Chapter 3
Novel Inductive and Transductive Transfer Learning Approaches Based on Support Vector Learning

Zhaohong Deng and Shitong Wang

Abstract In this chapter, two novel transfer learning approaches based on support vector learning are involved. For inductive transfer learning, the knowledge-leverage-based TSK fuzzy system (KL-TSK-FS) is proposed, which demonstrates the good privacy-protection abilities and strong adaptability for the situations where the data are only partially available from the target domain while some useful knowledge of the source domains is available. For transductive transfer learning, domain adaptation kernelized support vector machine (DAKSVM) and its two extensions are proposed, which can reduce the distribution gap between different domains in an RKHS as much as possible by integrating the large margin learner with the proposed generalized projected maximum distribution distance (GPMDD) metric.

3.1 Introduction

3.1.1 Background

Recently, transfer learning has been studied extensively for different applications [1], such as text classification and indoor WiFi location estimation. Referring to Fig. 3.1 and the explanations given in Table 3.1, transfer learning is an approach to obtain an effective model of data from the target domain by effectively leveraging the useful information from source domains in the learning procedure.

Situations requiring transfer learning are becoming common in real-world applications. The modeling of fermentation process [2] is one example where the transfer learning is required. In the target domain of a microbiological fermentation process,

Z. Deng • S. Wang (✉)
School of Digital Media, Jiangnan University, Wuxi, Jiangsu, P.R. China
e-mail: wxwangst@yahoo.com.cn

Y. Ma and G. Guo (eds.), *Support Vector Machines Applications*,
DOI 10.1007/978-3-319-02300-7_3,

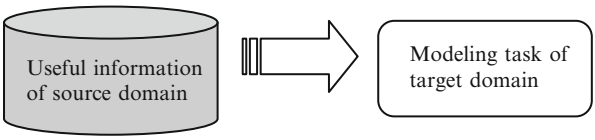

Fig. 3.1 An illustration of transfer learning for regression

Table 3.1 Some terms used for transfer leaning in the text

Terms	Explanations
Domain	A domain is a situation where a modeling task is to be accomplished. It is usually characterized by: (1) the *data* collected in this domain and (2) the *learning task* to be performed in this domain
Target domain	In transfer learning, it is referred to as the domain with insufficient data 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 learning task. There may be differences between the source domain and the target domain, but it is assumed that the source domain can provide some useful information for the modeling task of the target domain

the data collected may be insufficient or some of the data may be missing due to the deficiency of the sensor setup. Thus, we cannot effectively model the fermentation process for this domain with the collected data. However, data available from other similar microbiological fermentation process could be sufficient and considered as source domains for the target domain. Hence, transfer learning can be exploited to make use of the information from the source domain to improve the modeling effect of the target domain, thereby resulting in a model with better generalization capability. In this case, 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.

A comprehensive survey about transfer learning can be discovered in [1]. In general, the existing work about transfer learning can be categorized into three types: (1) transfer learning for classification [3–15]; (2) transfer learning for unsupervised learning (clustering [16, 17] and dimensionality reduction [18, 19]); and (3) transfer learning for regression [20–24]. According the setting whether there are the labeled data in the target domain available, all the transfer learning methods can also be classified as inductive transfer learning methods and transductive learning methods. While there are a few labeled data for the supervised learning in the inductive transfer learning methods, all the data are unlabeled in the target domain for the transductive learning methods and the unsupervised learning is implemented accordingly While there are a few labeled data for the inductive transfer learning, all the data are unlabeled in the target domain for the transductive learning method. Among the existing transfer methods, a lot of them are based on the support vector learning. In this chapter, we mainly focus on the novel support vector learning-based methods for the inductive and transductive learning.

3.1.2 Support Vector Learning-Based Inductive Transfer Learning

In the 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. The representative inductive transfer learning algorithms are reviewed below. Dai et al. [25] proposed a boosting algorithm with the support vector machine (SVM) as the learner, TrAdaBoost, which is an extension of the AdaBoost algorithm, to address the inductive transfer learning problems. TrAdaBoost attempts to iteratively reweight the source domain data to reduce the effect of the "bad" source data while encouraging the "good" source data to contribute more for the target domain. Wu and Dietterich [26] integrated the source domain (auxiliary) data and SVM framework for improving the classification performance. Evgeniou and Pontil [27] borrowed the idea of hierarchical Bayesian to SVMs for multitask learning. The proposed method assumed that the parameter, $\mathbf{w}$, in SVMs for each task can be separated into two terms. One is a common term over tasks and the other is a task-specific term.

3.1.3 Support Vector Learning-Based Transductive Transfer Learning

For support vector learning-based transductive transfer learning, a major computational problem is how to reduce the difference between the distributions of the source and target domains. There have existed several works describing how to measure the distance between distributions [28, 29]. Intuitively, discovering a good feature representation across domains is crucial [13, 30]. A good feature representation should be able to reduce the distribution discrepancy between two domains as much as possible, while at the same time preserving the underlying geometric structures (or scatter information) of both source and target domain data as much as possible. Ben-David et al. [31] used an example of hyperplane classifiers to show that the performance of the hyperplane classifier that could best separate the data could provide a good method for measuring the distribution distance for different data representations. Along these same lines, Gretton et al. [32] showed that for a given class of functions, the measure could be simplified by computing the discrepancy between two means of the distributions in a reproducing kernel Hilbert space (RKHS), thus resulting in the maximum mean discrepancy (MMD) measure. Inspired by the ideas of both transductive SVM (TSVM) and MMD, Brian et al. [28] proposed a so-called large margin kernel projected (LMPROJ) TSVM paradigm for domain adaptation problems based on the projected distance measure in an RKHS. The basic idea of LMPROJ is to minimize the distribution mean distance between source and target domain data by finding a feature translation in an RKHS. By the same way of LMPROJ, based on multiple kernel learning framework, Duan et al.

also proposed a domain transfer SVM (DTSVM) for domain adaptation learning (DAL) problem such as video concept detection. Further details about DTSVM can be found in [29].

3.1.4 Main Work in This Study

In this study, one support vector learning-based inductive learning approach and one support vector learning-based transductive learning approach are proposed, respectively.

3.1.4.1 Support Vector Learning-Based Inductive Transfer Learning with Knowledge-Leveraged Fuzzy Logic Systems

As support vector learning-based fuzzy system modeling is a type of important modeling methods [2, 33], it is promising to incorporate transfer learning with the fuzzy model. To the best of our knowledge, however, the study of transfer learning for support vector learning-based fuzzy system modeling has not yet been reported before. For support vector learning-based fuzzy system modeling, transfer learning is very useful in real-world modeling tasks where traditional fuzzy modeling methods may not work very well. For example, the trained fuzzy systems are much weaker in generalization capability when the training data are insufficient or only partially available [34, 35]. The situation is common in real-world applications in which the sensors and setups for data sampling are not steady due to noisy environment or other malfunctions that lead to insufficiency of data for the modeling task.

In order to tackle the problems with traditional support vector learning-based fuzzy system modeling as described above, a feasible remedy strategy is to boost up the performance by taking advantage of the useful information from source domains (or related domains), which can be the data in the domains, or the relevant knowledge like the density distribution and/or fuzzy rules. The simplest way to obtain the information from source domains is to directly use the data, collected from the source domains, but this approach leads to two major challenges. First, due to the necessity of privacy protection in some proprietary applications, such as the aforementioned fermentation process, the data of the source domains cannot always be obtained. Under this situation, the knowledge about the source domains, e.g. the density distribution and model parameters, can be obtained more easily to enhance the modeling of the target domain. Second, drifting phenomenon may exist between the source domain and the target domain, which makes it inappropriate to directly use the data from the former in the latter, or negative effect on the modeling task will be produced. These two issues should be properly addressed in order to develop an effective transfer learning modeling strategy for fuzzy systems.

In this study, a support vector learning-based fuzzy system modeling approach with knowledge-leverage capability from source domains is exploited for the inductive transfer learning. In view of its popularity, the Takagi–Sugeno–Kang-type fuzzy system (TSK-FS) is chosen to incorporate with a knowledge-leverage mechanism and hence the knowledge-leveraged TSK-type fuzzy system (KL-TSK-FS) is proposed. A novel objective criterion is proposed to integrate the model knowledge of the source domains and the data of the target domain, and the induced fuzzy rules of the model are learned accordingly. The knowledge of the source domain will effectively make up the deficiency in learning due to the lack of data in the target domain. Hence, the proposed system—KL-TSK-FS is more adaptive to the situations where the data are only partially available from the target domain while some useful knowledge of source domains is available. Besides, the proposed method is distinctive in preserving data privacy as only the knowledge (e.g., the corresponding model parameters) rather than the data of the source domain is used.

3.1.4.2 Support Vector Learning-Based Transductive Transfer Learning with DAL

As we may know well, mean (or expectation) and variance (or scatter) are two main features characterizing the distribution of samples which measure order one and order two statistics, respectively. However, most existing DAL methods for support vector learning-based transductive learning focus only on the first-order statistics matching which attempts to make the empirical means of the training and testing instances from source and target domain to be closer in an RKHS [36]. Intuitively, it is not enough to measure the distribution distance discrepancy between two domains to some extent only by considering the mean of the distribution of samples [13, 29, 36]. Hence, the state-of-the-art DAL MMD-based methods [28, 29, 37], which are only focused on the first-order statistics of the data distributions still have considerable limitation in the generalization capacity for specific domain adaptation transfer learning problems. What is more, since LMPROJ or DTSVM only focuses on the consistency of domain distributions in an RKHS, they sometimes project the data onto some noisy directions of separation which are completely irrelevant to the target learning task [13], and even result in poor performance.

In this study, we claim that it is indispensable to consider both mean and variance (or scatter) of data distribution in order to efficiently measure the distribution discrepancy between source and target domains. This motivates us to definitely utilize both MMD and scatter information of both domains to sufficiently evaluate their distribution discrepancy. In order to overcome the drawbacks of the MMD-based methods, we proposed a novel domain adaptation kernelized SVM (DAKSVM) using GPMDD discrepancy metric on RKHS embedding domain distributions, which can simultaneously consider both the distribution mean and scatter discrepancies between source and target domains. The idea is to find an RKHS for which the means and variances of the training and test data distributions are brought to be consistent, so that the labeled training data can be used to learn a

model for the test data. Particularly, we aim to obtain a linear kernel classifier based on the Representer Theorem [32], in an RKHS, such that it achieves a trade-off between the maximal margin between classes and the minimal discrepancy between the training and test distributions.

Compared with the existing state-of-the-art DAL methods, our main contributions include the following aspects: (1) the proposed methods inherit the potential advantages of classical TSVMs and MMD-based methods described as above, and further extend them to DAL; (2) as a novel large margin domain adaptation classifier, the proposed methods can reduce the distribution gap between different domains in an RKHS as much as possible, since they effectively integrate the large margin learner with the proposed GPMDD metric; (3) in addition, we propose two extensions to the standard formulation of DAKSVM based on both v-SVM and least-square SVM (LS-SVM), respectively.

The rest of this chapter is organized as follows. In Sect. 3.2, inductive transfer learning with support vector learning-based fuzzy systems is proposed; in Sect 3.3, transductive transfer learning with support vector learning-based domain adaptation transfer learning SVM by using the GPMDD metric is proposed; in Sect. 3.4, the experimental results about the proposed inductive transfer learning approach are reported; in Sect. 3.5, the experimental results about the proposed transductive transfer learning approach are reported; and The conclusions are given in the final section.

3.2 Inductive Transfer Learning with Support Vector Learning-Based Fuzzy Systems

3.2.1 Support Vector Learning-Based Fuzzy Systems

Support vector learning has been extensively used in the machine learning methods, such as kernel methods and other intelligence modeling methods. In this section, the support vector learning-based fuzzy systems, which have strong learning abilities and nicer interpretation properties, are introduced to develop the inductive transfer learning method.

3.2.1.1 Concept and Principle of TSK-FS

Classical fuzzy logic system models include the TSK model [38], Mamdani–Larsen (ML) model [39], and generalized fuzzy model [40]. Among them, the TSK model is the most popular one due to its effectiveness. In this study, the TSK model is adopted to develop the KL-TSK-FS for implementing the inductive transfer learning.

For TSK fuzzy logic systems, the most commonly used fuzzy inference rules are defined as follows:

TSK Fuzzy Rule R^k:

$$\text{IF } x_1 \text{ is } A_1^k \wedge x_2 \text{ is } A_2^k \wedge \cdots \wedge x_d \text{ is } A_d^k \tag{3.1}$$

$$\text{Then } f^k(\mathbf{x}) = p_0^k + p_1^k x_1 + \cdots + p_d^k x_d \quad k = 1, \cdots, K$$

In Eq. (3.1) A_i^{kK} 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 $\mathbf{x} = [x_1, x_2, \cdots, x_d]^T$, 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* is employed as the conjunction operator, *multiplicative implication* as the implication operator, and *additive disjunction* as the disjunction operator, the output of the TSK fuzzy model can be formulated as

$$y^0 = \sum_{k=1}^{K} \frac{\mu^k(\mathbf{x})}{\sum_{k'=1}^{K} \mu^{k'}(\mathbf{x})} \cdot f^k(\mathbf{x}) = \sum_{k=1}^{K} \tilde{\mu}^k(\mathbf{x}) \cdot f^k(\mathbf{x}), \tag{3.2a}$$

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

$$\mu^k(\mathbf{x}) = \prod_{i=1}^{d} \mu_{A_i^k}(x_i) \tag{3.2b}$$

and

$$\tilde{\mu}^k(\mathbf{x}) = \mu^k(\mathbf{x}) / \sum_{k'=1}^{K} \mu^{k'}(\mathbf{x}). \tag{3.2c}$$

A commonly used fuzzy membership function is the Gaussian membership function which can be expressed by

$$\mu_{A_i^k}(x_i) = \exp\left(\frac{-\left(x_i - c_i^k\right)^2}{2\delta_i^k}\right), \tag{3.2d}$$

where the parameters c_i^k, δ_i^k can be estimated by clustering techniques or other partition methods. For example, with fuzzy c-means (FCM) clustering, c_i^k, δ_i^k can be estimated as follows:

$$c_i^k = \sum_{j=1}^{N} u_{jk} x_{ji} / \sum_{j=1}^{N} u_{jk}, \tag{3.2e}$$

$$\delta_i^k = h \cdot \sum_{j=1}^{N} u_{jk} \left(x_{ji} - c_i^k\right)^2 / \sum_{j=1}^{N} u_{jk}, \tag{3.2f}$$

where u_{jk} denotes the fuzzy membership of the jth input data $\mathbf{x}_j = (x_{j1}, \cdots, x_{jd})^T$, belonging to the kth cluster obtained by FCM clustering [41]. Here, h is a scale parameter and can be adjusted manually.

When the antecedents of the TSK fuzzy model are determined, let

$$\mathbf{x}_e = \left(1, \mathbf{x}^T\right)^T, \tag{3.3a}$$

$$\tilde{\mathbf{x}}^k = \tilde{\mu}^k(\mathbf{x})\,\mathbf{x}_e, \tag{3.3b}$$

$$\mathbf{x}_g = \left(\left(\tilde{\mathbf{x}}^1\right)^T, \left(\tilde{\mathbf{x}}^2\right)^T, \cdots, \left(\tilde{\mathbf{x}}^K\right)^T\right)^T, \tag{3.3c}$$

$$\mathbf{p}^k = \left(p_0^k, p_1^k, \cdots, p_d^k\right)^T \tag{3.3d}$$

and

$$\mathbf{p}_g = \left(\left(\mathbf{p}^1\right)^T, \left(\mathbf{p}^2\right)^T, \cdots, \left(\mathbf{p}^K\right)^T\right)^T, \tag{3.3e}$$

then Eq. (3.2a) can be formulated as the following linear regression problem [33]

$$y^o = \mathbf{p}_g^T \mathbf{x}_g. \tag{3.3f}$$

Thus, the problem of TSK fuzzy model training can be transformed into the learning of the parameters in the corresponding linear regression model [2, 33].

3.2.1.2 Support Vector Learning-Based TSK-FS Training

Given a training dataset $D_{tr} = \{\mathbf{x}_i, y_i | \mathbf{x}_i \in R^d, y_i \in R, i = 1, \cdots, N\}$, for fixed antecedents obtained via clustering of the input space (or by other partition techniques), the least-square (LS) solution to the consequents is to minimize the following LS criterion function [29], that is,

$$\min_{\mathbf{p}_g} E = \sum_{i=1}^{N} \left(y_i^o - y_i\right)^2 = \sum_{i=1}^{N} \left(\mathbf{p}_g{}^T \mathbf{x}_{g\,i} - y_i\right)^2 = \left(\mathbf{y} - \mathbf{X}_g \mathbf{p}_g\right)^T \left(\mathbf{y} - \mathbf{X}_g \mathbf{p}_g\right), \tag{3.4}$$

where $\mathbf{X}_g = [\mathbf{x}_{g1}, \cdots, \mathbf{x}_{gN}]^T \in R^{N \times K \cdot (d+1)}$ and $\mathbf{y} = [y_1, \cdots, y_N]^T \in R^N$.

The most popular LS criterion-based TSK-FS training algorithm is the one used in the adaptive-network-based fuzzy inference systems [42]. For LS criterion-based algorithms, a main shortcoming is that they usually have weak robustness for modeling tasks involving noisy and/or small datasets. Besides the LS criterion-based

TSK-FS training methods, the more promising TSK-FS training methods are the support vector learning-based training algorithms, which are reviewed as follows.

Support Vector Learning-Based TSK-FS Training with L1-Norm Penalty

In addition to the LS criterion, another important criterion for TSK-FS training is the ε-insensitive criterion [33]. Given a scalar g and a vector $\mathbf{g}=[g_1,\cdots,g_g]^T$, the corresponding ε-insensitive loss functions take the following forms, respectively [33]: $|g|_\varepsilon = g-\varepsilon$ $(g>\varepsilon)$, $|g|_\varepsilon = 0$ $(g\le 0)$ and $|\mathbf{g}|_\varepsilon = \sum_{i=1}^{d}|g_i|_\varepsilon$. For the linear regression problem of the TSK-FS in Eq. (3.3f), the corresponding ε-insensitive loss-based criterion function [33] is defined as

$$\min_{\mathbf{p}_g} E = \sum_{i=1}^{N}|y_i^o - y_i|_\varepsilon = \sum_{i=1}^{N}\left|\mathbf{p}_g^T\mathbf{x}_{g_i} - y_i\right|_\varepsilon \tag{3.5a}$$

In general, the inequalities $y_i - \mathbf{p}_g^T\mathbf{x}_{gi} < \varepsilon$ and $\mathbf{p}_g^T\mathbf{x}_{gi} - y_i < \varepsilon$ are not satisfied for all data pairs $(\mathbf{x}_{gi},y_i)$. By introducing the slack variables $\xi_i^+ \ge 0$ and $\xi_i^- \ge 0$, Eq. (3.5a) can be equivalently written as

$$\min_{\mathbf{p}_g,\xi_i^+,\xi} E = \sum_{j=1}^{N}\left(\xi_i^+ + \xi_i^-\right) \tag{3.5b}$$

$$\text{s.t.} \quad \begin{cases} y_i - \mathbf{p}_g^T\mathbf{x}_{gi} < \varepsilon + \xi_i^+ \\ \mathbf{p}_g^T\mathbf{x}_{gi} - y_i < \varepsilon + \xi_i^- \end{cases}, \qquad \xi_i^+ \ge 0, \xi_i^- \ge 0 \qquad \forall i.$$

Further, by introducing the regularization term [30], Eq. (3.5b) is modified to become

$$\min_{\mathbf{p}_g,\boldsymbol{\xi}^+,\boldsymbol{\xi}^+} E = \frac{1}{\tau}\sum_{j=1}^{N}\left(\xi_i^+ + \xi_i^-\right) + \frac{1}{2}\mathbf{p}_g{}^T\mathbf{p}_g \tag{3.5c}$$

$$\text{s.t.} \quad \begin{cases} y_i - \mathbf{p}_g^T\mathbf{x}_{gi} < \varepsilon + \xi_i^+ \\ \mathbf{p}_g^T\mathbf{x}_{gi} - y_i < \varepsilon + \xi_i^- \end{cases}, \qquad \xi_i^+ \ge 0, \xi_i^- \ge 0 \qquad \forall i,$$

where $\tau > 0$ controls the trade-off between the complexity of the regression model and the tolerance of the errors. Here, ξ_i^+ and ξ_i^- can be taken as the L1-norm penalty terms and thus Eq. (3.5c) is an objective function based on L1-norm penalty terms. TSK training algorithm of this type is referred to as support vector learning-based L1-TSK-FS, which has the similar learning way as the classical SVM. The dual

optimization in Eq. (3.5c) is a quadratic programming (QP) problem, which can be expressed as

$$\begin{aligned}\max_{\boldsymbol{\alpha}^+,\boldsymbol{\alpha}^-} & -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\left(\alpha_i^+-\alpha_i^-\right)\left(\alpha_j^+-\alpha_j^-\right)\mathbf{x}_{gi}^T\mathbf{x}_{gi}-\sum_{i=1}^{N}\varepsilon\left(\alpha_j^++\alpha_j^-\right)\\ & +\sum_{i=1}^{N}y_i\left(\alpha_j^+-\alpha_j^-\right)\\ \text{s.t.}\quad & \sum_{i=1}^{N}\left(\alpha_j^+-\alpha_j^-\right)=0,\qquad \alpha_j^+,\alpha_j^-\in[0,\tau]\quad \forall i.\end{aligned} \tag{3.5d}$$

Compared with the LS-criterion-based algorithms, the support vector learning-based L1-TSK-FS with the ε-insensitive criterion has been shown to be more robust when dealing with noisy and small datasets.

Support Vector Learning-Based TSK-FLS Training with L2-Norm Penalty

Instead of the L1-norm penalty terms in Eq. (3.5c), another representative support vector learning-based TSK-FS learning method is the one developed by employing the L2-norm penalty terms [3]. The insensitive parameter ε is also added as a penalty term in the objective function. This is similar to the approaches used in other existing L2-norm penalty-based methods, e.g. L2-norm support vector regression (L2-SVR) [43]. For TSK fuzzy model training, the ε-insensitive objective function based on L2-norm penalty terms is then given by

$$\min_{\mathbf{p}_g,\boldsymbol{\xi}^+,\boldsymbol{\xi}^+,\varepsilon} \; g\left(\mathbf{p}_g,\boldsymbol{\xi}^+,\boldsymbol{\xi}^+,\varepsilon\right)=\frac{1}{\tau}\cdot\frac{1}{N}\sum_{j=1}^{N}\left(\left(\xi_i^+\right)^2+\left(\xi_i^-\right)^2\right)+\frac{1}{2}\mathbf{p}_g^T\mathbf{p}_g+\frac{2}{\tau}\cdot\varepsilon \tag{3.6a}$$

$$\text{s.t.}\quad \begin{cases} y_i-\mathbf{p}_g^T\mathbf{x}_{gi}<\varepsilon+\xi_i^+ \\ \mathbf{p}_g^T\mathbf{x}_{gi}-y_i<\varepsilon+\xi_i^- \end{cases}\quad \forall i.$$

Compared with the L1-norm penalty-based ε-insensitive criterion, the L2-norm penalty-based criterion is advantageous because of the following characteristics: (1) the constraints $\xi_i^+\geq 0$ and $\xi_i^-\geq 0$ in Eq. (3.5c) are not needed for the optimization; (2) the insensitive parameter ε can be obtained automatically by optimization without the need of manual setting. Similar properties can also be found in other L2-norm penalty-based machine learning algorithms, such as L2-SVR [43]. For convenience, the L2-norm penalty-based ε-insensitive TSK fuzzy model training

is referred to as L2-TSK-FS in this chapter. Based on the optimization theory, the dual problem in Eq. (3.6a) can be formulated as the following QP problem.

$$\begin{aligned}\max_{\boldsymbol{\alpha}^+,\boldsymbol{\alpha}^-} \quad & -\sum_{i=1}^{N}\sum_{j=1}^{N}\left(\alpha_i^+-\alpha_i^-\right)\left(\alpha_i^+-\alpha_i^-\right)\cdot\mathbf{x}_{gi}^T\mathbf{x}_{gj}-\sum_{i=1}^{N}\frac{N\tau}{2}\left(\alpha_i^+\right)^2-\sum_{i=1}^{N}\frac{N\tau}{2}\left(\alpha_i^-\right)^2 \\ & +\sum_{i=1}^{N}\alpha_i^+\cdot y_i\cdot\tau-\sum_{i=1}^{N}\alpha_i^-\cdot y_i\cdot\tau \\ \text{s.t.} \quad & \sum_{i=1}^{N}\left(\alpha_j^++\alpha_j^-\right)=1, \quad \alpha_j^+,\alpha_j^-\geq 0 \qquad \forall i\end{aligned} \tag{3.6b}$$

Notably, the characteristic of the QP problem in Eq. (3.6b) enables the use of core-set-based minimal enclosing ball (MEB) approximation technique to solve problems involving very large datasets [43]. The scalable L2-TSK-FS learning algorithm (STSK) has thus been proposed in this regard [3].

3.2.2 Inductive Transfer Learning with Support Vector Learning-Based TSK-FS

3.2.2.1 Framework of Knowledge-Leveraged Inductive Transfer Learning with TSK-FS

Most inductive transfer learning algorithms are developed to learn from the data in the source domain directly with some strategies. Recently, the transfer learning from the knowledge in the source domain rather than the original data is investigated with the knowledge-leveraged transfer learning framework [44], by observing the characteristics of two types of the learning ways below from the source domain, i.e., from the original data and from the induced knowledge.

1. For the data in the source domains, it is the original information and is also the most commonly used information for transfer learning. However, the data are not always available in some situations. For example, many data samples cannot be made open due to the necessity of privacy protection in the real world. Moreover, even if the data of source domains are available, it may not be always appropriate to directly adopt these data for the modeling task in the target domain due to the following issues: first, it is difficult to control and balance the similarity and difference of distributions of the source and target domains by using the data directly; secondly, there possibly exists a drifting between the distributions of different domains and thus some data from the source domain may result in an obvious negative influence on the modeling effect of the target domain.

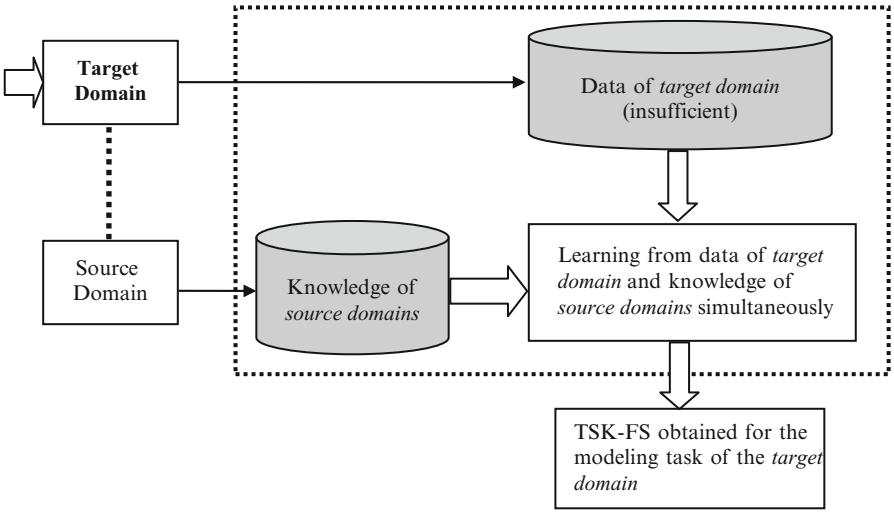

Fig. 3.2 Framework of knowledge-leveraged TSK-FS learning

2. For the knowledge in the source domains, it is another kind of important information. The types of knowledge are diverse, such as density distribution and model parameters. Most of them can be obtained by some learning procedures in the past. For example, the model parameters for the source domain can be learned by a certain modeling algorithm based on the data collected from that domain in a certain historical modeling task. Despite the fact that most of the knowledge obtained cannot be inversely mapped to the original data, it is a good property from a privacy preservation point of view and the important information from the source domains to improve the modeling effect of the target domain.

Thus, the characteristics above show that it should be more appropriate to exploit the use of knowledge rather than data from the source domains to enhance the modeling/learning performance of the models obtained in the target domain. As shown in Fig. 3.2, a generalized learning framework was proposed in [44] for knowledge-leveraged 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, the knowledge-leveraged inductive transfer learning for the support vector learning-based TSK-FS will be studied accordingly.

3.2.2.2 Inductive Transfer Learning with Support Vector Learning-Based TSK-FS

To take advantage of knowledge-leveraged learning mechanism for TSK-FS, KL-TSK-FS is proposed by using support vector learning and the L2-norm

penalty-based TSK-FS learning strategy with the corresponding knowledge-leverage mechanism. The goal is to effectively use the knowledge of the source domains to remedy the deficiency caused by data insufficiency in the target domain and develop an efficient learning algorithm for TSK-FS.

Objective Criterion Integrating the Knowledge of Source Domain

For a TSK-FS constructed by the support vector learning-based technique, the corresponding model parameters obtained in the source domains can be regarded as the knowledge. To develop an effective KL-TSK-FS for model learning of the target domain, we propose an optimization criterion which is integrated with the knowledge of the source domain as follows:

$$\min_{\mathbf{p}_g} \sum_{i=1}^{N} \left| \mathbf{p}_g{}^T \mathbf{x}_{gi} - y_i \right|_{\varepsilon} + \lambda (\mathbf{p}_g - \mathbf{p}_{g0})^T (\mathbf{p}_g - \mathbf{p}_{g0}) . \tag{3.7}$$

The optimization criterion in Eq. (3.7) contains two terms. The first term refers to the learning from the data of the target domain for the desired TSK-FS. This term is included so that the desired TSK-FS will fit the sampled training data of the target domain as accurate as possible. The second term refers to the knowledge-leverage of the source domain, with $\mathbf{p}_{g0}$ denoting model parameters learned from the source domains. The purpose is to estimate the desired parameters by approximating the model obtained from the source domains. The parameter λ in Eq. (3.7) is used to balance the influence of these two terms and the optimal value can be determined by using the commonly used cross-validation strategy in machine learning. As in L2-TSK-FS [20], we introduce the terms structure risk and ε-insensitive penalty into Eq. (3.7) to obtain the following objective criterion

$$\begin{aligned} \min_{\mathbf{p}_g, \xi^+, \xi^-, \varepsilon} & \frac{1}{\tau} \cdot \frac{1}{N} \sum_{i=1}^{N} \left(\left(\xi_i^+\right)^2 + \left(\xi_i^-\right)^2 \right) + \frac{1}{2} \left(\mathbf{p}_g^T \mathbf{p}_g\right) \\ & + \frac{2}{\tau} \cdot \varepsilon + \lambda (\mathbf{p}_g - \mathbf{p}_{g0})^T (\mathbf{p}_g - \mathbf{p}_{g0}) \\ \text{s.t.} & \begin{cases} y_i - \mathbf{p}_g^T x_{gi} < \varepsilon + \xi_i^+ \\ \mathbf{p}_g^T x_{gi} - y_i < \varepsilon + \xi_i^- \end{cases}, \forall i \end{aligned} \tag{3.8}$$

In fact, the former three terms in Eq. (3.8) are directly inherited from the L2-TSK-FS [20] and the last term is referred to as the knowledge-leverage term which is used to learn the knowledge from the source domains. Based on the objective criterion in Eq. (3.8), we can derive the corresponding learning rules for the proposed KL-TSK-FS.

Parameter Solution for KL-TSK-FS

Given the optimization problem in Eq. (3.8), Theorem 1 below is proposed for parameter solution.

Theorem 1 *The dual problem of Eq.* (3.8) *is a QP problem as shown in Eq.* (3.9).

$$\begin{aligned}\max_{\boldsymbol{\alpha}^-,\boldsymbol{\alpha}^+} & -\tfrac{1}{2(1+2\lambda)}\sum_{i=1}^{N}\sum_{i=1}^{N}\left(\alpha_i^- - \alpha_i^+\right)\left(\alpha_j^- - \alpha_j^+\right)\mathbf{x}_{gi}^T\mathbf{x}_{gj} - \tfrac{N\tau}{4}\sum_{i=1}^{N}\left(\left(\alpha_i^+\right)^2 + \left(\alpha_i^-\right)^2\right) \\ & - \tfrac{2\lambda}{1+2\lambda}\sum_{i=1}^{N}\left(\alpha_i^- - \alpha_i^+\right)\left(\mathbf{p}_{g0}^T\mathbf{x}_{gi} + y_i\right) + \tfrac{\lambda}{1+2\lambda}\mathbf{p}_{g0}^T\mathbf{p}_{g0} \\ \text{s.t.} \quad & \sum_{i=1}^{N}\alpha_i^- + \sum_{i=1}^{N}\alpha_i^+ = \frac{2}{\tau}, \qquad \alpha_i^- \geq 0, \alpha_i^+ \geq 0.\end{aligned} \tag{3.9}$$

Proof By using the Lagrangian optimization theorem, we can obtain the following Lagrangian function for Eq. (3.8)

$$\begin{aligned}L\left(\mathbf{p}_g, \boldsymbol{\xi}^+, \boldsymbol{\xi}^-, \varepsilon, \boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-\right) &= \frac{1}{\tau}\cdot\frac{1}{N}\sum_{i=1}^{N}\left(\left(\xi_i^+\right)^2 + \left(\xi_i^-\right)^2\right) \\ &+ \frac{1}{2}\left(\mathbf{p}_g^T\mathbf{p}_g\right) + \frac{2}{\tau}\cdot\varepsilon + \lambda\left(\mathbf{p}_g - \mathbf{p}_{g0}\right)^T\left(\mathbf{p}_g - \mathbf{p}_{g0}\right) \\ &+ \sum_{i=1}^{N}\alpha_i^+\left(y_i - \mathbf{p}_g^T\mathbf{x}_{gi} - \varepsilon - \xi_i^+\right) + \sum_{i=1}^{N}\alpha_i^-\left(\mathbf{p}_g^T\mathbf{x}_{gi} - y_i - \varepsilon - \xi_i^-\right).\end{aligned} \tag{3.10}$$

According to the dual theorem, the minimum of the Lagrangian function in Eq. (3.10) with respect to $\mathbf{p}_g, \boldsymbol{\xi}^+, \boldsymbol{\xi}^-, \varepsilon$ is equal to the maximum of the function with respect to $\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-$. Then the following equations can be considered as the necessary conditions of the optimal solution:

$$\frac{\partial L}{\partial \mathbf{p}_g} = \mathbf{p}_g + 2\lambda\left(\mathbf{p}_g - \mathbf{p}_{g0}\right) - \sum_{i=1}^{N}\left(\alpha_i^+ - \alpha_i^-\right)\mathbf{x}_{gi} = 0, \tag{3.11a}$$

$$\frac{\partial L}{\partial \xi_i^+} = \frac{2}{N\tau}\xi_i^+ - \alpha_i^+ = 0, \tag{3.11b}$$

$$\frac{\partial L}{\partial \xi_i^-} = \frac{2}{N\tau}\xi_i^- - \alpha_i^- = 0, \tag{3.11c}$$

$$\frac{\partial L}{\partial \varepsilon} = \frac{2}{\tau} - \sum_{i=1}^{N}\alpha_i^- - \sum_{i=1}^{N}\alpha_i^+ = 0. \tag{3.11d}$$

From Eqs. (3.11a)–(3.11d), we have

$$\mathbf{p}_g = \frac{2\lambda \mathbf{p}_{g0} + \sum_{i=1}^{N} \left(\alpha_i^+ - \alpha_i^-\right) \mathbf{x}_{gi}}{1+2\lambda}, \tag{3.12a}$$

$$\xi_i^+ = \frac{N\tau \cdot \alpha_i^+}{2}, \tag{3.12b}$$

$$\xi_i^- = \frac{N\tau \cdot \alpha_i^-}{2}, \tag{3.12c}$$

$$\sum_{i=1}^{N} \alpha_i^- + \sum_{i=1}^{N} \alpha_i^+ = \frac{2}{\tau}. \tag{3.12d}$$

Substituting Eqs. (3.12a)–(3.12d) into Eq. (3.10), we obtain the dual problem for Eq. (3.8), i.e.,

$$\begin{aligned} \max_{\boldsymbol{\alpha}^-, \boldsymbol{\alpha}^+} \; & \tfrac{-1}{2(1+2\lambda)} \cdot \sum_{i=1}^{N} \sum_{i=1}^{N} \left(\alpha_i^+ - \alpha_i^-\right) \left(\alpha_j^+ - \alpha_j^-\right) \mathbf{x}_{gi}^T \mathbf{x}_{gj} - \tfrac{N\tau}{4} \cdot \sum_{i=1}^{N} \left(\left(\alpha_i^+\right)^2 + \left(\alpha_i^-\right)^2 \right) \\ & - \tfrac{2\lambda}{1+2\lambda} \cdot \sum_{i=1}^{N} \left(\alpha_i^+ - \alpha_i^-\right) \mathbf{p}_{g0}^T \mathbf{x}_{gi} + \sum_{i=1}^{N} \left(\alpha_i^+ - \alpha_i^-\right) y_i + \tfrac{\lambda}{1+2\lambda} \cdot \mathbf{p}_{g0}^T \mathbf{p}_{g0} \\ \text{s.t.} \; & \sum_{i=1}^{N} \alpha_i^- + \sum_{i=1}^{N} \alpha_i^+ = \frac{2}{\tau}, \quad \alpha_i^- \geq 0, \alpha_i^+ \geq 0, \; \forall i. \end{aligned} \tag{3.12e}$$

Since the optimal solution of the dual problem, i.e., $(\boldsymbol{\alpha}^+)^*, (\boldsymbol{\alpha}^-)^*$, is independent of $\frac{\lambda}{1+2\lambda} \cdot \mathbf{p}_{g0}^T \mathbf{p}_{g0}$, Eq. (3.12e) is equivalent to the following equation:

$$\begin{aligned} \max_{\boldsymbol{\alpha}^-, \boldsymbol{\alpha}^+} \; & \tfrac{-1}{2(1+2\lambda)} \cdot \sum_{i=1}^{N} \sum_{i=1}^{N} \left(\alpha_i^+ - \alpha_i^-\right) \left(\alpha_j^+ - \alpha_j^-\right) \mathbf{x}_{gi}^T x_{gj} - \frac{N\tau}{4} \cdot \sum_{i=1}^{N} \left(\left(\alpha_i^+\right)^2 + \left(\alpha_i^-\right)^2 \right) \\ & - \tfrac{2\lambda}{1+2\lambda} \cdot \sum_{i=1}^{N} \left(\alpha_i^+ - \alpha_i^-\right) \left(p_{g0}^T x_{gi} + y_i\right) + \sum_{i=1}^{N} \left(\alpha_i^+ - \alpha_i^-\right) y_i \\ \text{s.t.} \; & \sum_{i=1}^{N} \alpha_i^- + \sum_{i=1}^{N} \alpha_i^+ = \frac{2}{\tau}, \alpha_i^- \geq 0, \alpha_i^+ \geq 0. \end{aligned} \tag{3.12f}$$

Thus, Theorem 1 is hold.

It is clear from the above results that the optimization problem in Eq. (3.9) for TSK-FS training can be transformed into a QP problem that can be directly solved by the traditional QP solutions [45].

With the optimal solution $(\boldsymbol{\alpha}^+)^*, (\boldsymbol{\alpha}^-)^*$ of the dual problem in Eq. (3.9), we can get the optimal solution of the primal problem in Eq. (3.8) based on the relations presented in Eqs. (3.12a)–(3.12d). The optimal model parameters of trained TSK-FS, i.e., $(\mathbf{p}_g)^*$, is then given by

$$(\mathbf{p}_g)^* = \frac{2\lambda \mathbf{p}_{g0} + \sum_{i=1}^{N} \left(\left(\alpha_i^+\right)^* - \left(\alpha_i^-\right)^* \right) \mathbf{x}_{gi}}{1 + 2\lambda}, \tag{3.13a}$$

which can be further expressed as

$$(\mathbf{p}_g)^* = \gamma \mathbf{p}_{g0} + (1\text{-}\gamma)\, \mathbf{p}_{gc}, \tag{3.13b}$$

with $\gamma = \frac{2\lambda}{1+2\lambda}$, $\mathbf{p}_{gc} = \sum_{i=1}^{N} \left(\left(\alpha_i^+\right)^* - \left(\alpha_i^-\right)^* \right) \mathbf{x}_{gi}$.

From Eq. (3.13b), we can see that the final optimal parameter $(\mathbf{p}_g)^*$ obtained for the desired TSK-FS contains two parts, i.e. $\gamma \cdot \mathbf{p}_{g0}$ and $(1-\gamma) \cdot \mathbf{p}_{gc}$. While $(1-\gamma) \cdot \mathbf{p}_{gc}$ can denote the knowledge learned from the data of the target domain, $\gamma \cdot \mathbf{p}_{g0}$ can be taken as the knowledge inherited from the source domains. Thus, the final model parameter $(\mathbf{p}_g)^*$ is a balance between these two kinds of knowledge.

Learning Algorithm For KL-TSK-FS

Based on the findings in the previous section, the learning algorithm of the proposed KL-TSK-FS is developed and described as follows:

Algorithm KL-TSK-FS	
Step 1	Introduce the knowledge of the source domains, i.e., the model parameter.
Step 2	Set the balance parameters τ, λ in Eq. (3.8).
Step 3	Use the antecedent parameters of the fuzzy model obtained from the source domains and Eqs. (3.2d) and (3.3e) to construct the dataset $\mathbf{x}_{gi}$ for the corresponding model task, i.e., the linear regression model in Eq. (3.3f), associated with the fuzzy system to be constructed for the target domain.
Step 4	Use Eqs. (3.9) and (3.13a) to obtain the final consequent parameters $(\mathbf{p}_g)^*$ of the desired TSK-FS in the target domain.
Step 5	Use the antecedent parameters inherited from the source domains and the consequent parameters obtained in step 4 to generate the fuzzy system for the target domain.

Computational Complexity Analysis

The computational complexity of the above algorithm is analyzed as follows. The whole algorithm includes two main parts: (1) acquisition of the antecedent parameters of the fuzzy system and (2) learning of the consequent parameters. For the first part, since the antecedent parameters are inherited directly from the reference scene as the available knowledge, the computational complexity is O(1). For the second, the consequent parameters are obtained by solving the QP problem in Eq. (3.9) and the complexity is usually O(N^2) for typical QP problems. However, it can be further reduced to O(N) with some sophisticated algorithms, such as the working set-based algorithm [33]. Therefore, the computational complexity of the proposed algorithm is between O(N) and O(N^2), depending on the QP solutions used. In this study, we adopt the working set-based QP solution [33] for solving the QP problem concerned.

3.3 Transductive Transfer Learning with DAKSVM

3.3.1 Concepts and Problem Formulation

In this subsection, we introduce several definitions to clarify our terminology and propose our algorithm and analysis on the domain adaptation transfer learning problems.

Definition 1 (Domain) A domain D is composed of both feature space χ and marginal probabilistic distribution $P(\mathbf{X})$, i.e., $D = \{\chi, P(\mathbf{X})\}$, where $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{N} \in \chi$.

Definition 2 (Task) Given a specific domain $D = \{\chi, P(\mathbf{X})\}$, a task is composed of both tag space $\mathbf{Y}$ and target prediction function $f(\cdot)$, i.e., $T = \{\mathbf{Y}, f(\cdot)\}$, where $f(\cdot)$ learned from the training dataset $\{\mathbf{x}_i, y_i\}$, where $\mathbf{x}_i \in \mathbf{X}$, $y_i \in \mathbf{Y}$. The function $f(\cdot)$ can be used to make prediction for the tag $f(\mathbf{x})$ corresponding with $\mathbf{X}$. From a probabilistic point of view $f(\mathbf{x}) = P(y|\mathbf{x})$.

Definition 3 (Domain Adaptation Learning, DAL) Given a source domain D_s with its learning task T_s and target domain D_t with its learning task T_t, respectively, we refer to domain adaptation learning (DAL) as the following problem: given a set of labeled training dataset $\mathbf{X}_s = \{(\mathbf{x}_i, y_i)\}_i \in D_s \times \{\pm 1\}$, where $y_i \in \mathbf{Y}_s \subset \mathbf{Y}$ is the class label corresponding to $\mathbf{x}_i$, from source domain D_s. Thus, we need to make prediction $f_t(\cdot)$ for some unlabeled test dataset $X_t = \{\mathbf{x}_j\}_j \in D_t$ from target domain D_t. D_s with its task T_s and D_t with its task T_t are different, respectively, in the same feature space. When $D_s = D_t$ and $T_s = T_t$, DAL will be degenerated into classical machine learning problems.

Given an input space $\mathbf{X}$ and a label set $\mathbf{Y}$ of classes, a classifier is a function as $f(\mathbf{x}) : \mathbf{X} \to \mathbf{Y}$ which maps data $\mathbf{x} \in \mathbf{X}$ to label set $\mathbf{Y}$. In the context, let us consider two

datasets $\mathbf{X}_s = \{(\mathbf{x}_{s1}, y_{s1}), \ldots, (\mathbf{x}_{sn}, y_{sn})\}$ drawn from $\mathbf{X} \times \mathbf{Y}$ with probabilistic distribution $P_s(\mathbf{x}_s, y_s)$ and $\mathbf{X}_t = \{\mathbf{x}_{t1}, \ldots, \mathbf{x}_{tm}\}$ drawn from $\mathbf{X}$ with probabilistic distribution $P_t(\mathbf{x}_t, y_t)$ where y_t needs to be predicted, which are composed of n source domain and m target domain patterns, respectively, and usually $0 \leq m << n$. $\mathbf{x}_s$ and $\mathbf{x}_t$ are denoted by d-dimensional feature vectors with respect to $\mathbf{X}_s$ and $\mathbf{X}_t$, respectively. The classical large margin learning machines (such as SVMs) work well under such hypothesis as $P_s(\mathbf{x}_s, y_s) = P_t(\mathbf{x}_t, y_t)$. However, DAL can make accurate prediction for the unlabeled target data to some extent by learning a classifier under even such hypothesis as $P_s(\mathbf{x}_s, y_s) \neq P_t(\mathbf{x}_t, y_t)$. The performance of DAL depends on both the complexity of the investigated problems and the correlation between $P_s(\mathbf{x}_s, y_s)$ and $P_t(\mathbf{x}_t, y_t)$ [6]. In this chapter, the proposed method is formulated under the following hypothesis:

1. There are only one source domain and one target domain sharing the same feature space in DAL problems, which is the most popular hypothesis used by the state-of-the-art methods.
2. A training dataset $\mathbf{X}_s = \{(\mathbf{x}_{si}, y_{si})\}_i$ is available for D_s while a testing dataset $\mathbf{X}_t = \{(\mathbf{x}_{tj}, y_{tj})\}_j$ is available for D_t with y_{tj} which is unknown.
3. $P_s(\mathbf{x}_s, y_s) \neq P_t(\mathbf{x}_t, y_t)$ and $P_s(y_s|\mathbf{x}_s) \neq P_t(y_t|\mathbf{x}_t)$.

3.3.2 *Distribution Discrepancy Metrics on RKHS Embedding Domain Distributions*

Kernel methods are broadly used as an effective way of constructing nonlinear algorithms from linear ones by embedding datasets into some higher dimensional RKHSs [46]. A generalization of this idea is to embed probabilistic distributions into RKHS, giving us a linear method for dealing with higher order statistics [47, 48]. Let a complete inner product space H of functions F, and for $g \in F$, $g : \mathbf{X} \to \mathbf{R}$, where $\mathbf{X}$ is a nonempty compact set, if the linear dot function mapping $g \to g(\boldsymbol{x})$ exists for all $\mathbf{x} \in \mathbf{X}$, we call H as an RKHS. Under the aforementioned conditions, $g(\boldsymbol{x})$ can be denoted as an inner product: $g(\mathbf{x}) = < g, \phi(\mathbf{x}) >_H$, where $\phi : \mathbf{X} \to H$ denotes the feature space projection from $\mathbf{x}$ to H. And the inner product of the images of any points $\mathbf{x}$ and $\mathbf{x}'$ in feature space is called kernel $k(\mathbf{x}, \mathbf{x}') = < \phi(\mathbf{x}), \phi(\mathbf{x}') >_H$. It is pointed out in [48] that RKHS with Gaussian kernel is universal.

Definition 4 (Integral Probability Metric on RKHS Embedding Distributions [2]) Given the set Θ of all Borel probabilistic measures defined on the topological space M, and the RKHS (H, k) of functions on M with k as its reproducing kernel. For any $P \in \Theta$, denotes by $Pk := \int_M k(.,\boldsymbol{x}) dP(\boldsymbol{x})$. If k is measurable and bounded, then we may define the embedding of P in H as $Pk \in H$. Then, the RKHS embedding distributions distance between two such mappings associated with $P, Q \in \Theta$ is defined as follows:

$$\gamma_k(P, Q) = ||Pk - Qk||_H \tag{3.14}$$

We may say k is a characteristic kernel (CK) if the mapping $P \mapsto Pk$ is injective [48], in which case $\gamma_k(P,Q) = 0$ if and only if $P = Q$ [49]. Hence γ_k is viewed as the distance metric on Θ. The RKHS embedding distributions cannot be distinguished when k is not a CK, thus leading to the failure of RKHS embedding distribution measure. Hence, it is a key factor for the success of RKHS embedding distribution measure that whether k is a CK or not. Fortunately, many popular kernel functions, such as polynomial kernel function, Gaussian kernel function, and Laplace kernel function, are all CK and universal ones [48]. Particularly, it is worth noting that Gaussian kernel mapping can provide us an effective RKHS embedding skill for the consistency estimation of the probability distribution distance between different domains [48]. Hence, in the sequel, we adopt the Gaussian kernel function $k_\sigma(\mathbf{x},\mathbf{z}) = \exp\left(-\frac{1}{2\sigma^2}||\mathbf{x}-\mathbf{z}||^2\right)$, where $\mathbf{x},\mathbf{z} \in \mathbf{X}$, and σ denotes the kernel bandwidth, as the reproducing kernel in Hilbert space in this work. It is worthy to note that instead of using a fixed and parameterized kernel, one can also use a finite linear combination of kernels to compute γ_k.

For domain adaptation transfer learning problems, let D_s and D_t denote source and target domain, respectively, and $\mathbf{X}_s \in D_s$, $\mathbf{X}_t \in D_t$ denote sample from D_s and D_t, respectively, with probability measures P_s and P_t, respectively. Let P_{x_s,x_t} denote the joint probability measure of $\mathbf{X}_s \times \mathbf{X}_t$. Assume all measures are Borel ones and $\mathbf{X}_s$, $\mathbf{X}_t$ are two compact sets. Besides, let an RKHS H of a class of functions F with kernel k, then for $g \in F$, $g: \mathbf{X} \to \mathbf{R}$, where $\mathbf{X}$ is a nonempty compact set, there exists the reproducing property as follows: $< g(\cdot), k(\mathbf{x},\cdot) > = g(\mathbf{x})$, $< k(\mathbf{x},\cdot), k(\mathbf{x}',\cdot) > = k(\mathbf{x},\mathbf{x}')$, where $<,>$ denotes inner product operator. Thus, by Definition 1, the unbiased empirical estimator of maximum mean distance (MMD) on RKHS embedding domain distributions is defined as [50]:

$$MMD(F,\mathbf{X}_s,\mathbf{X}_t) = \left\| \frac{1}{n}\sum_{i=1}^{n} \varphi(\mathbf{x}_i) - \frac{1}{m}\sum_{j=1}^{m} \varphi(\mathbf{z}_j) \right\|^2, \tag{3.15}$$

where $\mathbf{x}_i \in \mathbf{X}_s$, $\mathbf{z}_j \in \mathbf{X}_t$.

Specifically, by Definition 1, we can have the following definitions on RKHS embedding distribution distance metric.

Definition 5 (Projected Maximum Mean Distance Metric on RKHS Embedding Domain Distributions) Let linear function f: $f(\mathbf{x}) = \langle \mathbf{w}, \phi(\mathbf{x}) \rangle$, where $\mathbf{w}$ is a projection vector. Then the projected maximum mean distance metric on RKHS embedding domain distributions is defined as follows:

$$\begin{aligned} \gamma_{KM}(f,\mathbf{X}_s,\mathbf{X}_t) &= \left\| \frac{1}{n}\sum_{i=1}^{n} \mathbf{w}^T\varphi(\mathbf{x}_i) - \frac{1}{m}\sum_{j=1}^{m} \mathbf{w}^T\varphi(\mathbf{z}_j) \right\|^2 \\ &= \mathbf{w}^T \left(\frac{1}{n}\sum_{i=1}^{n} \varphi(\mathbf{x}_i) - \frac{1}{m}\sum_{j=1}^{m} \varphi(\mathbf{z}_j) \right) \left(\frac{1}{n}\sum_{i=1}^{n} \varphi(\mathbf{x}_i) - \frac{1}{m}\sum_{j=1}^{m} \varphi(\mathbf{z}_j) \right)^T \mathbf{w}, \end{aligned} \tag{3.16}$$

where $\boldsymbol{x}_i \in \mathbf{X}_s$, $\boldsymbol{z}_j \in \mathbf{X}_t$.

Definition 6 (Projected Maximum Scatter Distance Metric on RKHS Embedding Domain Distributions) Let linear function $f: f(\mathbf{x}) = \langle \mathbf{w}, \phi(\mathbf{x}) \rangle$, where $\mathbf{w}$ is a projection vector. Then, along the same line of Definition 2, the projected maximum scatter distance metric on RKHS embedding domain distributions is defined as

$$\gamma_{KS}(f, \mathbf{X}_s, \mathbf{X}_t) = \mathbf{w}^T \left| \frac{1}{n} \sum_{i=1}^{n} \varphi(\mathbf{x}_i) [\varphi(\mathbf{x}_i)]^T - \frac{1}{m} \sum_{j=1}^{m} \varphi(\mathbf{z}_j) [\varphi(\mathbf{z}_j)]^T \right| \mathbf{w}, \tag{3.17}$$

where $\mathbf{x} \in \mathbf{X}_s, \mathbf{z} \in \mathbf{X}_t$.

Definition 7 (GPMDD Metric on RKHS Embedding Domain Distributions) By Definitions 2 and 3, generalized projected maximum distribution distance metric on RKHS embedding domain distributions with probabilistic distribution $p, q \in P$ is defined as

$$\gamma_{KMS}(f, \mathbf{X}_s, \mathbf{X}_t) = (1-\lambda) \gamma_{KM}(f, \mathbf{X}_s, \mathbf{X}_t) + \lambda \gamma_{KS}(f, \mathbf{X}_s, \mathbf{X}_t), \tag{3.18}$$

where $\lambda \in [0,1]$ and when $\lambda = 0$, $\gamma_{KMS} = \gamma_{KM}$. The parameter λ is treated as a balance between probabilistic distribution mean and scatter (or variance). When λ increases, γ_{KMS} is biased in favor of preserving the distribution scatter consistency between both domains. When λ decreases, γ_{KMS} is biased in favor of preserving the distribution mean consistency between both domains. Hence, the proposed method can preserve both the distributions consistency between domains and the discriminative information in both domains.

It can be guaranteed by the following theorem that the GPMDD between both domains can be measured sufficiently.

Theorem 1 [51] Let F is a unit ball defined in some universal RKHS H with a kernel $k(\cdot,\cdot)$, which are all defined in a compact metric space. And let $\mathbf{X}_s$, $\mathbf{X}_t$ are two compact sets generated from Borel probability metrics p and q, respectively, in the metric space with p and q. Then $\gamma_{KMS}(F, \mathbf{X}_s, \mathbf{X}_t) = 0$ if and only if $p = q$.

3.3.3 Domain Adaptation Kernelized Support Vector Machine

Inspired by the idea of manifold regularization, MMD-based methods for transductive transfer learning (e.g., LMPROJ [28] and DTSVM [29], etc.) can be formulated as follows:

$$\begin{aligned} f = & \min_{\mathbf{w} \in H_K} C \sum_{i=1}^{n} \xi_i + \tfrac{1}{2} \|\mathbf{w}\|_K^2 + \lambda \gamma_{KM}(f, \mathbf{X}_s, \mathbf{X}_t) \\ s.t. \quad & y_i \left(\mathbf{w}^T \phi(\mathbf{x}_i) + b \right) \geq 1 - \xi_i, \\ & \xi_i \geq 0, i = 1, \ldots, n \end{aligned} \tag{3.19}$$

where $\mathbf{w}$ is a normal projection vector, k is a kernel with ϕ a kernel mapping, H_K is a set of functions in the kernel space, λ is a balance parameter, and $\gamma_{KM}(f,\mathbf{X}_s,\mathbf{X}_t)$ is the projected distribution mean distance metric between source and target domains, where $\mathbf{x}_i \in \mathbf{X}_s$.

However, Eq. (3.6) discloses a key limitation of MMD-based methods to some extent, i.e., they ignore considering sufficiently the potential scatter statistics, which may include underlying discriminative information in both domains for DAL, such that they may lead to "overfitting" phenomenon in some specific pattern recognition applications. Therefore, in this chapter, we propose a robust DAKSVM regularized by GPMDD metric on RKHS embedding domain distributions, which partially extends the ideas of classical SVMs and MMD. The key goals of our methods are to find a feature transform such that the mean and variance distances between the distributions of the testing and training data are minimized sufficiently, while at the same time maximizing the class margin or certain classification performance criterion for the training data, thus learning a robust model to effectively make prediction for target domain.

3.3.3.1 Objective Function of DAKSVM

For simplicity, firstly we only consider binary pattern classification problems, and secondly we propose a so-called least-square DAKSVM (LSDAKSVM) based on the classical LS-SVM [52] as an extension to the standard DAKSVM method for multi-class pattern classification problems.

For DAL problems, DAKSVM aims to find a linear transform $f(\mathbf{x}) = \mathbf{w}^T \varphi(\mathbf{x})$ in a universal RKHS with Gaussian kernel mapping, where $\mathbf{w}$ is a linear projection vector, in order to minimize the distribution discrepancy between-domain as well as to reduce the empirical risk of the classification decision function as much as possible, thus implementing transfer learning in cross-domains. DAKSVM can be formulated as

$$\min f = C\sum_{i=1}^{n} V(\mathbf{x}_i, y_i, f) + \gamma_{KMS}(f, \mathbf{X}_s, \mathbf{X}_t), \tag{3.20}$$

where $\mathbf{x}_i \in \mathbf{X}_s$ is a set of training data and matrix $\phi(\mathbf{X}_s) = (\phi(\mathbf{x}_{s1}), \phi(\mathbf{x}_{s2}), \ldots, \phi(\mathbf{x}_{sn}))$, $y_i \in \mathbf{Y}_s$ is the class label corresponding to $\mathbf{x}_i$, $C > 0$ is a regularization coefficient, and V measures the fitness of the function in terms of predicting the class labels for the training data and is called the risk function. The hinge loss function is a commonly used risk function in the form of $V = (1 - y_i f(x_i))_+$ [53] in which $(x)_+ = x$ if $x \geq 0$ and zero otherwise.

Therefore, the linear function f in Eq. (3.20) represented by a vector $\mathbf{w}$ can be represented as

$$\begin{aligned} &\arg\min_{\mathbf{w},b,\xi} f = C\sum_{i=1}^{n} \xi_i + \gamma_{KMS}(f, \mathbf{X}_s, \mathbf{X}_t) \\ &\text{s.t. } y_i((w, \varphi(\mathbf{x}_i)) + b) \geq 1 - \zeta_i, i = 1, 2, \ldots, n \end{aligned} \tag{3.21}$$

In order to solve the primal in Eq. (3.21) effectively, we introduce the following revised Representer Theorem for DAL problems as follows:

Theorem 2 (Representer Theorem [54] for DAL) For a DAL problem, let $\psi : [0,\infty) \to \mathbf{R}$ denote a strictly monotonic increasing function, $\mathbf{X} = \mathbf{X}_s \cup \mathbf{X}_t$ be a dataset, and $c : (\mathbf{X} \times \mathbf{R}^2)^n \to \mathbf{R} \cup \{\infty\}$ be any loss function. Then the regularized risk function is defined as

$$R(f) = c\left((x_i, y_i, f(x_i))_{i=1}^n\right) + \psi\left(||f||_H^2\right), \tag{3.22}$$

where $f \in H$ is represented as

$$f(x) = \sum_{i=1}^{m} \beta_i k(x_i, x) + \sum_{j=1}^{n} \beta_j k(z_j, x), \tag{3.23}$$

where k is a kernel, $\mathbf{x}_i \in \mathbf{X}_s$, $\mathbf{y}_i \in \mathbf{Y}_s$, $\mathbf{z}_j \in \mathbf{X}_t$ and β_i is a coefficient.

By Theorem 2, we can have the following theorem.

Theorem 3 The primal of DAKSVM can be formulated as

$$\min_{\boldsymbol{\beta}, \boldsymbol{\xi}, b} f = \tfrac{1}{2}\boldsymbol{\beta}^T \Omega \boldsymbol{\beta} + C \sum_{i=1}^{N} \xi_i, \tag{3.24a}$$

$$\text{s.t. } y_i \left(\sum_{j=1}^{n+m} \beta_j k_\sigma(\mathbf{x}_i, \mathbf{x}_j) + b \right) \geq 1 - \xi_i, \quad i = 1, \ldots, n,$$

where $\mathbf{x}_i \in \mathbf{X}_s$, $\mathbf{x}_j \in \mathbf{X}_s \cup \mathbf{X}_t$, Ω is a positive semi-definite kernel matrix with

$$\Omega = (1-\lambda)\Omega_1 + \lambda \Omega_2 \tag{3.24b}$$

where Ω_1 is a $(n+m) \times (n+m)$ symmetrical positive semi-definite kernel matrix defined as

$$\Omega_1 = \frac{1}{n^2}\mathbf{K}_s[1]^{n\times n}\mathbf{K}_s^T + \frac{1}{m^2}\mathbf{K}_t[1]^{m\times m}\mathbf{K}_t^T - \frac{1}{nm}\left(\mathbf{K}_s[1]^{n\times m}\mathbf{K}_t^T + \mathbf{K}_t[1]^{m\times n}\mathbf{K}_s^T\right) \tag{3.24c}$$

and Ω_2 is a $(n+m) \times (n+m)$ symmetrical positive semi-definite kernel matrix defined as

$$\Omega_2 = \left| \frac{1}{n}\mathbf{K}_s\mathbf{K}_s^T - \frac{1}{m}\mathbf{K}_t\mathbf{K}_t^T \right| \tag{3.24d}$$

where $\mathbf{K}_s$ is a $(n+m) \times n$ kernel matrix for the training data, $\mathbf{K}_t$ is a $(n+m) \times m$ kernel matrix for test data, and $[\mathbf{1}]^{k\times l}$ is a $k \times l$ matrix of all ones.

Theorem 4 The dual of the primal in Eq. (3.24) can be formulated as

$$\begin{aligned} &\min_{\boldsymbol{\alpha}} \tfrac{1}{2}\boldsymbol{\alpha}^T \mathbf{H}^{\phi}\boldsymbol{\alpha} - 1^T\boldsymbol{\alpha} \\ &\text{s.t. } 0 \le \alpha_i \le C, \ \ i = 1,\ldots,n, \\ &\sum_{i=1}^{n} \alpha_i y_i = 0, \end{aligned} \tag{3.25}$$

where $\mathbf{H}^{\phi} = \tilde{\mathbf{Y}}\mathbf{K}_s^T(\Omega)^{-1}\mathbf{K}_s\tilde{\mathbf{Y}}$, and $\tilde{\mathbf{Y}} = diag\,(y_1, y_2, \ldots, y_n)$, $y_i \in \mathbf{Y}_s$.

By the same way of the classical SVM, the biased variable b^{φ} in the kernel space can be formulated as

$$b^{\phi} = -\frac{1}{2}\left(\frac{1}{|\mathbf{X}_{s+}|}\sum_{\mathbf{x}\in\mathbf{X}_{s+}}\sum_{j=1}^{n+m}\beta_j k_{\sigma}(\mathbf{x}_j,\mathbf{x}) + \frac{1}{|\mathbf{X}_{s-}|}\sum_{\mathbf{x}\in\mathbf{X}_{s-}}\sum_{j=1}^{n+m}\beta_j k_{\sigma}\left(\mathbf{x}_j,\mathbf{x}\right)\right) \tag{3.26a}$$

Meanwhile, we can get the solution of β with dual theory as follows:

$$\beta = (\Omega)^{-1}\mathbf{K}_s\widetilde{\mathbf{Y}}\alpha \tag{3.26b}$$

3.3.3.2 Learning Algorithm of DAKSVM

The proposed DAKSVM algorithm can be summarized as follows.

Algorithm DAKSVM	
Input:	Dataset matrix $\mathbf{X} = (\{\mathbf{x}_i, y_i\}_{i=1}^{n}, \{\mathbf{z}_j\}_{j=1}^{m})$, $\mathbf{x}_i \in \mathbf{X}_s$, $\mathbf{y}_i \in \mathbf{Y}_s$, $\mathbf{z}_j \in \mathbf{X}_t$, set Gaussian kernel bandwidths σ, σ/γ, respectively, in γ_{KM} and γ_{KS} of GPMDD.
Output:	Decision function $f(\mathbf{x})$.
Step 1:	Determine the parameter γ in γ_{KS} of GPMDD such that the scatter consistency between source and target domains is maximized.
Step 2:	Compute the matrices Ω_1 and Ω_2, respectively, by Eqs. (3.24a) and (3.24b). In terms of λ given by users to construct matrix $\Omega = (1-\lambda)\Omega_1 + \lambda\Omega_2$.
Step 3:	For the given C, find out the optimal vector $\boldsymbol{\beta}$ by applying Theorem 4 to solve the corresponding dual. And then recover the optimal normal vector $\mathbf{w}$ and bias b^{ϕ} by $\boldsymbol{\beta}$;
Step 4:	Output the decision function $f(\mathbf{x}) = \mathbf{w}^T\phi(\mathbf{x}) + b^{\phi}$.

3.3.4 Variants and Extensions

3.3.4.1 Least-Square DAKSVM

One variant of DAKSVM is the LSDAKSVM which is also based on the idea of LS-SVM [52], which can be formulated as:

$$\begin{aligned} &\arg\min_{\mathbf{w},b,\xi} f = \frac{C}{2}\sum_{i=1}^{n}\xi_i^2 + \gamma_{KMS}(p,q) \\ &\text{s.t.}\quad (\mathbf{w},\phi(\mathbf{x}_i)) + b = y_i - \xi_i, i = 1,2,\ldots,n. \end{aligned} \tag{3.27}$$

Along the same line of DAKSVM, the primal of Eq. (3.17) is defined as

$$\min_{\boldsymbol{\beta},\xi,b} f = \frac{1}{2}\boldsymbol{\beta}^T\Omega\boldsymbol{\beta} + \frac{C}{2}\sum_{i=1}^{n}\xi_i^2, \tag{3.28}$$

$$\text{s.t.}\quad \sum_{j=1}^{n+m}\beta_j k_{\sigma/\gamma}(\mathbf{x}_i,\mathbf{x}_j) + b = y_i - \xi_i, \quad \xi_i \geq 0, i = 1,\ldots,n.$$

Theorem 5 (Analytic Solution to Binary Class Case) Given the parameter $\lambda \in [0,1]$, for a binary classification problem, the optimal solution of Eqs. (3.27) and (3.28) is equivalent to the linear system of equations with respect to variable α as follows:

$$\begin{bmatrix} 0 & \mathbf{1}_n^T \\ \mathbf{1}_n & \tilde{\Omega} \end{bmatrix} \begin{bmatrix} b \\ \boldsymbol{\alpha} \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{Y}_s \end{bmatrix}, \tag{3.29}$$

where $\mathbf{1}_n = [1,\ldots,1]^T$, $\boldsymbol{\alpha} = [\alpha_1,\ldots,\alpha_n]^T$, $\mathbf{Y}_s = [y_1,\ldots,y_n]^T$, $\tilde{\Omega} = \mathbf{K}_s^T(\Omega)^{-1}\mathbf{K}_s + \frac{\mathbf{I}_n}{C}$, $\mathbf{I}_n$ is an n-dimensional identity matrix.

As for multi-class classification problems, the traditional skills are to separate a multi-class classification problem into several binary classification problems in one-against-one (OAO) or one-against-all (OAA) way. However, the main drawbacks of these skills deal with high computational complexity and imbalance between classes. Hence, here we introduce the vector labeled outputs into the solution of LSDAKSVM, which can make the corresponding computational complexity independent of the number of classes and require no more computations than a single binary classifier [55]. Furthermore, Szedmak and Shawe-Taylor [55] pointed out that this technique does not reduce the classification performance of a learning model but in some cases can improve it, with respect to OAO and OAA. Therefore, we represent the class labels according to the one-of-c rule, namely, if training sample $\boldsymbol{x}_i$ $(i = 1,\ldots,n)$ belongs to the kth class, then the class label of $\boldsymbol{x}_i$ is

$Y_i = \left[\underbrace{0,\ldots,1}_{k},\ldots,0\right]^T \in \mathbf{R}^c$, where the kth element is one and all the other elements are zero. Hence, for some multi-class classification problems, the optimal problem of LSDAKSVM can be formulated as

$$\min_{\tilde{\boldsymbol{\beta}},\boldsymbol{\xi},b} f = \tfrac{1}{2}\tilde{\boldsymbol{\beta}}^T\Omega\tilde{\boldsymbol{\beta}} + \tfrac{C}{2}\sum_{i=1}^{n}\xi_i^2, \tag{3.30}$$

$$\text{s.t. } \tilde{\boldsymbol{\beta}}^T\mathbf{K}_s + b = Y_i - \xi_i, i = 1,\ldots,n,$$

where $\tilde{\boldsymbol{\beta}} \in R^{n\times c}$, $b \in \mathbf{R}^c$.

Theorem 6 (Analytic Solution to Multi-Class Case) Given the parameter $\lambda \in [0,1]$, for a multi-class classification problem, the optimal solution of Eq. (3.30) is equivalent to the linear system of the following equation.

$$\begin{bmatrix} b & \boldsymbol{\alpha} \end{bmatrix}\begin{bmatrix} 0 & 1_n^T \\ 1_n & \tilde{\Omega} \end{bmatrix} = \begin{bmatrix} 0_c & \tilde{\mathbf{Y}}_s \end{bmatrix}, \tag{3.31}$$

where $\mathbf{0}_c = [0,\ldots,0]^T$, $\boldsymbol{\alpha} = [\alpha_1,\ldots,\alpha_n]^T$, $\tilde{\mathbf{Y}}_s = [Y_1, Y_2, \ldots, Y_n]^T$, $\tilde{\Omega}$ is the same as in Theorem 6.

Theorems 5 and 6 actually provide us the LSDAKSVM versions for both binary and multi-class classification problems, respectively. It is clearly shown from Eqs. (3.20) and (3.23) that LSDAKSVM keeps the same solution framework for both binary and multi-class cases.

3.3.4.2 μ-Domain Adaptation Kernelized Support Vector Machine

The v-SVM [56] is a typical extension of SVM for classification in which Schölkopf et al. introduced a new parameter v instead of C in SVM to control the number of support vectors and the training errors. More details about v-SVM can be found in [56]. Hence, as the second variant of DAKSVM based on v-SVM, μ-DAKSVM can be formulated as:

$$\min_{\boldsymbol{\beta},\boldsymbol{\xi},b} f = \tfrac{1}{2}\boldsymbol{\beta}^T\Omega\boldsymbol{\beta} - \mu\rho + \tfrac{1}{N}\sum_{i=1}^{n}\xi_i, \tag{3.32}$$

$$\text{s.t. } y_i\left(\sum_{j=1}^{N}\beta_j k_{\sigma/\gamma}(\mathbf{x}_i,\mathbf{x}_j) + b\right) \geq \rho - \xi_i, \;\; i = 1,\ldots,n,$$

where the variables $N = n + m$, $\rho \geq 0$, $\mu > 0$ and $\xi_i \geq 0$ have the same meaning as in v-SVM. Similar to v-SVM, the dual of the primal in Eq. (3.32) can be formulated as:

$$\begin{aligned} &\min_{\alpha} \tfrac{1}{2}\alpha^T \mathbf{H}^{\phi} \alpha \\ &\text{s.t. } 0 \leq \alpha_i \leq \frac{1}{N}, \ \ i = 1, \ldots, n, \\ &\sum_{i=1}^{n} \alpha_i y_i = 0, \\ &\sum_{i=1}^{n} \alpha_i \geq \mu, \end{aligned} \tag{3.33}$$

where $\mathbf{H}^{\varphi} = \widetilde{\mathbf{Y}}\mathbf{K}_s^T \left(\overline{\Omega}\right)^{-1} \mathbf{K}_s \widetilde{\mathbf{Y}}$, and $\widetilde{\mathbf{Y}} = diag\,(y_1, y_2, \ldots, y_n)$, $y_i \in \mathbf{Y}_s$.

3.3.5 Computational Complexity Analysis

In terms of Algorithm 1, DAKSVM and its variants can be implemented by using standard SVM solver (e.g., LibSVM [57]) with the quadratic form induced by matrix Ω aforementioned above, and using the optimal solution to obtain the expansion coefficients by Eqs. (3.35) and (3.13)–(3.15) respectively. It is worth noting that our algorithms compute the inverse of a dense Gram matrix Ω which leads to $O((n+m)^3)$ training complexity comparable to SVM. This seems to be impractical for large datasets. However, for highly sparse datasets, for example, in text categorization problems, effective conjugate gradient schemes can be used in a large-scale implementation [58]. For the nonlinear case, one may obtain approximate solutions (e.g., using greedy, matching pursuit techniques) where the optimization problem is solved over the span of a small set of basis functions instead of using the full representation in $f(x) = \boldsymbol{w}^T \phi(\boldsymbol{x})$. Besides, CVM [59] may be an alternative choice in addressing scalability issues occurring in SVM learning. The testing complexity of DAKSVM depends on the number of support vector learned from the training stage. In fact, the proposed method DAKSVM and its variants take less than half a minute to finish the whole prediction for test samples from target domain in most of the following experiments.

3.4 Experimental Results of KL-TSK-FS

3.4.1 Experimental Settings

The proposed inductive transfer learning method KL-TSK-FS is evaluated by using both synthetic and real-world datasets. Details about the evaluation will be described in detail in Sects. 3.4.2 and 3.4.3, respectively. For clarity, the notations for the datasets and their definitions are listed in Table 3.2. Here, datasets generated from the source domain and the target domain are denoted by D1 and D2, respectively. The proposed support learning-based KL-TSK-FS algorithm is evaluated from the following two aspects.

1. *Comparison with traditional support vector learning-based L2-TSK-FS*. The performance of KL-TSK-FS is compared comprehensively with that of three L2-TSK-FS methods implemented under different conditions. That is, four TSK-FS systems are constructed by (a) L2-TSK-FS based on the data in the source domain, (b) L2-TSK-FS based on the data in the target domain, (c) L2-TSK-FS based on the data in both the target domain and the source domain, and (d) the proposed KL-TSK-FS. They are denoted by L2-TSK-FS(D1), L2-TSK-FS (D2), L2-TSK-FS (D1 + D2), and KL-TSK-FS(D2 + Knowledge), respectively. With these four fuzzy systems, the testing data, i.e. D2_test, of the target domain are used to evaluate their generalization capability.
2. *Comparison with regression methods designed for datasets with missing or noisy data*. Three related regression methods are employed to compare with the proposed KL-TSK-FS for performance evaluation. The three methods include: (a) TS-fuzzy system-based support vector regression (TSFS-SVR) [60]; (b) fuzzy system learned through fuzzy clustering and SVM (FS-FCSVM) [61]; and (c) Bayesian task-level transfer learning for nonlinear regression method (HiRBF) [20].

The methods adopted for performance comparison from these two aspects are summarized in Table 3.3. The following generalization performance index J is used in the experiments [2],

Table 3.2 Notations of the adopted datasets and their definitions

Notation	Definitions
D1	Dataset generated from the source domain
D2	Dataset generated from the target domain for training
D2_test	Dataset generated from the target domain for testing
r	Relation parameter between the source domain and the target domain, which is used to construct the synthetic datasets

Table 3.3 The methods adopted for performance comparison

Support vector learning and L2-norm penalty-based TSK-FS modeling methods		Four methods designed for noisy/missing data	
(1) The proposed KL-TSK-FS (D2 + Knowledge)	(2) L2-TSK-FS (D1) [2]	(1) The proposed KL-TSK-FS	(5) TSFS-SVR [60]
	(3) L2-TSK-FS (D2) [2]		(6) FS-FCSVM [61]
	(4) L2-TSK-FS (D1 + D2) [2]		(7) HiRBF [20]

$$J = \sqrt{\frac{\frac{1}{N}\sum_{i=1}^{N}\left(y_i' - y_i\right)^2}{\frac{1}{N}\sum_{i=1}^{N}\left(y_i - \bar{y}\right)^2}}, \tag{3.34}$$

where N is the number of test datasets, y_i is the output for the ith test input, $y_i^{'}$ is the fuzzy model output for the ith test input, and $\bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i$. The smaller the value of J, the better the generalization performance.

In the experiments, the hyperparameters of all the methods adopted are determined by using the fivefold cross-validation strategy with the training datasets. All the algorithms are implemented using MATLAB on a computer with Intel Core 2 Duo P8600 2.4 GHz CPU and 2GB RAM.

3.4.2 Synthetic Datasets

3.4.2.1 Generation of Synthetic Datasets

Synthetic datasets are generated to simulate the domains in the study and the following requirements need to be satisfied: (1) the source domain should be related to the target domain, i.e., the source and target domains are different but related; (2) some of the data of the target domain are not available or missing. In other words, the data available from the target domain are insufficient.

Based on the above requirements, the function $Y = f(x) = \sin(x) * x, x \in [-10, 10]$ is used to describe the source domain and to generate the dataset D1. On the other hand, the function $y = r * f(x) = r * \sin(x) * x, x \in [-10, 10]$ is used to describe the target domain and to generate the dataset D2 and testing dataset D2_test for the target domain. Here, r is a relation parameter between the source domain and the target domain. The parameter is used to control the degree of similarity/difference between these two domains. When $r = 1$, the two domains are identical. On the other hand,

Table 3.4 Details of the synthetic datasets

Source domain	Target domain		
Dataset	Training set		Testing set
Size of dataset	Interval with missing data	Size of dataset	Size of dataset
400	[−6, −3] and [0, 4]	144	200
Relation parameter between the two domains: $r = 0.9$, 0.85, 0.8, 0.75 and 0.7			

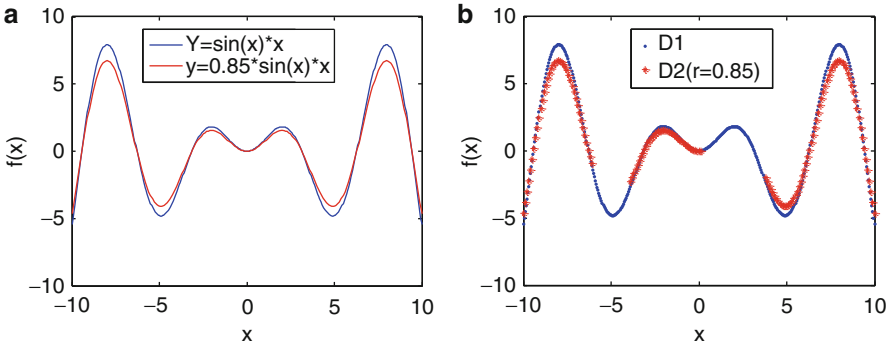

Fig. 3.3 Functions representing two different domains with the relation parameter $r = 0.85$ and the corresponding sampled data from these domains: (**a**) the functions representing the source domain (Y) and the target domain (y); (**b**) the data of the source domain and the training data of the target domain with missing data in the intervals [−6, −3] and [0, 4]

the lack of information (data insufficiency) is simulated by introducing intervals with missing data into the training set generated for the target domain. The settings for generating the synthetic datasets are described in Table 3.4. For example, the two functions used to simulate the two related domains, with the relation parameter $r = 0.85$, are shown in Fig. 3.3a. The datasets of the source domain and the training sets of the target domain, generated with the same relation parameter (i.e. $r = 0.85$), are shown in Fig. 3.3b. The figures also show the two data-missing intervals [−6, −3] and [0, 4] introduced into the dataset.

3.4.2.2 Comparing with the Traditional Support Vector Learning-Based L2-TSK-FS Modeling Methods

The performance of the proposed KL-TSK-FS and the three traditional L2-norm penalty-based TSK-FS modeling methods is evaluated and compared using the synthetic datasets. The experimental results are shown in Table 3.5 and Fig. 3.4. In Table 3.5 and other tables in this paper, the bold values denote the best results obtained among all the methods. The following observations can be made from the results:

Table 3.5 Generalization performance (J) of the proposed KL-TSK-FS method and the traditional L2-TSK-FS methods on the synthetic datasets

Interval with data missing	Relation parameter (r)	L2-TSK-FS (D1)	L2-TSK-FS (D2)	L2-TSK-FS (D1 + D2)	KL-TSK-FS (D2 + Knowledge)
[−6, −3] and [0, 4]	0.9	0.1343	0.2858	0.1012	**0.0501**
	0.85	0.1908	0.2813	0.1434	**0.0516**
	0.8	0.2574	0.2864	0.1983	**0.1094**
	0.75	0.3525	0.2841	0.2627	**0.1534**
	0.7	0.4406	0.2821	0.3432	**0.2388**

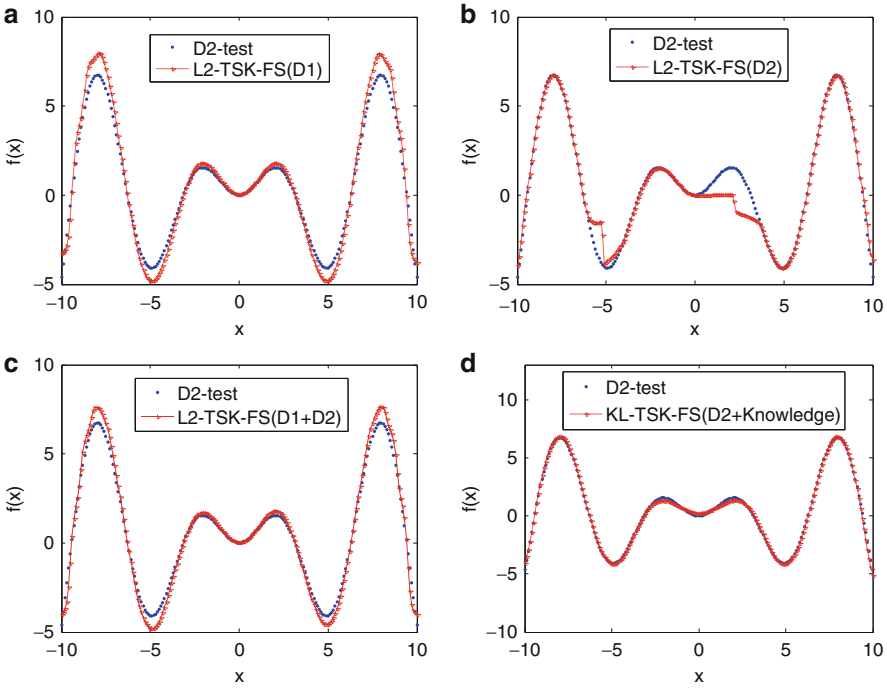

Fig. 3.4 Modeling results of the proposed KL-TSK-FS method and three traditional L2-TSK-FS methods by using the synthetic datasets shown in Fig. 3.5b: (**a**) L2-TSK-FS based on the data of the source domain (D1); (**b**) L2-TSK-FS based on the data of the target domain (D2); (**c**) L2-TSK-FS based on the data of both the reference and target domains (D1 + D2); (**d**) the proposed KL-TSK-FS (D2 + Knowledge)

1. It can be seen from Table 3.4 that the generalization performance of the knowledge-leverage-based fuzzy system KL-TSK-FS is better than that of the traditional L2-TSK-FS methods.
2. Figure 3.4a shows the modeling results of the L2-TSK-FS obtained by using the data of the source domain only. The results indicate that drifting exists between the source domain and the target domain, as evident from the discrepancies

between the two curves in the figure. Hence, the generalization performance of the TSK-FS obtained by L2-TSK-FS from the source domain is weak for the target domain. The findings show that the use of the data of the source domain alone is not appropriate for the modeling of the target domain.

3. Figure 3.4b shows the modeling results of the L2-TSK-FS obtained by using the data of the target domain only. The results indicate that the generalization performance of the TSK-FS obtained by L2-TSK-FS is even much weaker for the target domain. An obvious reason is that the data in the training set is insufficient, which degrades the generalization capability of the obtained TSK-FS. The prediction performance is especially poor in the intervals with missing data in the training dataset.
4. Figure 3.4c shows the modeling results of the L2-TSK-FS obtained by using the data of both the target domain and the source domain. Although the data of both domains have been used for training, the generalization performance of the obtained TSK-FS is still not good enough for the target domain. This can be explained by two reasons. First, drifting occurs between the reference and target domains, i.e., not all data in the source domain are useful for the modeling task of the target domain. Some of them may even have negative influence. Second, the size of the source domain is larger than that of the target domain, which makes the obtained TSK-FS more apt to approximate the source domain rather than the target domain.
5. Figure 3.4d shows the modeling results of the proposed KL-TSK-FS. The following observations can be made by comparing its results with the results of the three L2-TSK-FS methods, respectively. First, by inspecting Fig. 3.4a, d, we see that the KL-TSK-FS demonstrates better prediction results than the L2-TSK-FS which only uses the data of source domain. Second, it is evident from Fig. 3.4b, d that, by introducing the knowledge-leverage mechanism, the proposed KL-TSK-FS has effectively remedied the deficiency of the L2-TSK-FS obtained by the data of the target domain. By comparing Fig. 3.4c, d, we also find that the KL-TSK-FS has demonstrated better generalization performance than the L2-TSK-FS which employs the data of both the reference and target domains. It is noteworthy to point out that the KL-TSK-FS also has better privacy-protection capability than the methods that use the data of source domains directly. When the data in the source domains are not available due to the necessity of privacy protection, or in situations where knowledge are only partially revealed, methods requiring the data of all domains are no longer feasible. Therefore, the proposed KL-TSK-FS is particularly suitable for these situations attributed to its distinctiveness in privacy protection.

3.4.2.3 Comparing with Regression Methods Designed for Missing or Noisy Data

The performance of the proposed KL-TSK-FS method is evaluated by comparing its performance with that of three regression methods designed for handling

Table 3.6 Generalization performance (J) of the proposed KL-TSK-FS method and three related regression methods on the synthetic datasets

Interval with missing data	Relation parameter (r)	TSFS-SVR	FS-FCSVM	HiRBF	KL-TSK-FS
$[-6, -3]$ and $[0, 4]$	0.9	0.2972	0.3161	0.2621	**0.0501**
	0.85	0.2989	0.3179	0.2619	**0.0516**
	0.8	0.2983	0.3170	0.2687	**0.1094**
	0.75	0.2933	0.3167	0.2639	**0.1534**
	0.7	0.2970	0.3185	0.2611	**0.2388**

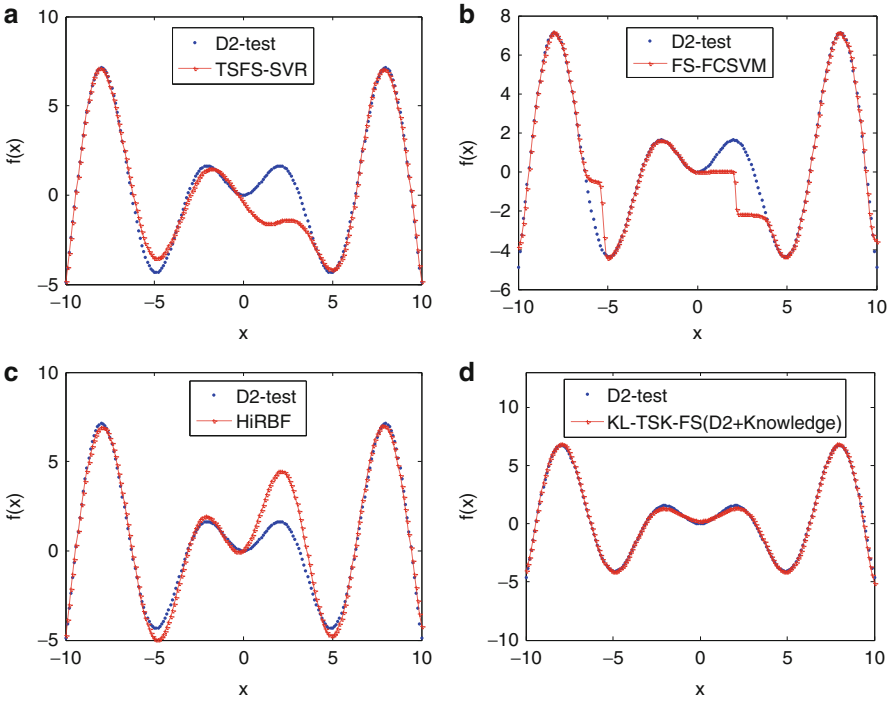

Fig. 3.5 Modeling results of the proposed KL-TSK-FS method and three related regression methods using the synthetic datasets in Fig. 3.5b: **a** TSFS-SVR, **b** FS-FCSVM, **c** HiRBF, and **d** KL-TSK-FS

noisy/missing data, i.e., TSFS-SVR, FS-FCSVM, and HiRBF. The evaluation is performed on the synthetic datasets. The experimental results are shown in Table 3.6 and Fig. 3.5, and the following observations can be obtained:

1. KL-TSK-FS has demonstrated better generalization performance than the other three related methods.
2. The results in Fig. 3.5a, b show that the support vector learning-based fuzzy modeling methods TSFS-SVR and FS-FCSVM are able to give better generalization

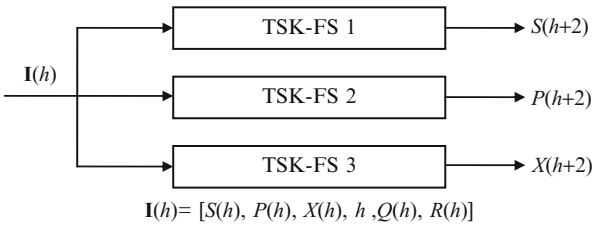

Fig. 3.6 Illustration of the glutamic acid fermentation process prediction model based on the TSK-FS

performance to a certain extent. For example, although the data in the interval $[-6, -3]$ are missing, these two methods still demonstrate promising generalization capability at this interval. However, the generalization abilities of these two methods in the other data-missing interval $[0, 4]$ are not satisfactory.

3. Although the transfer learning-based method HiRBF has used the data in both the target domain and the source domain in the training, it is evident form Fig. 3.5c that this method cannot effectively cope with the problem caused by the missing data, still exhibiting poor generalization ability in the two data-missing intervals.
4. Figure 3.5d shows that the proposed method KL-TSK-FS is able to give acceptable generalization capability in the two data-missing intervals, indicating that the method has effectively leveraged the useful knowledge from the source domain and remedy the generalization abilities in the training procedure.

3.4.3 Real-World Datasets

3.4.3.1 The Glutamic Acid Fermentation Process Modeling

To further evaluate the performance of the proposed method, an experiment is conducted to apply the method to model a biochemical process with real-world datasets [2]. The datasets adopted originates from the glutamic acid fermentation process, which is a multiple-input–multiple-output system. 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 $Q(h)$, where $h = 0, 2, \cdots, 28$. The output variables are glucose concentration $S(h+2)$, thalli concentration $X(h+2)$, and glutamic acid concentration $P(h+2)$ at a future time $h+2$. The TSK-FS-based biochemical process prediction model is illustrated in Figs. 3.6. The data in this experiment were collected from 21 batches of fermentation processes, with each batch containing 14 effective data samples. In this experiment, in order to match the situation discussed in this study, the data are divided into two domains, i.e., the source domain and the target domain, as described in Table 3.7.

Table 3.7 The fermentation process modeling datasets

		Data of target domain	
	Data of source domain (D1)	Training set (D2)[a]	Testing set (D2_test)
Batches	1–16	17–19	20–21
Size of dataset	224	30	28

[a]For training set of the target domain, information is missing at time $h = 6, 8, 10, 12$

Table 3.8 Generalization performance (J) of the proposed KL-TSK-FS method and the traditional L2-TSK-FS methods in fermentation process modeling

Output	L2-TSK-FS (D1)	L2-TSK-FS (D2)	L2-TSK-FS (D1 + D2)	KL-TSK-FS (D2 + Knowledge)
$S(h+2)$	0.2792	0.3944	0.2804	**0.1239**
$X(h+2)$	0.8342	1.1203	1.0642	**0.4548**
$P(h+2)$	0.2842	0.3255	0.2533	**0.1482**

3.4.3.2 Comparing with the Traditional L2-TSK-FS Modeling Methods

The experimental results of fermentation process modeling using the proposed inductive transfer learning method KL-TSK-FS and the traditional L2-TSK-FS are given in Table 3.8 and Fig. 3.7. The findings are similar to those presented in section IV-B for the experiments performed on the synthetic datasets. The modeling results of the KL-TSK-FS are better than that of the three traditional L2-TSK-FS methods. As the proposed method can effectively exploit not only the data of the target domain but also the useful knowledge of the source domains, the obtained TSK-FS has demonstrated better adaptive abilities. It can be seen from the experimental results that, even if the data in the training data of the target domain are missing, the generalization capability of the TSK-FS obtained by the proposed KL-TSK-FS does not degrade significantly. This remarkable feature is very valuable for the task of biochemical process modeling since the lack of sampled data is common due to poor sensitivity of sensors in the noisy environment.

3.4.3.3 Comparing with the Regression Methods Designed for Missing or Noisy Data

The experimental results of fermentation process modeling using the proposed inductive transfer learning method KL-TSK-FS and three regression methods (i.e., TSFS-SVR, FS-FCSVM, and HiRBF) are shown in Table 3.9 and Fig. 3.8. Similar to the findings presented in Sect. 3.4.2.3 for the experiments conducted with the synthetic datasets, in general, the proposed KL-TSK-FS has demonstrated better generalization performance than the other three regression methods in fermentation

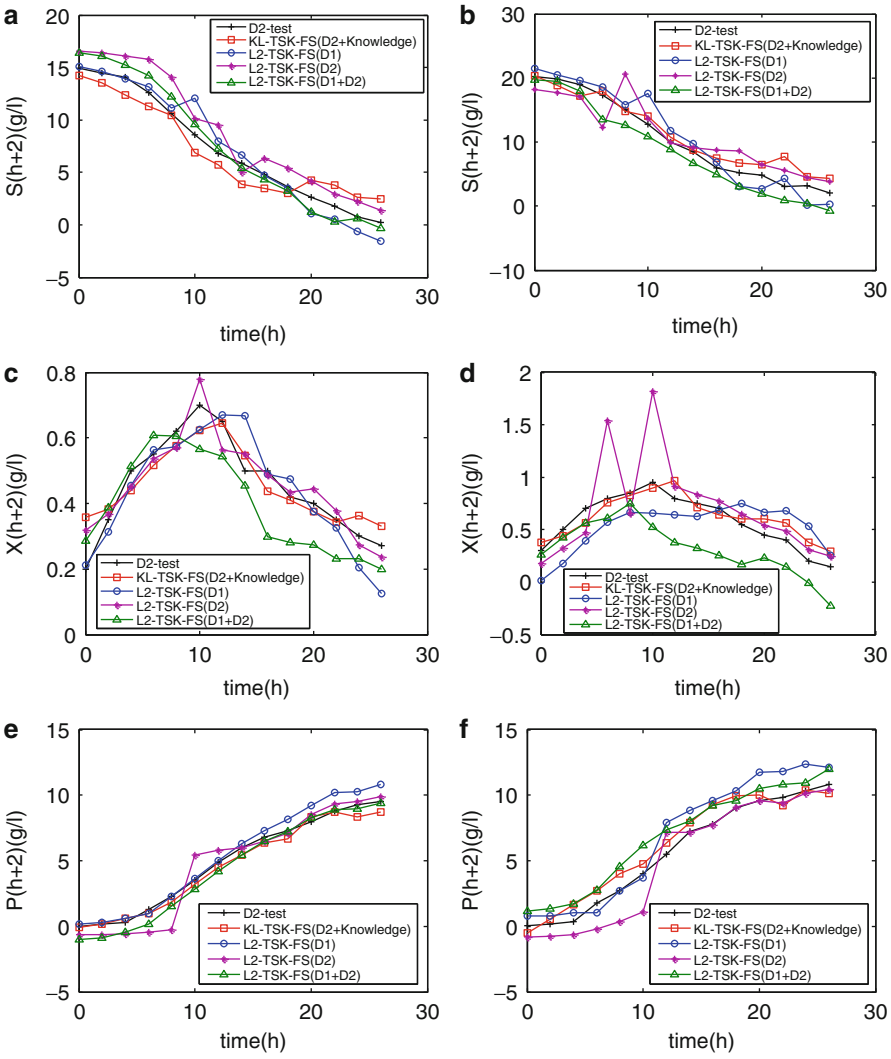

Fig. 3.7 Performance comparison between the proposed KL-TSK-FS method and three traditional L2-TSK-FS methods in fermentation process modeling: the prediction results of **a** S(h + 2) for the 20th batch; **b** S(h + 2) for the 21st batch; **c** X(h + 2) for the 20th batch; **d** X(h + 2) for the 21st batch; **e** P(h + 2) for the 20th batch; and **f** P(h + 2) for the 21st batch

process modeling. This can be explained again by the fact that the proposed KL-TSK-FS has effectively leveraged the useful knowledge from the source domain in the training procedure such that the influence of the missing data can be properly reduced.

Table 3.9 Generalization performance (J) of the proposed KL-TSK-FS method and several related regression methods in fermentation process modeling

Output	TSFS-SVR	FS-FCSVM	HiRBF	KL-TSK-FS
$S(h+2)$	0.3452	0.3750	0.3510	**0.1239**
$X(h+2)$	0.7295	0.6118	0.7026	**0.4548**
$P(h+2)$	0.3574	0.4144	0.4117	**0.1482**

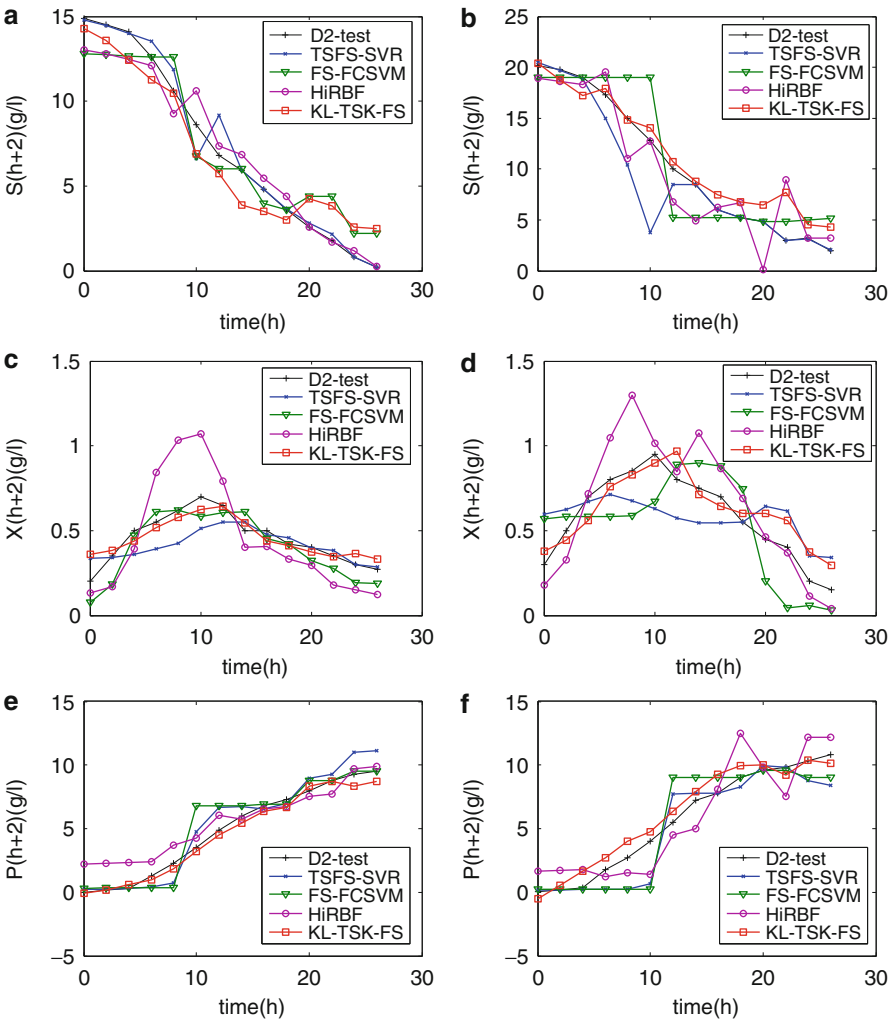

Fig. 3.8 Performance comparison between the proposed KL-TSK-FS method and three regression methods in fermentation process modeling: the prediction results of **a** S(h + 2) for the 20th batch; **b** S(h + 2) for the 21st batch; **c** X(h + 2) for the 20th batch; **d** X(h + 2) for the 21st batch; **e** P(h + 2) for the 20th batch; and **f** P(h + 2) for the 21st batch

3.5 Experimental Results of DAKSVM

3.5.1 Experiment Settings

To evaluate the effectiveness of the proposed transductive learning method DAKSVM and its extensions for DAL problems, we systematically compare them with several state-of-the-art algorithms on different datasets. We investigate three classes of domain adaptation problems: (1) a series of two-dimensional synthetic problems having different complexities with a two-moon dataset, (2) several real-world cross-domain text classification problems with different domain adaptation datasets such as 20Newsgroups, Reuters, Email Spam Filtering, web query set, and Amazon sentiment reviews set, and (3) a real problem in the context of multi-class classification in intra-domain on face recognition with Yale and ORL datasets. For all these datasets, true labels are available for both source and target-domain instances. However, prior information related to the target domain D_t is considered only for an objective and quantitative assessment of the performances of the proposed algorithms.

We construct synthetic datasets (two-moon) to exhibit the performance of the proposed method and choose real-world datasets to show the classification performance of the proposed method DAKSVM and its extension μ-DAKSVM. We also carry out a multi-class classification experiment to show the performance of the proposed method LSDAKSVM in multi-class classification problems.

In the sequel, we will first describe the whole experimental details. Throughout this experimental part, we use standard Gaussian kernel function as for several related kernel methods such as SVM, TSVM, KMM, TCA, LMPROJ, and DTSVM. For multiple kernel learning in DTSVM, according to the setting in [29], we use four Gaussian base kernels with the bandwidth $1.2^{\delta}\sigma$, where δ is set as $\{0,0.5,1,1.5\}$. For our methods, we use the parameterized Gaussian kernel as $k_{\sigma/\gamma}(\mathbf{x},\mathbf{x}_i) = \exp(-\|\mathbf{x}-\mathbf{x}_i\|^2/2(\sigma/\gamma)^2)$ in γ_{KS} of GPMDD, where the kernel parameter σ can be obtained by minimizing MMD with the most conservative test, which follows the setting in [46]. Empirically, we first select σ as the square root of the mean norm of the training data for binary classification and $\sigma\sqrt{c}$(where c is the number of classes) for multi-class classification. The tunable parameter γ can be set by minimizing GPMDD with the most optimal target test.

Presently, how to choose the algorithm parameters for the kernel methods still keeps an open and hot topic. In general, the algorithm parameters are manfully set. In order to evaluate the performance of the algorithm, a strategy, as is pointed out in [62], is that a set of the prior parameters is first given and then the best cross-validation mean rate among the set is used to estimate the generalized accuracy. In this work we adopted this strategy. The fivefold cross validation is used on the training set for parameter selection. Finally, the mean of experimental results on the test data is used for the performance evaluation. We chose the percentage overall accuracy AC% (i.e., the percentage of correctly labeled samples over the number of the whole samples) as the classification accuracy measure.

In the context, SVMs (such as SVM or v-SVM, TSVM) is implemented by the state-of-the-art software package such as LIBSVM [57]. As the experiments in Sect. 3.4, all the algorithms are implemented using MATLAB on a computer with Intel Core 2 Duo P8600 2.4 GHz CPU and 2GB RAM.

3.5.2 Synthetic Datasets

3.5.2.1 Generation of Synthetic Datasets

In this subsection, we construct a serial of trials on two-moon datasets to justify our method DAKSVM. In this toy problem, a serial of two-moon datasets with different complexities are used to exhibit the generalization capability of the proposed method DAKSVM on domain adaptation transfer learning. We compare the proposed method DAKSVM with SVM and LMPROJ on this toy data.

A synthetic dataset containing 600 samples generated according to a bi-dimensional pattern of two intertwining moons associated with two specific information classes (300 samples each) is taken as the source domain data, as shown in Fig. 3.9a. Target data were generated by rotating anticlockwise the original source dataset 11 times by 10°, 15°, 20°, 25°, 30°, 35°, 40°, 45°, 50°, 55°, and 60°, respectively. Due to rotation, source and target-domain data exhibit different distributions. Particularly, the greater the rotation angle, the more complex the resultant domain adaptation problem, as confirmed by the values for Jensen–Shannon scatter (D_{JS}) [6] shown in Fig. 3.10a. The proposed DAKSVM algorithm is proved to be particularly effective for solving this kind of problems with high accuracy. Figure 3.9b, c shows the target domain data with the rotation angle 30° and 60°, respectively.

3.5.2.2 Comparing with the Related Methods

Figure 3.9d–i shows the learning accuracy rates of different methods on the datasets shown in Fig. 3.9b, c. And Fig. 3.10b shows the performance comparison among different methods on 11 target datasets aforementioned above. From Figs. 3.9b–d and 3.10b, we can observe that with appropriate learning parameters, the proposed method can obtain perfect separation between classes even if the rotation angles range from 10° to 50°. Besides, we can also observe several results as follows:

1. From Fig. 3.9d–i, we can observe that the accuracies of DAKSVM and LMPROJ are always higher than those by SVM according to a fivefold cross-validation on source domain data. This result shows that it is unsuitable for SVM on cross-domain learning. With Figs. 3.9 and 3.10, in some angles range (i.e., from 10° to 50°), the proposed method and LMPROJ can preserve the solution consistency

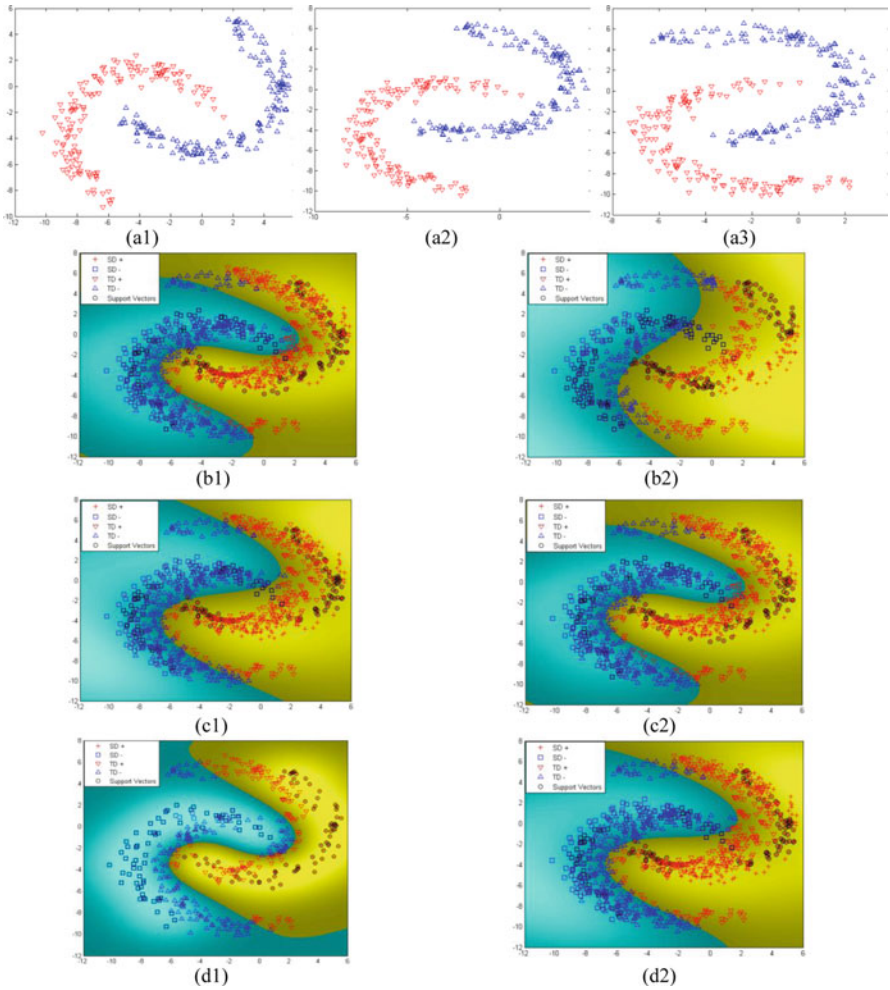

Fig. 3.9 Performance of different classifiers on two two-moon datasets with different complexities. **a** The original two-moon dataset; **b** rotation angle 30°; **c** rotation angle 60°; **d** classification accuracy for SVM: 95.4 %; **e** classification accuracy for SVM: 65 %; **f** classification accuracy for LMPROJ: 97.3 %; **g** classification accuracy for LMPROJ: 78.7 %; **h** classification accuracy for DAKSVM: 98.7 %; **i** classification accuracy for DAKSVM: 87.5 %

well with target domain to some extent, which shows that the proposed method is better than or at least comparable to LMPROJ in this experiment.

2. Figure 3.10b shows that for greater values of rotation angles (i.e., from 50° to 60°), the classification accuracy rates of all methods descend dramatically, which seems reasonable due to the increase of the complexity of the corresponding domain adaptation problems; however, the descendant rate of the proposed method is slower than others due to preserving the distribution consistency of

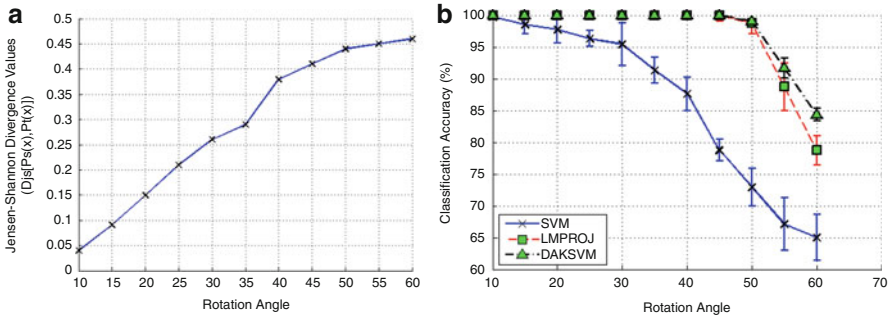

Fig. 3.10 Jenson–Shannon divergence values and classification accuracies on target domain data for different rotation angles: **a** Jenson–Shannon divergence values for different rotation angles (D_{JS}); **b** accuracies exhibited on target domain data for different rotation angles

both means and variances of different domains. When the rotation angle is big enough, all methods will not be able to keep the solution consistency with target domain. If this case happens, the hypothesis aforementioned above will not be satisfied.

3.5.3 Binary Class Text Classification Datasets

In this section, we demonstrate the overall efficiency and effectiveness of the proposed method DAKSVM and its variation μ-DKSVM on five different real-world domain adaptation tasks for text datasets such as 20Newsgroups, Reuters, mail spam filtering, web query classification, and Amazon sentiment reviews classification.

Except for SVM, KMM, DTSVM, LMPROJ, and the TSVM, we still choose for comparison another two algorithms from KDD'08. They are cross-domain spectral classifier [63] and locally weighted ensemble (LWE) classifier [9].

Unlike SVM and TSVM with default parameter values are adopted in most cases, in order to make our comparison fair, we report the best performance for each method over a range of parameter selections.

3.5.3.1 Dataset Settings

A brief description of each dataset and its setup is given in this subsection. Tables 3.10 and 3.11 summarize the datasets and give the indices to some of which we will refer in our experimental results. For example, dataset 6 is a 20Newsgroup dataset about *Rec.* vs. *Sci.* where the number of positive and negative training samples is 1,984 and 1,977, respectively, and the number of positive and negative class testing samples is 1,993 and 1,972, respectively.

Table 3.10 Cross-domain text classification tasks

Task		Datasets	Number of training samples		Number of testing samples	
			Positive class	Negative class	Positive class	Negative class
1	Reuters	Orgs vs. People	Documents from sub-categories		Documents from different sub-categories	
2		Orgs vs. Place				
3		People vs. Place				
4	20Newsgroup	Comp vs. Sci	1,958	1,972	2,923	1,977
5		Rec vs. Talk	1,993	1,568	1,984	1,658
6		Rec vs. Sci	1,984	1,977	1,993	1,972
7		Sci vs. Talk	1,971	1,403	1,978	1,850
8		Comp vs. Rec	2,916	1,993	1,965	1,984
9		Comp vs. Talk	2,914	1,568	1,967	1,685
10	Email spam filtering	User1 vs. User2	User1's emails		User2's emails	
11		User2 vs. User3	User2's emails		User3's emails	
12		User3 vs. User1	User3's emails		User1's emails	

Table 3.11 Web query text and sentiment reviews classification tasks

Task		Categories	Number of training samples	Number of testing samples
13	Web query	Business (B)	1,500	1,200
14		Computers (C)	1,500	1,000
15		Education (E)	2,210	2,500
16		Health (H)	1,180	1,190
17		Sports (S)	1,420	660
18	Amazon sentiment reviews	Books (B)	1,000	1,000
19		DVDs (D)	1,000	1,000
20		Electronics (E)	1,000	1,000
21		Kitchen (H)	1,000	1,000

20Newsgroups and Reuters

Reuters and 20Newsgroups are two cross-domain text classification datasets commonly used by the state-of-the-art DAL classifiers [9, 28–30, 36, 64]. These datasets both represent text categorization tasks, Reuters is made up of news articles with five top-level categories, among which, Orgs, Places, and People are the largest, and the 20Newsgroups dataset contains 20newsgroup categories each with approximately 1,000 documents. For these text categorization data, in each case the goal is to correctly discriminate between articles at the top level, e.g. "sci" articles vs. "talk" articles, using different sets of sub-categories within each top-category for training and testing, e.g. *sci.electronics* and *sci.med* vs. *talk.politics.misc* and *talk.religion.misc* for training and *sci.crypt* and *sci.space* vs. *talk.politics.guns* and *talk.politics.mideast* for testing. For more details about the sub-categories, see [65]. Each set of sub-categories represents a different domain in which different words will be more common. Features are given by converting the documents into bag-of-word representations which are then transformed into feature vectors using the term frequency, details about this procedure can also be found in [65]. Table 3.10 shows the more detailed information about the experimental datasets drawn from the aforementioned above datasets.

Email Spam Filtering

In email spam filtering datasets [66], there are three email subsets (denoted by User1, User2, and User3, respectively) annotated by three different users. In this trial, the task is to classify spam and non-spam emails. Since the spam and non-spam emails in the subsets have been identified by different users, the data distributions of the three subsets are different but related. Each subset has 2,500 emails, in which one half of the emails are non-spam (labeled as 1) and the other half of them are spam (labeled as -1). On this dataset, in terms of [54], we consider three settings: (1) User1 (source domain) and User2 (target domain); (2) User2 (source domain)

and User3 (target domain), and 3) User3 (source domain) and User1 (target domain). For each setting, the training dataset contains all labeled samples from the source domain. And the samples in the target domain are used as the unlabeled test ones. We report the experimental results with their means and the standard deviations of all methods. Again, the word-frequency feature is used to represent each document as in [66]. The more detailed information about the experimental datasets drawn from Email Spam Filtering datasets can be found in Table 3.2.

Web Query

We also construct a set of tasks on cross-domain query classification for a search engine, e.g. Google. We use a set of search snippets gathered from Google as our training data and some incoming unlabeled queries as the test data. The detailed descriptions of the procedure can be found in [67]. We use the labeled queries from AOL provided by [68] (http://grepgsadetsky.com/aol-data) for evaluation. We consider queries from five classes: *Business*, *Computer*, *Entertainment*, *Health*, *and Sports* which are shown in both training and test datasets. We form ten binary classification tasks for query classification [64]. The more detailed information can be seen in Table 3.11.

Sentiment Reviews

The data of sentiment domain adaptation [69] consist of Amazon product reviews for four different product types, including books, DVDs, electronics, and kitchen appliances. Each review consists of a rating with scores ranging from 0 to 5, a reviewer name and location, a product name, a review title and date, and the review text. Reviews with ratings higher than three are labeled as positive and reviews with ratings lower than three are labeled as negative, the rest are discarded since the polarity of these reviews is ambiguous. The details of the data in different domains are summarized in Table 3.11. The experimental settings are the same as in [69]. To study the performance of our methods in this task, we construct 12 pairs of cross-domain sentiment classification tasks as shown in Table 3.6, e.g., we use the reviews from domain A as the training data and then predict the sentiment of the reviews in the domain B.

3.5.3.2 Comparing with the Related Methods

Tables 3.12, 3.13, and 3.14 and Fig. 3.11 show the means and standard deviations of classification accuracies of different methods on the above domain adaptation transfer learning tasks, respectively. From these results, we can make several interesting observations as follows:

Table 3.12 Means and standard deviations (%) of classification accuracies (ACC) of all methods on the 20Newsgroups, Reuters datasets, and email spam filtering datasets

Datasets		Methods								
		SVM	TSVM	CDCS	LWE	LMPROJ	DTSVM	KMM	DAKSVM	μ-DAKSVM
Reuters	1	80.20*	81.84*	88.5	83.42*	84.63*	**88.77***	85.43*	87.64	88.13
		(±0.45)	(±1.26)	(±4.32)	(±2.01)	(±2.82)	(±**2.14**)	(±3.02)	(±0.42)	(±0.50)
	2	71.35*	75.80*	73.90*	69.70*	80.20*	81.52	79.38*	79.97	**82.64**
		(±2.18)	(±1.72)	(±2.14)	(±0.08)	(±1.16)	(±0.56)	(±0.01)	(±1.06)	(±**0.00**)
	3	65.36*	69.8	64.00*	68.52*	70.80*	74.12*	72.19*	74.4	**74.9**
		(±1.67)	(±0.46)	(±0.32)	(±1.49)	(±1.38)	(±3.26)	(±1.13)	(±**0.42**)	
20NG	4	72.53*	76.75*	69.80*	**85.24**	82.52*	83.52*	78.11*	84.68	85.02
		(±3.42)	(±0.54)	(±0.48)	(±**1.81**)	(±0.86)	(±2.74)	(±2.80)	(±0.63)	(±1.08)
	5	70.10*	73.40*	**82.92**	78.60*	79.30*	80.5*	79.11	82.36	82.67
		(±2.62)	(±2.02)	(±**1.06**)	(±0.34)	(±2.76)	(±2.82)	(±1.00)	(±0.15)	(±0.30)
	6	75.40*	83.9	64.00*	87.2	86.34*	**90.23**	87.52*	88.81	88.81
		(±0.51)	(±1.14)	(±3.1)	(±2.10)	(±3.10)	(±**0.82**)	(±2.69)	(±1.74)	(±0.6)
	7	78.00*	81.20*	70.84*	75.32*	84.68	84.84	81.47*	85.12	**85.37**
		(±0.04)	(±0.06)	(±1.62)	(±0.47)	(±2.11)	(±1.49)	(±3.27)	(±0.04)	(±**0.80**)
	8	83.80*	85.24*	82.72*	88.3	85.40*	91.76	89.46*	89.58	**91.84**
		(±1.13)	(±0.18)	(±0.34)	(±1.08)	(±1.08)	(±1.68)	(±2.07)	(±0.3)	(±**0.10**)
	9	92.70*	88.74*	90.20*	94.00*	93.43*	94.13	92.50*	96.72	**96.78**
		(±0.40)	(±0.70)	(±1.16)	(±2.06)	(±0.79)	(±0.4)	(±0.78)	(±0.60)	(±**0.20**)
Spam filtering	10	96.08	96.21	83.28*	93.51*	93.21*	96.89	96.21	96.49	**97.19**
		(±0.10)	(±0.22)	(±0.20)	(±0.40)	(±0.52)	(±0.1)	(±0.10)	(±0.03)	(±**0.06**)
	11	96.89	97	92.14*	98.74	94.0*	**97.65**	97.13	97.25	97.25
		(±0.0)	(±0.10)	(±1.02)	(±0.4)	(±0.00)	(±**0.20**)	(±0.05)	(±0.4)	(±0.41)
	12	91.7*	91.80*	90.02*	88.78*	88.79*	94.5	91.8*	93.2	**93.85**
		(±0.6)	(±0.3)	(±0.3)	(±0.01)	(±0.24)	(±0.60)	(±0.00)	(±0.20)	(±**0.10**)

Each result in the table is best among all the results obtained by using different parameters

The performance of μ-DAKSVM is statistically significant compared with other seven classifiers at p-value ≤ 0.05

Table 3.13 Means and standard deviations (%) of classification accuracies (ACC) of all methods on the web query dataset

	Datasets									
	Web query data									
Methods	B–C	B–H	B–H	B–S	C–E	C–H	C–S	E–H	E–S	H–S
SVM	82.52* (±1.14)	85.93* (±0.03)	90.94* (±2.67)	87.25 (±0.44)	87.34 (±4.12)	93.19* (±3.10)	84.68* (±2.01)	92.55* (±0.4)	81.59* (±0.2)	93.45* (±1.04)
TSVM	82.64 (±0.2)	85.42* (±0.2)	91.19* (±1.06)	82.60* (±2.50)	83.35* (±1.52)	89.70* (±2.00)	92.01* (±3.24)	93.11 (±0.03)	77.93* (±2.44)	80.84* (±1.60)
CDCS	84.34 (±2.74)	82.86* (±0.33)	96.44 (±3.16)	85.40* (±2.22)	78.70* (±0.50)	91.28* (±1.60)	89.40* (±0.62)	92.76* (±1.10)	**85.55** (±**0.00**)	91.25* (±0.40)
LWE	83.26* (±0.74)	86.72 (±3.02)	93.20* (±1.27)	82.56* (±0.80)	85.80 (±2.28)	93.66 (±1.40)	84.20* (±4.56)	94.40* (±0.72)	79.46* (±3.31)	94.71* (±0.20)
LMPROJ	84.68 (±0.4)	86.48* (±2.33)	94.82* (±1.08)	85.78* (±0.86)	88.52* (±3.26)	92.00* (±1.09)	86.12* (±4.20)	95.38* (±2.02)	82.70* (±1.17)	95.00* (±2.48)
DTSVM	84.22* (±3.12)	**88.36*** (±**1.32**)	96.61 (±0.08)	86.39* (±2.16)	**90.15** (±**0.26**)	94.08 (±0.40)	95.16 (±0.82)	93.08* (±0.00)	83.29* (±0.42)	97.33* (±1.28)
KMM	83.92* (±0.74)	86.86 (±0.30)	95.12* (±0.6)	81.22* (±3.06)	85.52* (±1.90)	93.00* (±0.01)	84.82* (±2.70)	95.11 (±0.02)	80.10* (±0.2)	96.32* (±1.42)
DAKSVM	86.36 (±0.40)	87.82 (±0.56)	98.23 (±1.02)	90.46 (±0.03)	88.78 (±0.00)	95.10 (±0.6)	96.94 (±0.90)	96.81 (±0.12)	83.20 (±0.62)	**98.27** (±**0.2**)
μ-DAKSVM	**86.76** (±**0.00**)	87.87 (±0.2)	**98.75** (±**0.4**)	**91.02** (±**1.01**)	88.78 (±0.00)	**95.42** (±**0.80**)	**97.54** (±0.05)	**96.81** (±0.3)	84.31 (±0.5)	**98.27** (±**0.4**)

Each result in the table is best among all the results obtained by using different parameters

*The performance of μ-DAKSVM is statistically significant compared with other seven classifiers at p-value ≤ 0.05

Table 3.14 Means and standard deviations (%) of classification accuracies (ACC) of all methods on the sentiment reviews dataset

	Datasets											
	Sentiment reviews data											
Methods	D–B	E–B	K–B	B–D	E–D	K–D	B–E	D–E	K–E	B–K	D–K	E–K
SVM	71.29* (±2.14)	67.89* (±1.32)	67.72* (±0.74)	75.69* (±3.22)	67.81* (±0.50)	71.77* (±4.22)	68.83* (±3.80)	69.31* (±0.62)	78.77* (±0.34)	73.13* (±2.18)	73.68* (±4.18)	80.46 (±0.22)
TSVM	71.52* (±0.72)	68.27 (±0.40)	69.96* ±2.18)	74.84* (±2.50)	64.51* (±3.58)	71.63* (±3.04)	71.13* (±1.48)	70.52* (±1.22)	80.16 (±0.30)	76.14* (±2.54)	76.57* (±0.96)	80.99 (±0.12)
CDCS	69.33* (±0.44)	72.30 (±1.05)	74.34 (±0.60)	82.30* (±0.14)	73.00* (±0.54)	80.66 (±0.82)	76.80 (±0.40)	70.36* (±2.08)	84.07* (±0.41)	79.02 (±0.36)	80.58* (±1.16)	**86.47** (±**3.05**)
LWE	74.54* (±1.40)	69.10 (±0.20)	77.60 (±1.01)	78.70* (±0.82)	69.40* (±2.60)	78.21* (±1.36)	78.90* (±0.08)	75.67 (±1.15)	84.73 (±0.68)	78.79* (±1.06)	83.19* (±0.01)	79.89 (±2.76)
LMPROJ	76.71* (±0.61)	71.29* (±1.34)	75.80* (±1.80)	83.65* (±1.74)	73.20* (±0.20)	81.14* (±1.52)	75.20 (±0.32)	77.30 (±2.26)	81.66* (±0.92)	80.04 (±0.00)	85.33 (±0.44)	81.38* (±0.80)
DTSVM	78.41 (±0.30)	72.58 (±0.40)	**78.88** (±**0.66**)	**86.69** (±**1.27**)	74.62* (±0.25)	82.49* (±0.50)	79.21 (±1.14)	77.54 (±0.05)	83.97 (±0.20)	**81.72** (±**2.02**)	87.07 (±0.84)	82.56* (±1.48)
KMM	74.02* (±1.31)	69.58* (±2.04)	77.18 (±0.56)	81.79* (±0.07)	73.16* (±0.05)	82.06* (±1.12)	70.88* (±2.06)	75.82* (±0.42)	83.50 (±0.20)	79.94* (±1.70)	84.27* (±1.46)	80.07* (±0.40)
DAKSVM	78.54 (±0.40)	73.09 (±0.01)	77.82 (±1.06)	86.34 (±0.6)	78.02 (±0.00)	83.07 (±0.56)	78.79 (±0.02)	79.21 (±0.72)	**84.25** (±**0.2**)	81.33 (±0.43)	88.14 (±0.6)	84.57 (±0.05)
μ-DAKSVM	**78.72** (±**0.25**)	**73.74** (±**0.56**)	78.03 (±0.50)	86.34 (±0.71)	**78.43** (±**0.34**)	**84.21** (±**0.20**)	**79.36** (±**0.18**)	**79.54** (±**0.00**)	**84.25** (±**0.3**)	81.68 (±0.08)	**89.36** (±**0.82**)	84.89 (±0.25)

Each result in the table is best among all the results obtained by using different parameters

*The performance of μ-DAKSVM is statistically significant compared with other seven classifiers at p-value ≤ 0.05

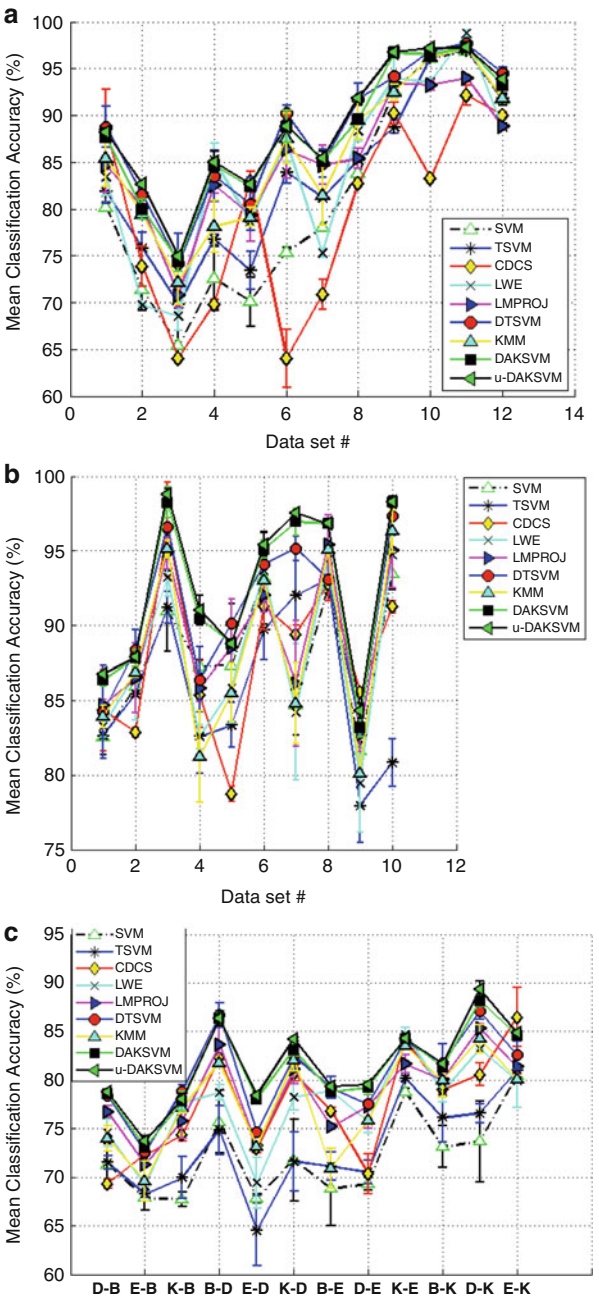

Fig. 3.11 Means and standard deviations (%) of classification accuracies (*ACC*) of all methods on text datasets. **a** Text datasets: Reuters, 20Newsgroups and mail spam filtering; **b** web query dataset; **c** sentiment classification dataset

1. From Tables 3.12, 3.13, and 3.14, we can see that our method achieves very promising result. The major limitation of LMPROJ, DTSVM, and KMM is that they only consider the first-order statistics and thus cannot well generalize their result. However, since our methods definitely consider both the second-order and the first-order statistics between the source and target domains, it yields better generalization capability. It can be observed that our method significantly outperforms other methods. These empirical results again show that considering second-order statistics as well as first-order statistics can help us improve the domain adaptation performance.
2. SVM and TSVM have the worst performance on almost all learning tasks compared to other classifiers, which is consistent with the experimental results of the above toy datasets. Though obtaining better classification on both 20Newsgroup and Reuters datasets, TSVM exhibits its worse classification performance on two web text classification tasks than other methods. It is worth noting that we obtain a little better results for SVM and TSVM than those typically reported in the previous literature on the same datasets used in our trials. This is because in order to make our comparison fair we reported the best results over a set of parameters for SVM and instead of selecting a default parameter on the training data to be performed.
3. In Tables 3.12, 3.13, and 3.14 and Fig. 3.11, we can also observe that although seven methods, i.e., CDCS, LWE, LMPROJ, DTSVM, KMM, DAKSVM, and its variation μ-DAKSVM, exhibit comparable classification capability on all text datasets, the proposed method DAKSVM and its variation μ-DAKSVM always keep significantly high classification accuracy in most cases, which implies that it is more stable than other methods, particularly on two web text classification datasets such as web query and sentiment reviews datasets.
4. The results in Tables 3.12, 3.13, and 3.14 and Fig. 3.11 also show that the proposed method DAKSVM and its variation μ-DAKSVM perform relatively better than MMD-based methods LMPROJ and KMM in almost all datasets, which justifies that the only emphasis on minimizing distribution mean discrepancy between both domains is far from sufficiency for domain adaptation transfer learning. Hence, we should introduce more underlying information, such as distribution scatter discrepancy minimization, into the regularization framework of the classifier to further enhance the classification performance. Besides, it is worth mentioning that DTSVM also obtains fairly robust performance on almost all datasets by adopting multiple kernel learning scheme. A possible explanation is that multiple kernel learning skill can improve learning capability for DAL.
5. μ-DAKSVM keeps obviously superior capability over DAKSVM in classification accuracy for almost all these datasets, which demonstrates that parameter μ can be used to enhance the generalization capability of DAKSVM. Therefore, we use μ-DAKSVM instead of DAKSVM for the performance evaluation hereafter.
6. In order to verify whether the proposed methods are significantly better than the other methods, we also performed the paired two-tailed t-test [70] on the classification results of the 10 runs to calculate the statistical significance of the proposed method μ-DAKSVM. The smaller the p-value, the more significant

the difference of the two average results is, and a p-value of 0.05 is a typical threshold which is considered to be statistically significant. Thus, in Tables 3.12, 3.13, and 3.14, if the p-value of each dataset is less than 0.05, the corresponding results will be denoted "*." Therefore, as shown in Tables 3.12, 3.13, and 3.14, we can clearly find that the proposed method μ-DAKSVM significantly outperforms other methods in most datasets.

3.5.4 Multi-Class Face Recognition Datasets

3.5.4.1 Dataset Settings

In this subsection, in order to evaluate the effectiveness of the proposed methods on multi-class classification problems, we investigate the performance of the proposed algorithms LSDAKSVM and μ-DAKSVM for face recognition on two benchmarking Yale and ORL face databases. The Yale face database was constructed at the Yale Center for Computation Vision and Control. There are 165 images about 15 individuals in this database where each person has 11 images. The images demonstrate face variations under lighting condition (left-right, center-light, right-light) and facial expression (normal, happy, sad, sleepy, surprised and wink) with or without glasses. Each image is cropped to be the size of 32×32 in our experiment. We randomly select 8 images of each individual to construct the source domain dataset; the ORL database contains 400 images grouped into 40 distinct subjects with 10 different images for each. The images are captured at different times, and for some subjects, the images may vary in facial expressions and facial details. All the images are taken against a dark homogeneous background with the tolerance for some side movement of about 20. The original images are all sized 112×92 pixels with 256 gray levels per pixel, which are further down-sampled into 32×32 pixels in our experiment. We randomly select eight images of each individual to construct the source domain training set. Figure 3.12a, c shows the cropped images of one person in Yale and ORL face databases, respectively.

The target datasets are generated by rotating anticlockwise the original source domain dataset three times by 10°, 30°, and 50°, respectively. Due to rotation, source and target-domain data exhibit different distributions. Particularly, the greater the rotation angle is, the more complex the resulting domain adaptation problem becomes. Thus we construct three face domain adaptation transfer learning problems for each face database. Figure 3.12b, d shows the face samples with rotation angle 10°, respectively.

3.5.4.2 Comparing with the Related Methods

We test the performance of LSDAKSVM and μ-DAKSVM in comparison with CDCS, LWE, DTSVM, and LMPROJ. In order to do a comprehensive comparison,

Fig. 3.12 Face examples from the face databases Yale and ORL. **a** Yale faces for an object; **b** Yale faces for an object with rotation angle 10°; **c** ORL faces for an object; **d** ORL faces for an object with rotation angle 10°

we also perform the baseline method LS-SVM for face recognition with different distributions. For the above multi-class classification tasks, μ-DAKSVM, CDCS, LWE, LS-SVM, DTSVM, and LMPROJ adopt OAO multi-class separation strategy to finish the corresponding multi-class classification tasks. For each evaluation, ten rounds of experiments are repeated with randomly selected training data, and the average result is recorded as the final classification accuracy in Table 3.7. Several attractive insights can be obtained from these results as follows:

1. The overall accuracy of LS-SVM is lower than any other classifier on all DAL tasks, which is consistent with SVM.
2. With the increase of rotation angle, the classification performance of all classifiers descends gradually. However, LSDAKSVM seems to decrease more slowly than other methods. Exceptionally, CDCS and DTSVM exhibit competitive performance to some extent compared to other methods, particularly on more complex datasets.
3. As shown in Table 3.15, we can observe that the LSDAKSVM method delivers more stable results across all the datasets and is competitive as the best method for the majority of all the other datasets. It obtains the best classification accuracy more times than any other method. Hence, as discussed in the above section, LSDAKSVM possesses overall DAL advantages over other methods in the sense of both computational complexity and classification accuracy.
4. Table 3.15 also shows that although LSDAKSVM seems to have overall advantage over μ-DAKSVM in classification accuracy, μ-DAKSVM is actually considerably comparable to LSDAKSVM.

Table 3.15 Means and standard deviations (%) of classification accuracies (ACC) of all methods on Yale and ORL with different rotation angles

Faces data		Method						
		LS-SVM	LMPROJ	LWE	CDCS	DTSVM	LSDAKSVM	μ-DAKSVM
Yale	10°	61.78 (±3.45)	68.45 (±0.56)	63.78 (±2.48)	62.47 (±1.10)	68.77 (±0.54)	**70.24** (±0.24)	69.93 (±0.2)
	30°	58.37 (±2.74)	64.13 (±1.07)	61.66 (±1.01)	60.70 (±0.4)	67.28 (±2.34)	**66.47** (±0.5)	66.2 (±0.6)
	50°	52.29 (±2.12)	62.08 (±1.18)	58.78 (±0.41)	60.20 (±0.34)	**65.63** (±1.14)	63.00 (±0.4)	63.7 (±0.34)
ORL	10°	76.30 (±1.00)	85.94 (±1.40)	80.90 (±0.6)	84.64 (±0.2)	84.84 (±0.00)	**86.28** (±0.04)	84.18 (±0.44)
	30°	70.72 (±3.04)	82.00 (±0.76)	79.33 (±1.20)	**83.71** (±2.10)	83.19 (±0.01)	83.10 (±1.06)	83.40 (±0.00)
	50°	65.70 (±0.62)	78.65 (±0.20)	72.22 (±3.54)	79.91 (±1.03)	80.01 (±1.14)	**81.46** (±0.02)	78.10 (±1.68)

The best results among all the results obtained with different parameters are listed in the table

3.6 Conclusions

In this chapter, we propose one inductive learning approach and one transductive learning approach based on support vector learning, respectively. On the one hand, the proposed inductive transfer learning method, i.e., KL-TSK-FS, is more adaptive to the situations where the data are only partially available from the target domain while some useful knowledge of the source domains is available. Besides, the proposed method is distinctive in preserving data privacy as only the knowledge (e.g., the corresponding model parameters) rather than the data of the source domain is adopted. On the other hand, the proposed transductive transfer learning method DAKSVM and its two extensions indeed inherit the potential advantages of classical TSVMs and MMD-based methods and are further extended to DAL. As a novel large margin domain adaptation classifier, the proposed methods can reduce the distribution gap between different domains in an RKHS as much as possible, since they effectively integrate the large margin learner with the proposed GPMDD metric, in which both the distribution mean discrepancy and the distribution scatter discrepancy on RKHS embedding domain distributions are *simultaneously* considered.

References

1. Pan, S.J., Yang, Q.: A survey on transfer learning. IEEE Trans. Knowl. Data Eng. **22**(10), 1345–1359 (2010)
2. Deng, Z.H., Choi, K.S., Chung, F.L., et al.: Scalable TSK fuzzy modeling for very large datasets using minimal-enclosing-ball approximation. IEEE Trans. Fuzzy Syst. **19**(2), 210–226 (2011)
3. Liao, X., Xue, Y., Carin, L.: Logistic regression with an auxiliary data source. In: Proceedings of 21st International Conference Machine Learning, pp. 505–512 (2005)
4. Huang, J., Smola, A., Gretton, A., et al.: Correcting sample selection bias by unlabeled data. In: Proceedings of 19th Annual Conference Neural Information Processing Systems, pp. 601–608 (2007)
5. Bickel, S., Brückner, M., Scheffer, T.: Discriminative learning for differing training and test distributions. In: Proceedings of 24th International Conference Machine Learning, pp. 81–88 (2007)
6. Sugiyama, M., Nakajima, S., Kashima, H., et al.: Direct importance estimation with model selection and its application to covariate shift adaptation. In: Proceedings of 20th Annual Conference Neural Information Processing Systems, Bangkok (December 2008)
7. Lawrence, N.D., Platt, J.C.: Learning to learn with the informative vector machine. In: Proceedings of 21st International Conference Machine Learning (July 2004)
8. Schwaighofer, A., Tresp, V., Yu, K.: Learning Gaussian process kernels via hierarchical Bayes. In: Proceedings 17th Annual Conference Neural Information Processing Systems, pp. 1209–1216 (2005)
9. Gao, J., Fan, W., Jiang, J., et al.: Knowledge transfer via multiple model local structure mapping. In: Proceedings of 14th ACM SIGKDD International Conference Knowledge Discovery and Data Mining, pp. 283–291 (August 2008)

10. Mihalkova, L., Huynh, T., Mooney, R.J.: Mapping and revising Markov logic networks for transfer learning. In: Proceedings of 22nd Association for the Advancement of Artificial Intelligence (AAAI) Conference on Artificial Intelligence, pp. 608–614 (July 2007)
11. Mihalkova, L., Mooney, R.J.: Transfer learning by mapping with minimal target data. In: Proceedings of Association for the Advancement of Artificial Intelligence (AAAI'08) Workshop Transfer Learning for Complex Tasks (July 2008)
12. Davis, J., Domingos, P.: Deep transfer via second-order Markov logic. In: Proceedings of Association for the Advancement of Artificial Intelligence (AAAI'08) Workshop Transfer Learning for Complex Tasks (July 2008)
13. Pan, S.J., Tsang, I.W., Kwok, J.T., et al.: Domain adaptation via transfer component analysis. IEEE Trans. Neural Netw. **22**(2), 199–210 (2011)
14. Duan, L.X., Xu, D., Tsang, I.W.: Domain adaptation from multiple sources: a domain-dependent regularization approach. IEEE Trans. Neural Netw. Learn. Syst. **23**(3), 504–518 (2012)
15. Duan, L.X., Tsang, I.W., Xu, D.: Domain transfer multiple kernel learning. IEEE Trans. Pattern Anal. Mach. Intell. **34**(3), 465–479 (2012)
16. Dai, W., Yang, Q., Xue, G., et al.: Self-taught clustering. In: Proceedings of 25th International Conference Machine Learning, pp. 200–207 (July 2008)
17. Jiang, W.H., Chung, F.L.: Transfer spectral clustering. In: Proceedings of European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD), Bristol (24–28 September 2012)
18. Wang, Z., Song, Y., Zhang, C.: Transferred dimensionality reduction. In: Proceedings of European Conference Machine Learning and Knowledge Discovery in Databases (ECML/PKDD'08), pp. 550–565 (September 2008)
19. Gong, B., Shi, Y., Sha, F., et al.: Geodesic flow kernel for unsupervised domain adaptation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2066–2073 (2012)
20. Yang, P., Tan, Q., Ding, Y.: Bayesian task-level transfer learning for non-linear regression. In: Proceedings of International Conference on Computer Science and Software Engineering, pp. 62–65 (2008)
21. Borzemski, L., Starczewski, G.: Application of transfer regression to TCP throughput prediction. In: Proceedings of First Asian Conference on Intelligent Information and Database Systems, pp. 28–33 (2009)
22. Mao, W., Yan, G., Bai, J., et al.: Regression transfer learning based on principal curve. Lect. Notes Comput. Sci. **6063**, 365–372 (2010)
23. Liu, J., Chen, Y., Zhang, Y.: Transfer regression model for indoor 3D location estimation. Lect. Notes Comput. Sci. **5916**, 603–613 (2010)
24. Pardoe, D., Stone, P.: Boosting for regression transfer. In: Proceedings of International Conference on Machine Learning, pp. 863–870 (2010)
25. Dai, W., Yang, Q., Xue, G., et al.: Boosting for transfer learning. In: Proceedings of 24th International Conference on Machine Learning, pp. 193–200 (June 2007)
26. Wu, P., Dietterich, T.G.: Improving SVM accuracy by training on auxiliary data sources. In: Proceedings of 21st International Conference on Machine Learning (July 2004)
27. Evgeniou, T., Pontil, M.: Regularized multi-task learning. In: Proceedings of 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 109–117 (August 2004)
28. Qduanz, B., Huan, J.: Large margin transductive transfer learning. In: Proceedings of 18th ACM conference on Information and knowledge management (CIKM), pp. 1327–1336. ACM, New York (2009)
29. Duan, L., Tsang, I.W., Xu, D., et al.: Domain transfer SVM for video concept detection. In: Proceedings IEEE International Conference on Computer Vision and Pattern Recognition, pp. 1375–1381 (2009)

30. Bruzzone, L., Marconcini, M.: Domain adaptation problems: a DASVM classification technique and a circular validation strategy. IEEE Trans. Pattern Anal. Mach. Intell. **32**(5), 770–787 (2010)
31. Ben-David, S., Blitzer, J., Crammer, K., et al.: Analysis of representations for domain adaptation. In: NIPS (2007)
32. Gretton, A., Harchaoui, Z., Fukumizu, K., Sriperumbudur, B.: A fast, consistent Kernel two-sample test. In: Advances in Neural Information Processing Systems, vol. 22, pp. 673–681. MIT, Cambridge (2010)
33. Leski, J.: TSK-fuzzy modeling based on ε-insensitive learning. IEEE Trans. Fuzzy Syst. **13**(2), 181–193 (2005)
34. Wang, L.X.: Adaptive fuzzy systems and control: design and stability analysis. Prentice-Hall, Upper Saddle River (1994)
35. Jang, J.S.R., Sun, C.T., Mizutani, E.: Neuro-fuzzy and soft-computing. Prentice-Hall, Upper Saddle River (1997)
36. Chen, B., Lam, W., Tsang, I.W., et al.: Location and scatter matching for dataset shift in text mining. In: Proceedings of the IEEE International Conference on Data Mining (IEEE ICDM 2010), Sydney (December 2010)
37. Huang, J., Smola, A., Gretton, A., et al.: Correcting sample selection bias by unlabeled data. In: Proceedings of Twentieth Annual Conference on Neural Information Processing Systems (2006)
38. Takagi, T., Sugeno, M.: Fuzzy identification of systems and its application to modeling and control. IEEE Trans. Syst. Man Cybern. **15**(1), 116–132 (1985)
39. Mamdani, E.H.: Application of fuzzy logic to approximate reasoning using linguistic synthesis. IEEE Trans. Comput. **C-26**(12), 1182–1191 (1977)
40. Azeem, M.F., Hanmandlu, M., Ahmad, N.: Generalization of adaptive neural-fuzzy inference systems. IEEE Trans. Neural Netw. **11**(6), 1332–1346 (2000)
41. Deng, Z.H., Choi, K.S., Chung, F.L., et al.: Enhanced soft subspace clustering integrating within-cluster and between-cluster information. Pattern Recognit. **43**(3), 767–781 (2010)
42. Jang, J.S.R.: ANFIS: adaptive-network-based fuzzy inference systems. IEEE Trans. Syst. Man Cybern. **23**(3), 665–685 (1993)
43. Tsang, I.W., Kwok, J.T., Zurada, J.M.: Generalized core vector machines. IEEE Trans. Neural Netw. **17**(5), 1126–1140 (2006)
44. Deng, Z.H., Jiang, Y.Z., Chung, F.L., et al.: Knowledge-leverage based fuzzy system and its modeling. IEEE Trans. Fuzzy Syst. (2012). doi:10.1109/TFUZZ.2012.2212444
45. Fan, R.E., Chen, P.H., Lin, C.J.: Working Set selection using second order information for training support vector machines. J. Mach. Learn. Res. **6**, 1889–1918 (2005)
46. Hofmann, T., Schölkopf, B., Smola, A.J.: Kernel methods in machine learning. Ann. Stat. **36**, 1171–1220 (2007)
47. Sriperumbadur, B.K., Fukumizu, K., Gretton, A., et al.: Kernel choice and classifiability for RKHS embeddings of probability distributions. In: Advances in Neural Information Processing Systems, vol. 22, pp. 1750–1758. MIT, Cambridge (2010)
48. Smola, A.J., Gretton, A., Song, L., et al.: A Hilbert space embedding for distributions. In: Proceedings of 18th International Conference on Algorithmic Learn. Theory, Sendai, pp. 13–31 (October 2007)
49. Gretton, A., Harchaoui, Z., Fukumizu, K., et al.: A fast, consistent kernel two-sample test. In: Advances in Neural Information Processing Systems, vol. 22, pp. 673–681. MIT, Cambridge (2010)
50. Borgwardt, K.M., Gretton, A., Rasch, M.J., et al.: Integrating structured biological data by kernel maximum mean discrepancy. Bioinformatics (ISMB) **22**(14), e49–e57 (2006)
51. Sriperumbudur, B.K., Gretton, A., Fukumizu, K., et al.: Hilbert space embeddings and metrics on probability measures. J. Mach. Learn. Res. **11**(3), 1517–1561 (2010)
52. Suykens, J.A.K., Vandewalle, J.: Least squares support vector machine classifiers. Neural Process. Lett. **9**(3), 293–300 (1999)

53. Wu, Y., Liu, Y.: Robust truncated hinge loss support vector machines. J. Am. Stat. Assoc. **102**(479), 974–983 (2007)
54. Scholkopf, B., Herbrich, R., Smola, A.J.: Generalized representer theorem. In: Proceedings of COLT'2001, pp. 416–426. Springer, Amsterdam (2001)
55. Szedmak, S., Shawe-Taylor, J.: Multiclass learning at one-class complexity. Technical Report No. 1508, School of Electronics and Computer Science, Southampton (2005)
56. Schölkopf, B., Smola, A.J., Williamson, R., et al.: New support vector algorithms. Neural Comput. **12**(5), 1207–1245 (2000)
57. Chang, C.C., Lin, C.J.: Training *v*-support vector classifiers: theory and algorithms. Neural Comput. **13**(9), 2119–2147 (2001)
58. Sindhwani, V., Belkin, M., Niyogi, P.: The geometric basis of semi-supervised learning. In: Chapelle, O., Schölkopf, B., Zien, A. (eds.) Semi-Supervised Learning. MIT, Cambridge (2006)
59. Tsang, I.W., Kwok, J.T.: Very large scale manifold regularization using core vector machines. In: NIPS 2005 Workshop on Large Scale Kernel Machines (2005)
60. Juang, C.F., Chiu, S.H., Shiu, S.J.: Fuzzy system learned through fuzzy clustering and support vector machine for human skin color segmentation. IEEE Trans. Syst. Man Cybern. **37**(6), 1077–1087 (2007)
61. Juang, C.F., Hsieh, C.D.: TS-fuzzy system-based support vector regression. Fuzzy Sets Syst. **60**(17), 2486–2504 (2009)
62. Abril, L.G., Angulo, C., Velasco, F., et al.: A note on the bias in SVMs for multi classification. IEEE Trans. Neural Netw. **19**(4), 723–725 (2008)
63. Ling, X., Dai, W., Xue, G., et al.: Spectral domain transfer learning. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, New York (2008)
64. Xiang, E.W., Cao, B., Hu, D.H., et al.: Bridging domains using world wide knowledge for transfer learning. IEEE Trans. Knowl. Data Eng. **22**(6), 770–783 (2010)
65. Dai, W., Xue, G.R., Yu, Y.: Co-clustering based classification for out-of-domain documents. In: Proceedings of the Thirteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Jose, pp. 210–219. ACM, New York (August 2007)
66. Bickel, S.: ECML-PKDD discovery challenge 2006 overview. In: Proceedings of ECML/PKDD on Discovery Challenge Workshop (2006)
67. Phan, X.H., Nguyen, M.L., Horiguchi, S.: Learning to classify short and sparse text & web with hidden topics from large-scale data collections. In: Proceedings of 17th International World Wide Web Conference (WWW'08), pp. 91–100 (April 2008)
68. Beitzel, S.M., Jensen, E.C., Frieder, O., et al.: Improving automatic query classification via semi-supervised learning. In: Proceedings of Fifth IEEE International Conference Data Mining (ICDM'05), pp. 42–49 (November 2005)
69. Blitzer, J., Dredze, M., Pereira, F., et al.: Boom-boxes and blenders: domain adaptation for sentiment classification. In: Proceedings of 45th Annual Meeting of the Association for Computational Linguistics (ACL'07), pp. 440–447 (June 2007)
70. Alpaydin, E.: Introduction to machine learning. MIT, Cambridge (2004)