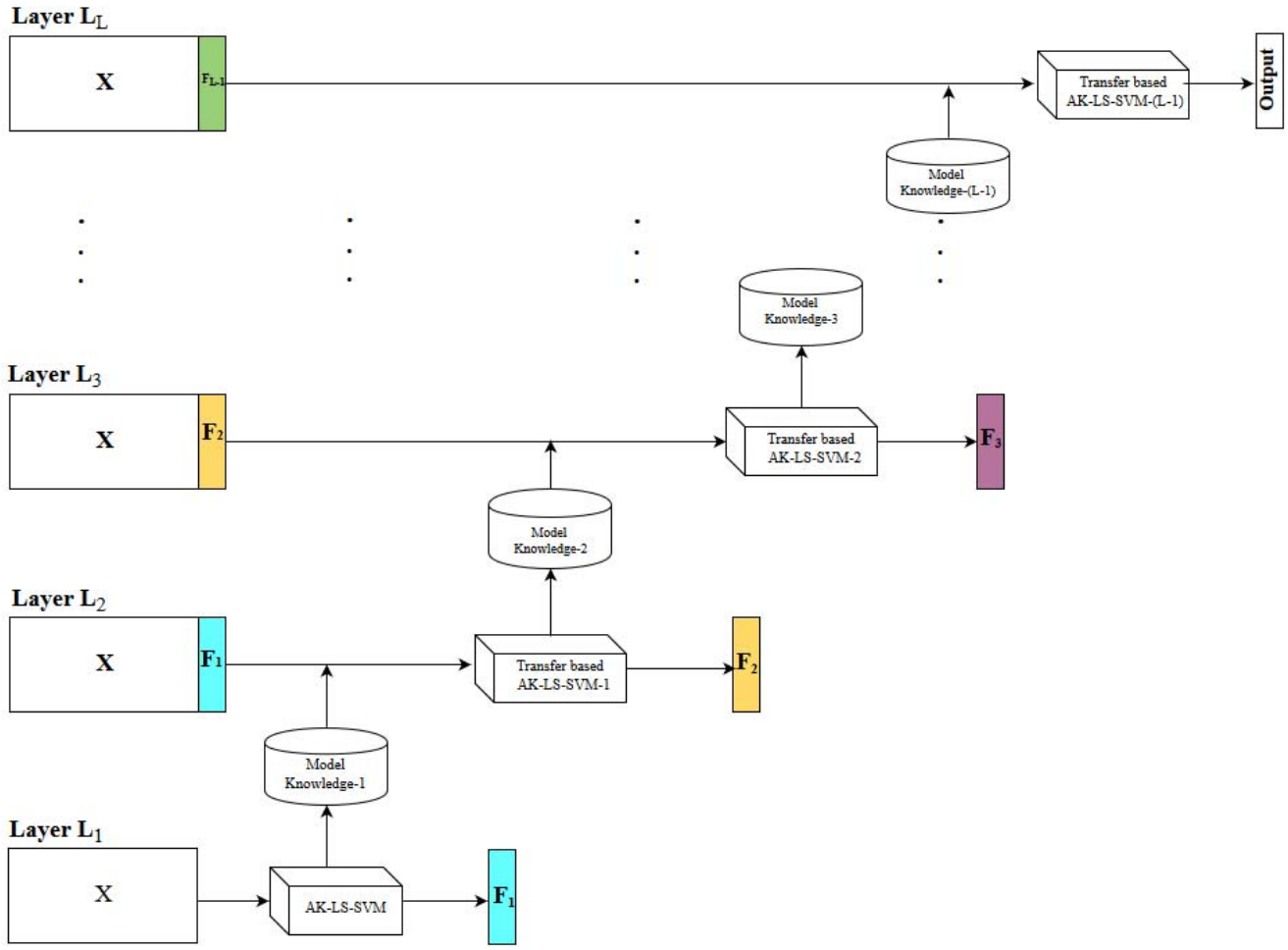

Fig. 2. Hierarchical architecture and learning process of DTA-LS-SVM.

In the whole framework, the stacked generalization principle and the unfolding of manifold structure existing in the original data input space by concatenating the predictions from the adjacent module of the previous layer guarantee the improvement on the generalization capability. Moreover, transfer learning is used to maintain similarity across the adjacent modules by leveraging the previously learned knowledge from the previous module such that the generalization capability also can be improved as well.

A. AK-LS-SVM and Its Adaptive Regularization

AK-LS-SVM is based on statistical learning theory which is formulated on the structural risk minimization principle. The learning process can be formalized as an optimization problem that minimizes the structural risk as follows:

$$\Omega(f) + C\sum_{i=1}^{N} R_{\text{emp}}(f(\boldsymbol{x_i}), y_i) \tag{4}$$

where $\Omega(f)$ is a regularization term that encodes some notion of smoothness for f and prevent overfitting. R_{emp} is a chosen convex loss function that evaluates the quality of f on the instance $\{\boldsymbol{x}_i, y_i\}$. C gives a tradeoff between the minimization of R_{emp} and the smoothness or simplicity enforced by a small $\Omega(f)$.

In the first processing layer L_1 in the framework of DTA-LS-SVM, a traditional AK-LS-SVM is used to find an optimal hyperplane $f(\boldsymbol{x}) = \boldsymbol{w}^T\varphi(\boldsymbol{x}) + b$. The regularization term is set to be $\Omega(f) = (1/2)\|\boldsymbol{w}\|^2$ and the loss function to be the weighted square loss $R_{\text{emp}}(f(\boldsymbol{x})_i, y_i) = (f(\boldsymbol{x}_i) - y_i)^2$. Therefore, the learning problem in (4) becomes

$$\begin{aligned} \min_{\boldsymbol{w},b} \; & J(\boldsymbol{w}, b) = \frac{1}{2}\|\boldsymbol{w}_1\|^2 + \frac{C_1}{2}\sum_{i=1}^{N}\xi_{i1}^2 \\ \text{s.t.} \; & y_i = \boldsymbol{w}_1^T\varphi(\boldsymbol{x}_{i1}) + b_1 + \xi_{i1} \\ & i = 1, 2, \ldots, n \end{aligned} \tag{5}$$

where C is the regularization parameter and $\varphi(\boldsymbol{x}_{i1}) = (\tilde{\varphi}(x_{11}^i), \tilde{\varphi}(x_{21}^i), \ldots, \tilde{\varphi}(x_{k1}^i), \ldots, \tilde{\varphi}(x_{d1}^i))$, and $\varphi(\boldsymbol{x}_{i1})$ is a feature mapping such that the additive kernel $\boldsymbol{K}$ below can be adopted in (5)

$$\boldsymbol{K}\left(\boldsymbol{x}_{i1}, \boldsymbol{x}_{j1}\right) = \varphi(\boldsymbol{x}_{i1})^T\varphi\left(\boldsymbol{x}_{j1}\right) = \textstyle\sum_{k=1}^{d} k\left(x_k^{i1}, x_k^{j1}\right). \tag{6}$$

In this paper, an additive Gaussian kernel [5] is adopted, i.e., $k(x_k^{i1}, x_k^{j1}) = e^{(-(x_k^{i1} - x_k^{j1})/\delta^2)}$, where δ is the kernel width.

Input	Features				
$\boldsymbol{X}'_l$	1	2	$\cdots$	d	$d+1$
$\boldsymbol{x}'_1$					$f_{l-1}(\boldsymbol{x}_1)$
$\boldsymbol{x}'_2$					$f_{l-1}(\boldsymbol{x}_2)$
$\vdots$					$\vdots$
$\boldsymbol{x}'_N$					$f_{l-1}(\boldsymbol{x}_N)$

Original data input $\boldsymbol{X}$ Augmented feature space $\boldsymbol{F}_{l-1}$

Fig. 3. Augmented space $\boldsymbol{X}'_l$ of $\boldsymbol{X}$.

From the next processing layer L_l, $(l = 2, 3, \ldots, L)$, transfer learning is used to leverage model knowledge from $D_{S(l-1)}$ (source domain) to D_{Tl} (target domain), For example, first, the optimal $\boldsymbol{w}_{S(l-1)}$ is found by minimizing (5) in $D_{S(l-1)}$, then when D_{Tl} is encountered, a model is constructed in which $\boldsymbol{w}_{Tl}$ gets as close as possible to the known $\boldsymbol{w}_{S(l-1)}$. Through editing the regularization term, the learning classification task from the second processing layer becomes the corresponding transfer-based AK-LS-SVM, that is

$$\frac{1}{2}\left\|\boldsymbol{w}_{Tl} - \boldsymbol{w}_{S(l-1)}\right\|^2 + \frac{C_l}{2}\sum_{i=1}^{N}\xi_{il}^2. \tag{7}$$

Moreover, the regularization term can be further revised to $\|\boldsymbol{w}_{Tl} - \lambda_l \boldsymbol{w}_{S(l-1)}\|$ by adding the weighting factor λ_l to control the degree to which the new model is close to the source model. This evaluates the similarity between $\boldsymbol{w}_{S(l-1)}$ and $\boldsymbol{w}_{Tl}$ in the optimization problem above.

B. Transfer Learning in DTA-LS-SVM and Its Fast Leave-One-Out Cross Validation

In this section, we describe in detail how to use transfer learning across modules in DTA-LS-SVM. As demonstrated in Fig. 2, the proposed classifier is built by multiple layers of modules, which from the second module, each learns a transfer-based AK-LS-SVM on the original data inputs with an additional feature whose values are the predictions from the module of the previous layer. The core motivation of DTA-LS-SVM is that when the deeper structure is well trained, it tends to do a better job at disentangling the underlying factors of variation [31]. The recursive leverage of previous predictions helps to move apart the manifolds in the original data in a stacked way such that a better separability can be achieved [21].

As mentioned in Section III-A, in the first processing layer L_1, a traditional AK-LS-SVM model is constructed using the additive kernel defined in (5) giving the decision function $f_1(\boldsymbol{x}_i) = \boldsymbol{w}_1^T\varphi(\boldsymbol{x}_i) + b_1$. Also the predicted label vector $\boldsymbol{F}_1$ of $\boldsymbol{X}$ is obtained, where $\boldsymbol{F}_1 = (f_1(\boldsymbol{x}_1), f_1(\boldsymbol{x}_2), \ldots, f_1(\boldsymbol{x}_N))$.

From the processing layer L_l $(l = 2, 3, \ldots L)$, the new data input $\boldsymbol{X}'_l$ is the augmentation of the original data input set $\boldsymbol{X}$ and the predicted label vector $\boldsymbol{F}_{l-1}$ from the previous layer, which can be denoted as $\boldsymbol{X} \oplus \boldsymbol{F}_{l-1}$ for simplicity. $\boldsymbol{X}'_l$ is illustrated in Fig. 3. The transfer-based AK-LS-SVM is used in L_l, which leverages the source model in L_{l-1} to have $\lambda_l \boldsymbol{w}_{l-1}$, which is represented as

$$\begin{aligned} \min\ & J(\boldsymbol{w}_l, b_l) = \frac{1}{2}\|\boldsymbol{w}_l - \lambda_l \boldsymbol{w}_{l-1}\|^2 + \frac{C_l}{2}\sum_{i=1}^{N}\xi_{li}^2 \\ \text{s.t.}\ & y_i = \boldsymbol{w}_l^T\varphi(\boldsymbol{x}'_i) + b_l + \xi_{li} \\ & i = 1, 2, \ldots, N. \end{aligned} \tag{8}$$

The corresponding Lagrangian $\mathcal{L}_l$ of (5) is

$$\begin{aligned} \mathcal{L}_l(\boldsymbol{w}_l, b_l, \boldsymbol{\xi}_l; \boldsymbol{\alpha}_l) &= J(\boldsymbol{w}_l, b_l) \\ &\quad + \sum_{i=1}^{N}\alpha_{li}(y_i - \boldsymbol{w}_l^T\varphi(\boldsymbol{x}'_i) - b_l - \xi_{li}) \end{aligned} \tag{9}$$

where $\boldsymbol{\alpha}_l \in \mathbf{R}^N$ is the vector of all Lagrangian multipliers. With respect to $\boldsymbol{w}_l$, ξ_{li}, and α_{li}, the optimal condition can be calculated by

$$\frac{\partial \mathcal{L}_l}{\partial \boldsymbol{w}_l} = 0 \Rightarrow \boldsymbol{w}_l = \lambda_l \boldsymbol{w}_{l-1} + \sum_{i=1}^{n}\alpha_{li}\varphi(\boldsymbol{x}'_i) \tag{10}$$

$$\frac{\partial \mathcal{L}_l}{\partial \xi_{li}} = 0 \Rightarrow \xi_{li} = \alpha_{li}/C_l \tag{11}$$

$$\frac{\partial \mathcal{L}_l}{\partial \alpha_{li}} = 0 \Rightarrow y_i = \boldsymbol{w}_l^T\varphi(\boldsymbol{x}'_i) + b_l + \xi_{li}. \tag{12}$$

Note that the optimal solution $\boldsymbol{w}_l$ in (10) is given by the sum of the source model scaled by parameter λ_l and a new model is built on the new data inputs. When λ_l is 0, $\boldsymbol{w}_l$ returns to the original formulation in a traditional AK-LS-SVM model; no learned knowledge can be leveraged from the previous source model.

Combining (10), (11) with (12), we obtain

$$\sum_{i=1}^{N}\alpha_{li}\varphi(\boldsymbol{x}'_i)^T\varphi\left(\boldsymbol{x}'_j\right) + b_l + \alpha_{li}/C_l = y_i - \lambda_l \boldsymbol{w}_{l-1}^T\varphi(\boldsymbol{x}'_i). \tag{13}$$

The above equation can be written in matrix form

$$\begin{bmatrix} \widetilde{\boldsymbol{K}}_l + C_l^{-1}\mathbf{I} & \mathbf{1} \\ \mathbf{1}^T & 0 \end{bmatrix}\begin{bmatrix} \boldsymbol{\alpha}_l \\ b_l \end{bmatrix} = \begin{bmatrix} \boldsymbol{Y} - \lambda_l \widetilde{\boldsymbol{Y}}_l \\ 0 \end{bmatrix} \tag{14}$$

where $\widetilde{\boldsymbol{K}}_l = [\boldsymbol{K}(\boldsymbol{x}'_i, \boldsymbol{x}'_j)]_{N*N}$, $\boldsymbol{I}$ is a diagonal matrix with unity diagonal entries, $\boldsymbol{Y}$ is the actual labels of training samples, and $\widetilde{\boldsymbol{Y}}_l$ is the predicted labels of training samples that are obtained from the source model, i.e., $\boldsymbol{Y} = [y_1; \cdots; y_N]$, $\widetilde{\boldsymbol{Y}}_l = [y'_{l1}; \cdots; y'_{lN}] = [\boldsymbol{w}_{l-1}^T\varphi(\boldsymbol{x}'_{l1}); \cdots; \boldsymbol{w}_{l-1}^T\varphi(\boldsymbol{x}'_{lN})] = \left[\sum_{i=1}^{N}\alpha_{(l-1)i}\boldsymbol{K}(\boldsymbol{x}_{(l-1)i}, \boldsymbol{x}'_{l1}), \cdots, \sum_{i=1}^{N}\alpha_{(l-1)i}\boldsymbol{K}(\boldsymbol{x}_{(l-1)i}, \boldsymbol{x}'_{lN})\right]$. Here, $\widetilde{\boldsymbol{Y}}_l$ can be obtained in a kernel form as long as the previous model $\boldsymbol{w}_{l-1}$ and the current model $\boldsymbol{w}_l$ use the same kernel. Therefore, for the safe use of the adopted Gaussian additive kernel, δ in each model must be the same.

$\mathbf{H}_l$ represents the first matrix on the left-hand side of (14); the model parameters can be calculated simply using a matrix inversion

$$\begin{bmatrix} \boldsymbol{\alpha}_l \\ b_l \end{bmatrix} = \mathbf{Q}\begin{bmatrix} \boldsymbol{Y} - \lambda_l \widetilde{\boldsymbol{Y}}_l \\ 0 \end{bmatrix} \tag{15}$$

where $\mathbf{Q}_l = \mathbf{H}_l^{-1}$. It can be observed that λ_l, $\boldsymbol{\alpha}_l$, and b_l can be calculated accordingly from (15), and $\boldsymbol{w}_l$ and b_l from (10) and (12), respectively.

In order to find the optimal value for parameter λ_l efficiently and effectively, we propose a fast leave-one-out cross-validation method for parameter tuning.

$\mathbf{H}_l$ is decomposed into its block representation and the first row and the first column are isolated, that is

$$\mathbf{H}_l = \begin{bmatrix} \widetilde{\boldsymbol{K}}_l + C_l^{-1}\mathbf{I} & \mathbf{1} \\ \mathbf{1}^T & 0 \end{bmatrix} = \begin{bmatrix} h_{11l} & \mathbf{h}_{1l}^T \\ \mathbf{h}_{1l} & \mathbf{H}_{(-1)l} \end{bmatrix} \tag{16}$$

$\boldsymbol{\alpha}_{l(-i)}$ and $b_{l(-i)}$ represent the model parameters in the ith iteration of the leave-one-out cross validation procedure. In the first iteration, where the first training sample is excluded, we have

$$\begin{bmatrix} \boldsymbol{\alpha}_{(-1)l} \\ b_{(-1)l} \end{bmatrix} = \mathbf{Q}_{(-1)l} \begin{bmatrix} \boldsymbol{Y}_{(-1)} - \lambda_l \widetilde{\boldsymbol{Y}}_{(-1)l} \\ 0 \end{bmatrix} \tag{17}$$

where $\mathbf{Q}_{(-1)l} = \mathbf{H}_{(-1)l}^{-1}$ and $\boldsymbol{Y}_{(-1)} = (y_2, y_3, \ldots, y_N, 0)^T$. We denote the predicted label on the ith sample excluded from the training dataset by $\tilde{y}_{il}$, and the predicted label for the first training sample becomes

$$\begin{aligned} \tilde{y}_{1l} &= \mathbf{h}_{1l}^T \begin{bmatrix} \boldsymbol{\alpha}_{(-1)l} \\ b_{(-1)l} \end{bmatrix} + \lambda_l \widetilde{\boldsymbol{Y}}_l \\ &= \mathbf{h}_{1l}^T \mathbf{Q}_{(-1)l} \big(\boldsymbol{Y}_{(-1)} - \lambda_l \widetilde{\boldsymbol{Y}}_{(-1)} \big) + \lambda_l \widetilde{\boldsymbol{Y}}_l. \end{aligned} \tag{18}$$

Considering the last N equations in (14), we get $[\mathbf{h}_{1l}\, \mathbf{H}_{(-1)l}][\boldsymbol{\alpha}_l^T\ b_l]^T = (\boldsymbol{Y}_{(-1)} - \lambda_l \widetilde{\boldsymbol{Y}}_{(-1)})$, and

$$\begin{aligned} \tilde{y}_{1l} &= \mathbf{h}_{1l}^T \mathbf{Q}_{(-1)l} \big[\mathbf{h}_{1l}\, \mathbf{H}_{(-1)l} \big] [\alpha_{1l}, \ldots, \alpha_{Nl}, b_l]^T + \lambda_l \widetilde{\boldsymbol{Y}}_{(-1)} \\ &= \mathbf{h}_1^T \mathbf{Q}_{l(-1)} \mathbf{h}_1 \alpha_1 + \mathbf{h}_1^T [\alpha_{2l}, \ldots, \alpha_{Nl}, b_l]^T + \lambda_l \widetilde{\boldsymbol{Y}}_{(-1)}. \end{aligned} \tag{19}$$

From (14), the first equation of the system is $y_1 - \lambda_l \boldsymbol{y}_{1(-1)l} = h_{11l}\alpha_{1l} + \mathbf{hl}_1^T[\alpha_{2l}, \alpha_{3l}, \ldots, \alpha_{Nl}, b_l]^T$, and hence $\tilde{y}_1 = y_1 - \alpha_{1l}(h_{11l} - \mathbf{h}_{1l}^T \mathbf{Q}_{(-1)l} \mathbf{h}_{1l})$. Finally, by using $\mathbf{Q}_l = \mathbf{H}_{(-1)l}$ and the block matrix inversion lemma we can obtain

$$\mathbf{Q}_l = \begin{bmatrix} v^{-1} & -v^{-1}\mathbf{h}_{1l}\mathbf{Q}_{(-1)l} \\ \mathbf{Q}_{(-1)l} + v^{-1}\mathbf{Q}_{(-1)l}\mathbf{h}_{1l}^T\mathbf{h}_{1l}\mathbf{Q}_{(-1)l} & -v^{-1}\mathbf{Q}_{(-1)l}\mathbf{h}_{1l}^T \end{bmatrix} \tag{20}$$

where $v = h_{11l} - \mathbf{h}_{1l}^T \mathbf{Q}_{(-1)l} \mathbf{h}_{1l}$. Since the system of linear equations in (14) is insensitive to permutations of the ordering of the equations, then

$$\tilde{y}_{il} = y_i - \alpha_{il} / \mathbf{Q}_{iil} \tag{21}$$

By defining $[\boldsymbol{\alpha}_l'^T, b_l']^T = \mathbf{Q}_l[\mathbf{y}^T, 0]$, $[\boldsymbol{\alpha}_l''^T, b_l''] = \mathbf{Q}_l[\boldsymbol{Y}^T, 0]$, and $\boldsymbol{\alpha}_l = \boldsymbol{\alpha}_l' - \lambda_l \boldsymbol{\alpha}_l''$, then we can obtain

$$\tilde{y}_{il} = y_i - \frac{\alpha_{il}'}{\mathbf{Q}_{iil}} + \frac{\lambda_l \alpha_{il}''}{\mathbf{Q}_{iil}}. \tag{22}$$

It can be seen from (22) that $\boldsymbol{\alpha}_l$ and λ_l have a linear relationship, which means that after determining λ_l, the learning model can be obtained as well. The optimal λ_l is supposed to keep the same sign of $\tilde{y}_{il}$ and y_i for all training samples. However, it might cause many local minima issues due to nonconvex formulation. Thus, in the end the following loss function is adopted, which is similar to the hinge loss:

$$l(\tilde{y}_{il}, y_i) = |1 - \tilde{y}_{il} y_i|_+ = \left| y_i \frac{\alpha_{il}' - \lambda_l \alpha_{il}''}{\mathbf{Q}_{iil}} \right|_+ \tag{23}$$

where $|x|_+ = \max\{0, x\}$. This is a convex upper bound to the leave-one-out misclassification loss, and it prefers solutions in which $\tilde{y}_i$ has an absolute value equal to or larger than 1 and the same sign as y_i. Finally, the objective function is

$$\begin{aligned} &\sum_{i=i}^{N} l(\tilde{y}_{il}, y_i) \\ &\text{s.t. } 0 \le \lambda_l \le D \end{aligned} \tag{24}$$

where D is a constant. A regularization based on this can induce numerical stability. This optimization process can be implemented by a projected subgradient descent algorithm, and the pseudocode in given in Algorithm 1. After obtaining λ_l, $\boldsymbol{\alpha}_l$, and b_l can be calculated accordingly from (15), and then $\boldsymbol{w}_l$ and b_l from (10), (11), and (12).

As a result, we can obtain the decision function $f_l(\boldsymbol{x}_{il}') = \boldsymbol{w}_l^T \varphi(\boldsymbol{x}_{il}') + b_l$ on L_l ($l \ge 2$), and the predicted label vector $\boldsymbol{F}_l = (f_l(\boldsymbol{x}_{1l}'), f_l(\boldsymbol{x}_{2l}'), \ldots, f_l(\boldsymbol{x}_{Nl}'))$ for $\boldsymbol{X}_l'$. The processing layer continues to be added until the prediction accuracy performance has no improvement or the improvement is negligible, (i.e., $||\boldsymbol{F}_l - \boldsymbol{F}_{l-1}||_F^2 < \epsilon$). Here, the complete learning algorithm of the proposed classifier DTA-LS-SVM is given in Algorithm 2 that outputs the final decision function. Please note that in this paper we select the parameter C_l from comparatively big intervals, i.e., $C_l \in \{1, 10, 50, 100, 150, 200, 250, 500\}$, to guarantee diversities between the modules from adjacent layers. However, C_l can also be selected from different intervals depending upon the piratical situations.

In DTA-LS-SVM, transfer learning is embedded from the second layer. Referring to (10), we can observe that the classification on the lth ($l >= 2$) layer is in fact achieved in a way like a combination of multiple kernel functions from different layers. According to the excellent generalization performances of multikernel classifiers [32]–[34], the module from the second layer in the proposed approach is supposed to obtain a better generalization performance than the previous one. In this sense, the learning process under such a stacked architecture tends to be greedy. Our experiments show that normally $L = 3, 4$, or 5 is an appropriate reference for small or medium datasets. If L is too big, it might lead to overfitting issues.

The proposed classifier DTA-LS-SVM can be extended to explain the multiclassification tasks. The one-against-all strategy may be used to find the multiple decision functions that separate one class from the remaining classes. In the end, the predicted label of the new input data sample $\boldsymbol{x}_i$ is determined by $\max_{k=1,\ldots,M} y_k(\boldsymbol{x}_i)$, where M denotes the number of the classes.

C. Computational Complexity

One highlight of the proposed classifier DTA-LS-SVM is its fast computational ability of the leave-one-out cross validation in the deep architecture. Its computational cost can be represented as $O(N^3 + (L-1)(N^3 + N))$. First it includes the traditional AK-LS-SVM model construction on the first layer. Therefore, the complexity of this part is $O(N^3)$. From the layer ($l \ge 2$), the computational cost of each module consists of two parts. The first part includes the calculation of the matrix $\mathbf{Q}_l$ by the inverse related to the training set on the L_l, in which the corresponding computational complexity is $O(N^3)$.

Algorithm 1 Learning Algorithm of Transfer Additive LS-SVM

Input: $\boldsymbol{w}_{l-1}$, $\boldsymbol{X}'_l$, $\boldsymbol{Y}$, C_l and kernel width σ

Output: λ_l

Procedure

Step 1: Calculate $\boldsymbol{Q}_l$ by using $\boldsymbol{Q}_l = \mathbf{H}_{(-1)l}$ and Eq. (17).
Step 2: Calculate $\boldsymbol{\alpha}'_l$, $\boldsymbol{\alpha}''_l$
Step 3: $t = 1$
Step 4: Repeat
$\tilde{y}_{il} = y_i - \frac{\alpha'_{li}}{Q_{iil}} + \frac{\lambda_l \alpha''_{il}}{Q_{iil}}$, $i = 1, 2, \ldots, N$
$d_i \leftarrow \mathbf{1}\{\tilde{y}_{il} y_i > 0\}$, $i = 1, 2, \ldots, N$
$\lambda_l \leftarrow \lambda_l - \frac{1}{\sqrt{t}} d_i y_i \frac{\alpha''_{il}}{Q_{iil}}$
If $\lambda_l > D$ then $\lambda_l \leftarrow D$
End if
$\lambda_l \leftarrow max(\lambda_l, 0)$
$t \leftarrow t + 1$
Step 5: Until convergence
Step 6: Output λ_l

Algorithm 2 Learning Algorithm of DTA-LS-SVM

Input: training set $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N]$, $\boldsymbol{x}_i \in \boldsymbol{R}^d$, output set $\boldsymbol{Y} = [y_1, y_2, \cdots, y_N]$, $y_i \in \{+1, -1\}$ for binary classification, kernel width δ, number of layers L, $l = 1$

Output: The stacked structure of DTA-LS-SVM with tuned parameter values

Procedure

Step 1:
1.1 Choose the regularization parameter C_1 randomly from seven values, i.e., $C_1 \in \{1, 10, 50, 100, 150, 200, 250, 500\}$.
1.2 Construct the 1st module using the additive kernel LS-SVM shown in Eq. (4) and obtain $\boldsymbol{w}_1$, b_1 and the predicted labels $\boldsymbol{F_1}$ $(f_1(\boldsymbol{x}_{11}), f_1(\boldsymbol{x}_{21}), \cdots, f_1(\boldsymbol{x}_{N1}))$.
Step 2: $l = l + 1$
Step 3: For $l = 2 : L$ do
3.1 $\boldsymbol{X}'_l = \boldsymbol{X} \oplus \boldsymbol{F}_{l-1}$
3.2 Choose the regularization parameter C_l randomly from seven values, i.e., $C_l \in \{1, 10, 50, 100, 150, 200, 250, 500\}$.
3.3 Construct the lth module by invoking Algorithm 1 on $\boldsymbol{X}'_l$ and obtain λ_l.
3.4 Calculate $\boldsymbol{w}_l$ using Eq. (12), Eq. (13), Eq. (15) and the predicted labels $\boldsymbol{F}_l$ $(f_l(\boldsymbol{x}_{1l}), f_l(\boldsymbol{x}_{2l}), \cdots, f_l(\boldsymbol{x}_{Nl}))$.
Step 4: Calculate $\triangle_F = ||\boldsymbol{F}_l - \boldsymbol{F}_{l-1}||_F^2$
Step 5: If $\triangle_F \leq \epsilon$ (a given threshold)
End else
Step 6: $l = l + 1$
Step 7: Output the stacked structure of the proposed classifier DTA-LS-SVM with tuned parameter values and the decision function in the L-th module as the final decision function.

The second part includes the computational complexity of each iteration in Algorithm 1 to optimize (24), which can be represented as $O(N)$. Therefore, the entire computational cost of DTA-LS-SVM becomes $O(N^3 + (L-1)(N^3 + N)) = O(LN^3 + (L-1)N)$.

Let us consider the traditional leave-one-out cross-validation strategy on SVM. Theoretically, it takes $O(N^3)$ to train an SVM. By using specific speed-up strategies [35], the training time can be accelerated to $O(N) - O(N^{2.3})$ such that the leave-one-out cross validation time on SVM becomes $O(N * N) - O(N * N^{2.3}) = O(N^2) - O(N^{3.3})$. Considering the grid search for generalization parameter C (s_1 grid values) and kernel width σ (s_2 grid values), the computational complexity of SVM becomes $s_1 s_2 O(N^2) - s_1 s_2 O(N^{3.3})$. In general, s_1 and s_2 are normally greater than 3 in the experiments, and the number L of layers in DTA-LS-SVM is small ($3 \leq L \leq 5$). Therefore, although it seems that the computational complexity of SVM may be less than that of DTA-LS-SVM, our experiments reveal that the actual running time of SVM with grid search actually is much longer than that of DTA-LS-SVM.

In terms of LS-SVM, the computational complexity to train an LS-SVM is $O(N^3)$ due to the calculation of the matrix $\mathbf{Q}$ by the inverse of $\mathbf{H}$. Therefore, the computational complexity of leave-one-out cross validation on LS-SVM becomes $s_1 s_2 O(N * N^3) = s_1 s_2 O(N^4)$. Moreover, LS-SVM could use the fast leave-one-out cross validation strategy discussed in Section III-B. Referring to (22) and (23) with λ_l equal to 0, the computational complexity of LS-SVM could be accelerated into $s_1 s_2 O(N^3 + N)$. In summary, the proposed classifier DTA-LS-SVM has a superior advantage in the running speed, compared with SVM and LS-SVM.

D. Extension on Imbalanced Datasets

The proposed classifier DTA-LS-SVM is based on the AK-LS-SVM framework. However, it is often the case in many real-world scenarios that the datasets are imbalanced. This is especially true when dealing with datasets involving medical problems. For example, a dataset will be imbalanced due to far more cases of a diagnosis of malignancy within cancer cases compared to those that are benign or cancer free. We found that DTA-LS-SVM can be naturally extended to its cost-sensitive or imbalanced version such that our method is also suitable for imbalanced datasets.

The problem with imbalanced datasets is that the decision boundary tends to get too close to the positive class (i.e., minority class). Therefore, the decision boundary needs to be pushed away from positive instances. One solution is to give different error costs to the positive and negative classes [36]. There are several ways to achieve this goal. However, in this paper we only change the objective functions of AK-LS-SVM into (25) for an imbalanced dataset

$$\min_{\boldsymbol{w},b} \; J(\boldsymbol{w}, b) = \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{C}{2}\left[\frac{N^-}{N}\sum_{i=1}^{N^+}\xi_i^2 + \frac{N^+}{N}\sum_{i=N^++1}^{N}\xi_i^2\right]$$
$$\text{s.t.} \;\; y_i = \boldsymbol{w}^T\varphi(\boldsymbol{x}_i) + b + \xi_i \tag{25}$$

where N^+ and N^- represents the numbers of samples in the positive and negative classes, respectively, in the N samples, i.e., when (25) is applied to AK-LS-SVM, we term DTA-LS-SVM as iDTA-LS-SVM for an imbalanced dataset.

Similar to the mathematical derivations from (9) to (15), we easily find that only $\mathbf{H}_l$ in the first matrix in (14) needs to be changed into the following representation:

$$\mathbf{H}_l = \begin{bmatrix} \widetilde{\boldsymbol{K}}_l + C_l^{-1}\mathbf{E} & \mathbf{1} \\ \mathbf{1}^T & 0 \end{bmatrix} \tag{26}$$

where

$$\mathbf{E} = \begin{pmatrix} \frac{N^-}{N} & 0 & 0 & 0 & 0 & 0 \\ 0 & \ddots & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{N^-}{N} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{N^+}{N} & 0 & 0 \\ 0 & \ddots & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{N^+}{N} \end{pmatrix} \tag{27}$$

and the remaining derivations remain the same such that iDTA-LS-SVM can be applied on imbalanced datasets. By comparing $\mathbf{H}_l$ in (26) with the first matrix in (14), we can find that only the difference is between $\mathbf{E}$ in (26) and $\mathbf{I}$ in (14). Only under the condition that N^- equals N^+, $\mathbf{E}$ degenerates into $\mathbf{I}$ such that iDTA-LS-SVM can handle balanced datasets as well. In addition, it is obvious that iDTA-LS-SVM has the same computational complexity of leave-one-out cross validation as that for DTA-LS-SVM.

IV. Experiments and Results

In the experiments, the proposed classifiers DTA-LS-SVM and iDTA-LS-SVM were evaluated on balanced and imbalanced UCI datasets and one real world dataset. In order to make the comparison fair, their classification performances are compared with those using the traditional LS-SVM and SVM with additive kernels. During the data preparation process, all values in the datasets were normalized. For any original UCI datasets with missing data, the records with missing values were first removed and then the processed datasets were used in the experiments. To evaluate methods on balanced datasets, the accuracy and F-score were used as metrics. On imbalanced datasets, only the F-score was used as a metric. This considers for example, that with an imbalance ratio 99:1, a method labeling each data input with a positive class is 99% accurate, but it is useless as a classifier to find out the negative class. Therefore, metrics precision and recall were adopted for imbalanced datasets. Precision is defined as the number of correct positive results divided by the number of all positive results, while recall is defined as the number of correct positive results divided by the number of positive results that should have been found. The F-score is a weighted average of the precision and recall

$$\text{Precision} = \frac{\text{true positive}}{\text{true positive} + \text{false positive}} \tag{28}$$

$$\text{Recall} = \frac{\text{true positive}}{\text{true positive} + \text{false negative}} \tag{29}$$

TABLE I
UCI Datasets Description

Type	Dataset	Sample size	Feature	Class(%)
Imbalanced	*breast cancer*	683	9	65.52 34.48
	Pima Indians	768	8	65.02 34.98
	Indians Liver	579	10	71.50 28.50
	Australian	690	14	44.50 55.50
Balanced	*diabetic*	1151	19	53.08 46.92
	credit approval	653	15	45.33 54.67
	mammographic	830	5	48.55 51.45

$$\text{F-score} = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}}. \tag{30}$$

The F-score was used to measure the performances on imbalanced datasets in Lin *et al.* [37], Zou *et al.* [38], and Tan [39]. In this paper, this metric was also used to evaluate iDTA-LS-SVM and comparative methods. In the experiments, each dataset was randomly split into the corresponding training and testing sets in the ratio 7:3. This process was repeated ten times such that every sample from the dataset might have a chance to be used in the training and testing sets. In the end, the mean and standard deviation of metrics(s) on training and testing sets were calculated. All the experiments were implemented using 64-bit MATLAB on a computer with an Intel Core i5-6300 2.40 GHz CPU and 8.00 GB RAM.

A. UCI Public Datasets

In this section, DTA-LS-SVM and iDTA-LS-SVM with the traditional LS-SVM and SVM using additive kernels are compared. Seven public UCI datasets were adopted in which three were imbalanced and four were balanced. These UCI datasets are summarized in Table I.

In the experiments, different additive kernels were tried to find the most suitable one for DTA-LS-SVM, iDTA-LS-SVM and the comparative methods on each dataset. Here, only the experimental results using the selected kernel (Gaussian additive kernel) are displayed. For DTA-LS-SVM and iDTA-LS-SVM, the number of modules is set to 3 or 4 due to the small or medium sample size of the adopted datasets. ϵ is set to 0.1. δ is set to be the average value of the standard deviations for all respective features. For the comparative methods, the grid search algorithm with ten-fold cross-validation was used to find the optimal values for parameters C and δ which could give the best performances. The values of $\{1, 10, 50, 100, 150, 200, 250, 500\}$ and $\{0.1, 1, 5, 10, 20, 50, 100, 150, 200\}$ were searched for C and δ, respectively, in the experiments. Here, only the performances using the optimal parameters are displayed.

Table II shows the experimental results of DTA-LS-SVM and the comparative methods on imbalanced datasets. Table III

TABLE II
PERFORMANCE RESULTS ON BALANCED UCI DATASETS

Datasets	Metrics	Performances					
		DTA-LS-SVM		LS-SVM		SVM	
		training	testing	training	testing	training	testing
Australian	Accuracy	0.8936±0.0099	**0.8678±0.0212**	0.8583±0.0098	0.8500±0.0266	0.8589±0.0075	0.8572±0.0174
	F1-score	0.9050±0.0091	**0.8766±0.0215**	0.8665±0.0060	0.8630±0.0193	0.8620±0.0082	0.8619±0.0190
diabetic	Accuracy	0.7720±0.0096	**0.7395±0.0197**	0.7391±0.0221	0.7220±0.0242	0.7275±0.0068	0.7188±0.0194
	F1-score	0.7668±0.0163	**0.7368±0.0202**	0.7434±0.0380	0.7262±0.0281	0.7262±0.0112	0.7153±0.0194
credit approval	Accuracy	0.8985±0.0075	**0.8786±0.0161**	0.8611±0.0138	0.8536±0.0203	0.8678±0.0087	0.8648±0.0230
	F1-score	0.9065±0.0095	**0.8895±0.0154**	0.8683±0.0097	0.8632±0.0205	0.8684±0.0079	0.8682±0.0192
mammographic	Accuracy	0.8231±0.0078	**0.8273±0.0233**	0.8155±0.0103	0.8121±0.0313	0.8189±0.0116	0.8104±0.0349
	F1-score	0.8227±0.0102	**0.8321±0.0266**	0.8172±0.0173	0.8167±0.0357	0.8181±0.0120	0.8137±0.0350

TABLE III
PERFORMANCE RESULTS ON IMBALANCED UCI DATASETS

Datasets	Performances					
	iDTA-LS-SVM		LS-SVM		SVM	
	training	testing	training	testing	training	testing
breast cancer	0.9861±0.0064	**0.9801±0.0026**	0.9760±0.0074	0.9738±0.0069	0.9806±0.0041	0.9728±0.0123
Pima Indians	0.8748±0.0167	**0.8359±0.0097**	0.8401±0.0131	0.8284±0.0317	0.8336±0.0084	0.8285±0.0248
Indians liver	0.8438±0.0829	**0.8403±0.0083**	0.8390±0.0166	0.7986±0.0580	0.8343±0.0080	0.8323±0.0188

TABLE IV
PERFORMANCE RESULTS ON THE *Australian* DATASET WITH DIFFERENT RATIOS OF TRAINING AND TESTING DATA

Australia	Metrics	Performances							
		6:4		5:5		4:6		3:7	
		training	testing	training	testing	training	testing	training	testing
DTA-LS-SVM	Accuracy	0.9086±0.0087	**0.8572±0.0111**	0.9009±0.0148	**0.8577±0.0110**	0.9047±0.0215	**0.8589±0.0141**	0.9043±0.0175	**0.8551±0.0149**
	F1-score	0.9081±0.0083	**0.8716±0.0125**	0.9123±0.0121	**0.8727±0.0098**	0.9124±0.0199	**0.8742±0.0131**	0.9132±0.0171	**0.8703±0.0132**
LS-SVM	Accuracy	0.8425±0.0384	0.8417±0.0431	0.8417±0.0598	0.8304±0.0692	0.8413±0.0544	0.8283±0.0559	0.8449±0.0695	0.8265±0.0762
	F1-score	0.8586±0.0200	0.8595±0.0281	0.8631±0.0307	0.8546±0.0396	0.8600±0.0272	0.8519±0.0325	0.8651±0.0337	0.8496±0.0418
SVM	Accuracy	0.8621±0.0148	0.8504±0.0221	0.8649±0.0143	0.8487±0.0139	0.8717±0.0202	0.8498±0.0176	0.8797±0.0251	0.8478±0.0161
	F1-score	0.8663±0.0139	0.8564±0.0194	0.8716±0.0150	0.8529±0.0126	0.8788±0.0176	0.8563±0.0164	0.8871±0.0242	0.8556±0.0155

TABLE V
PERFORMANCE RESULTS ON THE *Diabetic* DATASET WITH DIFFERENT RATIOS OF TRAINING AND TESTING DATA

Diabetic	Metrics	Performances							
		6:4		5:5		4:6		3:7	
		training	testing	training	testing	training	testing	training	testing
DTA-LS-SVM	Accuracy	0.7990±0.0082	**0.7386±0.0139**	0.8028±0.0125	**0.7241±0.0164**	0.8052±0.0121	**0.7159±0.0132**	0.8183±0.0150	**0.7076±0.0129**
	F1-score	0.7974±0.0090	**0.7331±0.0117**	0.7986±0.0154	**0.7127±0.0197**	0.8084±0.0186	**0.7071±0.0132**	0.8115±0.0240	**0.6965±0.0162**
LS-SVM	Accuracy	0.7445±0.0216	0.7180±0.0208	0.7431±0.0166	0.7163±0.0245	0.7315±0.0286	0.6991±0.0237	0.7209±0.0143	0.6815±0.0237
	F1-score	0.7451±0.0333	0.7270±0.0250	0.7377±0.0409	0.7096±0.0338	0.7355±0.0402	0.7021±0.0333	0.7268±0.0482	0.6770±0.0412
SVM	Accuracy	0.7330±0.0134	0.7015±0.0199	0.7297±0.0136	0.7038±0.0145	0.7193±0.0100	0.6987±0.0151	0.7075±0.0216	0.6875±0.0210
	F1-score	0.7350±0.0219	0.7038±0.0147	0.7262±0.0226	0.6936±0.0276	0.7339±0.0154	0.6965±0.0095	0.7004±0.0355	0.6733±0.0432

shows the experimental results of iDTA-LS-SVM and the comparative methods on balanced datasets. During the training process of both DTA-LS-SVM and iDTA-LS-SVM classifiers, the first module (i.e., the first layer) was constructed to obtain the training accuracy after comparing the predicted and actual labels. If the training accuracy of the adjacent higher layer was improving continuously, one more layer was added and training continued. If the training accuracy of the adjacent layer remained the same or even dropped, the process was stopped. To elaborate, one complete training process on the *Australian* dataset is described. For this dataset, the training (testing) accuracy is 0.8589 (0.8173), respectively, after the first module, and 0.8734 (0.8365), respectively, after the second module (i.e., the second layer). Since the accuracies are increasing, training of the third and forth module continued, and 0.8880 (0.8702) accuracy for the third module and 0.8734 (0.8365) accuracy for the fourth module was obtained. After the third module, a decrease in the accuracy can be seen. Therefore, the accuracy performances on the third module are used, which indeed outperforms the comparative methods. From the performance results shown in Table II, DTA-LS-SVM obtains considerably higher testing accuracies than those using the comparative methods on the adopted balanced UCI datasets. In terms of the experimental results on the imbalanced datasets displayed in Table III, iDTA-LS-SVM maintains the advantage on the F-scores compared to other methods. From these two tables, the performance increases from DTA-LS-SVM and iDTA-LS-SVM are not dramatic. However, compared to LS-SVM, it becomes consistently better. In some cases, LS-SVM performs worse than SVM, but DTA-LS-SVM and iDTA-LS-SVM always perform better. Overall, the proposed classifiers DT-AK-LS-SVM and iDTA-LS-SVM exhibited good generalization performances on both balanced and imbalanced datasets.

Moreover, to observe the behavior of the proposed classifiers, we divided the training and testing sets at different ratios on each UCI dataset to evaluate their classification performances. The training and testing sets ratios were set to 6:4,

TABLE VI
PERFORMANCE RESULTS ON THE *credit approval* DATASET WITH DIFFERENT RATIOS OF TRAINING AND TESTING DATA

credit approval	Metrics	Performances							
		6:4		5:5		4:6		3:7	
		training	testing	training	testing	training	testing	training	testing
DTA-LS-SVM	Accuracy	0.8985±0.0071	**0.8664±0.0115**	0.9110±0.0100	**0.8621±0.0116**	0.9031±0.0119	**0.8615±0.0112**	0.9190±0.0149	**0.8614±0.0126**
	F1-score	0.9074±0.0075	**0.8773±0.0122**	0.9206±0.0107	**0.8708±0.0124**	0.9120±0.0107	**0.8720±0.0097**	0.9270±0.0134	**0.8687±0.0168**
LS-SVM	Accuracy	0.8509±0.0305	0.8481±0.0556	0.8426±0.0763	0.8202±0.0890	0.8295±0.0798	0.8120±0.0997	0.8508±0.0649	0.8328±0.0630
	F1-score	0.8623±0.0182	0.8608±0.0406	0.8614±0.0382	0.8411±0.0506	0.8609±0.0396	0.8405±0.0556	0.8673±0.0363	0.8560±0.0423
SVM	Accuracy	0.8681±0.0050	0.8622±0.0127	0.8696±0.0164	0.8612±0.0169	0.8778±0.0149	0.8564±0.0108	0.8677±0.0164	0.8537±0.0193
	F1-score	0.8695±0.0059	0.8628±0.0149	0.8716±0.0159	0.8625±0.0167	0.8802±0.0167	0.8604±0.0117	0.8754±0.0172	0.8600±0.0132

TABLE VII
PERFORMANCE RESULTS ON THE *mammographic* DATASET WITH DIFFERENT RATIOS OF TRAINING AND TESTING DATA

mammographic	Metrics	Performances							
		6:4		5:5		4:6		3:7	
		training	testing	training	testing	training	testing	training	testing
DTA-LS-SVM	Accuracy	0.8325±0.0045	**0.8187±0.0171**	0.8234±0.0120	**0.8224±0.0118**	0.8241±0.0173	**0.8106±0.0138**	0.8261±0.0240	**0.8095±0.0165**
	F1-score	0.8317±0.0078	**0.8093±0.0149**	0.8214±0.0130	**0.8234±0.0132**	0.8216±0.0264	**0.8091±0.0213**	0.8275±0.0267	**0.8050±0.0171**
LS-SVM	Accuracy	0.8213±0.0137	0.8051±0.0173	0.8212±0.0086	0.8039±0.0181	0.8250±.0180	0.8036±0.0186	0.8309±0.0172	0.8024±0.0149
	F1-score	0.8202±0.0166	0.8017±0.0201	0.8151±0.0081	0.7988±0.0183	0.8240±0.0202	0.8006±0.0178	0.8326±0.0209	0.7975±0.0153
SVM	Accuracy	0.8139±0.0125	0.8060±0.0284	0.8089±0.0124	0.8000±0.0153	0.8108±0.0197	0.7972±0.0215	0.8092±0.0277	0.7914±0.0153
	F1-score	0.8165±0.0119	0.8079±0.0264	0.8135±0.0138	0.7978±0.0159	0.8110±0.0187	0.8008±0.0189	0.8052±0.0296	0.7867±0.0234

TABLE VIII
PERFORMANCE RESULTS ON THE *breast cancer* DATASET WITH DIFFERENT RATIOS OF TRAINING AND TESTING DATA

breast cancer	F-score							
	6:4		5:5		4:6		3:7	
	training	testing	training	testing	training	testing	training	testing
iDTA-LS-SVM	0.9818±0.0038	**0.9798±0.0053**	0.9821±0.0040	**0.9787±0.0044**	0.9859±0.0065	**0.9796±0.0031**	0.9874±0.0043	**0.9775±0.0053**
LS-SVM	0.9735±0.0051	0.9742±0.0115	0.9684±0.0183	0.9649±0.0236	0.9750±0.0044	0.9675±0.0111	0.9745±0.0093	0.9711±0.0057
SVM	0.9811±0.0042	0.9711±0.0063	0.9797±0.0081	0.9749±0.0064	0.9821±0.0042	0.9726±0.0083	0.9854±0.0088	0.9718±0.0043

TABLE IX
PERFORMANCE RESULTS ON THE *Pima Indians* DATASET WITH DIFFERENT RATIOS OF TRAINING AND TESTING DATA

Pima Indians	F-score							
	6:4		5:5		4:6		3:7	
	training	testing	training	testing	training	testing	training	testing
iDTA-LS-SVM	0.8665±0.0157	**0.8345±0.0171**	0.8697±0.0088	**0.8345±0.0066**	0.8700±0.0129	**0.8336±0.0121**	0.8918±0.0188	**0.8254±0.0088**
LS-SVM	0.8399±0.0141	0.8265±0.0239	0.8409±0.0179	0.8295±0.0205	0.8360±0.0252	0.8263±0.0189	0.8385±0.0280	0.8222±0.0178
SVM	0.8379±0.0104	0.8185±0.0138	0.8417±0.0175	0.8198±0.0161	0.8426±0.0150	0.8174±0.0075	0.8478±0.0134	0.8141±0.0135

TABLE X
PERFORMANCE RESULTS ON THE *Indians liver* DATASET WITH DIFFERENT RATIOS OF TRAINING AND TESTING DATA

Indians liver	F-score							
	6:4		5:5		4:6		3:7	
	training	testing	training	testing	training	testing	training	testing
iDTA-LS-SVM	0.8387±0.0104	**0.8375±0.0135**	0.8422±0.0085	**0.8373±0.0078**	0.8382±0.0113	**0.8336±0.0096**	0.8353±0.0177	**0.8294±0.0177**
LS-SVM	0.9197±0.0583	0.8285±0.0153	0.8714±0.0537	0.8117±0.0212	0.8850±0.0779	0.7995±0.0542	0.9203±0.0760	0.7813±0.0542
SVM	0.8369±0.0106	0.8294±0.0160	0.8420±0.0133	0.8253±0.0133	0.8410±0.0175	0.8288±0.0117	0.8363±0.0203	0.8226±0.0088

5:5, 4:6, and 3:7 on each UCI datasets. Tables IV–VII display the experimental results of DTA-LS-SVM and the comparative methods on all the balanced datasets in terms of accuracy and F score. Tables VIII–X contain the experimental results of iDTA-LS-SVM and the comparative methods for all the imbalanced datasets, in terms of F-scores. On each table, the best results have been highlighted in bold. In terms of training accuracy and F-score, when changing the ratios, DTA-LS-SVM and iDTA-LS-SVM do not exhibit an increasing trend as seen in LS-SVM and SVM when the size of the training set increases. The reason might be that DTA-LS-SVM and iDTA-LS-SVM essentially are multiple kernel combination methods, which have a stronger representation capability for smaller training datasets. Importantly, it is easily seen that after changing the ratios of training and testing sets, the proposed classifiers DTA-LS-SVM and iDTA-LS-SVM still outperform LS-SVM and SVM on the testing accuracies of all seven datasets. This can be seen, for example, in Table V for the *mammographic* dataset at the 5:5 ratio; DTA-LS-SVM not only performs the best in terms of overall accuracy but also has the highest F score on the testing sets. Another example is for the imbalanced *Pima Indians* dataset in Table IX in which iDTA-LS-SVM maintains an advantage over the comparative methods in terms of the F-score on the testing set. Overall, the experimental results demonstrate that DTA-LS-SVM and iDTA-LS-SVM are tolerant to the change of training sample size and is a favorable choice for balanced and imbalanced datasets in terms of the generalization capability.

In terms of the average training and testing time, Table XI in the 7:3 ratio case shows that DTA-LS-SVM and iDTA-LS-SVM take much less training time compared to the other methods. As explained in Section III-C, the running time of

TABLE XI
TRAINING AND TESTING TIME (SECONDS) ON SEVEN UCI DATASETS

Type	Datasets	Running time					
		(i)DTA-LS-SVM		LS-SVM		SVM	
		training	testing	training	testing	training	testing
Imbalanced	*breast cancer*	2.278	1.979	8776.1	1.399	19553	2.498
	Pima Indians	2.731	2.376	10512	1.447	28950	3.018
	Indians liver	1.886	1.679	6488.1	1.249	13304	2.087
	australian	3.472	2.935	12103	1.835	22059	3.209
Balanced	*diabetic*	12.698	10.600	73863	6.568	129790	12.943
	credit approval	3.638	2.782	11660	2.203	18138	3.190
	mammographic	2.279	1.933	8340.3	1.034	29387	3.212

TABLE XII
PERFORMANCE RESULTS ON THE *MIHC* DATASET USING iDTA-LS-SVM

MIHC	Performances			
	iDTA-LS-SVM			
	Part (a)		Part (b)	
	training	testing	training	testing
Accuracy	0.7261±0.0094	0.7331±0.0218	0.7210±0.0102	**0.7425±0.0266**
F1-score	0.8413±0.0063	0.8492±0.0147	0.8358±0.0100	**0.8553±0.0233**

DTA-LS-SVM and iDTA-LS-SVM is the summation of the running time in each module plus the running time for the parameter tuning of λ_l $(l = 1, 2, \ldots, L)$ using a fast leave-one-out cross validation strategy. The regularization parameter C_l $(l = 1, 2, \ldots, L)$ in each module and the kernel width δ can be randomly chosen instead of using model selection strategy, which also significantly reduces the running time. Conversely, LS-SVM and SVM take much more time to produce the optimal values for C and δ by searching from 8 and 9 grid values, respectively, which is much more computationally expensive. After this comparison, DTA-LS-SVM and iDTA-LS-SVM exhibited superior advantages in running times compared to other methods.

B. Real-World Dataset: the Community Healthcare Dataset

In this section, a real community healthcare dataset was used in the experiments to investigate performances of the proposed classifiers DTA-LS-SVM and iDTA-LS-SVM.

This dataset was collected in the nurse-led PolyU-Henry G. Leong Mobile Integrative Health Care Centre (MIHC) [40] in August 2013. It contains 444 patient records with 33 features, such as demographic, socio-economic, social relationship, and social participation data. Additionally, information on the patients' health history, such as smoking and drinking habits, chronic illnesses, and data from a series of health assessments with descriptions was included. Due to the nature of both the tests performed on the mobile clinic and the patients' themselves, some values in the dataset were missing. For example, language barriers affect the communications between nurses and patients. The missing values in the dataset were filled in by using the K-nearest neighbor (KNN) imputation method [41] with their corresponding values from the nearest-neighbor columns using the corresponding Euclidean distance.

The label information is the overall quality of life (QOL) score on a 1–5 scale of 444 patients obtained using the World Heath Organization questionnaire on QOL: from the Hong Kong short version framework [42], [43]. After analysis, the proportion of the two constructed classes ("poor" and "good") was 122 and 322, respectively. We aim to construct a classifier to predict the QOL of elderly patients (poor or good) using these 33 features from this *MIHC* dataset.

Since the two classes were originally imbalanced, iDTA-LS-SVM was used on the datasets, and its classification performance was compared with those using DTA-LS-SVM, LS-SVM, and SVM. In addition, for iDTA-LS-SVM and DTA-LS-SVM, experiments were divided into two parts for more detailed comparisons.

1) *Part (a):* In layer 1 of DTA-LS-SVM and iDTA-LS-SVM, the complete portion of the *MIHC* dataset, which had no missing data was used to train an AK-LS-SVM model. From layer 2, the whole dataset after imputation and its corresponding predictions from the previous model were used as the new data input.
2) *Part (b):* In layer 1 of DTA-LS-SVM and iDTA-LS-SVM, the whole *MIHC* dataset was used after imputation as the input.

The number of processing layers in both DTA-LS-SVM and iDTA-LS-SVM was set to 3 in these experiments. The ratio of training and testing data sets was set to 7:3. The classification accuracies and the running time of DTA-LS-SVM, iDTA-LS-SVM and the comparative methods on the training and testing datasets are listed in Tables XII–XIV. From the experimental results, iDTA-LS-SVM in *Part* (*b*) has achieved the best testing accuracy (0.7425) and an F-score of 0.8553 compared to DTA-LS-SVM, LS-SVM and SVM on the *MIHC* dataset. In *Part* (*a*), it achieved an accuracy of 0.7331 and an F-score of 0.8492, which are still higher than other methods. It shows that iDTA-LS-SVM is more suitable for classification on imbalanced datasets. The traditional LS-SVM had the worst accuracy and F-score on the testing dataset, iDTA-LS-SVM based on the combination of several AK-LS-SVM modules achieved the best accuracy. We believe that this performance improvement has arisen from

TABLE XIII
PERFORMANCE RESULTS ON THE *MIHC* DATASET USING DTA-LS-SVM AND THE OTHER COMPARATIVE METHODS

MIHC	Performances							
	DTA-LS-SVM				LS-SVM		SVM	
	Part (a)		Part (b)					
	training	testing	training	testing	training	testing	training	testing
Accuracy	0.7203±0.0133	0.7266±0.0309	0.7294±0.0146	0.7224±0.0391	0.7510±0.0452	0.7050±0.0398	0.7274±0.0166	0.7201±0.0384
F1-score	0.8374±0.0090	0.8380±0.0205	0.8430±0.0101	0.8379±0.0262	0.8452±0.0103	0.8268±0.0439	0.8435±0.0057	0.8341±0.0133

TABLE XIV
TRAINING AND TESTING TIME (SECONDS) ON THE *MIHC* DATASET

MIHC	iDTA-LS-SVM				DTA-LS-SVM				LS-SVM		SVM	
	Part (a)		Part (b)		Part (a)		Part (b)					
	training	testing	traning	testing	training	testing	training	testing	training	testing	training	testing
Time (s)	5.530	5.201	5.837	5.437	4.343	4.019	4.673	4.217	4950.6	2.398	1157.8	4.183

the stacked generalization principle and the transfer learning mechanism. In terms of the running time, DTA-LS-SVM and iDTA-LS-SVM remains superior over LS-SVM and SVM.

In addition, we also notice that in the experimental results of DTA-LS-SVM and iDTA-LS-SVM, both accuracy and F-score on the testing data set are higher than those on the training data set. There might be two reasons to explain these results. First, DTA-LS-SVM and iDTA-LS-SVM used the stacked hierarchical architecture to enhance generalization performances and this is reflected in the obtained results. Second, in the data preparation, the KNN imputation method was used to fill in missing values in the *MIHC* dataset. Following this procedure, the distributions in the training and testing sets might be mismatched. Referring to González and Abu-Mostafa's [44] conclusions, we postulate that mismatched distributions also occurred in this experiment, which may have caused a higher testing accuracy and F-score in DTA-LS-SVM and iDTA-LS-SVM.

C. Statistical Analysis

In order to detect significant differences among the experimental results of the proposed methods and the comparative methods, we also carried out the Friedman ranking test followed by Holm *post-hoc* test [45], [46] for multiple comparisons on the testing sets of four balanced and four imbalanced datasets in which the ratio of training and testing data sets is 7:3. The Friedman ranking test was used to evaluate whether there was a statistically significant difference among these methods. The null hypothesis is that there is no statistically difference. If the p-value is smaller than 0.5, the null hypothesis is rejected. The Holm *post-hoc* test was used to further verify if there was a statistical difference between the best Friedman ranking method and each remaining method. We used $\alpha = 0.05$ as the level of confidence in all cases. First, we conducted two Friedman ranking tests to assess whether there are significant differences between: 1) DTA-LS-SVM and the comparative methods on four balanced UCI datasets in terms of accuracy and F1-score and 2) iDTA-LS-SVM and the comparative methods on three imbalanced UCI datasets and one real-world *TRUS* dataset in terms of F1-score. Here, experimental results of both parts (a) and (b) on the *TRUS* dataset were included.

TABLE XV
AVERAGE RANKINGS OF DTA-LS-SVM AND THE COMPARATIVE METHODS ON BALANCED DATASETS IN TERMS OF ACCURACY (p-VALUE $= 0.049787$)

Methods	Ranking
DTA-LS-SVM	1
LS-SVM	2.5
SVM	2.5

TABLE XVI
HOLM POST-HOC COMPARISON RESULTS FOR DTA-LS-SVM AND THE OTHER METHODS IN TERMS OF ACCURACY WITH $\alpha = 0.05$

i	Methods	z-value	p-value	$Holm = \alpha/i$
2	LS-SVM	2.12132	0.033895	0.025
1	SVM	2.12132	0.033895	0.05

TABLE XVII
AVERAGE RANKINGS OF DTA-LS-SVM AND THE COMPARATIVE METHODS ON BALANCED DATASETS IN TERMS OF F1-SCORE (p-VALUE= 0.038774)

Methods	Ranking
DTA-LS-SVM	1
LS-SVM	2.25
SVM	2.75

The ranking results of Friedman test (1) in terms of accuracy and F1-score are shown in Tables XV and XVII, respectively. The results reveal that there are significant differences in the both performance metrics between DTA-LS-SVM and other comparative methods. Then we conducted the Holm *post-hoc* tests to compare the best ranking method DTA-LS-SVM with LS-SVM and SVM in terms of accuracy and F1-score, and presented the results in Tables XVI and XVIII, where the methods are ranked according to the obtained z-values. Holm *post-hoc* test rejects the hypothesis of equivalence for the methods with $p < \alpha/i$. According to the results in these tables, we know that DTA-LS-SVM is at least comparable to LS-SVM and SVM on balanced datasets in terms of accuracy; and is at least comparable to LS-SVM and statistically outperforms SVM on balanced datasets in terms of F1-score.

The ranking results of Friedman test (2) are shown in Table XIX. The results reveal that there are significant differences in terms of F1-score between iDTA-LS-SVM and the other comparative methods. Then we conducted the Holm

TABLE XVIII
HOLM POST-HOC COMPARISON RESULTS FOR DTA-LS-SVM AND THE OTHER METHODS IN TERMS OF F1-SCORE WITH $\alpha = 0.05$

i	Methods	z-value	p-value	$Holm = \alpha/i$
2	SVM	2.474874	0.013328	0.025
1	LS-SVM	1.767767	0.0771	0.05

TABLE XIX
AVERAGE RANKINGS OF iDTA-LS-SVM AND THE COMPARATIVE METHODS ON IMBALANCED DATASETS IN TERMS OF F1-SCORE (p-VALUE = 0.022371)

Methods	Ranking
iDTA-LS-SVM	1
LS-SVM	2.6
SVM	2.4

TABLE XX
HOLM POST-HOC COMPARISON RESULTS FOR iDTA-LS-SVM AND THE OTHER METHODS WITH $\alpha = 0.05$

i	Methods	z-value	p-value	$Holm = \alpha/i$
2	LS-SVM	2.529822	0.011412	0.025
1	SVM	2.213594	0.026857	0.05

post-hoc test to compare the best ranking method iDTA-LS-SVM with LS-SVM and SVM, and presented the results in Table XX. According to the obtained results, iDTA-LS-SVM outperforms the other methods on imbalanced datasets in terms of F1-score in our experiments.

In summary, the proposed methods are at least comparable to or even better than LS-SVM and SVM in terms of accuracy and/or F1-score with the faster learning speed.

V. CONCLUSION

AK-LS-SVM has been applied recently in many classification tasks. In this paper, to improve its generalization performance and learning speed, we proposed the novel classifier DTA-LS-SVM to be applied to balanced datasets, which follows the stacked generalization principle and transfer learning mechanism, and its extended version iDTA-LS-SVM on imbalanced datasets. DTA-LS-SVM and iDTA-LS-SVM stack multiple AK-LS-SVMs layer-by-layer, where the prediction from the previous module becomes one additional feature space together with the original data inputs to train the next module of the next layer. Moreover, transfer learning is embedded into the deep architecture to guarantee consistency across adjacent modules, and thus the classification capability of the module at the higher layer can be further enhanced. In addition, transfer learning based on AK-LS-SVM can perfectly formulate a fast leave-one-out cross validation strategy for the learning parameter tuning such that even though the kernel width and the regularization parameter are randomly selected, the performance of the proposed method can remain outstanding. We compared the proposed method with the traditional LS-SVM and SVM using additive Gaussian kernels on seven public UCI datasets and one real world community healthcare dataset. The experimental results indicate that the proposed classifiers DTA-LS-SVM and iDTA-LS-SVM have the superior advantages on the generalization performance and the running time. Moreover, the experimental results also reveal that both DTA-LS-SVM and iDTA-LS-SVM have the potential to be applied in a real world application.

Even though the proposed methods show a promising performance, this paper has some limitations that will be addressed in the future work. For example, DT-AK-LS-SVM has to use the same kernel in every processing layer. The proposed model will be investigated using different kernels in the stacked architecture to further improve the generalization performance.

REFERENCES

[1] G. C. Cawley, "Leave-one-out cross-validation based model selection criteria for weighted LS-SVMs," in *Proc. 6th Joint Int. Conf. Neural Netw.*, Vancouver, BC, Canada, 2006, pp. 1661–1668.
[2] H. Yang and J. Wu, "Practical large scale classification with additive kernels," in *Proc. 4th Asian Conf. Mach. Learn.*, Singapore, 2012, pp. 523–538.
[3] A. Vedaldi and A. Zisserman, "Efficient additive kernels via explicit feature maps," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 3, pp. 480–492, Mar. 2012.
[4] S. Maji, A. C. Berg, and J. Malik, "Efficient classification for additive kernel SVMs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 1, pp. 66–77, Jan. 2013.
[5] D. K. Duvenaud, H. Nickisch, and C. E. Rasmussen, "Additive Gaussian processes," in *Proc. Adv. Neural Inf. Process. Syst.*, Granada, Spain, 2011, pp. 226–234.
[6] G. Wang, Z. Deng, and K.-S. Choi, "Tackling missing data in community health studies using additive LS-SVM classifier," *IEEE J. Biomed. Health Inform.*, to be published, doi: 10.1109/JBHI.2016.2634587.
[7] C. M. Salgado *et al.*, "Takagi–Sugeno fuzzy modeling using mixed fuzzy clustering," *IEEE Trans. Fuzzy Syst.*, to be published, doi: 10.1109/TFUZZ.2016.2639565.
[8] B. Fan, X. Lu, and H.-X. Li, "Probabilistic inference-based least squares support vector machine for modeling under noisy environment," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 46, no. 12, pp. 1703–1710, Dec. 2016.
[9] L. Bi, O. Tsimhoni, and Y. Liu, "Using the support vector regression approach to model human performance," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 41, no. 3, pp. 410–417, May 2011.
[10] L. Wang *et al.*, "A UKF-based predictable SVR learning controller for biped walking," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 43, no. 6, pp. 1440–1450, Nov. 2013.
[11] A. Abdullah, R. C. Veltkamp, and M. A. Wiering, "An ensemble of deep support vector machines for image categorization," in *Proc. 1st Int. Conf. Soft Comput. Pattern Recognit.*, Malacca, Malaysia, 2009, pp. 301–306.
[12] Y. Tang, "Deep learning using linear support vector machines," in *Proc. Workshop Challenges Represent. Learn. (ICML)*, Atlanta, GA, USA, 2013.
[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.
[14] G. E. Hinton, "Deep belief networks," *Scholarpedia*, vol. 4, no. 5, p. 5947, 2009.
[15] R. Salakhutdinov and G. E. Hinton, "Deep Boltzmann machines," in *Proc. 12th Int. Conf. Artif. Intell. Stat.*, vol. 1. Clearwater, FL, USA, 2009, pp. 448–455.
[16] M. Wang, H.-X. Li, X. Chen, and Y. Chen, "Deep learning-based model reduction for distributed parameter systems," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 46, no. 12, pp. 1664–1674, Dec. 2016.
[17] L. Deng and D. Yu, "Deep convex net: A scalable architecture for speech pattern classification," in *Proc. 12th Annu. Int. Conf. Speech Commun. Assoc.*, Florence, Italy, 2011, pp. 2285–2288.
[18] D. H. Wolpert, "Stacked generalization," *Neural Netw.*, vol. 5, no. 2, pp. 241–259, 1992.
[19] L. Deng, G. Tur, X. He, and D. Hakkani-Tur, "Use of kernel deep convex networks and end-to-end learning for spoken language understanding," in *Proc. 4th IEEE Workshop Spoken Lang. Technol.*, Miami, FL, USA, 2012, pp. 210–215.
[20] B. Hutchinson, L. Deng, and D. Yu, "Tensor deep stacking networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1944–1957, Aug. 2013.

[21] O. Vinyals, Y. Jia, L. Deng, and T. Darrell, "Learning with recursive perceptual representations," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 2825–2833.
[22] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.
[23] W. Dai, Q. Yang, G.-R. Xue, and Y. Yu, "Boosting for transfer learning," in *Proc. 24th Int. Mach. Learn. Conf.*, Corvallis, OR, USA, 2007, pp. 193–200.
[24] J. Jiang and C. Zhai, "Instance weighting for domain adaptation in NLP," in *Proc. 45th Annu. Meeting Assoc. Comput. Linguist.*, vol. 7. Prague, Czech Republic, 2007, pp. 264–271.
[25] A. Quattoni, M. Collins, and T. Darrell, "Transfer learning for image classification with sparse prototype representations," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Anchorage, AK, USA, 2008, pp. 1–8.
[26] C. Wang and S. Mahadevan, "Manifold alignment using procrustes analysis," in *Proc. 25th Int. Conf. Mach. Learn.*, Helsinki, Finland, 2008, pp. 1120–1127.
[27] M. Stark, M. Goesele, and B. Schiele, "A shape-based object class model for knowledge transfer," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Kyoto, Japan, 2009, pp. 373–380.
[28] J. Gao, W. Fan, J. Jiang, and J. Han, "Knowledge transfer via multiple model local structure mapping," in *Proc. 14th ACM Int. Conf. Knowl. Disc. Data Min.*, Las Vegas, NV, USA, 2008, pp. 283–291.
[29] J. Lu *et al.*, "Transfer learning using computational intelligence: A survey," *Knowl. Based Syst.*, vol. 80, pp. 14–23, May 2015.
[30] D. Rafailidis and A. Nanopoulos, "Modeling users preference dynamics and side information in recommender systems," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 46, no. 6, pp. 782–792, Jun. 2016.
[31] Y. Bengio, G. Mesnil, Y. Dauphin, and S. Rifai, "Better mixing via deep representations," in *Proc. 30th Int. Conf. Mach. Learn.*, Atlanta, GA, USA, 2013, pp. 552–560.
[32] G. R. G. Lanckriet, T. De Bie, N. Cristianini, M. I. Jordan, and W. S. Noble, "A statistical framework for genomic data fusion," *Bioinformatics*, vol. 20, no. 16, pp. 2626–2635, 2004.
[33] T. Senechal *et al.*, "Combining AAM coefficients with LGBP histograms in the multi-kernel SVM framework to detect facial action units," in *Proc. 9th IEEE Int. Conf. Autom. Face Gesture Recognit. Workshops*, Santa Barbara, CA, USA, 2011, pp. 860–865.
[34] C.-Y. Yeh, C.-W. Huang, and S.-J. Lee, "A multiple-kernel support vector regression approach for stock market price forecasting," *Expert Syst. Appl.*, vol. 38, no. 3, pp. 2177–2186, 2011.
[35] I. W.-H. Tsang, J. T.-Y. Kwok, and J. M. Zurada, "Generalized core vector machines," *IEEE Trans. Neural Netw.*, vol. 17, no. 5, pp. 1126–1140, Sep. 2006.
[36] R. Batuwita and V. Palade, "Class imbalance learning methods for support vector machines," in *Imbalanced Learning: Foundations, Algorithms, and Application*, H. He and Y. Ma, Eds. Hoboken, NJ, USA: Wiley, 2013, ch. 5, pp. 83–99.
[37] S.-J. Lin, C. Chang, and M.-F. Hsu, "Multiple extreme learning machines for a two-class imbalance corporate life cycle prediction," *Knowl. Based Syst.*, vol. 39, pp. 214–223, Feb. 2013.
[38] Q. Zou, S. Xie, Z. Lin, M. Wu, and Y. Ju, "Finding the best classification threshold in imbalanced classification," *Big Data Res.*, vol. 5, pp. 2–8, Sep. 2016.
[39] S. Tan, "Neighbor-weighted K-nearest neighbor for unbalanced text corpus," *Expert Syst. Appl.*, vol. 28, no. 4, pp. 667–671, 2005.
[40] K.-S. Choi, R. K. P. Wai, and E. Y. T. Kwok, "Healthcare information system: A facilitator of primary care for underprivileged elderly via mobile clinic," in *Proc. 1st Int. Conf. Smart Health*, Beijing, China, 2013, pp. 107–112.
[41] G. E. Batista and M. C. Monard, "A study of k-nearest neighbour as an imputation method," in *Proc. 2nd Int. Conf. Hybrid Intell. Syst.*, vol. 87. Santiago, Chile, 2002, pp. 251–260.
[42] K. F. Leung, W. W. Wong, M. S. Tay, M. M. Chu, and S. S. Ng, "Development and validation of the interview version of the Hong Kong Chinese WHOQOL-BREF," *Qual. Life Res.*, vol. 14, no. 5, pp. 1413–1419, 2005.
[43] K. F. Leung, M. Tay, S. W. Cheng, and F. Lin, *Hong Kong Chinese Version World Health Organization Quality of Life Measure-Abbreviated Version*, WHOQOL-BREF (HK), Geneva, Switzerland, 1997.
[44] C. R. González and Y. S. Abu-Mostafa, "Mismatched training and test distributions can outperform matched ones," *Neural Comput.*, vol. 27, no. 2, pp. 365–387, 2015.
[45] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, Jan. 2006.
[46] S. Garcia and F. Herrera, "An extension on 'statistical comparisons of classifiers over multiple data sets' for all pairwise comparisons," *J. Mach. Learn. Res.*, vol. 9, pp. 2677–2694, Dec. 2008.

Guanjin Wang received the bachelor's and master's degrees in information technology and systems from Monash University, Melbourne, VIC, Australia, in 2012 and 2014, respectively.

She is currently a joint Ph.D. student with the Faculty of Engineering and Information Technology, University of Technology Sydney, Broadway, NSW, Australia, and the School of Nursing, Hong Kong Polytechnic University, Hong Kong. Her current research interest includes machine learning, computational intelligence, and health informatics.

Guangquan Zhang received the Ph.D. degree in applied mathematics from the Curtin University of Technology, Perth, WA, Australia, in 2001.

He is currently an Associate Professor and the Director of the Decision Systems and e-Service Intelligence Laboratory, Centre for Artificial Intelligence, Faculty of Engineering and Information Technology, University of Technology Sydney, Broadway, NSW, Australia. He has authored four monographs, five reference books, and over 400 papers, including over 200 refereed journals. His current research interests include multiobjective and group decision making, decision support systems, fuzzy measure and optimization, concept drift, and uncertain information processing.

Dr. Zhang was a recipient of seven Australian Research Council Discovery Grants and several other research grants.

Kup-Sze Choi (M'97) received the Ph.D. degree in computer science and engineering from the Chinese University of Hong Kong, Hong Kong.

He is currently an Associate Professor with the School of Nursing, as well as the Director of the Centre for Smart Health and the PolyU-Henry G. Leong Mobile Integrative Health Centre, Hong Kong Polytechnic University, Hong Kong. His current research interests include virtual reality and artificial intelligence, and their applications in medicine and healthcare.

Jie Lu (SM'13) received the Ph.D. degree in information systems from the Curtin University of Technology, Perth, WA, Australia, in 2000.

She is currently a Distinguished Professor and the Associate Dean in research excellence with the Faculty of Engineering and Information Technology, and the Director of the Centre for Artificial Intelligence, University of Technology Sydney, Broadway, NSW, Australia. She has authored ten research books and over 400 papers in international journals and conferences. Her current research interests include decision support systems, fuzzy transfer learning, recommender systems, concept drift, and e-Service intelligence.

Dr. Lu was a recipient of eight Australian Research Council Discovery Grants. She serves as the Editor-in-Chief for *Knowledge-Based Systems* (Elsevier) and the *International Journal of Computational Intelligence Systems* (Atlantis). She is a fellow of IFSA.