Applied Intelligence (2021) 51:8451–8465
https://doi.org/10.1007/s10489-021-02194-9

Multi-source fast transfer learning algorithm based on support vector machine

Peng Gao [1,2] · Weifei Wu [1] · Jingmei Li [1]

Accepted: 4 January 2021 / Published online: 6 April 2021

Abstract
Knowledge in the source domain can be used in transfer learning to help train and classification tasks within the target domain with fewer available data sets. Therefore, given the situation where the target domain contains only a small number of available unlabeled data sets and multi-source domains contain a large number of labeled data sets, a new Multi-source Fast Transfer Learning algorithm based on support vector machine(MultiFTLSVM) is proposed in this paper. Given the idea of multi-source transfer learning, more source domain knowledge is taken to train the target domain learning task to improve classification effect. At the same time, the representative data set of the source domain is taken to speed up the algorithm training process to improve the efficiency of the algorithm. Experimental results on several real data sets show the effectiveness of MultiFTLSVM, and it also has certain advantages compared with the benchmark algorithm.

Keywords Multi-source transfer learning · Support vector machine · Classification

1 Introduction

As one of the fastest-growing technical fields today, machine learning is also the core of artificial intelligence and data science. It solves the problem of how to automatically improve computers through experience [1]. Machine learning has been widely applied in intrusion detection [2, 3], computer vision [4], data mining [5], text classification [6], spam detection [7] and pattern recognition [8], and other fields. However, the further development of machine learning in these fields is restricted by the shortcomings of traditional machine learning methods. Usually, two basic assumptions in traditional machine learning should be met for the current traditional machine learning classification tasks: there are enough data samples in the training data set to train a high-precision classifier; training and test data comes from the same feature space and has the same distribution. For practical application, training and test data usually come from different domains, with differing marginal probabilities, or conditional probabilities, so the data distribution is also different. When the distribution is changed, most machine learning algorithms need to re-collect training data. In many real-world applications, the cost of re-collecting training data and reconstructing the model is very expensive, or even impossible [9].

In this case, transferring learning between learning task domains is desirable. The research motivation of transfer learning is that previously learned knowledge can be used by people to better solve new problems [9, 10], and its purpose is to build a model for the target domain by using labeled information in another related domain (source domain). Therefore transfer learning is defined in Wikipedia as "Transfer learning is a new machine learning method that takes existing knowledge to solve problems in different but similar fields. It no longer follows the two basic assumptions in traditional machine learning. Instead, the existing knowledge is transferred to solve the problem of only a small amount of labeled sample data in the target field [11]. The difference between traditional machine learning and transfer learning is shown in Fig. 1. It can be seen from Fig. 1a that each learning task in traditional machine learning starts from zero, while in (b) the knowledge from previous learning tasks

✉ Jingmei Li
lijingmei@hrbeu.edu.cn

Peng Gao
gaopeng1979@hrbeu.edu.cn

Weifei Wu
wuweifei@hrbeu.edu.cn

[1] College of Computer Science and Technology, Harbin Engineering University, Harbin, China

[2] Technology Development Cente, Heilongjiang Broadcasting Station, Harbin, China

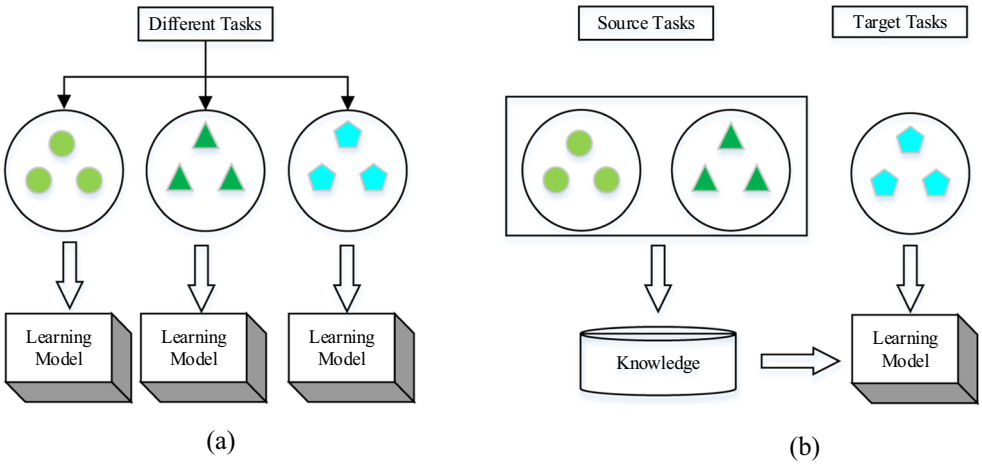

Fig. 1 Difference between transfer learning and traditional machine learning

can be transferred to the current target learning task by using transfer learning. The representative algorithms related to transfer learning research are as follows: Gao et al. [12] proposed an integrated framework of a locally weighted combination of multiple models (LWE). LWE can integrate the advantages of various algorithms and label learning in multiple training domains into one model and dynamically assign weights according to the predictive ability of each model per each instance. Pan et al. [13] proposed a new learning method for domain adaptation transfer components analysis(TCA). TCA tries to learn several transfer components across domains in the Reproducing Kernal Hilbert Spaces (RKHS) by the maximum mean discrepancy (MMD) error rate. In the subspace spanned by these components, data attributes are preserved, and data distribution in different domains is close to each other. An adaptive regularization transfer learning algorithm (ARTL) is based on the structural risk minimization principle and a regularization theory that was proposed by Long et al. [14]. Li et al. [15] implemented the transfer learning algorithm RankRE-TL based on the transfer learning mechanism of knowledge used and the error set selection method with rank reduction. A new SVM-based model transfer method was proposed in [16], in which a large boundary classifier is trained on the labeled target sample and adjusted by an offset of the source classifier. This method is called Heterogeneous Max-margin Classifier Adaptation Method (HMCA). Xie et al. [17] proposed a new method of supervised domain adaptation by dual support vector machines, called adaptive dual support vector machines for an aggregation domain.

However, in the above transfer learning algorithm, only the knowledge in a source domain is transferred to the target domain. In transfer learning, the performance of the target classifier largely depends on the correlation between the source domain and the target domain. If the correlation between the target domain and the source domain is strong, it will help to improve the learning effect in the target domain, otherwise, it will reduce the learning effect of the target domain, leading to the phenomenon of negative transfer [12]. One strategy to reduce this negative transfer is to import knowledge from multi-sources to increase the chance of discovering a source domain closely related to the target domain [18]. The typical research work for a multi-source domain transfer learning algorithm is as follows: A novel task-based instance transfer enhancement technology TransferBoost was proposed in [19], which selectively transfers knowledge from the source domain to the target task. It has been enhanced at both the instance level and the task level. The source tasks that show good portability to target tasks can be assigned higher weights and the weight of each instance in each source task can be adjusted through AdaBoost. Yao et al. [20] introduced multi-source domains to solve the problem of negative transfers and proposed two new algorithms, Multi-source-TrAdaBoost and TaskTrAdaBoost. Duan et al. [21] proposed a multi-source domain adaptation method (DAM) by using a set of pre-trained classifiers (called auxiliary/source classifiers) that use labeling patterns from multi-source domains to learn from robust decision function for pattern labeling prediction in the target domain (called target classifier). A incomplete multi-source transfer learning algorithm IMTL is proposed through two directions of knowledge transfer (ie, cross-domain transfer and cross-source transfer from each source to target) [22]. In [23], this paper present a Bayesian framework for transfer learning using neural networks that considers single and multiple sources of data. Other researches on multi-source transfer learning in literature [24–26]. Today, transfer learning has been applied in the fields of speech recognition, computer vision, information retrieval, natural language processing, adaptive update map coverage, fault diagnosis, automatic detection of COVID-19 infection and other fields [27–33].

In this paper, the structural risk minimization theory, support vector machine, and source transfer learning theory are taken. Aiming for an application scenario where there is only a small amount of labeled data in the target domain and a large amount of labeled data in multi-source domains, a new multi-source fast transfer algorithm-MultiFTLSVM is proposed. The idea of the MultiFTLSVM algorithm is to integrate the knowledge within the target domain and the labeled data in

multi-source domains into a structural risk minimization framework of support vector machines. The knowledge that needs to be transferred in the source domain is selected by constructing a similar distance term and the MMD between the target domain and each source domain, and the AESVM algorithm is used to reduce the sample size of the source domain to improve algorithm training efficiency, and then construct an optimizable objective function. The theoretical proof of the objective function shows that the solution process is a quadratic programming problem with an optimal solution.

Compared with previous work, the contributions of this paper include:

1) Existing data sets are taken to reduce the cost of collecting data sets.
2) All training samples no longer require AESVM to train the learning model, which can greatly reduce the size of the training samples so that the training cost of the learning model is reduced.
3) To prevent negative transfer and improve classification performance, multi-source transfer learning simultaneously extracts knowledge from multi-source domains to assist the learning task in the target domain. In this process, the knowledge of the similarity between each source domain and sample and the target domain is transferred to the target domain to the greatest extent.

The rest of the paper is arranged as follows: The related work of multi-source transfer learning, approximate pole support vector machine (AESVM), and maximum mean discrepancy (MMD) are reviewed in Section 2. The construction and training process of multi-source fast transfer learning is introduced in detail in Section 3. The effectiveness of the algorithm on the 20-Newsgroups text data set, sentiment analysis data set and the spam data set is verified in Section 4. The main work of the paper is summarized in Section 5.

2 Overview of related work

In this part, we give a brief introduction to multi-source transfer learning and group probability. In the introduction of group probability, we focus on IC technology and the group probability classification algorithm IC-SVM that is technology is based on.

2.1 Multi-source transfer learning

Transfer learning has been widely studied for many years since it was proposed in NIPS-95 in 1995. Compared with traditional machine learning algorithms, transfer learning has significant advantages. Useful knowledge can be taken from the source domain to significantly improve the learning performance of the target domain and greatly reduce costly data labeling work. There is no need for training data and test data to satisfy the same distribution. At present, most transfer learning tasks only transfer the knowledge in a source domain to the target domain [9]. However, in real-world applications, we can easily collect auxiliary data from multi-source domains. Therefore, the study of transfer learning in multi-source domains has gradually aroused the interest of researchers [18]. As shown in Fig. 2, the relationship between multi-source domains and target domains is taken for the multi-source transfer to improve the predictive performance of the target domain on samples and assist the target domain to establish a prediction model.

In Fig. 2,(D_{S_1}, T_{S_1}),(D_{S_2}, T_{S_2}),…, (D_{S_n}, T_{S_n}) represents n source domains and their corresponding learning tasks.(D_T, T_T) represents the target domain and its corresponding learning tasks,f_t represents the target domain classifier obtained by training the data sets in the target domain D_T and the source domain $D_{S_i}(i = 1, \ldots, n)$.

Generally, multi-source transfer learning algorithms can be divided into two categories: methods based on boosting [19, 20] and methods based on regularization [21–24]. Additionally, [25] divided the multi-source transfer learning into multi-source sample transfer learning, parameter-based multi-source transfer learning, and feature-based multi-source transfer methods. The regularized multi-source transfer method needs to design a regularization term, while the boosting multi-source transfer method needs to adjust the weight of different domains or instances to achieve its purpose of transferring knowledge. The focus of these methods is sample mobility, instead of studying which source domain has better mobility. When there are multi-source domains, how to determine which source domain has better mobility is an important issue [26].

2.2 Approximate pole support vector

Given that the basic idea of support vector machines is to find a hyperplane that represents the largest interval between two types, from a geometric point of view, the calculation of the maximum boundary hyperplane is equivalent to calculating the nearest sample between the two convex hulls [34, 35]. For SVM, a prerequisite for achieving better training results is a large number of training samples. A large number of training samples not only requires a lot of manpower to label but also a lot of time is then consumed in the training phase, so the training efficiency of SVM is not very satisfactory.

A training data set $X = \{x_1, x_2, \ldots, x_n\}$is given, and the corresponding class label set is $Y = \{y_1, y_2, \ldots, y_n\}, y_i \in \{1, -1\}$. AESVM optimization problem is described in Eq. (1):

$$\min_{w,b} F_{AESVM}(w, b) = \frac{1}{2} w^T w + \frac{C}{M} \sum_{i=1}^{M} \beta_i l(w, b, \varphi(x_i)) \quad (1)$$

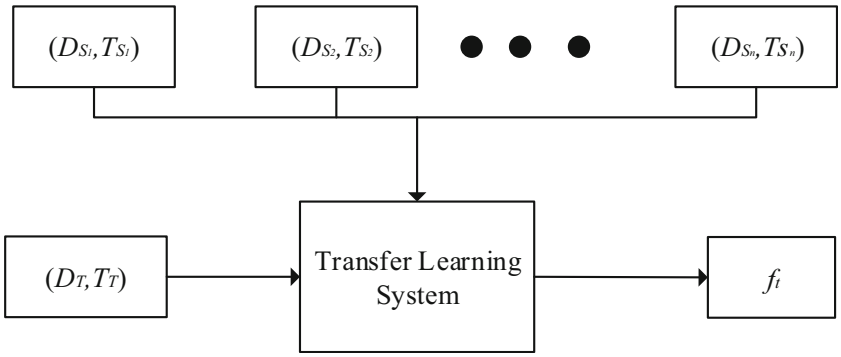

Fig. 2 Multi-source transfer learning

In Eq. (1), $M$is the number of samples of representative training data set X^* selected in the data set X, parameters w, b, i and C are support vector normal vector, displacement item, sample No. and regularization coefficient, the vector $\beta = [\beta_1, \beta_2, \ldots \beta_M]$is the weight vector corresponding to the representative data set sample, such as Eq. (9), M is the number of the representative data set, l is the hinge loss function, $l(w, b, \phi(x_i) = \max\{0, 1 - y_i(w^T\phi(x_i) + b)\}$, $x_i \in X^*$, and $\phi(\cdot)$ is the nonlinear mapping function. The kernel function can be written as $K_{i,j} = k(x_i, x_j) = \phi(x_i)^T\phi(x_j)$.

To obtain the representative data set X^*, the data set Xneeds to be grouped according to a determined separation strategy $X = \{X_1, X_2, \ldots, X_{n/V}\}$. n is the number of samples in the data set X,V is the maximum number of samples in each group and $X_q(q = 1, 2, \ldots, n/V)$ represents the group q. The sample data in each group has a high similarity, while the sample data between different groups had a low similarity. The concept of similarity is that the distance between sample data in a subset is less than the distance between sample data in different subsets. In each subset X_q, the representative data set X^{**} and the corresponding weight vector β_q are calculated. Finally, the representative data sets X^{**} are obtained from all subsetsX_q and are then merged into the representative data setsX^*.

The specific process of obtaining representative data sets from the data set X is as follows.

Firstly, the initial representative dataset X^{**} is calculated by using the SVDD algorithm [36] and the sample x_i $\left(x_i \in X_q \text{ and } x_i \notin X_q^*\right)$ should determine whether or not it belongs to the representative data set X^{**}, which is formally described as Eq. (2) .

$$\begin{cases} \max\limits_{x_i \in X_q, x_i \notin X_q^*} f\left(\varphi(x_i), X_q^*\right) = \min\limits_{\mu_{it}} \left\| \varphi(x_i) - \sum_{j=1}^{|X_q^*|} \mu_{i,t}\varphi\left(x_j\right) \right\|^2 \le \varepsilon \\ s.t. 0 \le \mu_{i,j} \le 1, \sum\limits_{j=1}^{|X_q^*|} \mu_{i,j} = 1, x_j \in X_q^* \end{cases} \tag{2}$$

In Eq. (2), $|X_q^*|$ is the number of samples in the sample set X_q^*, ε is a small normal vector artificially given, $\mu_{i,j}$ is the coordination coefficient, and j is the serial number of samples in the sample set X_q^*. For x_i meeting Eq. (2) in X_q, the extended representative sample set X_q^* is $X_q^* = X_q^* \cup \{x_i\}$. For all samples $x_i\left(x_i \in X_q \text{and} x_i \notin X_q^*\right)$, Eq. (3) is calculated by Eq. (2):

$$\varphi(x_i) = \sum_{x_i \in X_q} \gamma_{i,j}\varphi(x_i) + \tau_i \tag{3}$$

In Eq. (3), $\gamma_{i,j} = \begin{cases} \mu_{i,j}, x_j \in X_q^* \text{and} x_i \in X_q \\ 0, \end{cases}$ τ_i is the approximate error vector $\|\tau_i\|^2 \le \varepsilon$ in Eq. (3). The weight vector corresponding to the representative data set $\gamma_{i,j}$ in Eq. (4):

$$\beta_j = \sum_{i=1}^{n} \gamma_{i,j} \tag{4}$$

2.3 Maximum mean discrepancy

In transfer learning, the difference in sample distribution leads to the problem of negative transfer. Therefore, it is necessary to select a convenient distribution distance measurement. MMD is an effective measure to estimate the distance between two different distribution in the Hilbert Spaces. The value is calculated by the distance distribution found in the given functions. The function can best separate the two kinds of distribution and is limited to a unit ball in RKHS.

The set D_S containing n_s training samples and the set D_T containing n_t test samples are given. The formal definition of nonlinear mapping function and MMD in Hilbert Spaces is as follows:

$$MMD_H(D_S, D_T) = \left\| \frac{1}{n_s} \sum_{i=1}^{n_s} \phi\left(x_i^s\right) - \frac{1}{n_t} \sum_{j=1}^{n_t} \phi\left(x_j^t\right) \right\|_H \tag{5}$$

In Eq. (5), we find that the empirical estimate of the difference between the two distributions is considered as the distance between the two data distributions in the Hilbert Spaces, and an MMD value close to zero indicates that the two distributions are matched. Recently, the MMD measurement method is often used to calculate distribution values between domains in transfer learning.

3 Multi-source fast transfer support vector machine algorithm (MultiFTLSVM)

This section describes the group probability multi-source transfer algorithm in detail. The algorithm framework is shown in Fig. 3. As shown in Fig. 3, the input information for the MultiTLGP framework consists of two parts, labeled samples contained in the M source domains and quantity labeled samples contained in the target domain. For the sake of convenience, the dichotomy is considered.

N source domains are defined as $D_S = \left\{ D_{S_i} = \left(x_j^{S_i}, y_j^{S_i} \right)_{j=1}^{n_{S_i}}, i = 1, \ldots, N \right\}$, where $x_j^{S_i}$ is the j^{th} sample in the S_i^{th} source domain, and $y_j^{S_i}$ is the corresponding label of $x_j^{S_i}$. n_{S_i} is the number of the S_i^{th} source domain. Joint distribution probability of D_{S_i} is P_{S_i}. Similarly, the target domain is defined as $D_T = \left(x_i^T \right)_{i=1,..,n_T}$, and the corresponding joint distribution probability is P_T. $P_{S_i}(x^{S_i})$ and $P_T(x^T)$ are the marginal probability from the source domain D_{S_i} and the target domain D_T, and $P_{S_i}(x^{S_i}) \neq P_T(x^T)$. The MultiFTLSVM algorithm fully addresses the difference between the source domain samples and the target domain by reducing the marginal probability difference.

Figure 3 shows the framework of the MultiFTLSVM algorithm. Firstly, the weights of each source domain sample's marginal probability are calculated. Then, the proposed objective function is combines with support vector machine, structural risk minimization theory, and similarity distance minimization. Finally, the objective function is calculated, proved, and solved. The detailed process of MultiFTLSVM construction is as follows.

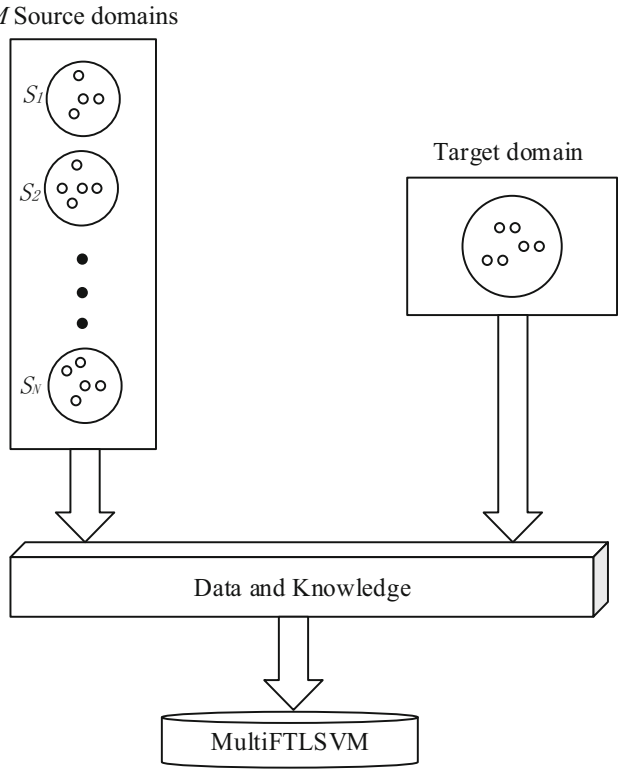

Fig. 3 Framework of MultiFTLSVM

3.1 Re-weighting data samples based on marginal probability differences

For the convenience of calculation, the similarity of weights $\gamma_j^{S_i}$ between samples and target domains in each source domain D_{S_i} with Eq. (5) showing the MMD measuring method in 2.3. Eq. (5) is modified as follows:

$$\min_{\alpha^{S_i}} \left\| \frac{1}{n_{s_i}} \sum_{j=1}^{n_{S_i}} \gamma_j^{S_i} \phi\left(x_j^{s_i} \right) - \frac{1}{n_T} \sum_{j=1}^{n_T} \phi\left(x_j^T \right) \right\|_H \tag{6}$$

ϕ is the mapping function in Hilbert Spaces H. n_{s_i} is the number of samples in the source domain D_{S_i}. n_T is the number of samples in the target domain D_T, and γ^{S_i} is the weight vector of the dimension n_{S_i}. The minimization of Eq. (6) is a standard quadratic programming problem, which can be solved by many existing solutions.

3.2 Objective function construction of MultiFTLSVM

Based on 3.1 and support vector machines, the objective function of MultiFTLSVM is constructed and combined with structural risk minimization theory and similarity distance minimization, as follows:

$$\begin{aligned} \min_{f_t, f_s \in H_k} & \frac{1}{2N} \sum_{i=1}^{N} \left\| f_{S_i} \right\|_K^2 + \frac{1}{N} \sum_{i=1}^{N} \frac{C_{S_i}}{M_i} \sum_{j=1}^{M_i} \beta_j^{S_i} l_{S_i}\left(f_{S_i}, y_j \right) + \frac{1}{2} \left\| f_t \right\|_K^2 \\ & + C_t \sum_{i=1}^{n_T} l_t(f_t, y_i) + \frac{\lambda}{2N} \sum_{i=1}^{N} d\left(f_t, f_{S_i} \right) \end{aligned} \tag{7}$$

f_S is the vector of decision function in N source domains, and the decision function in the target domain f_t is the same. $\left\| f_{S_i} \right\|_K^2$ and $\left\| f_t \right\|_K^2$ are the structural risk items controlling classifier complexity in the source domain and target domain respectively. $\|f\|^2$ is two norm functions. C_{S_i} and C_t are the regularization coefficients. $l()$ is the loss function. The function $d()$ is used to quantify the difference between the two domains. λ is the compromise item. M_i represents the number of representative data sets in the S_i^{th} source domain. $\beta^{S_i} = \left[\beta_1^{S_i}, \beta_2^{S_i}, \ldots, \beta_{M_i}{}^{S_i} \right]$ represents corresponding weights of representative data sets in samples from each source domain.

Equation (7) contains three items. The first item $\left(\frac{1}{2N} \sum_{i=1}^{N} \left\| f_{S_i} \right\|_K^2 + \frac{1}{M} \sum_{i=1}^{M} \frac{C_{S_i}}{M_i} \sum_{j=1}^{M_i} l_{S_i}\left(f_{S_i}, y_j \right) \right)$ represents knowledge learned from each source domain. The second $\left(\frac{1}{2} \left\| f_t \right\|_K^2 + C_t \sum_{i=1}^{n_T} l_t(f_t, y_i) \right)$ represents knowledge learned

from the target domain. The third $\left(\frac{\lambda}{2N}\sum_{i=1}^{N} d(f_t, f_{S_i})\right)$ represents the regularization term, which ensures good generalization performance by minimizing the differences between each source domain and target domain. $\frac{\lambda}{2N}\sum_{i=1}^{N} d(f_t, f_{S_i})$ is expressed as $\frac{\lambda}{2N}\sum_{i=1}^{N} \|w_t - \beta^{s_i} w_{s_i}\|^2$ with simple quadratic distance measurement. To solve the issue of negative transfer, we use the weight γ^{S_i} and β^{S_i} from 3.1 and replace the weight vector β^{S_i} of the representative data set with it, as shown in Eq. (8).

$$\rho^{S_i} = c_1\beta^{S_i} + c_2\gamma^{S_i}\kappa(D_{S_i}) \tag{8}$$

In Eq. (8), c_1 and c_2 represent the coefficients, $c_1 + c_2 = 1$, $\kappa(D_{S_i})$ is the mapping function of samples from the source domain D_{S_i} and their corresponding representative data set samples.

In conclusion, Eq. (7) is rewritten as Eq. (9).

$$\begin{aligned}\min_{w_t,b_t,w_s,b_s} &\frac{1}{2}\|w_t\|^2 + C_t\sum_{i=1}^{n_T} l_t\left(w_t^T\varphi(x) + b_t, y_i\right) + \frac{1}{2N}\sum_{i=1}^{N}\|w_{S_i}\|^2 \\ &+ \frac{1}{N}\sum_{i=1}^{N}\frac{C_{S_i}}{M_i}\sum_{j=1}^{M_i}\rho^{S_i} l_{S_i}\left(w_{S_i}^T\varphi(x) + b_{S_i}, y_j\right) + \frac{\lambda}{2N}\sum_{i=1}^{N}\|w_t - w_{S_i}\|^2\end{aligned} \tag{9}$$

In Eq. (9), a hinge loss function is introduced in each source domain and target domain. Therefore, Eq. (9) can be transformed into the optimization problem shown in Eq. (10):

$$\begin{aligned}\min_{w_t,b_t,w_s,b_s} &\frac{1}{2}\|w_t\|^2 + C_t\sum_{i=1+\sum_j^N M_j}^{\sum_j^N M_j+n_T}\xi_i + + \frac{1}{2N}\sum_{i=1}^{N}\|w_{s_i}\|^2 + \frac{1}{N}\sum_{i=1}^{N}\frac{C_{S_i}}{M_i}\sum_{j=1}^{M_i}\rho_j^{S_i}\xi_j^{S_i} + \\ &+ \frac{\lambda}{2N}\sum_{i=1}^{N}\|w_t - w_{S_i}\|^2 \\ s.t.\ & y_j^{S_i}\left(w_{s_i}^T\varphi\left(x_j^{S_i}\right) + b_{s_i}\right) \geq 1-\xi_j^{S_i}, j = 1,\ldots,n_{s_i}, s_i = 1,\ldots,N \\ & \tilde{y}_i\left(w_t^T\varphi(x_i^t) + b_t\right) \geq 1-\xi_i, i = 1,\ldots,n_T\end{aligned} \tag{10}$$

In Eq. (10), $\xi_j^{s_i}$ ($\xi_j^{s_i} \geq 0$) and ξ_i ($\xi_i \geq 0$) are relaxation variables. The first constraint ensures that the learning tasks from each source domain are classified as correctly as possible. The second constraint ensures that the learning tasks in the target area are classified as correctly as possible.

3.3 Objective function theorem proofing

Theorem 1: The dual problem of Eq. (10) is a quadratic programming (QP) problem, as shown in Eq. (11).

$$\begin{aligned}&\min_{\Gamma}\frac{1}{2}\mathbf{\Gamma}^T\tilde{\mathbf{K}}\mathbf{\Gamma} + \tilde{\mathbf{e}}^T\mathbf{\Gamma} \\ &s.t.\mathbf{f}^T\mathbf{\Gamma} = 0 \\ &\mathbf{\Gamma} = [\alpha^{s_1}, \alpha^{s_2}, ..., \alpha^{s_N}, \alpha]^{\mathrm{T}}, \\ &0 \leq \mathbf{\Gamma} \leq [\underbrace{C_{s_1}/\rho^{S_1}M_1, ..., C_{s_1}\rho^{S_1}/M_1}_{M_1}, \cdots, \underbrace{C_{s_N}/\rho^{S_N}M_1, ..., C_{s_N}/\rho^{S_N}M_1}_{M_{S_N}}, \underbrace{C_t, ..., C_t}_{n_T}], \\ &\mathbf{f}^{\mathrm{T}} = [y_1^{S_1}, ..., y_{n_{S_1}}^{S_1}, \cdots, y_1^{S_N}, ..., y_{n_{S_N}}^{S_N}, \underbrace{1, ..., 1}_{n_T}], \\ &\tilde{\mathbf{e}} = [\underbrace{0, ..., 0}_{M_1}, \cdots, \underbrace{0, ..., 0}_{M_N}, \tilde{\mathbf{y}}] \\ &\tilde{\mathbf{K}} = \begin{bmatrix} \frac{N+\lambda}{1+2\lambda N}\mathbf{K}_{s_1,s_1} + \frac{\lambda}{N}, ..., \frac{N+\lambda}{1+2\lambda N}\mathbf{K}_{s_1,s_N} + \frac{\lambda}{N}, -\frac{\lambda}{1+2\lambda N}\mathbf{K}_{s_1,t} \\ \cdots \\ \frac{N+\lambda}{1+2\lambda N}\mathbf{K}_{s_N,s_1} + \frac{\lambda}{N}, ..., \frac{N+\lambda}{1+2\lambda N}\mathbf{K}_{s_N,s_N} + \frac{\lambda}{N}, -\frac{\lambda}{1+2\lambda N}\mathbf{K}_{s_N,t} \\ -\frac{\lambda}{1+2\lambda N}\mathbf{K}_{s_1,t}^T, ..., -\frac{\lambda}{1+2\lambda N}\mathbf{K}_{s_N,t}^T, \frac{N+\lambda}{1+2\lambda N}\mathbf{K}_{t,t} \end{bmatrix}_{(\sum_{i\in N}M_i+n_T)\times(\sum_{i\in N}M_i+n_T)} \\ &\mathbf{K}_{s_i,s_i} = (y_j^{S_i}y_q^{S_i}k(\mathbf{x}_j^{S_i}\mathbf{x}_q^{S_i}))_{j,q=1,2,...,M_i}, \\ &\mathbf{K}_{s_i,t} = (y_j^{S_i}\sum_{q\in n_T}\tilde{y}_i k(\mathbf{x}_j, \mathbf{x}_q))_{j=1,..,M_i,q=1,...,n_T}, \\ &\mathbf{K}_{t,t} = (\sum_{i\in n_T}\sum_{j=n_T}k(\mathbf{x}_i, \mathbf{x}_j))_{i,j=1,...,n_T}.\end{aligned} \tag{11}$$

Proof: The Lagrangian function of Eq. (10) is as follows:

$$\begin{aligned} &L(w_t, w_s, b_t, b_s, \xi, \xi^s, \alpha, \alpha^s, r, r^s) = \\ &\frac{1}{2}\|w_t\|^2 + \frac{1}{2N}\sum_{i=1}^{N}\|w_{S_i}\|^2 + C_t \sum_{i=\sum_{j=1}^{N} M_j+1}^{\sum_{j=1}^{N} M_j+n^T} \xi_i + \frac{1}{N}\sum_{i=1}^{N}\frac{C_{S_i}}{M_i}\sum_{j=1}^{M_i}\rho_j^{S_i}\xi_j^{S_i} \\ &+\frac{\lambda}{2N}\sum_{i=1}^{N}\|w_t - w_{S_i}\|^2 - \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M_i} r_j^{S_i}\xi_j^{S_i} - \sum_{i=\sum_{j=1}^{N} M_j+1}^{\sum_{j=1}^{N} M_j+n^T} r_i\xi_i \\ &-\sum_{i=1}^{N}\sum_{j=1}^{M_i}\alpha_j^{S_i}\left(y_j^{S_i}\left(w_{s_i}^T\varphi\left(x_j^{S_i}\right)+b_{s_i}\right)-1+\xi_j^{S_i}\right) \\ &-\sum_{i=\sum_{j=1}^{N} M_j+1}^{\sum_{j=1}^{N} M_j+n^T}\alpha_i\left(\widetilde{y}_i\left(w_t^T\varphi\left(x_i^t\right)+b_t\right)-1+\xi_i\right) \end{aligned} \tag{12}$$

Where, $\alpha^{S_i} = \left(\alpha_1^{S_i}, \alpha_2^{S_i}, \ldots, \alpha_{M_i}{}^{S_i}\right), \alpha = (\alpha_1, \alpha_2, \ldots, \alpha_{n_T})$ and $r^{S_i} = \left(r_1^{S_i}, r_2^{S_i}, \ldots, r_{M_i}{}^{S_i}\right), r = (r_1, r_2, \ldots, r_{n_T})$ are the Lagrange multipliers, according to Karush-Kuhn-Tucker (KKT) conditions [17]:

$$\frac{\partial L}{\partial \xi_j^{S_i}} = 0 \Rightarrow \sum_{i=1}^{N}\sum_{j=1}^{M_i}\left(r_j^{S_i} + \alpha_j^{S_i}\right) = \frac{1}{N}\sum_{i=1}^{N}\frac{C_{S_i}}{M_i}\sum_{j=1}^{M_i}\rho_j^{S_i} \tag{13}$$

$$\frac{\partial L}{\partial \xi_i} = 0 \Rightarrow \sum_{i=\sum_{j=1}^{N} M_j+1}^{\sum_{j=1}^{N} M_j+n^T}(\alpha_i + r_i) = \sum_{i=\sum_{j=1}^{N} M_j+1}^{\sum_{j=1}^{N} M_j+n^T} C_t \tag{14}$$

$$\frac{\partial L}{\partial \mathbf{w}_{s_i}} = 0 \Rightarrow \frac{1}{N}\sum_{i=1}^{N}\mathbf{w}_{s_i} - \frac{\lambda}{N}\sum_{i=1}^{N}(\mathbf{w}_t - \mathbf{w}_{s_i}) - \sum_{i=1}^{N}\sum_{j=1}^{M_i}\alpha_j^{S_i} y_j^{S_i}\varphi\left(\mathbf{x}_j^{S_i}\right) = 0 \tag{15}$$

$$\frac{\partial L}{\partial w_t} = 0 \Rightarrow w_t + \frac{\lambda}{N}\sum_{i=1}^{N}(w_t - w_{s_i}) - \sum_{i=\sum_{j=1}^{N} M_j+1}^{\sum_{j=1}^{N} M_j+n_T}\alpha_i\widetilde{y}_i\varphi(x_j) = 0 \tag{16}$$

$$\frac{\partial L}{\partial b_{s_i}} = 0 \Rightarrow \sum_{i=1}^{M}\sum_{j=1}^{n_{s_i}}\alpha_j^{S_i} y_j^{S_i} = 0 \tag{17}$$

$$\frac{\partial L}{\partial b_t} = 0 \Rightarrow \sum_{i=\sum_{j=1}^{N} M_j+1}^{\sum_{j=1}^{N} M_j+n_T}\alpha_i\widetilde{y}_i = 0 \tag{18}$$

Equation (13)˜(18) can be substituted back into Eq. (10) and simplified to obtain Eq. (11) of the dual problem. Theorem 1 is thusly proved.

Theorem 2: The quadratic programming form of the optimization problem of Eq. (11) is a standard convex quadratic programming problem.

Proof: The matrix $\widetilde{\mathbf{K}}$ can be broken down into the form $\widetilde{\mathbf{K}} = \widetilde{\mathbf{K}}_1 + \widetilde{\mathbf{K}_2} + \widetilde{\mathbf{K}_3} + \widetilde{\mathbf{K}_4}$. Where, the forms of $\widetilde{\mathbf{K}}_1$, $\widetilde{\mathbf{K}_2}$, $\widetilde{\mathbf{K}_3}$ and $\widetilde{\mathbf{K}_4}$ are as follows:

$$\widetilde{K}_1 = \frac{\lambda}{1+2\lambda N}\begin{bmatrix} K_{s_1,s_1}, \ldots, K_{s_1,s_N}, -K_{s_1,t} \\ \cdots \\ K_{s_M,s_1}, \ldots, K_{s_N,s_N}, -K_{s_N,t} \\ -K_{s_1,t}^T, \ldots, -K_{s_N,t}^T, K_{t,t} \end{bmatrix}_{\left(\sum_{i\in N} M_i + n_T\right)\times\left(\sum_{i\in N} M_i + n_T\right)}$$

$$\widetilde{K}_2 = \frac{N}{1+2\lambda N}\begin{bmatrix} K_{s_1,s_1}, \ldots, K_{s_1,s_N}, 0 \\ \cdots \\ K_{s_N,s_1}, \ldots, K_{s_N,s_N}, 0 \\ 0, \ldots, 0, 0 \end{bmatrix}_{\left(\sum_{i\in N} M_i + n_T\right)\times\left(\sum_{i\in N} M_i + n_T\right)}$$

$$\widetilde{K}_3 = \frac{\lambda}{N}\begin{bmatrix} 1, \ldots, 1, 0, \\ \cdots \\ 1, \ldots, 1, 0 \\ 0, \ldots, 0, 0 \\ 0, \ldots, 0, 0 \end{bmatrix}_{\left(\sum_{i\in N} M_i + n_T\right)\times\left(\sum_{i\in N} M_i + n_T\right)}$$

$$\widetilde{K}_4 = \frac{N}{1+2\lambda N}\begin{bmatrix} 0, \ldots, 0, 0 \\ \cdots \\ 0, \ldots, K_{t,t} \end{bmatrix}_{\left(\sum_{i\in N} M_i + n_T\right)\times\left(\sum_{i\in N} M_i + n_T\right)}$$

For the matrix $\widetilde{\mathbf{K}}_1$, let $Q_1 = \sqrt{\frac{\lambda}{1+2\lambda N}}\left(y_1^{S_1}\varphi\left(x_1^{S_1}\right), \ldots, y_{M_1}{}^{S_1}\varphi(x_{M_1}{}^{S_1}), \ldots, y_1^{S_N}\varphi\left(x_1^{S_N}\right), \ldots, y_{M_N}{}^{S_N}\varphi(x_{M_N}{}^{S_N}), -\sum_{i\in n_T}\varphi(x_i), \ldots, -\sum_{i\in n_T}\varphi(x_i)\right)$ be the symmetric and positive semidefinite matrix. It is obvious that $\widetilde{\mathbf{K}}_1 = \mathbf{Q}_1^T\mathbf{Q}_1$, so $\widetilde{\mathbf{K}}_1$ is the symmetric semidefinite matrix and $\widetilde{\mathbf{K}_2}$, $\widetilde{\mathbf{K}_3}$ and $\widetilde{\mathbf{K}_4}$ are symmetric semidefinite matrices. Therefore, $\widetilde{\mathbf{K}}$ is the symmetric semidefinite matrix. Eq. (14) is a standard convex quadratic programming problem. Theorem 2 is thusly proved.

Theorem 3: The solution to the quadratic programming problem of Eq. (11) is the optimal solution.

Proof:Since Eq. (9) is a convex quadratic programming problem and the KKT condition is also a sufficient condition, the obtained solution is the optimal solution. The solution of convex quadratic programming refers to [37].

The optimal value $\mathbf{\Gamma}^* = (\alpha^{s_1}, \alpha^{s_2}, \ldots, \alpha^{s_N}, \alpha)^{\mathrm{T}}$ of $\mathbf{\Gamma}$ is calculated by Eq. (11), and the optimal solutions of $\mathbf{w}_t$ and b_t parameters are as follows:

$$w_t^* = \frac{\lambda N}{1+2\lambda N}\sum_{i=1}^{N}\sum_{j=1}^{M_i}\widetilde{\alpha}_j^{s_i}\rho_j^{S_i}y_j^{S_i}\varphi\left(x_j^{S_i}\right) + \frac{N+\lambda}{1+2\lambda N}\sum_{i=1+\sum_{j=1}^{N} M_j}^{\sum_{j=1}^{N} M_j+n_T}\widetilde{\alpha}_i\sum_{j\in n_T}\varphi(x_j) \tag{19}$$

$$\begin{aligned} b_t^* = y_i &- \frac{\lambda N}{1+2\lambda N}\sum_{i'=1}^{N}\sum_{j=1}^{M_{i'}}\rho_j^{S_{i'}}\alpha_j^{S_{i'}}y_j\sum_{q\in S_{i'}}k(x_j, x_q) \\ &- \frac{\lambda+N}{1+2\lambda N}\sum_{i'=1}^{N}\sum_{j=1+\sum_{l\in M} M_l}^{\sum_{l\in M} M_l+n_T}\widetilde{\alpha}_j\sum_{j'\in n_T}\sum_{q\in n_T}k\left(x_{j'}, x_q\right) \end{aligned} \tag{20}$$

Finally, the decision function for the MultiFTLSVM algorithm is as follows:

$$f(x) = \mathbf{w}_t \varphi(x) + b_t \tag{21}$$

The optimal solutions in Eqs. (19) and (20) contain information in M source domains and target domain. For example, in $\mathbf{w}_t^*$, $\frac{\lambda N}{1+2\lambda N} \sum_{i=1}^{N} \sum_{j=1}^{M_i} \widetilde{\alpha}_j^{S_i} \rho_j^{S_i} y_j^{S_i} \varphi\left(x_j^{S_i}\right)$ is the knowledge learned from the source domain, and $\frac{N+\lambda}{1+2\lambda N} \sum_{i=1+\sum_{j=1}^{N} M_j}^{\sum_{j=1}^{N} M_j + n_T} \widetilde{\alpha}_i \sum_{j \in n_T} \varphi(x_j)$ is the knowledge learned from the target domain.

3.4 MultiFTLSVM algorithm process

From Sections 3.1 through 3.3, the specific process and training steps of the MultiTLGP algorithm are shown in Table 1.

4 Results of experiments

In this section, to test the generalization performance of the MultiFTLSVM algorithm, we compare MultiFTLSVM with reference algorithms STL-SVM [38], RankRE-TL [15], HMCA [16], STIL [39], MultiDTNN [40], FastDAM [21], IMTL [22], SSL-MSTL [26] and SVM [41] on 20-Newsgroups, emotion analysis and spam datasets.

4.1 Experimental settings

To ensure the impartiality of the experiment, all experiments adopted the five-time cross-validation strategy and the experimental result is the final comparison result after 2 repeats of the strategy. For 5 transfer learning algorithms, all labeled source domain data and 5% unlabeled target data were randomly selected as training data sets. For the non-transfer learning algorithm SVM, we only used labeled data from the target domain for training. In the experiment, we took classification accuracy and the mean value of the corresponding standard deviation after running the calculation 10 times as the criteria of the evaluation algorithm. Classification accuracy is expressed as follows [14, 17, 37]:

$$Accuracy = \frac{|x : x \in D_t \wedge f(x) = y(x)|}{|x : x \in D_t|} \times 100\%$$

D_t is the target domain data set, $f(x)$ is the sample class x label predicted by the classifier, and $y(x)$ is the real class label of the sample x.

In the experiment, all kernel functions use the Gaussian function $k(x_i, x_j) = \exp(-\|x_i - x_j\|/2\sigma^2)$. The values of parameters C_t, C_{S_i} and λ are obtained from the grid $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10, 10^1, 10^2, 10^3, 10^4\}$ by gird search methods often used in machine learning. In addition to the above parameters, the other parameters of the reference algorithms are the same as those in corresponding literature. The hardware environment of all experiments was Intel Core

Table 1 Steps of MultiTLGP algorithm

Traning of MultiTLGP algorithm
Input: N source domains $D_S = \{D_{S_i} = (x_j^{S_i}, y_j^{S_i})_{j=1}^{n_{S_i}}, \ i = 1, \ldots, N\}$, and the number of labeled samples in each source domain is n_{S_i}. Target domain is $D_T = (x_i^T)_{i=1,\ldots,n_T}$.
Output: Decision function $f(x)$
Step 1: Calculate Eq (5) to get the sample weight vector for each source domain.
Step 2: Calculate the representative data sets and corresponding weight vectors for each source domain.
Step 3: Calculate Eq (8) to get the weight combination.
Step 4: Use QP solver to calculate Eq (11), and get $\boldsymbol{\Gamma}$.
Step 5: Calculate Eq (19) to get $\mathbf{w}_t$.
Step 6: Calculate Eq (20) to get b_t.
Step 7: Output the decision function in Eq (21).

(TM)i3, 3.6GHz, 8GB, Windows 10 OS, and running MATLAB R2014b.

4.2 Data sets used for experiments

20-Newsgroups [14], emotion analysis [14], and spam [15] are commonly used applications for transfer learning, so all experiments in this paper were carried out based on these 3 data sets.

1) 20-newsgroups

The 20-Newsgroups data set contains about 20,000 documents divided into four categories comp(c), rec(r), sci(s), and talk(t), each of which can be subdivided into four subcategories. The details of the data sets are shown in Table 2. In this experiment, the dichotomous task group is constructed by randomly selecting two of the four categories, one of which is positive and the other negative. Each task group is specifically comp vs rec, comp vs sci, comp vs talk, rec vs sci, rec vs talk, and sci vs talk. Common construction methods for cross-domain task groups were, there were 4 subclasses A1, A2, A3, and A4, while B has four subclasses B1, B2, B3, and B4 in each task group A vs B. Two subcategories (A1 and A3) from A and two subcategories (B1 and B2) from B are randomly selected to form the target domain data set, and the remaining data sets in A and B constitute the source domain data set. Each task group A vs B can generate $C_4^2 \times C_4^2 = 36$ classification tasks. The target domain data set and source domain data set obtained by the above construction methods ensure a correlation between the target domain and the source domain because they come from the same category. This also ensures target domain and source domain heterogeneity because they come from different subcategories. See Table 2 for details.

2) Emotion analysis data set

The emotion analysis data set consists of reviews from four different types of Amazon products, books, DVD, electronics, and kitchen supplies, which represents 4 domains – Books (B), DVDs (D), Electronics (E) and Kitchen (K). The content of each review contains: name, title, name of reviewer, date, place, and review. We take products rated by the evaluators with a rating of more than 3 stars (0–5 stars) as positive examples and those rated less than 3 stars as negative examples, with vague evaluation being discarded. In these 4 domains, there were 2000 annotated examples and about 4000 unannotated examples, with roughly the same number of positive and negative examples. The details of the data set are shown in Table 3.

3) Spam data set

The spam data set is distributed by the ECML/PKDD 2006 Knowledge Discovery Challenge and consists of four separate user mailboxes: personal mailbox U1, personal mailbox U2, personal mailbox U3, and public mailbox U4. There are 1250 spam and 1250 normal mailboxes in each personal mailbox and 2000 spam and 2000 normal mailboxes in each public

Table 2 20-Newsgroups Data set

Data set	Category	Subcategory	Samples	Features
20-Newsgroups	comp	comp.graphics	970	25,804
		comp.os.ms-windows.misc	963	
		comp.sys.ibm.pc.hardware	979	
		comp.sys.mac.hardware	958	
	rec	rec.autos	987	
		rec.motorcycles	993	
		rec.sport.baseball	991	
		rec.sport.hokey	997	
	sci	sci.crypt	989	
		sci.electronics	984	
		sci.med	987	
		sci.space	985	
	talk	talk.politics.guns	909	
		talk.politics.mideast	940	
		talk.politics.misc	774	
		talk.religion.misc	627	

Table 3 Emotion analysis data set

Domain	Comments	Training	Test	Positive sample ratio	Features
Books (B)	6465	2000	4465	50%	30,000
DVDs (D)	5586	2000	3586	50%	30,000
Electronics (E)	7681	2000	5681	50%	30,000
Kitchen (K)	7945	2000	5945	50%	30,000

mailbox. The personal and public mailboxes are expressed by an item frequency vector. The probability distribution of emails within each group is similar, but the difference between groups is large. Therefore, the six classification tasks that were constructed across groups in this paper are: U1 → U4, U2 → U4, U3 → U4, U4 → U1, U4 → U2 and U4 → U3. In the above representation modes of classification tasks, for example, inU1 → U4, U1 is the source domain, and U4 is the target domain. The details of the data set are shown in Table 4.

4.3 Results of experiments and analysis

In this section, we compare the mean classification accuracy and training time with a standard difference of MultiFTLSVM algorithm and reference algorithm on 3 real data sets and analyzed the results.

For the 20-Newsgroups data set, we selected one from data set r, s and t as the target domain. Five of the single source domain transfer learning algorithms SVM, STL-SVM, RankRE-TL, STIL and HCMA can only use one data set as a source domain, while five multi-source transfer learning algorithms FastDAM, IMIL, MultiDTNN、SSL-MSTL, and MultiFTLSVM can simultaneously use 3 data sets as its source domain. In the emotion analysis data set, 3 data sets were constructed with Books, DVDs, Electronics, and Kitchen as the target domain. Single source domain transfer learning algorithms can only select one data set from these 3 data sets as the source domain, while a multi-source transfer learning algorithm can simultaneously select 3 source domains. Similarly, for the spam data set, multi-source transfer learning algorithms used 3 personal mailboxes as 3 data sets and public mail data sets as the target domain. Single source domain transfer learning algorithms can only take one of the 3 personal data sets as the source domain and used the public mailbox as the target domain.

Table 4 Spam data set

Domain	Emails	Positive sample	Negative sample	Features
U1	4000	2000	2000	206,908
U2	2500	1250	1250	206,908
U3	2500	1250	1250	206,908
U4	2500	1250	1250	206,908

In Table 5, the experimental results on the data set 20-Newsgroups can draw the following conclusions: The classification accuracy of the MultiFTLSVM algorithm on the 9 cross-domain classification tasks has been improved compared with the benchmark algorithm, and the average accuracy has exceeded 95%. Compared with the non-migration algorithm, the average accuracy of SVM has increased by more than 10%, which also shows that the migration learning algorithm has considerable advantages over the traditional machine learning algorithm; compared with the single-source domain migration learning algorithm STL-SVM, STIL, RankRE-TL and HCMA, the average accuracy rate is improved; in the multi-source transfer learning algorithms MultiDTNN, FastDAM, IMTL, SSL-MSTL and MultiFTLSVM, the algorithm proposed in the article also has certain advantages. Because SVM does not have the ability of cross-domain transfer learning, the average classification accuracy is the lowest; single-source transfer learning algorithm is better than SVM; multi-source transfer learning algorithm is better than single-source transfer learning algorithm, and the algorithm proposed in the article is the best. The difficulty of transfer learning for the 12 cross-domain learning tasks is closely related to the similarity of the text content. It can be seen that the higher the similarity of the text content of the classification task, the higher the classification accuracy of the transfer learning algorithm.

For the experimental results on the sentiment analysis and spam data sets in Table 6, MultiFTLSVM has the highest average accuracy of all algorithms, and it has certain advantages compared to the non-transfer learning algorithm or the transfer learning algorithm in the benchmark algorithm: Compared with non-transfer learning SVM, the average accuracy rate is increased by about 12%; compared with TL-SVM, STIL, RankRE-TL, HCMA, MultiDTNN, FastDAM, IMTL and SSL-MSTL, the average accuracy rate is improved. In the classification accuracy of each cross-domain classification task, MultiFTLSVM is the highest compared with all benchmark algorithms.

According to the experimental analysis, we can draw the following conclusions:

Table 5 The average classification accuracy (%) and standard deviation of the algorithm on 20Newsgroups

Algorithms	r->s	c->s	t->s	c->r	s->r	t->r	s->t	c->t	r->t
SVM	73.81 (0.59)	75.38 (0.65)	75.47 (0.77)	87.51 (0.89)	71.39 (1.05)	84.32 (0.92)	76.82 (0.87)	95.43 (0.97)	83.26 (1.23)
STL-SVM	91.52 (1.06)	87.88 (1.12)	91.23 (1.23)	97.32 (1.18)	81.28 (1.03)	91.35 (1.18)	92.57 (1.21)	98.25 (1.01)	97.06 (1.17)
STIL	86.65 (1.26)	81.76 (1.68)	83.23 (1.57)	89.57 (1.64)	84.14 (1.57)	88.65 (1.55)	82.71 (1.62)	93.56 (1.53)	89.28 (1.75)
RankRE-TL	90.28 (0.98)	90.01 (1.05)	91.57 (0.52)	96.38 (0.28)	89.15 (1.13)	93.35 (0.64)	92.71 (0.53)	97.13 (0.29)	92.25 (0.69)
HCMA	81.27 (0.58)	88.28 (0.65)	87.29 (0.71)	94.65 (0.75)	87.32 (0.86)	92.24 (0.54)	88.97 (0.62)	96.87 (0.47)	90.34 (0.53)
	{r,c,t}->s			{r,s,t}->r			{s,c,r}->t		
MultiDTNN	96.21 (0.35)	94.76 (0.41)	95.88 (0.39)	97.45 (0.67)	92.76 (0.72)	98.34 (0.88)	95.46 (0.87)	97.65 (0.72)	98.86 (0.83)
FastDAM	95.34 (0.57)	92.98 (0.51)	94.32 (0.58)	94.36 (0.53)	91.25 (0.49)	97.02 (0.67)	94.76 (0.58)	96.75 (0.64)	96.89 (0.69)
IMTL	96.56 (0.43)	95.84 (0.48)	96.15 (0.51)	98.87 (0.45)	93.74 (0.52)	99.25 (0.47)	95.83 (0.65)	98.87 (0.57)	98.47 (0.55)
SSL-MSTL	94.33 (0.38)	93.21 (0.39)	95.42 (0.46)	98.75 (0.56)	92.65 (0.65)	98.82 (0.72)	94.28 (0.78)	97.98 (0.69)	97.54 (0.63)
MultiFTLSVM	97.65 (0.37)	96.13 (0.42)	97.26 (0.43)	99.23 (0.45)	95.23 (0.51)	99.57 (0.49)	96.75 (0.47)	99.58 (0.54)	99.85 (0.61)

(1) Based on the accuracy of average classification, it can be observed from Tables 5 and 6 that transfer learning algorithms can help classification tasks from the target domain by using knowledge from the source domain. Therefore, it has better classification effects than merely using data set training classifiers from the target domain by non-transfer learning algorithm SVM. In addition, compared with single source transfer learning algorithms

Table 6 The average classification accuracy (%) and standard deviation of the algorithm on spam email and emotion analysis

Algorithms	B->K	D->K	E->K	U1->U4	U2->U4	U3->U4
SVM	59.26 (2.78)	60.83 (2.74)	63.24 (2.73)	79.54 (2.52)	78.25 (2.51)	80.75 (2.68)
STL-SVM	64.21 (2.12)	63.34 (2.25)	68.12 (2.23)	86.23 (2.19)	84.63 (2.65)	86.34 (2.21)
STIL	65.11 (2.23)	65.13 (2.12)	67.28 (2.36)	87.21 (1.27)	85.87 (1.21)	87.21 (1.29)
RankRE-TL	63.78 (2.65)	65.12 (2.64)	69.57 (2.63)	87.54 (2.21)	86.12 (2.98)	88.63 (2.78)
HCMA	62.52 (2.67)	64.15 (2.66)	66.55 (2.64)	86.45 (2.37)	85.28 (2.12)	86.54 (1.98)
	{B,D,E}->K			{U1,U2,U3}->U4		
MultiDTNN	71.87 (2.13)	74.67 (2.24)	77.56 (2.11)	94.87 (2.48)	91.23 (2.23)	94.98 (2.32)
FastDAM	67.45 (2.31)	69.17 (2.26)	74.32 (2.19)	92.35 (2.28)	91.27 (2.35)	94.56 (2.42)
IMTL	70.45 (2.13)	73.98 (2.11)	76.15 (1.98)	94.87 (2.14)	93.23 (2.18)	96.11 (2.13)
SSL-MSTL	68.33 (2.28)	71.26 (2.17)	75.43 (2.21)	93.47 (2.26)	92.65 (2.28)	95.32 (2.25)
MultiFTLSVM	**72.65** (1.89)	75.13 (1.95)	78.23 (1.86)	95.43 (2.11)	94.32 (2.15)	97.86 (2.10)

STL-SVM, STIL, RankRE-TL and HCMA, multi-source transfer learning algorithms MultiDTNN, FastDAM, IMTL and SSL-MSTL showed obvious advantages in terms of classification effects. Finally, since combination weight information obtained from MMD is applied to effectively handle negative transfer from the proposed MultiFTLSVM algorithm in this paper, the classification effects of such an algorithm is superior to the majority of multi-source transfer learning algorithms regarding all learning tasks.

(2) As for algorithm operation time, the training time of SVM is relatively fast since SVM value just uses training data of the target domain. Since supplementary samples of the source domain are used in transfer learning, the training time of transfer learning algorithms increases when compared with that of the non-transfer learning algorithms. Since multi-source transfer learning uses more than two source domains to assist with the training of the target domain, its training time increases when compared with that of single source transfer learning. Among all of the multi-source transfer learning algorithms, MultiFTLSVM utilizes representative source domain data to shorten the training data set scale, which is conducive to reducing training time. Therefore, its training time is promoted in comparison to that of multi-source transfer learning algorithms from the benchmark algorithms Tables 7 and 8.

4.4 Parameter sensitivity analysis

In this section, we performed a sensitivity analysis of three parameters in the MultiFTLSVM objective function, namely the regular parameter of the target domain C_t, the mean value of regular parameters in the source target C_S, and the compromise item λ, to describe their impacts to the algorithm's performance. As for each parameter, we fix another two parameters as the optimum values determined by cross-validation and observed the parameter's impacts on classification results when using different values. The results of the experiment are seen in Figs. 4, 5 and 6.

Conclusions drawn from Figs. 4, 5 and 6 are as follows:

(1) We first fix $\lambda = 10, C_t = 10$and then search the value of C_s on the grid of $C_s \in \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10, 10^{1}, 10^{2}, 10^{3}, 10^{4}\}$, and record the experimental results on the real data set as shown in Fig. 5. Figure 5 shows C_s that with different values, the proposed classification effect is also different; We can see that when $C_s = 100$, the classification effect of the algorithm is the best on 14 cross-domain tasks. In the same way, we fix $\lambda = 10$, $C_s = 100$ and we use the same method to get the experimental results when C_t taking different values as shown in Fig. 4. From Fig. 4, we conclude that when $C_t = 10$, the algorithm performed the best classification on most cross-domain classification tasks. After the above analysis,

Table 7 Average score training time (s) and standard deviation of the algorithm on 20Newsgroups

Algorithms	r->s	c->s	t->s	c->r	s->r	t->r	s->t	c->t	r->t
SVM	1.35 (0.14)	1.31 (0.15)	1.41 (0.16)	1.28 (0.16)	1.32 (0.15)	1.25 (0.12)	1.43 (0.15)	1.47 (0.13)	1.26 (0.11)
STL-SVM	7.23 (0.73)	8.23 (0.89)	6.34 (0.98)	5.34 (0.69)	7.43 (0.72)	6.12 (0.63)	8.32 (0.77)	9.13 (0.76)	11.23 (0.86)
STIL	9.12 (0.63)	8.23 (0.61)	7.26 (0.59)	8.21 (0.58)	8.11 (0.57)	7.43 (0.72)	9.21 (0.82)	10.23 (0.83)	12.34 (0.93)
RankRE-TL	12.65 (0.98)	11.13 (1.12)	13.86 (1.13)	10.15 (1.01)	11.76 (1.17)	9.43 (1.14)	14.36 (1.21)	15.67 (1.23)	13.86 (1.19)
HCMA	8.11 (0.43)	7.32 (0.45)	9.63 (0.48)	7.98 (0.41)	9.86 (0.54)	8.23 (0.47)	11.35 (0.63)	12.56 (0.68)	12.04 (0.64)
	{r,c,t}->s			{r,c,t}->r			{r,c,t}->t		
MultiDTNN	72.23 (3.23)	65.23 (3.12)	69.32 (3.23)	55.67 (2.12)	62.34 (2.24)	52.34 (2.98)	76.21 (4.76)	73.87 (4.18)	67.76 (4.65)
FastDAM	119.56 (2.76)	112.53 (2.55)	116.38 (2.63)	110.45 (1.98)	111.38 (1.95)	109.87 (1.92)	122.26 (2.78)	124.33 (2.81)	123.15 (2.79)
IMTL	18.23 (1.25)	13.65 (1.18)	15.63 (1.24)	12.54 (1.19)	13.24 (1.21)	11.35 (1.16)	19.12 (1.76)	20.58 (1.79)	19.77 (1.77)
SSL-MSTL	1.34 (0.85)	1.21 (0.84)	1.29 (0.87)	1.16 (0.96)	1.28 (1.03)	0.95 (0.82)	1.36 (1.16)	1.42 (1.18)	1.41 (1.17)
MultiFTLSVM	1.15 (0.68)	0.98 (0.62)	1.11 (0.65)	0.88 (0.52)	0.96 (0.55)	0.93 (0.54)	1.24 (0.72)	1.28 (0.74)	1.26 (1.72)

Table 8 The average training time (s) and standard deviation of the algorithm on the sentiment analysis data set and spam data set

Algorithms	B->K	D->K	E->K	U1->U4	U2->U4	U3->U4
SVM	2.45 (0.15)	2.56 (0.14)	2.62 (0.16)	1.83 (0.15)	1.82 (0.14)	1.81 (1.12)
STL-SVM	4.56 (0.87)	6.78 (0.92)	5.98 (0.92)	6.11 (0.87)	4.98 (0.97)	5.01 (0.89)
STIL	10.23 (0.68)	12.34 (0.82)	13.76 (0.83)	7.97 (0.84)	6.83 (0.87)	6.33 (0.98)
RankRE-TL	13.78 (1.15)	14.12 (1.13)	16.57 (1.12)	9.98 (0.97)	9.69 (0.96)	9.15 (0.95)
HCMA	8.27 (0.73)	8.38 (0.72)	8.55 (0.71)	7.45 (0.76)	7.32 (0.77)	7.29 (0.75)
	{B,D,E}->K			{U1,U2,U3}->U4		
MultiDTNN	87.23 (4.36)	76.34 (4.27)	71.23 (5.12)	65.83 (4.31)	68.67 (4.21)	53.87 (4.34)
FastDAM	129.78 (2.65)	132.86 (2.75)	136.34 (2.87)	121.43 (2.43)	119.54 (2.42)	117.24 (2.34)
IMTL	20.45 (1.35)	23.98 (1.51)	26.15 (1.43)	21.25 (1.23)	19.86 (1.19)	18.43 (1.18)
SSL-MSTL	1.98 (0.49)	2.32 (0.51)	2.65 (0.52)	2.28 (0.64)	2.21 (0.63)	2.19 (0.61)
MultiFTLSVM	1.65 (0.59)	1.58 (0.56)	1.29 (0.55)	1.12 (0.42)	0.95 (0.41)	0.92 (0.46)

when C_S and C_t take different values, the average classification accuracy of the MultiFTLSVM method on 14 cross-domain tasks has a significant difference. We can find that MultiFTLSVM has a certain value range for the regularization parameters C_S and C_t Sensitive, the parameter values when the classification effect reaches the best can be obtained on different cross-domain tasks.

(2) For parameter λ, fix $C_s = 100$ and $C_t = 10$ to obtain the experimental results in the same way as in (1) as shown in Fig. 6. When the value of λ is 10, MultiFTLSVM achieves the best classification effect on 14 cross-domain tasks. We can get the following conclusion by analyzing the results in Fig. 6. If the value of λ is too small, the difference between the source domain and the target domain will be ignored and negative migration will occur, so the classification effect will not change; on the contrary, when the value of λ is too large, It can make the distribution difference between the source domain and the target domain more obvious, which results in less knowledge in the source domain that can be

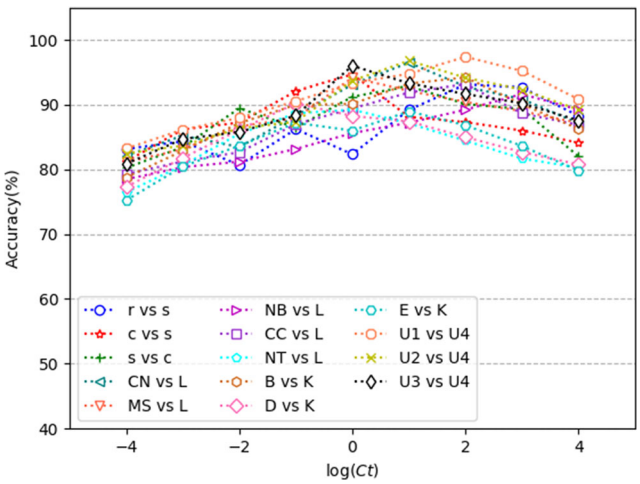

Fig. 4 Sensitivity of Parameter C_t in MultiFTLSVM Algorithm in 20-Newsgroups, Emotion Analysis, and Spam Data Sets

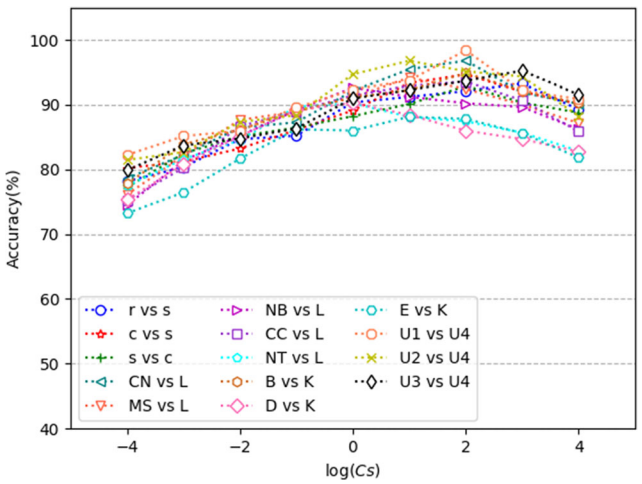

Fig. 5 Sensitivity of Parameter C_s in MultiFTLSVM Algorithm in 20-Newsgroups, Emotion Analysis, and Spam Data Sets

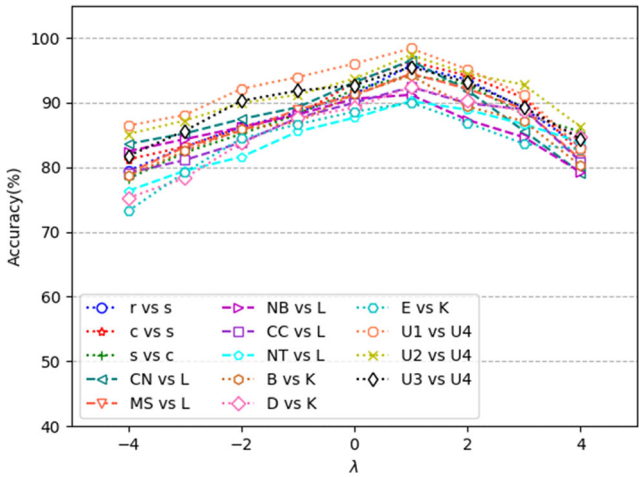

Fig. 6 Sensitivity of Parameter λ in MultiFTLSVM Algorithm in 20-Newsgroups, Emotion Analysis, and Spam Data Sets

transferred to the target domain, and the classification effect is also not good.

In short, the MultiFTLSVM algorithm is very sensitive to the regularization coefficients λ, C_t and C_s within a certain range, which means that it is very important to determine the optimal values of these parameters through effective strategies.

5 Conclusions

In this paper, we propose a multi-source fast transfer learning algorithm based on support vector machines, MultiFTLSVM, to provide multi-source domains for application in transfer learning. Firstly, the similarity weight of each source domain sample and the target domain is calculated with the purpose of resolving the minimum marginal probability differences, based negative transfer problem algorithm. Then the approximate pole support vector machine is used to obtain representative data sets from each source domain that are relatively important to the model training and corresponding weights, enhancing the training efficiency of the algorithm. Finally, knowledge from the target domain and combinatorially weighted multi-source domains are combined to be integrated into the structural risk minimization framework of the support vector machine. Then an objective function is constructed and a theoretical demonstration is performed. Results from experiments using the 20-Newsgroups data set, emotion analysis data set, and spam data set indicated that MultiFTLSVM is superior to the benchmark algorithms in regard to classification accuracy rate and training efficiency. Although the results indicated that the MultiFTLSVM algorithm has better results than the benchmark algorithms, it still needs to be further studied in the future with regard to the following aspects: extension of MultiFTLSVM in multi-class problems; increasing the number of source domains is an interesting challenge.

Acknowledgements This work has been supported by China's national key research and development plan.(2016YFB0801004).

References

1. Jordan MI, Mitchell TM (2015) Machine learning: trends, perspectives, and prospects [J]. Science 349(6245):255–260
2. Ashfaq RAR, Wang XZ, Huang JZ et al (2016) Fuzziness based semi-supervised learning approach for intrusion detection system [J]. Inf Sci 378(C):484–497
3. Li J, Wu W, Xue D (2020) An intrusion detection method based on active transfer learning[J]. Intell Data Anal 2020:363–383
4. Athanasios V, Nikolaos D, Anastasios D et al (2018) Deep learning for computer vision: a brief review[J]. Comput Intell Neurosci 2018:1–13
5. Nguyen G, Dlugolinsky S, Bobák M et al (2019) Machine learning and deep learning frameworks and libraries for large-scale data mining: a survey[J]. Artif Intell Rev 52(2019):77–124
6. Kadhim AI (2019) Survey on supervised machine learning techniques for automatic text classification[J]. Artif Intell Rev 52:273–292
7. Kumari KRV, Kavitha CR (2018) Spam Detection Using Machine Learning in R[C]//International Conference on Computer Networks and Communication Technologies, Lecture Notes on Data Engineering and Communications Technologies, April26 -27, Coimbatore, Tamil Nadu, India Springer, 55–64
8. Chen CLP (2015) Deep learning for pattern learning and recognition[C]// IEEE Jubilee International Symposium on Applied Computational Intelligence & Informatics, Timisoara, Romania, May 21–23 May, IEEE, 17–17
9. Pan SJ, Yang Q (2010) A survey on transfer learning [J]. IEEE Trans Knowled Data Eng 22(10):1345–1359
10. Day O, Khoshgoftaar TM (2017) A survey on heterogeneous transfer learning [J]. J Big Data 4(1):29
11. Weiss K, Khoshgoftaar TM, Wang DD (2016) A survey of transfer learning [J]. J Big Data 3(1):9
12. Gao J, Fan W, Jiang J, et al. (2008) Knowledge transfer via multiple model local structure mapping[C]// 14th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, August 24–27, Las Vegas, NV, United States. ACM, 283–291
13. Pan SJ, Tsang IW, Kwok JT, Yang Q (2011) Domain adaptation via transfer component analysis[J]. IEEE Trans Neural Netw 22(2): 199–210
14. Long M, Wang J, Ding G, Pan SJ, Yu PS (2014) Adaptation regularization: a general framework for transfer learning [J]. IEEE Trans Knowl Data Eng 26(5):1076–1089
15. Li M, Dai Q (2018) A novel knowledge-leverage-based transfer learning algorithm [J]. Appl Intell 48(8):2355–2372
16. Mozafari AS, Jamzad M (2016) A SVM-based model-transferring method for heterogeneous domain adaptation [J]. Pattern Recogn 52:142–158
17. Xie X, Sun S, Chen H, Qian J (2018) Domain adaptation with twin support vector machines[J]. Neural Process Lett 48(2):1213–1226
18. Sun S, Shi H, Wu Y (2015) A survey of multi-source domain adaptation [J]. Inf Fusion 24:84–92
19. Eaton E, des Jardins M (2011) Selective transfer between learning tasks using task-based boosting[C]// Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011, San Francisco, California, USA, August 7–11, DBLP, 337–342

20. Yao Y, Doretto G (2010) Boosting for transfer learning with multiple sources[C]//2010 IEEE computer society conference on computer vision and pattern recognition, san Franccisco, CA, USA, June 13–18, IEEE, 1855-1862
21. Duan L, Xu D, Tsang IW (2012) Domain adaptation from multiple sources: a domain-dependent regularization approach [J]. IEEE Trans Neural Netw Learn Syst 23(3):504–518
22. Ding Z, Shao M, Fu Y (2018) Incomplete multisource transfer learning [J]. IEEE Trans Neural Netw Learn Syst 29(2):310–323
23. Chandra R, Kapoor A (2020) Bayesian neural multi-source transfer learning[J]. Neurocomputing 378:54–64
24. Liu J, Li J, Lu K (2017) Coupled local-global adaptation for multi-source transfer learning[J]. Neurocomputing 275:247–254
25. Wu Q, Zhou X, Yan Y, Wu H, Min H (2017) Online transfer learning by leveraging multiple source domains[J]. Knowl Inf Syst 52:687–707
26. Fang M, Guo Y, Zhang X, Li X (2015) Multi-source transfer learning based on label shared subspace[J]. Pattern Recogn Lett 51:101–106
27. Huang Z, Siniscalchi SM, Lee CH (2016) A unified approach to transfer learning of deep neural networks with applications to speaker adaptation in automatic speech recognition[J]. Neurocomputing 2016(218):448–459
28. Ma YX, Xu JY, Wu XY, Wang F, Chen W (2017) A visual analytical approach for transfer learning in classification[J]. Inf Sci 2017(390):54–69
29. Lian QS, Shi BS, Chen SZ (2017) Transfer orthogonal sparsifying transform learning for phase retrieval [J]. Digit Signal Process 2017(62):11–25
30. Yan HB (2016) Transfer subspace learning for cross-dataset facial expression recognition[J]. Neurocomputing 2016(208):165–173
31. Ohata EF et al (2021) Automatic detection of COVID-19 infection using chest X-ray images through transfer learning [J]. IEEE/CAA J Autom Sin 8(1):239–248
32. Li W, Sai G, Zhang X, Chen T Transfer learning for process fault diagnosis: knowledge transfer from simulation to physical processes[J]. Comput Chem Eng 139:106904
33. Wu W, Peng M, Chen W, Yan S (2020) Unsupervised deep transfer learning for fault diagnosis in fog radio access networks[J]. IEEE Internet Things J 7(9):8956–8966
34. Chau AL, Li X, Yu W (2013) Convex and concave hulls for classification with support vector machine [J]. Neurocomputing 122(1):198–209
35. Dong JX, Krzyżak A, Suen CY (2003) A fast SVM training algorithm [J]. Int J Pattern Artif Intell 17(3):367–384
36. Ni T, Gu X, Wang J, Zheng Y, Wang H (2018) Scalable transfer support vector machine with group probabilities[J]. Neurocomputing 273:570–582
37. Xie X, Sun S (2019) Multi-view support vector machines with the consensus and complementarity information [J]. IEEE Trans Knowl Data Eng 32:2401–2413. https://doi.org/10.1109/TKDE.2019.2933511
38. Li J, Wu W, Xue D (2020) Research on transfer learning algorithm based on support vector machine [J]. J Intell Fuzzy Syst 38(4):4091–4106
39. Xie G, Sun Y, Lin M et al. (2017) A Selective Transfer Learning Method for Concept Drift Adaptation[C]//14th International Symposium, ISNN 2017, Sapporo, Hakodate, and Muroran, Hokkaido, Japan, June 21–26, Springer, 353–361
40. Li J, Wu W, Xue D, Gao P (2019) Multi-source deep transfer neural networks algorithm [J]. Sensors:19(18)
41. Chang CC, Lin CJ (2011) LIBSVM: a library for support vector machines[J]. ACM Trans Intell Syst Technol 2(3):1–27

Peng Gao A PhD student in Computer Technology at Harbin Engineering University. Senior engineer of Heilongjiang Radio and Television Station.