https://doi.org/10.1016/j.neucom.2019.11.010

Output based transfer learning with least squares support vector machine and its application in bladder cancer prognosis

Guanjin Wang[a,b,*], Guangquan Zhang[b], Kup-Sze Choi[c], Kin-Man Lam[d], Jie Lu[b]

[a] *Discipline of Information Technology, Mathematics & Statistics, Murdoch University, Perth, Australia*
[b] *Centre for Artificial Intelligence, School of Computer Science, Faculty of Engineering and Information Technology, University of Technology Sydney, Broadway, NSW, 2007, Australia*
[c] *Centre for Smart Health, School of Nursing, The Hong Kong Polytechnic University, Hong Kong, China*
[d] *Department of Surgery, Tseung Kwan O Hospital, Hong Kong, China*

Abstract

Two dilemmas frequently occur in many real-world clinical prognoses. First, the on-hand data cannot be put entirely into the existing prediction model, since the features from new data do not perfectly match those of the model. As a result, some unique features collected from the patients in the current domain of interest might be wasted. Second, the on-hand data is not sufficient enough to learn a new prediction model. To overcome these challenges, we propose an output-based transfer learning approach with least squares support vector machine (LS-SVM) to make the maximum use of the small dataset and guarantee an enhanced generalization capability. The proposed approach can learn a current domain of interest with limited samples effectively by leveraging the knowledge from the predicted outputs of the existing model in the source domain. Also, the extent of output knowledge transfer from the source domain to the current one can be automatically and rapidly determined using a proposed fast leave-one-out cross validation strategy. The proposed approach is applied to a real-world clinical dataset to predict 5-year

* *Email address:* Guanjin.Wang@murdoch.edu.au (Guanjin Wang)

overall and cancer-specific mortality of bladder cancer patients after radical cystectomy. The experimental results indicate that the proposed approach achieves better classification performances than the other comparative methods and has the potential to be implemented into the real-world context to deal with small data problems in cancer prediction and prognosis.

Keywords: transfer learning, machine learning, least squares support vector machine, cancer prediction

---

1. Introduction

Accurate prediction and prognosis plays an important role in the cancer field [1][2]. The successful implementation can help: 1) doctors to make prompt treatment decisions, 2) related health care industry to forecast the needs of their targeted populations and allocate resources accordingly, and 3) patients to better understand their conditions. For example, bladder cancer patients with poor prognosis will be advised to have follow-up examinations, such as cystoscopic or excretory urography.

However, there is a common dilemma that different hospitals or institutions may not exactly use the same clinical assessments for a particular cancer [3]. Thus, this leads to a possible situation that the existing prediction model or on-line tool were trained using a different feature set from the on-hand data. Although the most common features such as age, gender, tumour stage, etc. were remained, it is important to note that as time passes, the importance or relevance of the clinical measures might change, and the performance of the prediction model might be enhanced by adding more potential features.

Another significant dilemma is lack of data. If there are insufficient training samples in the current domain of interest, the performance of the constructed prediction model would substantially deteriorate. This is particularly common in the medical applications. For example, patients may not wish their medical records to be publicly available, or they drop out in the clinical follow-up study. In addition, due to the small sample size of the training data, it is important to employ the leave-one-out cross validation to guarantee that the selected parameters of the model can give the optimal generalization capability. This is because that the relatively high variance of the leave-one-out estimator can be offset by the stability resulting from the greater size of the training partition than using the traditional k-fold cross

validation [4]. However, how to greatly reduce the high computational cost of leave-one-out cross validation is another issue that is worthy of study.

To solve the above issues, in this study, transfer learning is used to take the advantage of the output knowledge from an existing model or on-line tool (source domain), which was trained on a large readily dataset to facilitate the learning process on a much smaller target domain with limited samples (target domain). More specific, our problem is heterogeneous transfer learning since the feature space between two domains may be different. Ideally, a subset of robust features is shared in both source and target domains while a unique subset of features is only kept in the target domain. The source and target data form an inverted pyramid dataset, as shown in Fig. 1. Due to the obvious commonality between the source and target data, the output knowledge from the prediction models in both domains should remain similar and mutually transferable. To achieve this, we propose a novel output-based transfer least squares support vector machine (LS-SVM) [5] approach in two versions. According to the concept of cascaded methods [6, 7], the proposed approach keeps the essence of cascaded methods and has the following main contributions:

(1) The proposed approach can learn knowledge from the probabilistic output predicted from the existing model or on-line tool to facilitate the learning process on the on-hand small bladder cancer dataset. It can be readily extended to learn a partial knowledge such as the sign of the output predicted from the existing model as well.

(2) The proposed approach can autonomously and quickly determine the influence level on the target domain made by the knowledge learned from the output using a proposed fast leave-one-out cross validation strategy.

(3) The proposed approach do not need to know the existing model's details and its training data to achieve transfer learning from the source domain to target domain. This makes it very practical to be implemented in the real-world scenarios such as bladder cancer prediction where the data and modeling details are kept in private.

The paper is organized as follows. Related work is introduced in Section 2. In Section 3, two versions of the proposed output-based transfer LS-SVM approach are presented. A fast leave-one-out cross validation strategy for parameter tuning is also developed. In Section 5, we give the experimental results on a real-world bladder cancer dataset. Lastly, the conclusions and future work are given in Section 6.

2. Related Work

This section presents the problem of transfer learning and its application in the medical field.

2.1. Transfer learning

Traditional machine learning techniques have been widely used in many fields; however, most hold the assumption that the training and testing data retain the same feature space and have the same data distribution. Therefore, once the feature space or the feature distribution of the new collected data changes, the prediction models cannot be used and must be re-constructed from scratch from new training data, which are very time consuming and impractical to collect. To solve this problem, transfer learning is introduced. The aim is to extract the knowledge from the related domain and apply it to the learning process in the current domain of interest which has few available data.

In transfer learning, a domain is denoted by $D = \{\chi, P(X)\}$. χ represents the feature space, and $P(X)$ represents the marginal probability distribution where $X = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n\} \in \chi$. Given a source domain D_S and learning task T_S, and a target domain D_T and learning task T_T, transfer learning can improve the learning process in D_T using the leveraged knowledge in D_S and T_S. In this work, $D_S \neq D_T$, $T_S = T_T$ and the condition $D_S \neq D_T$ implies that $\chi_S \neq \chi_T$.

There are three important problems to solve in transfer learning [8][9]: 1) what to transfer, 2) how to transfer, and 3) when to transfer. "What to transfer" focuses on which part of the knowledge or how much knowledge it is planned to leverage across domains. Based on this, transfer learning approaches in the literature can be classified into four categories.

The first category is instance transfer, which assumes that certain amount of data in the source domain can be useful for learning in the target domain, via instance re-weighting and importance sampling techniques. For example, in [10], a nonparametric method was proposed to directly obtain resampling weights with no distribution estimation. Xia et al. [11] presented another novel method to re-weight the training instance using in-target-domain probability by positive and unsupervised learning. Wang et al. [12] proposed a new transfer learning method with partial related 'instance-feature' knowledge to avoid discarding the partial related or unrelated knowledge in each source instance. Such partial structure is first explored by co-clustering

among both instances and features, then the source instances are reconstructed based on the exploited 'instance-feature' knowledge and the related target instances together such that the source instances are more related to the target ones for knowledge transfer later.

The second category is feature representation transfer, which aims to learn an appropriate common feature representation for the target domain such that the difference between the source and target domain is decreased, resulting in a decrease in classification error. The knowledge to be transferred across domains is embedded in the common feature representation. For example, transfer component analysis (TCA), was discovered in [13], whereby the distance between domains can be reduced in a latent space for domain adaptation. Duan et al. [14] proposed a method which augments the heterogeneous features from the source and target domains by utilizing two novel feature mapping functions. After that, the SVMs can be incorporated with newly generated feature representations for classification across domains with different feature spaces. In [15], a novel collaborative filtering method called Feature Subspace Transfer (FST) was proposed based on the user feature subspace transfer model for recommender systems, where the source and target features have different dimensions. Zuo et al. developed a fuzzy regression transfer learning in Takagi-Sugeno fuzzy models [16] and a granular fuzzy regression domain adaptation in Takagi-Sugeno fuzzy models [17], which both modify the input space of data through through mappings so that the fuzzy rules of the existing model become more compatible for solving tasks in the domain of interest.

The third category is relational knowledge transfer, which assumes that the relationship between the data in the source and target domain is similar. The knowledge to be transferred is the relationship between the data. In this context, statistical relational learning techniques are used to solve problems. For example, Mihalkova et al. [18] proposed a Markov logic networks (MLN)-based transfer system which maps the predicates in the source MLN to the target domain, and then edits the mapping structure to improve its performance.

The last category is parameter transfer, which assumes that the source and target domains share some parameters or priors of the models. The knowledge to be transferred is embedded in the shared parameters or priors. For example, a novel model proposed in [19] attempted to learn a shared covariance function on input dependent features and a 'free-form' covariance matrix of tasks. In [20], Gao et al. proposed a locally weighted ensemble

framework to leverage knowledge from multiple models for transfer learning. The weights are dynamically determined based on every model's predictive ability in each testing sample. Li et al. [21] proposed a transfer learning based extreme learning machines (ELM) called TL-ELM to use a small amount of target domain labeled data and a big number of source domain data to enhance the classification performance. The transferring knowledge is produced from exploiting the source model parameter knowledge reserved in the initial trained source ELM classifier. In [22][23], a knowledge-leverage-based Takagi-Sugeno-Kang fuzzy system (KL-TSK-FS) and an advanced version were proposed for parameter learning of the TSK-FS model on the target domain by utilizing existing knowledge in the source domain.

Of the existing transfer learning approaches, the parameter-transfer learning is most related to the output-based transfer LS-SVM classifier proposed in this paper. In general, most approaches aim to find the shared parameters/hyperparameters of the models. In the proposed classifier, since the existing model in the source domain is unknown, we focus on the output obtained from the source domain, and aim to learn a weighting parameter on the output of the target domain influenced by the predicted output of the existing model. The basic assumption behind is that the output from the models in both domains should be similar. This is because in this study, we only consider the inverted pyramid dataset in which a subset of features is shared in the feature space of both the source and target domains, such that there is a similarity between them. We can assume that the output from the ideal classifier on the target domain will retain a certain degree of consistency with the output of the existing model on the source domain. On the other hand, we find that there is an extra subset of unique features only in the target domain in the inverted pyramid dataset. Therefore, the problem in this study also involves the transfer of knowledge across different feature spaces, which is referred to as heterogeneous transfer learning [8].

"How to transfer" focuses on how to develop the model to achieve the knowledge transfer. "When to transfer" focuses on the circumstances in which transfer learning may or may not be used. If the source and target domain are not relevant to one another, brute-force transfer might fail, or even result in negative transfer. In this study, as explained above, the source and target domain are related to each other since only inverted pyramid datasets are adopted.

2.2. Transfer learning in medical applications

Transfer learning has been applied in various areas of medicine, as shown in the literature. For example, Silver and Mercer [24] proposed a task rehearsal method (TRM) which leverages the representation of a previously learned task as a source of inductive bias to make more accurate hypotheses for new tasks in three heart disease domains. In addition, they proposed a machine lifelong learning system composed of the task rehearsal method and ηMTL to learn a series of medical diagnostic tasks in a sequence. The source knowledge is stored in a domain knowledge database which contains the representations of successfully learned neural network models of the secondary tasks. Through ηMTL, this source knowledge is selectively transferred to help the learning process on the primary task based on a measure of task relatedness. Task rehearsal sequentially retains the task knowledge and uses it to generate virtual examples for learning a new task with ηMTL. The proposed system was practically applied to a number of medical diagnostic tasks. Another transfer learning based model was proposed in [25] in which knowledge previously learnt from each task using different drug combination are joined together to help predict the outcome of HIV therapy. The accuracy of the proposed method was increased by 10%-14% compared with the traditional modelling methods for individual tasks. Similarly, Zhou et al. [26][27] considered the prediction problem as a learning from multiple tasks in which each prediction at each time point is taken as a task. The authors proposed two novel transfer learning based models to predict MMSE/ADAS-Cog scores of Alzheimer's disease over the next four years using the baseline MRI features. In [28], a transfer-learning-based intelligent recognition method was proposed for epilepsy detection which learns the useful knowledge between the source and target domains by calculating the maximal mean discrepancy. Christodoulidis et al. [29] presented a method that improves the accuracy and stability of a deep convolutional neural network (CNN) to analyze lung tissue pattern, demonstrating the potential of transfer learning in the field of medical image analysis. However, there is little literature on utilizing transfer learning techniques for bladder cancer prediction and prognosis. This presents the opportunity to apply transfer learning to this specific area.

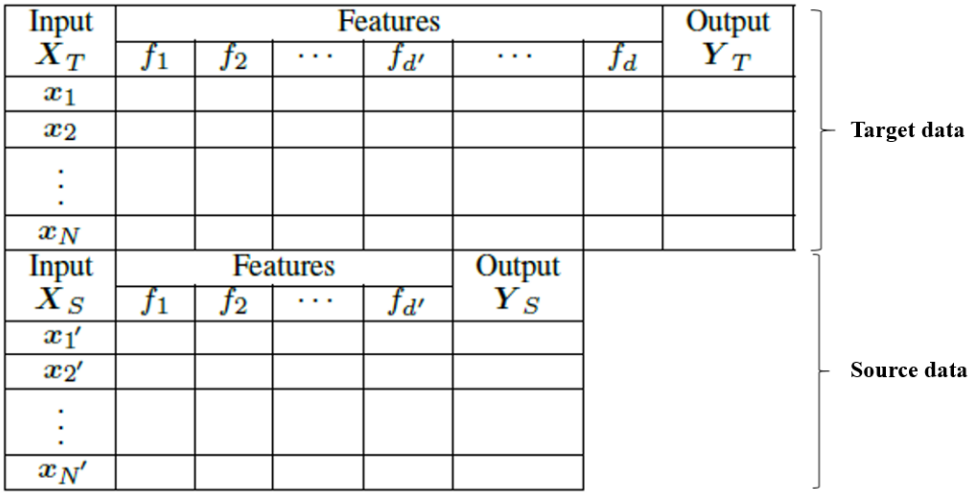


Figure 1: The inverted pyramid dataset in which $d' < d$

3. Output-based transfer LS-SVM classifier

3.1. Inverted pyramid dataset

In this work, we denote the data in the target domain as $D_T = \{(\boldsymbol{x}_1, y_1), \cdots, (\boldsymbol{x}_i, y_i), \cdots, (\boldsymbol{x}_N, y_N)\}$, where $\boldsymbol{x}_i = (x_1^i, x_2^i, \cdots, x_d^i) \in X_T \subset \mathbf{R}^d$ and $y_i \in Y_T = \{-1, 1\}$. X_T is the input dataset and Y_T is the corresponding class label set. Each sample $\boldsymbol{x}_i$ contains d features, i.e., f_1, f_2,$\cdots$, f_d. The data in the source domain is a subset of the target domain which only contains the same group of features with the existing model. We denote the data in the source domain as $D_S = \{(\boldsymbol{x}_1', y_1), \cdots, (\boldsymbol{x}_i', y_i), \cdots, (\boldsymbol{x}_N', y_N)\}$, where $\boldsymbol{x}_i' = (x_1^i, x_2^i, \cdots, x_{d'}^i) \in X_S \subset \mathbf{R}^{d'}$ and $y_i \in Y_S = [0, 1]$. X_S is the input dataset in the source domain and Y_S is the corresponding probabilistic output set obtained from the existing model. Each sample $\boldsymbol{x}_i'$ contains $d^{'}$ features, i.e., f_1, f_2, $\cdots$, $f_{d'}$ $(d^{'} < d)$. We want to find a decision function $F : X_T \rightarrow Y_T$, such that it can find the matching y for any new incoming sample $\boldsymbol{x}$.

We can stack D_T onto D_S as demonstrated in Fig. 1. Because it is shaped like an inverted pyramid, we call the adopted dataset in the proposed approach the *inverted pyramid dataset.*

3.2. Framework of the proposed approach

The framework of the proposed output-based transfer approach is illustrated in Fig. 2. As discussed in the last section, following Fig. 2, the

dataset is first transformed into the inverted pyramid dataset, which contains the target data D_T and the source data D_S with the same feature space of the existing model. After applying the existing prediction model or on-line tool on D_S, we obtain the corresponding probabilistic outputs. The proposed output-based transfer LS-SVMs classifier is then taken to help the classification in the target domain by not only making the full use of the data from D_T but also leveraging the output knowledge from the existing prediction model or on-line tool.

3.3. Handling probabilistic outputs from the existing model

Most prediction models or on-line tools in the medical field only produce probabilistic outputs using traditional statistical methods. Since the classification on the target domain is achieved using a different classification method with its class label set, the proposed approach is also designed to directly handle the probabilistic outputs from the existing model for convenience in the learning process in the target domain.

We put the source data D_S into the existing prediction model and obtain the corresponding probabilistic output p_i ($0 \leq p_i \leq 1$, $i = 1, 2, ..., N$). p_i and $1 - p_i$ are the probabilities of $\boldsymbol{x}_i$ which are classified into the positive or negative class. We set a threshold θ to 0.5 such that $\boldsymbol{x}_i$ is classified as positive class if its output probability is greater than 0.5. For example, if the sample $\boldsymbol{x}_i$ obtains the probabilistic output 0.65 from an existing predictive model, the probability of $\boldsymbol{x}_i$ being classified into the positive class or negative class is 0.65 or 0.35 ($1 - 0.65 = 0.35$) respectively. Based on the threshold θ, we classify $\boldsymbol{x}_i$ as the positive class ($0.65 - 0.5 = 0.15 > 0$) instead of the negative class ($0.35 - 0.5 = -0.15 < 0$).

Because the sign of $(2p_i - 1)$ reflects which class $\boldsymbol{x}_i$ belongs to, it can be used to retain the consistency between the output of the existing model and the adopted classification method, i.e., LS-SVM. This processed probabilistic output is the knowledge we want to effectively leverage in the target domain for classification.

3.4. Output-based transfer LS-SVM classifier in the target domain

After handling the probabilistic output from the existing prediction model, we can construct a model in the target domain in which the signs of the classification outputs and those of the processed probabilistic output from the existing source model are retained as much the same as possible.

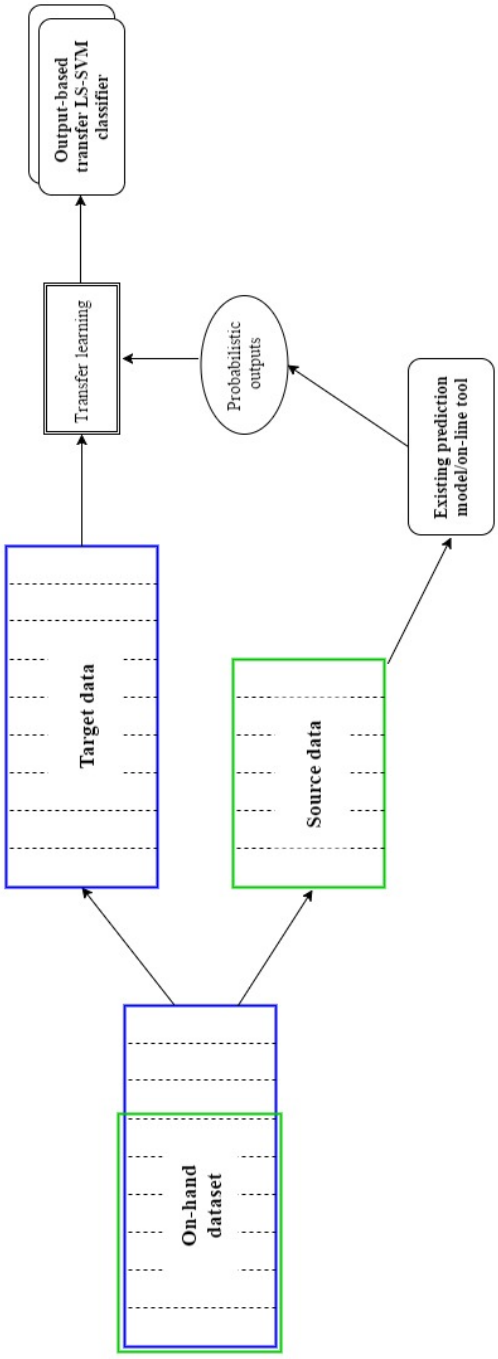


Figure 2: Framework of the proposed output-based transfer approach

In the proposed approach, the classification method for model construction on the target domain is LS-SVM. LS-SVM has two simplifications based on traditional SVM [30]. First, the inequality constraints in SVM are replaced by the equality constraints. Second, the hinge loss function in SVM is replaced by a squared loss function in the objective function. The optimization problem in LS-SVM is thus simplified by these updates and can be solved using a linear system instead of quadratic programming in SVM. At the same time, the performance of LS-SVM is comparable to SVM. The analytical solution of LS-SVM can assist with the efficient performance of the leave-one-out cross validation strategy, which is used for parameter tuning in the proposed approach and will be discussed in Section 4. The objective function of the standard LS-SVM is formulated as follows:

$$\begin{aligned} \min_{\boldsymbol{w},b} \quad & \frac{1}{2}\boldsymbol{w}^2 + \frac{C}{2}\sum_{i=1}^{N}\xi_i^2 \\ \text{s.t} \quad & y_i = \boldsymbol{w}^T\varphi(\boldsymbol{x}_i) + b + \xi_i, i = 1, 2, ..., N \end{aligned} \tag{1}$$

where C is the given trade-off parameter. The input $\boldsymbol{x}_i$ can be classified into its positive class or negative class in terms of the decision function $F(\boldsymbol{x}_i)$, i.e.,:

$$\boldsymbol{w}^T\varphi(\boldsymbol{x}_i) + b = \begin{cases} > 0, & \text{positive class} \\ < 0, & \text{negative class} \end{cases}$$

Therefore, to retain the same signs of both y_i and $(2p_i - 1)$ $(i = 1, 2..., N)$ as much as possible, $\sum_{i=1}^{N}(y_i - \xi_i)(2p_i - 1)$ should be made as large as possible.

We add a weighting parameter to reflect the influence level of the processed probabilistic outputs from the existing prediction model on the predicted output from the target domain. This weighting parameter can be selected by the fast leave-one-out cross validation strategy and is discussed in Section 4. Here, we present two versions of the proposed classifier in the target domain. Based on the equality constraints, the first version is developed to encourage the flexibility in its generalization capability, and the second version is developed to encourage the flexibility in its loss function.

First version:

In the first version, the objective function based on the LS-SVM becomes:

$$\begin{aligned} \min_{\boldsymbol{w},b} \quad & \frac{1}{2}\boldsymbol{w}^2 + \frac{C}{2}\sum_{i=1}^{N}\xi_i^2 - \mu\sum_{i=1}^{N}(y_i - \xi_i)(2p_i - 1) \\ \text{s.t} \quad & y_i = \boldsymbol{w}^T\varphi(\boldsymbol{x}_i) + b + \xi_i, i = 1, 2, ..., N \end{aligned} \tag{2}$$

where μ is a weighting parameter. Since

$$\begin{aligned}&\frac{1}{2}\boldsymbol{w}^2+\frac{C}{2}\sum_{i=1}^{N}\xi_i^2-\mu\sum_{i=1}^{N}(y_i-\xi_i)(2p_i-1)=\\&\frac{1}{2}\boldsymbol{w}^2+\frac{C}{2}\sum_{i=1}^{N}\xi_i^2+\mu\sum_{i=1}^{N}\xi_i(2p_i-1)-\mu\sum_{i=1}^{N}y_i(2p_i-1)\end{aligned}\tag{3}$$

and $\mu\sum_{i=1}^{N}y_i(2p_i-1)$ is a constant, thus after adding up the constant $\sum_{i=1}^{N}(\frac{\mu}{2C}(2p_i-1))^2$, we have the equivalent objective function as follows:

$$\begin{aligned}\min_{\boldsymbol{w},b}\quad&\frac{1}{2}\boldsymbol{w}^2+\frac{C}{2}\sum_{i=1}^{N}(\xi_i+\frac{\mu}{2C}(2p_i-1))^2\\\text{s.t}\quad&y_i=\boldsymbol{w}^T\varphi(\boldsymbol{x}_i)+b+\xi_i, i=1,2,...,N\end{aligned}\tag{4}$$

where $\frac{\mu}{2C}$ represents the influence level of the probabilistic output of the existing predictive model on the target domain. We can easily observe that if we set μ to 0, Eq. (4) becomes the objective function of the standard SVM in Eq. (1).

The Lagrangian J of Eq. (4) is

$$J=\tfrac{1}{2}\boldsymbol{w}^2+\tfrac{C}{2}\textstyle\sum_{i=1}^{N}(\xi_i+\frac{\mu}{2C}(2p_i-1))^2+\sum_{i=1}^{N}\alpha_i(y_i-\boldsymbol{w}^T\varphi(\boldsymbol{x}_i)-b-\xi_i)\tag{5}$$

where $\boldsymbol{\alpha}=(\alpha_1,\alpha_2,...,\alpha_N)^T$ is the vector of all Lagrangian multipliers. With respect to $\boldsymbol{w}$, ξ_i, b, α_i, we have

$$\frac{\partial L}{\partial \boldsymbol{w}}=0\quad\Rightarrow\boldsymbol{w}=\sum_{i=1}^{N}\alpha_i\varphi(\boldsymbol{x}_i)\tag{6}$$

$$\frac{\partial L}{\partial \xi_i}=0\quad\Rightarrow\xi_i=\frac{1}{C}[\alpha_i-\frac{\mu}{2}(2p_i-1)]\tag{7}$$

$$\frac{\partial L}{\partial b}=0\quad\Rightarrow\sum_{i=1}^{N}\alpha_i=0\tag{8}$$

$$\frac{\partial L}{\partial \alpha_i}=0\quad\Rightarrow y_i=\boldsymbol{w}^T\varphi(\boldsymbol{x}_i)+b+\xi_i\tag{9}$$

Combining Eq. (6) and Eq. (7) with Eq. (9), we obtain

$$\sum_{j=1}^{N} \alpha_i \varphi(\boldsymbol{x}_j)^T \varphi(\boldsymbol{x}_i) + b + \frac{\alpha_i}{C} = y_i + \frac{\mu}{2C}(2p_i - 1) \tag{10}$$

Using the kernel trick, we replace $\varphi(\boldsymbol{x}_j)^T \varphi(\boldsymbol{x}_i)$ with $K(\boldsymbol{x}_j, \boldsymbol{x}_i)$, and then write the linear equation in Eq. (10) in the following matrix form

$$\begin{bmatrix} \mathbf{K} + \frac{1}{C}\Lambda & \mathbf{1} \\ \mathbf{1}^T & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha} \\ b \end{bmatrix} = \begin{bmatrix} \boldsymbol{y} + \frac{\mu}{2C}\boldsymbol{M} \\ 0 \end{bmatrix} \tag{11}$$

where Λ is a matrix in which each diagonal entry is one and all other entries are zero, $\boldsymbol{y}$ is the output vector of all the samples in the training dataset, i.e, $\boldsymbol{y} = (y_1, y_2, \cdots, y_N)^T$ and $\boldsymbol{M} = \big(2p_1 - 1, 2p_2 - 1, ..., 2p_N - 1\big)^T$.

Lastly, the model parameters can be calculated simply by using the matrix inversion:

$$\begin{bmatrix} \boldsymbol{\alpha} \\ b \end{bmatrix} = \mathbf{P} \begin{bmatrix} \boldsymbol{y} + \frac{\mu}{2C}\boldsymbol{M} \\ 0 \end{bmatrix} \tag{12}$$

where $\mathbf{P} = \mathbf{V}^{-1}$ and $\mathbf{V}$ is the first matrix on the left in Eq. (11). Once we have obtained μ, $\boldsymbol{\alpha}$ and b can be calculated by using Eq. (12). Combined with Eq. (6), we can easily obtain the decision function for the new sample $\boldsymbol{x}_t$ as follows:

$$\begin{aligned} F_1(\boldsymbol{x}_t) &= \boldsymbol{w}^T \varphi(\boldsymbol{x}_t) + b \\ &= \sum_{i=1}^{N} \alpha_i K(\boldsymbol{x}_i, \boldsymbol{x}_t) + b \end{aligned} \tag{13}$$

Second version:

In the second version, in terms of the equality constraints, we replace $(y_i - \xi_i)$ in the first version with $(\boldsymbol{w}^T \varphi(\boldsymbol{x}_i) + b)$, therefore the objective function based on LS-SVM becomes:

$$\begin{aligned} \min_{\boldsymbol{w},b} \quad & \frac{1}{2}\boldsymbol{w}^2 + \frac{C}{2}\sum_{i=1}^{N} \xi_i^2 - \lambda \sum_{i=1}^{N} (2p_i - 1)(\boldsymbol{w}^T \varphi(\boldsymbol{x}_i) + b) \\ \text{s.t} \quad & y_i = \boldsymbol{w}^T \varphi(\boldsymbol{x}_i) + b + \xi_i, i = 1, 2, ..., N \end{aligned} \tag{14}$$

where λ is a weighting parameter.

The Lagrangian J of Eq. (14) is

$$J = \tfrac{1}{2}\boldsymbol{w}^2 + \tfrac{C}{2}\textstyle\sum_{i=1}^{N} \xi_i^2 - \lambda \sum_{i=1}^{N} (2p_i - 1)(\boldsymbol{w}^T \varphi(\boldsymbol{x}_i) + b) + \sum_{i=1}^{N} \alpha_i (y_i - \boldsymbol{w}^T \varphi(\boldsymbol{x}_i) - b - \xi_i) \tag{15}$$

where $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, ..., \alpha_N)^T \in \mathbf{R}^{N\times 1}$ is the vector of all Lagrangian multipliers. With respect to $\boldsymbol{w}$, ξ_i, b, α_i, we have

$$\frac{\partial L}{\partial \boldsymbol{w}} = 0 \quad \Rightarrow \boldsymbol{w} = \lambda \sum_{i=1}^{N} (2p_i - 1)\varphi(\boldsymbol{x}_i) + \sum_{i=1}^{N} \alpha_i \varphi(\boldsymbol{x}_i) \tag{16}$$

$$\frac{\partial L}{\partial \xi_i} = 0 \quad \Rightarrow \xi_i = \frac{\alpha_i}{C} \tag{17}$$

$$\frac{\partial L}{\partial b} = 0 \quad \Rightarrow \lambda \sum_{i=1}^{N} (1 - 2p_i) = \sum_{i=1}^{N} \alpha_i \tag{18}$$

$$\frac{\partial L}{\partial \alpha_i} = 0 \quad \Rightarrow y_i = \boldsymbol{w}^T \varphi(\boldsymbol{x}_i) + b + \xi_i \tag{19}$$

Combining Eq. (16) and Eq. (17) with Eq. (19), we obtain

$$\sum_{i=1}^{N} \alpha_i \varphi(\boldsymbol{x}_i)^T \varphi(\boldsymbol{x}_j) + b + \frac{\alpha_j}{C} = y_j - \lambda \sum_{i=1}^{N} (2p_i - 1)\varphi(\boldsymbol{x}_i)^T \varphi(\boldsymbol{x}_j) \tag{20}$$

Using the kernel trick, we replace $\varphi(\boldsymbol{x}_j)^T \varphi(\boldsymbol{x}_i)$ with $K(\boldsymbol{x}_j, \boldsymbol{x}_i)$, and then write the linear equation in Eq. (20) in the following matrix form:

$$\begin{aligned} \begin{bmatrix} \mathbf{K} + \frac{1}{C}\Lambda & \mathbf{1} \\ \mathbf{1}^T & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha} \\ b \end{bmatrix} &= \begin{bmatrix} \boldsymbol{y} - \lambda \boldsymbol{Z} \\ \lambda \sum_{i=1}^{N} (2p_i - 1) \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{y} \\ 0 \end{bmatrix} - \lambda \begin{bmatrix} \boldsymbol{Z} \\ \sum_{i=1}^{N} (1 - 2p_i) \end{bmatrix} \end{aligned} \tag{21}$$

where Λ is a matrix in which each diagonal entry is one and all other entries are zero, $\boldsymbol{y}$ is the output vector of all the samples in the training dataset, i.e., $\boldsymbol{y} = (y_1, y_2, \cdots, y_N)^T$, and $\boldsymbol{Z} = \left(\sum_{j=1}^{N} (2p_j - 1)K(\boldsymbol{x}_j, \boldsymbol{x}_1), \sum_{j=1}^{N} (2p_j - 1)K(\boldsymbol{x}_j, \boldsymbol{x}_2), \cdots, \sum_{j=1}^{N} (2p_j - 1)K(\boldsymbol{x}_j, \boldsymbol{x}_N) \right)^T$.

Lastly, the model parameters can be calculated simply by using the matrix inversion:

$$\begin{aligned} \begin{bmatrix} \boldsymbol{\alpha} \\ b \end{bmatrix} &= \mathbf{P} \begin{bmatrix} \boldsymbol{y} - \lambda \boldsymbol{Z} \\ \lambda \sum_{i=1}^{N} (1 - 2p_i) \end{bmatrix} \\ &= \mathbf{P} \begin{bmatrix} \boldsymbol{y} \\ 0 \end{bmatrix} - \lambda \mathbf{P} \begin{bmatrix} \boldsymbol{Z} \\ \sum_{i=1}^{N} (1 - 2p_i) \end{bmatrix} \end{aligned} \tag{22}$$

where $\mathbf{P} = \mathbf{V}^{-1}$ and $\mathbf{V}$ is the first matrix on the left in Eq. (11). Once we have obtained μ, $\boldsymbol{\alpha}$ and b can be calculated from Eq. (22). Combined with Eq. (16), the decision function for the new sample $\boldsymbol{x}_t$ becomes

$$\begin{aligned} F_2(\boldsymbol{x}_t) &= \boldsymbol{w}^T\varphi(\boldsymbol{x}_t) + b \\ &= \lambda\sum_{i=1}^{N}(2p_i - 1)K(\boldsymbol{x}_i, \boldsymbol{x}_t) + \sum_{i=1}^{N}\alpha_i K(\boldsymbol{x}_i, \boldsymbol{x}_t) + b \\ &= \sum_{i=1}^{N}(\lambda(2p_i - 1) + \alpha_i)K(\boldsymbol{x}_i, \boldsymbol{x}_t) + b \end{aligned} \tag{23}$$

Before ending up this section, we have the following remarks:

Remark 1: Although the first and second versions are proposed based on the equality constraints, we can observe the difference between them. According to Eq. (2), the second and the third terms actually forms a convex loss function, while the first and third terms in Eq. (14) actually forms a convex function about $\boldsymbol{w}$ which indeed controls the generalization capability of the second version. Therefore, we can take different values of λ and μ to provide different generalization capabilities or loss functions of the proposed classifier in practical applications. Only when $\lambda = \mu$ with the same C, the first version is equivalent to the second version theoretically.

Remark 2: We can use the signs of the probabilistic outputs to achieve the partial knowledge transfer by slightly changing the proposed approach. The value of $(2p_i - 1)$ in Eq. (2) and Eq. (14) can be replaced by the sign ($+1$ or -1) of $(2p_i - 1)$. It is expected that this sign knowledge transfer approach is less effective than the proposed approach as the sign of $(2p_i - 1)$ is less informative than its actual value. We will report the corresponding experimental results in Section 5.

4. Fast leave-one-out cross validation strategy for parameter tuning

We know from Section III that the classification performance of the proposed approach depends on the value of the parameter μ. The traditional leave-one-out cross validation strategy is commonly used as an unbiased estimator for parameter tuning during model construction; however, leave-one-cross validation is very time-consuming. In this section, we propose a fast version of the leave-one-out cross validation strategy for our proposed

approach to find the optimal value of μ and λ in Eq. (12) and Eq. (21) respectively of the two versions.

First version:

According to Eq. (12), we decompose $\mathbf{V}$ into block presentation with the isolation of the first row and column as follows:

$$\mathbf{V} = \begin{bmatrix} \mathbf{K} + \frac{1}{C}\Lambda & \mathbf{1} \\ \mathbf{1}^T & 0 \end{bmatrix} = \begin{bmatrix} v_{11} & \mathbf{v}_1^T \\ \mathbf{v}_1 & \mathbf{V}_{(-1)} \end{bmatrix} \tag{24}$$

We denote $\boldsymbol{\alpha}_{(-i)}$ and $b_{(-i)}$ as the model parameters during the i-th iteration of the leave-one-out cross validation. In the first iteration, we have:

$$\begin{bmatrix} \boldsymbol{\alpha}_{(-1)} \\ b_{(-1)} \end{bmatrix} = \mathbf{P}_{(-1)} \left(\boldsymbol{y}_{(-1)} + \frac{\mu}{2C} \boldsymbol{M}_{(-1)} \right) \tag{25}$$

where $\mathbf{P}_{(-1)} = \mathbf{V}_{(-1)}^{-1}$ and $\boldsymbol{y}_{(-1)} = [y_2, y_3, ..., y_N, 0]^T$. We denote the predicted label of the i-th sample excluded from the training dataset as $\tilde{y}_i$, so the predicted label of the first training sample becomes

$$\begin{aligned} \tilde{y}_1 &= \mathbf{v}_1^T \begin{bmatrix} \boldsymbol{\alpha}_{(-1)} \\ b_{(-1)} \end{bmatrix} - \frac{\mu}{2C} m_1 \\ &= \mathbf{v}_1^T \mathbf{P}_{(-1)} \left(\boldsymbol{y}_{(-1)} + \frac{\mu}{2C} \boldsymbol{M}_{(-1)} \right) - \frac{\mu}{2C} m_1 \end{aligned} \tag{26}$$

where m_1 denotes the first element of the vector $\boldsymbol{M}$, i.e., $m_1 = 2p_1 - 1$. Considering the last N equations in Eq. (11), we obtain $\left[\mathbf{v}_1 \, \mathrm{V}_{(-1)}\right] \left[\boldsymbol{\alpha}^T, \, b\right]^T = \left(\boldsymbol{y}_{(-1)} + \frac{\mu}{2C} \boldsymbol{M}_{(-1)}\right)$, and Eq. (26) can be written as

$$\begin{aligned} \tilde{y}_1 &= \mathbf{v}_1^T \mathbf{P}_{(-1)} \left[\mathbf{v}_1 \, \mathbf{V}_{(-1)}\right] [\alpha_1, \cdots, \alpha_N, b]^T - \frac{\mu}{2C} m_1 \\ &= \mathbf{v}_1^T \mathbf{P}_{(-1)} \mathbf{v}_1 \alpha_1 + \mathbf{v}_1^T [\alpha_2, \cdots, \alpha_N, b]^T - \frac{\mu}{2C} m_1 \end{aligned} \tag{27}$$

In Eq. (11), the first equation of the system is $y_1 + \frac{\mu}{2C} m_1 = v_{11}\alpha_1 + \mathbf{v}_1^T [\alpha_2, \alpha_3, \cdots, \alpha_N, b]^T$. Combined with Eq. (26), we obtain $\tilde{y}_1 = y_1 - \alpha_1 (v_{11} - \mathbf{v}_1^T \mathbf{P}_{(-1)} \mathbf{v}_1)$. Lastly, by using $\mathbf{P} = \mathbf{V}^{-1}$ and the block matrix inversion lemma, we obtain

$$\mathbf{P} = \begin{bmatrix} u^{-1} & -u^{-1} \mathbf{v}_1 \mathbf{P}_{-1} \\ \mathbf{P}_{(-1)} + u^{-1} \mathbf{P}_{(-1)} \mathbf{v}_1^T \mathbf{v}_1 \mathbf{P}_{(-1)} & -u^{-1} \mathbf{P}_{(-1)} \mathbf{v}_1^T \end{bmatrix} \tag{28}$$

where $u = v_{11} - \mathbf{v}_1^T \mathbf{P}_{(-1)} \mathbf{v}_1$. Since the system of linear equations in Eq. (11) is not sensitive to permutations of the ordering of the equations, we obtain

$$\tilde{y}_i = y_i - \alpha_i / P_{ii} \tag{29}$$

where P_{ii} denotes the i-th diagonal entry of $\mathbf{P}$. By defining $\left[\boldsymbol{\alpha}'^T, b'\right]^T = \mathbf{P}\left[\boldsymbol{y}^T, 0\right]$, $\left[\boldsymbol{\alpha}''^T, b''\right] = \mathbf{P}\left[\boldsymbol{M}^T, 0\right]$, and $\boldsymbol{\alpha} = \boldsymbol{\alpha}' + \frac{\mu}{2C}\boldsymbol{\alpha}''$, then we obtain

$$\tilde{y}_i = y_i - \frac{\alpha_i^{'}}{P_{ii}} - \frac{\frac{\mu}{2C}\alpha_i^{''}}{P_{ii}} \tag{30}$$

It can be seen from Eq. (30) that $\vec{\alpha}$ and μ have a linear relationship, which indicates that we can obtain the learning model if μ is determined. It is assumed that the optimal μ will retain the same sign of $\tilde{y}_i$ and y_i for all the samples in the training dataset. However, this might result in many local minima issues due to its non-convex formulation. Thus, we adopt the following loss function, which is similar to the hinge loss function:

$$l(\tilde{y}_i, y_i) = |1 - \tilde{y}_i y_i|_+ = \left| y_i \frac{\alpha_i^{'} - \frac{\mu}{2C}\alpha_i^{''}}{P_{ii}} \right|_+ \tag{31}$$

where $|x|_+ = \max\{0, x\}$. This is a convex upper bound to the leave-one-out misclassification loss. It prefers the solutions in which $\tilde{y}_i$ has an absolute value that is equal to or bigger than 1 and retain the same sign of y_i. In the end the objective function becomes:

$$\begin{aligned} &\sum_{i=1}^{N} l(\tilde{y}_i, y_i) \\ &\text{s.t} \quad 0 \leq \mu \leq D \end{aligned} \tag{32}$$

where D is a constant. This optimization process can be implemented by a projected sub-gradient descent algorithm. The pseudo-code is given in Algorithm 1.

Second version:

Similar to the first version, according to Eq. (22), we decompose $\mathbf{V}$ in Eq. (12) into block presentation with the isolation of the first row and column in Eq. (24).

We denote $\boldsymbol{\alpha}_{(-i)}$ and $b_{(-i)}$ as the model parameters during the i-th iteration of the leave-one-out cross validation. In the first iteration, we have:

$$\begin{bmatrix} \boldsymbol{\alpha}_{(-1)} \\ b_{(-1)} \end{bmatrix} = \mathbf{P}_{(-1)} \left(\boldsymbol{y}_{(-1)} - \lambda \boldsymbol{Z}_{(-1)} \right) \tag{33}$$

where $\mathbf{P}_{(-1)} = \mathbf{V}_{(-1)}^{-1}$ and $y_{(-1)} = [y_2, y_3, ..., y_N, 0]^T$. The predicted label of the i-th sample excluded from the training dataset is denoted as $\tilde{y}_i$. Similar to the derivations (see Eqs. (26)-(29)) in the first version, the predicted label of the first training sample becomes

$$\begin{aligned} \tilde{y}_1 &= \mathbf{v}_1^T \begin{bmatrix} \boldsymbol{\alpha}_{(-1)} \\ b_{(-1)} \end{bmatrix} + \lambda z_1 \\ &= \mathbf{v}_1^T \mathbf{P}_{(-1)} \left(\boldsymbol{y}_{(-1)} - \lambda \boldsymbol{Z}_{(-1)} \right) + \lambda z_1 \end{aligned} \tag{34}$$

where z_1 denotes the first element of the vector $\boldsymbol{Z}$, i.e., $z_1 = \sum_{j=1}^{N} (2p_j - 1) K(\boldsymbol{x}_j, \boldsymbol{x}_1)$. Considering the last N equations in the system of Eq. (21), it is clear that $\left[\mathbf{v}_1 \, \mathrm{V}_{(-1)} \right] \left[\boldsymbol{\alpha}^T, \, b \right]^T = \left(\boldsymbol{y}_{(-1)} - \lambda \mathbf{Z}_{(-1)} \right)$. Therefore Eq. (34) can be written as

$$\begin{aligned} \tilde{y}_1 &= \mathbf{v}_1^T \mathbf{P}_{(-1)} \left[\mathbf{v}_1 \, \mathbf{V}_{(-1)} \right] [\alpha_1, \cdots, \alpha_N, b]^T + \lambda z_1 \\ &= \mathbf{v}_1^T \mathbf{P}_{(-1)} \mathbf{v}_1 \alpha_1 + \mathbf{v}_1^T [\alpha_2, \cdots, \alpha_N, b]^T + \lambda z_1 \end{aligned} \tag{35}$$

Note that the first equation in the linear system in Eq. (21) is $y_1 - \lambda z_1 = v_{11} \alpha_1 + \mathbf{v}_1^T [\alpha_2, \alpha_3, \cdots, \alpha_N, b]^T$. Combining it with Eq. (35), we obtain

$$\tilde{y}_1 = y_1 - \alpha_1 (v_{11} - \mathbf{v}_1^T \mathbf{P}_{(-1)} \mathbf{v}_1) \tag{36}$$

Finally, by using $\mathbf{P} = \mathbf{V}^{-1}$ and the block matrix inversion lemma, we obtain

$$\mathbf{P} = \begin{bmatrix} u^{-1} & -u^{-1} \mathbf{v}_1 \mathbf{P}_{-1} \\ \mathbf{P}_{(-1)} + u^{-1} \mathbf{P}_{(-1)} \mathbf{v}_1^T \mathbf{v}_1 \mathbf{P}_{(-1)} & -u^{-1} \mathbf{P}_{(-1)} \mathbf{v}_1^T \end{bmatrix} \tag{37}$$

where $u = v_{11} - \mathbf{v}_1^T \mathbf{P}_{(-1)} \mathbf{v}_1$. Since the system of linear equations in Eq. (21) is not sensitive to permutations of the ordering of the equations, we obtain

$$\tilde{y}_i = y_i - \alpha_i / P_{ii} \tag{38}$$

By defining $\left[\boldsymbol{\alpha}^{'T}, b^{'} \right]^T = \mathbf{P} \left[\boldsymbol{y}^T, 0 \right]$, $\left[\boldsymbol{\alpha}^{''T}, b^{''} \right] = \mathbf{P} \left[\boldsymbol{M}^T, 0 \right]$, and $\boldsymbol{\alpha} = \boldsymbol{\alpha}^{'} - \lambda \boldsymbol{\alpha}^{''}$, we obtain

$$\tilde{y}_i = y_i - \frac{\alpha_i^{'}}{P_{ii}} + \frac{\lambda \alpha_i^{''}}{P_{ii}} \tag{39}$$

Algorithm 1: Projected Sub-gradient Descent Algorithm
Input: $\boldsymbol{\alpha}^{'}$, $\boldsymbol{\alpha}^{''}$
Initialize: $\mu \leftarrow 0$ and $t \leftarrow 1$
Repeat
$\tilde{y}_i = y_i - \frac{\alpha_i^{'}}{\mathbf{P}_{ii}} - \frac{\frac{\mu}{2C}\alpha_i^{''}}{\mathbf{P}_{ii}}$, $i = 1, 2, ..., N$
$d_i \leftarrow \mathbf{1}\{\tilde{y}_i y_i > 0\}$, $i = 1, 2, ..., N$
$\mu \leftarrow \mu - \frac{1}{\sqrt{t}} \sum_{i=1}^{N} d_i y_i \frac{\alpha_i^{''}}{\mathbf{P}_{ii}}$
If $\mu > D$ then $\mu \leftarrow D$
End if
$\mu \leftarrow \max(\mu, 0)$
$t \leftarrow t + 1$
Until convergence
Output: μ

We can observe from Eq. (39) that $\boldsymbol{\alpha}$ and μ also have a linear relationship, which indicates that we can obtain the learning model once μ has been determined. We adopt the same loss function in the first version and present it below:

$$l(\tilde{y}_i, y_i) = |1 - \tilde{y}_i y_i|_+ = \left| y_i \frac{\alpha_i^{'} + \lambda \alpha_i^{''}}{P_{ii}} \right|_+ \tag{40}$$

The objective function becomes:

$$\begin{aligned} &\sum_{i=1}^{N} l(\tilde{y}_i, y_i) \\ \text{s.t} \quad &0 \leq \mu \leq D \end{aligned} \tag{41}$$

where D is a constant. This optimization process can be implemented by a projected sub-gradient descent algorithm. The pseudo-code is given in Algorithm 2.

4.1. Computational complexity

Compared with traditional cross validation, the proposed fast leave-one-out cross validation strategy features the fast computational ability in both versions. Its computational cost contains two parts, which can be represented as $O(N^3 + N)$. The first part calculates the matrix $\mathbf{P}$ by the inverse, which is related to the training dataset in the target domain, therefore the

Algorithm 2: Projected Sub-gradient Descent Algorithm
Input: $\boldsymbol{\alpha}^{'}$, $\boldsymbol{\alpha}^{''}$
Initialize: $\lambda \leftarrow 0$ and $t \leftarrow 1$
Repeat
$\tilde{y}_i = y_i - \frac{\alpha_i^{'}}{\mathbf{P}_{ii}} + \frac{\lambda \alpha_i^{''}}{\mathbf{P}_{ii}}$, $i = 1, 2, ..., N$
$d_i \leftarrow \mathbf{1}\{\tilde{y}_i y_i > 0\}$, $i = 1, 2, ..., N$
$\lambda \leftarrow \lambda - \frac{1}{\sqrt{t}} \sum_{i=1}^{N} d_i y_i \frac{\alpha_i^{''}}{\mathbf{P}_{ii}}$
If $\lambda > D$ then $\lambda \leftarrow D$
End if
$\lambda \leftarrow \max(\lambda, 0)$
$t \leftarrow t + 1$
Until convergence
Output: λ

corresponding computational complexity becomes $O(N^3)$. The second part consists of the computational complexity of each iteration in Algorithm 1 for optimizing Eq. (32) or in Algorithm 2 for optimizing Eq. (41), which can be both represented as $O(N)$.

In terms of traditional cross validation with grid search, the whole time complexity from $[\mu_1, \mu_2, ..., \mu_T]$ for μ in the first version of the proposed classifier in Eq. (12), or from $[\lambda_1, \lambda_2, ..., \lambda_T]$ for λ in the second version of the proposed classifier in Eq. (22), would become $T * O(N^3 * N) = T * O(N^4)$, which is computationally much more expensive than $O(N^3 + N)$ in the proposed fast cross validation strategy.

5. Experimental results

5.1. The clinical dataset and the existing on-line model

A real clinical dataset collected in a urology unit in Hong Kong between 2003 and 2011 was used in the experiment [31][32]. The dataset contains 117 records of bladder cancer patients after radical cystectomy. In this dataset, ninety-nine patients are male. The mean age of patients is 68 years old (SD=10 years). The mean follow-up time is two years and seven months (SD=29 months). 71 patients had undergone open radical cystectomy, and 96 patients had ileal conduit diversion. The 30-day mortality rate, five-year cancer-specific mortality rate, other-cause mortality rate, and the overall

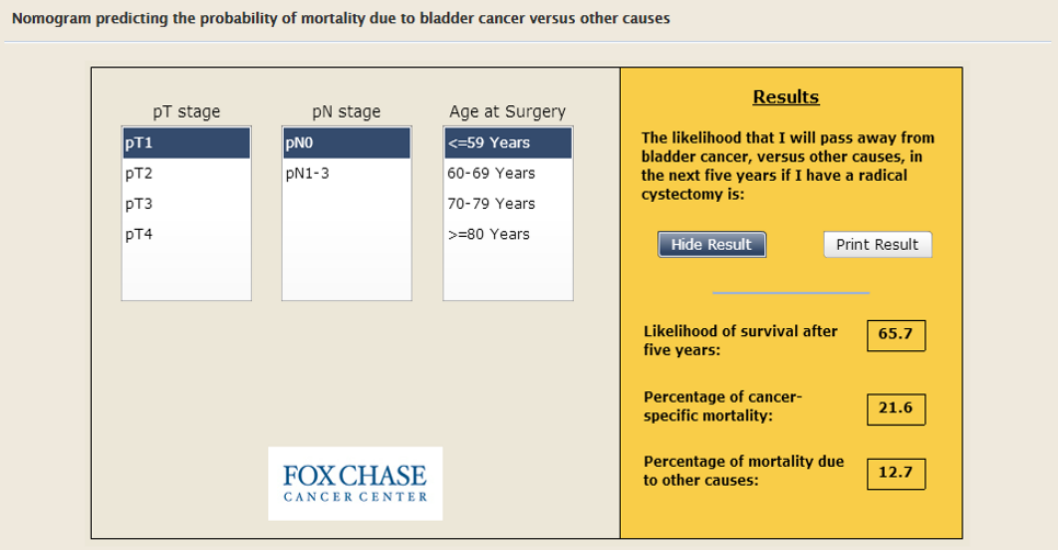


Figure 3: Online Nomogram predicting the probability of mortality due to bladder cancer versus other causes

mortality rate was 3%, 33%, 22% and 55% respectively. Other features include tumor stage, and grade, lymph node stage and preoperative serum albumin level. More details about this clinical dataset is presented in Table 1 [32]. Our goal is to predict the 5-year overall mortality and cancer-specific mortality of bladder cancer patients after radical cystectomy. Table 2 lists the data input and output we used for model construction.

The existing on-line model is selected from CancerNomograms.com [33], which was built using a smoothed Poisson regression model to predict the probability of overall mortality, cancer-specific mortality, and mortality due to other causes after five years. This on-line model is trained on 11,260 bladder cancer patients treated with radical cystectomy between 1988 and 2006 in the United States [34]. These patients were further stratified into 20 groups based on age, tumor stage, and lymph node stage after radical cystectomy. The user interface of this on-line model is screenshotted in Fig. 3.

From the observation, the adopted clinical dataset and the existing on-line model share the same feature groups, including 'age at operation,' 'tumor stage' and 'lymph node stage.' Therefore, the clinical dataset can be transformed into the inverted pyramid dataset to fit into our proposed framework.

Table 1: Patient demographics and clinicopathological characteristics

Demographics/Characteristics	No. (%) of patients or mean ± standard deviation		
	Overall	Age<=75 years	Age>75 years
No. of patients	117(100)	83(71)	34(29)
Mean age (years)	68±10	64±9	80±4
Gender			
Male	99 (85)	72 (87)	27 (79)
Female	18 (15)	11 (13)	7 (21)
Cystectomy			
Open	71 (61)	52 (63)	19 (56)
Laparoscopic/ robotic-assisted	46 (39)	31 (37)	15 (44)
Urinary diversion			
Ileal conduit	96 (82)	62 (75)	34 (100)
Neo-bladder/ continence diversion	21 (18)	21 (25)	0
Hospital stay duration (mouths)			
Mean	22±17	23±18	22±15
Median	18	17 (14-26)	18 (12-24)
Preoperative serum albumin level (g/L)	38±6	39±6	36±7
CCI			
0	77 (66)	60 (72)	17 (50)
1-2	38 (32)	22 (27)	16 (47)
>=3	2 (2)	1 (1)	1 (3)
Tumour grade			
G0	5 (4)	5 (6)	0
G2	24 (21)	17 (20)	7 (21)
G3	69 (59)	48 (58)	21 (62)
CIS	4 (3)	4 (5)	0
N/A	15 (13)	9 (11)	6 (18)
Tumour stage			
NMIBC	34 (29)	25 (30)	9 (26)
T0	11	6	5
Tis	7	7	0
Ta	4	3	1
T1	12	9	3
MIBC	82 (70)	57 (69)	25 (74)
T2	32	23	9
T3	32	23	9
T4	18	11	7
N/A			
Lymph node			
N0	88 (75)	65 (78)	23 (68)
N1	6 (5)	5 (6)	1 (3)
N2	14 (12)	8 (10)	6 (18)
N3	1 (1)	0	1 (3)
N/A	8 (7)	5 (6)	3 (9)
Follow-up (months)			
Mean	31±29	34±31	24±23
Range	(0-100)	(0-100)	(0-77)

Table 2: Input and output for the model construction

Input	**Value(s)**
Gender	1 (female) 2 (male)
Age at operation	Normalized to [0, 1]
Surgery Type	1 (open surgery) 2 (laparoscopic surgery) 3 (robotic surgery)
Preoperative serum albumin level	Normalized to [0,1]
Tumor stage	1 (T1) 2 (T2) 3 (T3) 4 (T4)
Lymph node stage	0 (N0) 1 (N1) 2 (N2) 3 (N3)
Overall cancer stage	1 (Stage I) 2 (Stage II) 3 (Stage III) 4 (Stage IV)
Follow up period	Normalized to [0,1]
Grade	1 (Grade 1) 2 (Grade 2) 3 (Grade 3)
Type of diversion	1 (ideal conduit) 2 (neo bladder)
Ouput	**Value(s)**
5-year mortality	0 (alive) 1 (died from bladder cancer) 2 (died from other causes)

5.2. Experimental Design

The primary purpose of the experiment is to evaluate the performance of the proposed approach (i.e., proposed classifier v1 (probability) and proposed classifier v2 (probability)) for predicting 5-year overall mortality and 5-year cancer-specific mortality after radical cystectomy. We compared the proposed approach with ten methods: proposed classifier v1 (sign), proposed classifier v2 (sign), large margin projected transductive transfer learning LMPROJ[35], LS-SVM [5], standard SVM [36], random forest, AdaBoost, back-propagation neural network (BPNN) [37], K nearest neighbor (KNN) algorithm [38] and NB∇C4.5 [7]. The first two methods are the same as the proposed classifier v1 (probability) and classifier v2 (probability) except that they transfer the knowledge learned from the signs of the outputs rather than the probabilistic outputs from the source model. We want to see if there is a performance difference between the probabilistic output transfer and the sign knowledge transfer under our proposed framework. The LMPROJ technique can minimize the distribution distance between the source and target domains by finding a feature transformation. Random forest and AdaBoost are the classical ensemble learning algorithms. The difference is that random forest trains individual models in a parallel way, while AdaBoost trains individual models in a sequential way where each model learns from mistakes made by the previous round. NB∇C4.5 is a cascade classification algorithm following the theory of Cascade Generalization [6]. A Bayesian classifier (NB) and a decision tree classifier (C4.5) are combined under an ensemble scheme to give a class decision. Notably, the C4.5 has additional input feature, which is the predicted outputs from the NB classifier. Other traditional machine learning methods, such as LS-SVM, SVM, BPNN, and KNN are also selected as comparative methods.

Before training our proposed models, the first step is to feed common features from the on-hand clinical data into the existing on-line tool to obtain the corresponding probabilistic predictions. This is the knowledge we want to leverage to assist the target domain construction. After that, the proposed classifier-v1 and classifier-v2, are applied to the whole clinical dataset for model construction with the help of output knowledge from the previous step. However, comparative methods only work on clinical data without learning probabilistic outputs.

To make our comparison fair, we used grid search with cross validation to discover the optimal parameters during the training process. More concretely, for the proposed models, LS-SVM and SVM, polynomial kernel was

selected [31] and the trade-off parameter C and the degree parameter γ were selected by searching $C \in \{1, 10, 50, 100, 150, 200, 250\}$ and $\gamma \in \{2e-5, 2e-4, 2e-3, 2e-2, 2e-1, 1\}$. For LMPROJ, λ, C and δ were selected from $\{10^-3, 10^-2, 10^-1, 10^0\}$, $\{10^2, 10^3, 10^4, 10^5, 10^6\}$ and $\{0.4, 0.6, 0.8, 1, 1.2, 1.4, 1.6\}$. For Random Forest, the number of trees were selected from $\{5, 10, 15, 20, 25, 30\}$ respectively. For AdaBoost, the number of learning cycles were selected from $\{10, 50, 100, 150\}$. For BPNN, the number of hidden neurons, the momentum and the learning rate were selected from $\{3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29\}$, $\{0, 0.2, 0.5, 0.9\}$ and $\{0.01$,0.05,$0.09\}$ respectively. For KNN, the number k of neighbors was experimentally selected from $\{5$,7,9,11,13,15,17,$19\}$. The detailed parameter settings are summarized in Table 3. All the experiments are implemented using 64-bit MATLAB on a computer with Intel Core i5-6300 2.40 GHz CPU and 8.00GB RAM.

5.3. Performance evaluation

The 10-fold cross validation was used for performance evaluation. This strategy randomly divides the dataset into ten subsets. The model was built on nine subsets and the remaining subset was used for testing. This process was repeated ten times, and the mean and standard deviation (SD) of accuracy, sensitivity, specificity, precision and AUC of ten models were calculated.

5.4. Classification performances

Tables 4 and 5 present the classification results of all the methods on the 5-year overall mortality and cancer-specific mortality, respectively. The corresponding ROC curves are shown in Fig. 4. A paired t-test was also used to compare the accuracy and AUC results between the proposed classifier and the other methods. Tables 4 and 5 list the corresponding paired t-test results, where the threshold of the p-value is set as 0.05. The superscript (+) means that the proposed classifier is significantly better than the other method due to a small p-value (<0.05). It can be observed that the proposed classifier v1 (probability) or the proposed classifier v2 (probability) achieved the highest classification accuracy in the different task. We conducted the paired t-test between the best classifier and every other method. The proposed classifier v1 (probability) or the proposed classifier v2 (probability) is statistically better than the proposed classifier v1 (sign) & the proposed classifier v2 (sign) in terms of accuracy. Besides, the proposed classifier v1 (probability) or the proposed classifier v2 (probability) achieved higher AUC values than

the proposed classifier v1 (sign) & the proposed classifier v2 (sign), although there is no statistically significant difference. At the same time, the proposed classifier v1 (probability) or the proposed classifier v2 (probability) is statistically better than the other comparative methods in terms of AUC in the different task. The performance of the comparative methods can be broadly categrozied into two tiers using an accuracy threshold of 0.7000. LMPROJ, LS-SVM, SVM, random forest, AdaBoost and KNN fall into the middle tier. In general the kernel based methods such as SVM and LS-SVMs performed better than the ensemble learning methods such as random forest and AdaBoost on the adopted datasets. BPNN and NB∇C4.5 are in the lower tier, whose accuracy are below 0.7000 in both tasks.

To further investigate the effectiveness of the proposed approach, we plot the t-SNE 2-D results with and without transfer for two prediction tasks in Fig. 5. Here, we only show the results of the proposed classifier v1 (probability). We observed that without output knowledge transfer, the two classes are not discriminated very well, while with output knowledge transfer, the data points are discriminated much better.

The experimental results indicate the proposed approach can achieve better classification performance on the clinical dataset than the traditional machine learning methods that do not benefit from learning knowledge in the source domain, showing its great potential for the real-world implementation. The proposed approach also outperformed the cascade method NB∇C4.5 although the latter one leverages the predictions from the Naive Bayes classifier to guide the final decisions at the higher level using C4.5. This might be due to the fact that the outcomes predicted from the base classifier in NB∇C4.5 did not supply enough valuable information to further enhance the prediction performance. In fact, the classifier levels and the selection of the combined classifiers can greatly influence the final classification results. As state in the above, we notice that the proposed approach also outperforms the proposed classifier v1 (sign) and the proposed classifier v2 (sign). We believe it is because that the direct use of the probabilistic output from the source model can help us precisely evaluate to what extent it will influence the output transfer term (third term in Eq. (2) and Eq. (14), respectively) in terms of each data point. That is, any data point's probabilistic output from the existing model may influence the learning of the weighting parameter at a certain degree. However, if we simply use the sign of $(2p-1)$, it only gives two absolute influence level ('+1' and '-1') instead of a specific influencing value. Besides, We used the online nomogram itself to predict the mortality due to

bladder cancer. It can only give 0.633 accuracy for predicting 5-year overall mortality and 0.725 accuracy for predicting 5-year cancer-specific mortality, showing our proposed output-transfer based approach has a big advantage in bladder cancer prognosis over the existing online model.

In addition, we noticed that there are similar works in the literature. For example, Fernandes et al.[39] proposed a SVMs based framework that regularizes the goal function with a penalty relevant with the source coefficient sign. However, how to obtain the value of α in the proposed objective function in Eq. (12) in [39] is a problem. Comparatively, in our approach, we can use the proposed fast leave-one-out cross validation strategy to automatically and simultaneously determine the value of the new parameter μ and λ perspectively, which has a superior advantage for implementation in the real-world scenarios. Kato et al. [40] imposed a hard constraint to restrict the sign of the predictor parameter $\boldsymbol{w}$. Instead, our approach focuses on the output knowledge instead of the predictor parameters, which allows the target and source feature sets being different. In summary, the main advantage of this study is that by using either version of the proposed approach, we are able to construct a reliable classifier on a small number of samples. The improved accuracy of prognosis could assist doctors to advise the best treatment plans for patients. Moreover, the proposed approach has the capability to work readily with the existing model or on-line tool in the medical field by leveraging their probabilistic output knowledge to help the learning process in the current domain of interest. Importantly, it is not necessary in this approach to know the details of the existing model and data trained on the model. It can still utilize transfer learning to improve generalization performance in the target domain. This is very practical in real-world scenarios where the data and its modeling are private. In other words, the proposed approach can be regarded as a module-based model which has the capacity be extended to various medical problems and situations.

Table 3: Parameter settings of the proposed and comparative methods

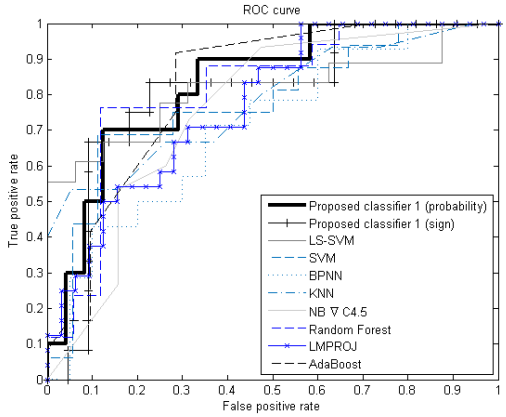
Models	Proposed classifier v1 & v2	LMPROJ	LS-SVM	SVM	Rondom Forest	AdaBoost	BPNN	KNN
Parameter settings	C=150 $\gamma = 2e-2$	$\lambda = 10^{-3}$ $C = 10^4$ $\delta = 1.2$	C=150 $\gamma = 2e-2$	C=200 $\gamma = 2e-2$	number of trees = 15	weak learner = 'Tree' learning cycle = 50	number of hidden neurons=15 learning rate=0.05 momentum=0.9	K=15

Table 4: Performance of the proposed classifiers and comparative methods in the prediction of 5-year overall mortality

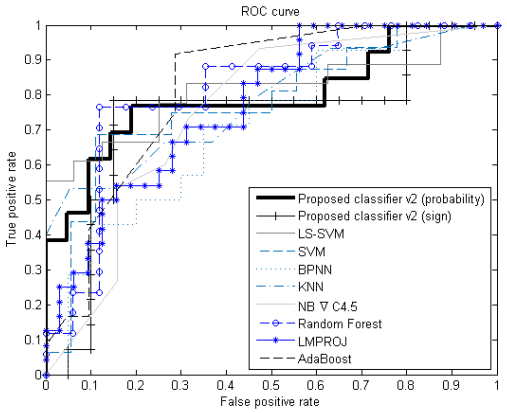
Performance		Proposed classifier v1 (probability)	Proposed classifier v2 (probability)	Proposed classifier v1 (sign)	Proposed classifier v2 (sign)	LMPROJ	LS-SVM	SVM	Random Forest	AdaBoost	BPNN	KNN	NB∇C4.5
Accuracy	Mean	**0.7697**	**0.7618**	0.7471	0.7500	0.7492	0.7424	0.7485	0.7353	0.7265	0.6758	0.7147	0.6882
	SD	0.0456	0.0621	0.0846	0.0541	0.0875	0.0860	0.0655	0.0572	0.0721	0.0516	0.0866	0.0886
	p-value	-	0.5338	$0.0443^{(+)}$	$0.0409^{(+)}$	$0.0406^{(+)}$	$0.0255^{(+)}$	$0.0486^{(+)}$	$0.0221^{(+)}$	$0.0032^{(+)}$	$0.0051^{(+)}$	$0.0027^{(+)}$	$0.0353^{(+)}$
	Max	0.8788	0.8824	0.8820	0.8235	0.9118	0.8182	0.7879	0.8529	0.8235	0.7879	0.8529	0.7941
	Min	0.6970	0.6176	0.6471	0.6471	0.6176	0.6364	0.6364	0.6765	0.6471	0.6061	0.5588	0.4412
Sensitivity	Mean	0.7848	0.7829	0.7646	0.7857	0.7719	0.7805	0.7551	0.7413	0.7447	0.6542	0.7370	0.7455
	SD	0.0678	0.0993	0.0934	0.1040	0.1319	0.1365	0.0806	0.1138	0.1170	0.0900	0.1524	0.1344
Specificity	Mean	0.7579	0.7433	0.7248	0.7226	0.7045	0.7216	0.7507	0.7212	0.7045	0.7119	0.7278	0.6897
	SD	0.0625	0.1121	0.0962	0.1087	0.1071	0.1315	0.1105	0.1294	0.1050	0.1368	0.1287	0.1136
Precision	Mean	0.7805	0.7834	0.7536	0.7827	0.7320	0.7462	0.7672	0.7976	0.7340	0.7485	0.7532	0.7164
	SD	0.0612	0.0901	0.1245	0.0888	0.0976	0.1261	0.1154	0.0830	0.0802	0.1124	0.1108	0.1125
AUC	Mean	**0.8385**	**0.8369**	0.8022	0.8148	0.7478	0.8016	0.7757	0.7720	0.7946	0.6667	0.7939	0.7454
	SD	0.0720	0.0839	0.0557	0.0700	0.0916	0.0676	0.0746	0.0500	0.0751	0.1210	0.0824	0.0871
	p-value	-	0.3985	0.0521	0.0510	$0.0326^{(+)}$	$0.0253^{(+)}$	$0.0424^{(+)}$	$0.0219^{(+)}$	$0.0474^{(+)}$	$0.0029^{(+)}$	$0.0406^{(+)}$	$0.0301^{(+)}$

Table 5: Performance of the proposed classifiers and comparative methods in the prediction of 5-year cancer-specific mortality

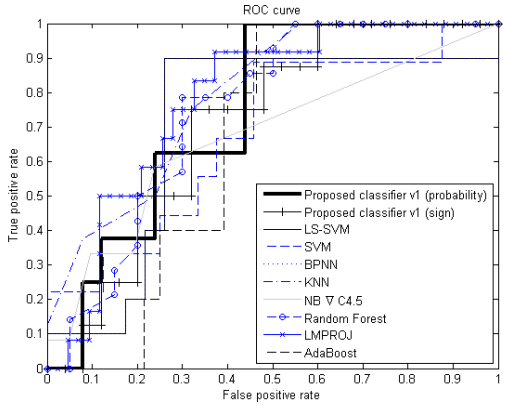
Performance		Proposed classifier v1 (probability)	Proposed classifier v2 (probability)	Proposed classifier v1 (sign)	Proposed classifier v2 (sign)	LMPROJ	LS-SVM	SVM	Random Forest	AdaBoost	BPNN	KNN	NB∇C4.5
Accuracy	Mean	**0.7485**	**0.7515**	0.7242	0.7424	0.7301	0.7182	0.7273	0.7121	0.6909	0.6667	0.7061	0.6788
	SD	0.0453	0.0682	0.0597	0.0848	0.0717	0.0623	0.0589	0.0627	0.0344	0.0742	0.0524	0.0478
	p-value	0.6704	-	$0.0261^{(+)}$	$0.0456^{(+)}$	$0.0261^{(+)}$	$0.0284^{(+)}$	$0.0363^{(+)}$	$0.0084^{(+)}$	$0.0382^{(+)}$	$0.0117^{(+)}$	$0.0285^{(+)}$	$0.025^{(+)}$
	Max	0.8182	0.8788	0.8182	0.9091	0.8182	0.8182	0.8182	0.8485	0.7273	0.7576	0.7879	0.7273
	Min	0.6970	0.6364	0.6364	0.6364	0.5909	0.6364	0.6364	0.6364	0.6364	0.5455	0.6061	0.5758
Sensitivity	Mean	0.9026	0.9238	0.9198	0.8984	0.9073	0.8680	0.7900	0.8353	0.8127	0.7366	0.9028	0.7978
	SD	0.0862	0.0744	0.0884	0.0851	0.0399	0.1018	0.0867	0.0816	0.0585	0.1154	0.0842	0.1223
Specificity	Mean	0.3871	0.3169	0.3094	0.2978	0.1597	0.3245	0.5249	0.4107	0.3180	0.4851	0.2702	0.4674
	SD	0.2056	0.2644	0.1418	0.1530	0.1687	0.2272	0.3536	0.1771	0.1989	0.2008	0.1368	0.2499
Precision	Mean	0.7894	0.7863	0.7537	0.7877	0.7513	0.7856	0.8877	0.7825	0.7956	0.8036	0.7395	0.8270
	SD	0.0743	0.0955	0.0938	0.0731	0.0704	0.0773	0.1019	0.0738	0.0750	0.0824	0.0717	0.0544
AUC	Mean	**0.7670**	**0.7766**	0.7604	0.7521	0.6994	0.7516	0.7416	0.7064	0.7029	0.6781	0.7390	0.6493
	SD	0.0567	0.0952	0.0613	0.1071	0.0875	0.0746	0.1221	0.0840	0.0672	0.0640	0.0926	0.0568
	p-value	0.9265	-	0.3574	0.6623	$0.0108^{(+)}$	$0.0475^{(+)}$	$0.0416^{(+)}$	$0.0387^{(+)}$	$0.0161^{(+)}$	$0.0082^{(+)}$	$0.0270^{(+)}$	$0.0013^{(+)}$



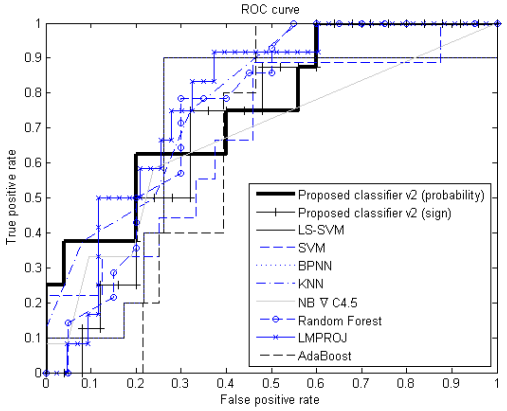
(a) 5-year overall mortality: proposed classifier-v1 and comparative methods



(b) 5-year overall mortality: proposed classifier-v2 and comparative methods



(c) 5-year cancer-specific mortality: proposed classifier-v1 and comparative methods

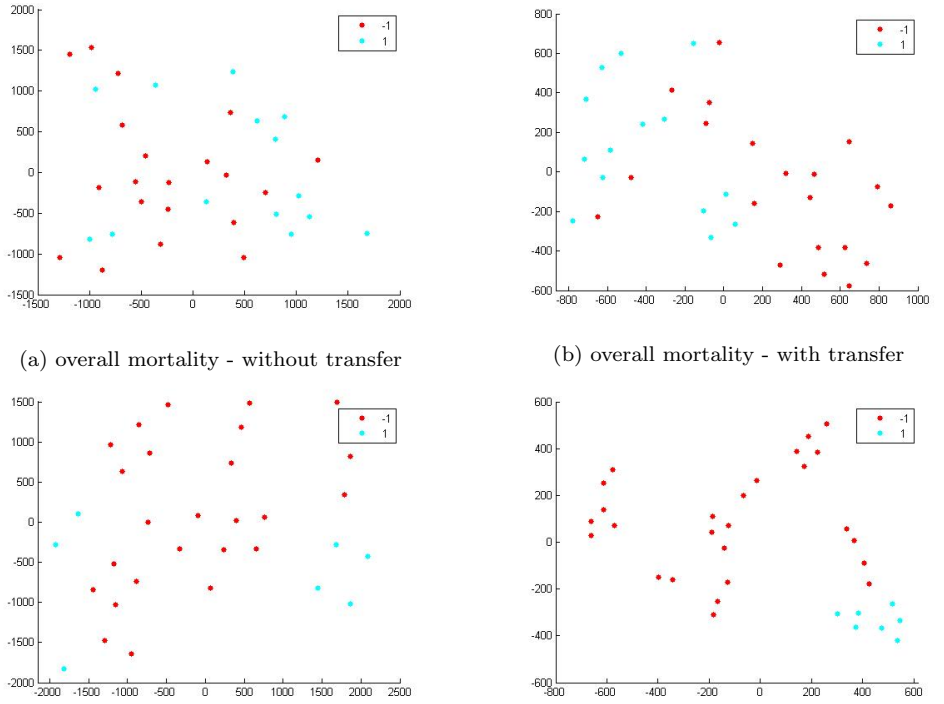


(d) 5-year cancer-specific mortality: proposed classifier-v2 and comparative methods

Figure 4: ROC curves

6. Conclusions and future work

Newly collected clinical data cannot always be put into the existing prediction model, as their features are not exactly matching those in the existing model. Besides, the new data may not be sufficient enough for learning a prediction model from scratch. To overcome these challenges, we propose a novel output-based transfer learning approach in two versions to make the maximum use of small data and guarantee an enhanced generalization capability. The proposed approach can leverage the probabilistic outputs predicted



(a) overall mortality - without transfer

(b) overall mortality - with transfer

(c) cancer-specific mortality - without transfer

(d) cancer-specific mortality - with transfer

Figure 5: Feature visualization: t-SNE without transfer (a) and with transfer (b) for predicting 5-year overall mortality; t-SNE without transfer (c) and with transfer (d) for predicting 5-year cancer-specific mortality.

from the existing model to facilitate better learning in the target domain. Also, our proposed approach has advantages in tuning the weighting parameter autonomously and quickly by using our proposed fast leave-one-out cross validation strategy. The proposed approach is applied to a real-world clinical dataset for predicting bladder cancer 5-year mortality after radical cystectomy. The experimental results show that the proposed approach can work readily with the existing on-line tool and obtain better classification performances than the other comparative methods.

In future, it is worthwhile to evaluate the performances of the proposed approach on different clinical datasets for cancer prediction and prognosis. Furthermore, we can expand our work by incorporating output knowledge learned from multiple existing models to enhance the generalization capability further.

Acknowledgments

The work was supported by the Australian Research Council (ARC) under Discovery Grant DP170101632, Natural Science Foundation of China (Grant No. 61300151), the Research Grants Council of the Hong Kong SAR (PolyU 152040/16E) and G. Wang is supported by Murdoch New Staff Startup Grant (SEIT NSSG).

References

[1] W. A. Knaus, D. P. Wagner, J. Lynn, Short-term mortality predictions for critically ill hospitalized adults: science and ethics, Science 254 (5030) (1991) 389–394 (1991).

[2] L. Ohno-Machado, Modeling medical prognosis: survival analysis techniques, Journal of Biomedical Informatics 34 (6) (2001) 428–439 (2001).

[3] J. A. Cruz, D. S. Wishart, Applications of machine learning in cancer prediction and prognosis, Cancer Informatics 2 (2007) 59–77 (2007).

[4] G. C. Cawley, Leave-one-out cross-validation based model selection criteria for weighted LS-SVMs, in: 2006 International Joint Conference on Neural Networks (IJCNN'06), IEEE, 2006, pp. 1661–1668 (2006).

[5] J. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, J. Vandewalle, Least Squares Support Vector Machine Classifiers, World Scientific, Singapore, 2002 (2002).

[6] J. Gama, P. Brazdil, Cascade generalization, Machine Learning 41 (3) (2000) 315–343 (2000).

[7] S. Karlos, N. Fazakis, S. Kotsiantis, K. Sgarbas, A semisupervised cascade classification algorithm, Applied Computational Intelligence and Soft Computing 2016 (2016) 4 (2016).

[8] S. J. Pan, Q. Yang, A survey on transfer learning, IEEE Transactions on Knowledge and Data Engineering 22 (10) (2010) 1345–1359 (2010).

[9] J. Lu, V. Behbood, P. Hao, H. Zuo, S. Xue, G. Zhang, Transfer learning using computational intelligence: a survey, Knowledge-Based Systems 80 (2015) 14–23 (2015).

[10] J. Huang, A. Gretton, K. M. Borgwardt, B. Schölkopf, A. J. Smola, Correcting sample selection bias by unlabeled data, in: Advances in Neural Information Processing Systems, 2006, pp. 601–608 (2006).

[11] B. Liu, W. S. Lee, P. S. Yu, X. Li, Partially supervised classification of text documents, in: 19th International Conference on Machine Learning, Vol. 2, 2002, pp. 387–394 (July 2002).

[12] Y. Wang, J. Zhai, Y. Li, K. Chen, H. Xue, Transfer learning with partial related 'instance-feature' knowledge, Neurocomputing (2018).

[13] S. J. Pan, I. W. Tsang, J. T. Kwok, Q. Yang, Domain adaptation via transfer component analysis, IEEE Transactions on Neural Networks 22 (2) (2011) 199–210 (2011).

[14] L. Duan, D. Xu, I. Tsang, Learning with augmented features for heterogeneous domain adaptation, in: Proceedings of the 29th International Conference on Machine Learning, Edinburgh, Scotland, UK, 2012, pp. 667–674 (2012).

[15] J. Wang, L. Ke, Feature subspace transfer for collaborative filtering, Neurocomputing 136 (2014) 1–6 (2014).

[16] H. Zuo, G. Zhang, W. Pedrycz, V. Behbood, J. Lu, Fuzzy regression transfer learning in Takagi-Sugeno fuzzy models, IEEE Transactions on Fuzzy Systems (2016).

[17] H. Zuo, G. Zhang, W. Pedrycz, V. Behbood, J. Lu, Granular fuzzy regression domain adaptation in Takagi-Sugeno fuzzy models, IEEE Transactions on Fuzzy Systems (2017).

[18] L. Mihalkova, T. Huynh, R. J. Mooney, Mapping and revising markov logic networks for transfer learning, in: 22nd Conference on Artificial Intelligence, Vol. 7, 2007, pp. 608–614 (July 2007).

[19] E. V. Bonilla, K. M. Chai, C. Williams, Multi-task Gaussian process prediction, in: Advances in Neural Information Processing Systems, 2007, pp. 153–160 (2007).

[20] J. Gao, W. Fan, J. Jiang, J. Han, Knowledge transfer via multiple model local structure mapping, in: Proceedings of the 14th ACM SIGKDD

International Conference on Knowledge Discovery and Data Mining, ACM, 2008, pp. 283–291 (2008).

[21] X. Li, W. Mao, W. Jiang, Extreme learning machine based transfer learning for data classification, Neurocomputing 174 (2016) 203–210 (2016).

[22] Z. Deng, Y. Jiang, K.-S. Choi, F.-L. Chung, S. Wang, Knowledge-leverage-based TSK fuzzy system modeling, IEEE transactions on Neural Networks and Learning Systems 24 (8) (2013) 1200–1212 (2013).

[23] Z. Deng, Y. Jiang, H. Ishibuchi, K.-S. Choi, S. Wang, Enhanced knowledge-leverage-based TSK fuzzy system modeling for inductive transfer learning, ACM Transactions on Intelligent Systems and Technology (TIST) 8 (1) (2016) 11 (2016).

[24] D. L. Silver, R. E. Mercer, The task rehearsal method of life-long learning: overcoming impoverished data, in: Conference of the Canadian Society for Computational Studies of Intelligence, Springer, 2002, pp. 90–101 (2002).

[25] S. Bickel, J. Bogojeska, T. Lengauer, T. Scheffer, Multi-task learning for HIV therapy screening, in: Proceedings of the 25th International Conference on Machine learning, ACM, 2008, pp. 56–63 (2008).

[26] J. Zhou, L. Yuan, J. Liu, J. Ye, A multi-task learning formulation for predicting disease progression, in: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2011, pp. 814–822 (2011).

[27] J. Zhou, J. Liu, V. A. Narayan, J. Ye, A. D. N. Initiative, et al., Modeling disease progression via multi-task learning, NeuroImage 78 (2013) 233–248 (2013).

[28] C. Yang, Z. Deng, K.-S. Choi, Y. Jiang, S. Wang, Transductive domain adaptive learning for epileptic electroencephalogram recognition, Artificial Intelligence in Medicine 62 (3) (2014) 165–177 (2014).

[29] S. Christodoulidis, M. Anthimopoulos, L. Ebner, A. Christe, S. Mougiakakou, Multi-source transfer learning with convolutional neural networks for lung pattern analysis, IEEE Journal of Biomedical and Health Informatics (2017).

[30] G. Wang, Z. Deng, K.-S. Choi, Tackling missing data in community health studies using additive ls-svm classifier, IEEE Journal of Biomedical and Health Informatics 22 (2) (2018) 579–587 (2018).

[31] G. Wang, K.-M. Lam, Z. Deng, K.-S. Choi, Prediction of mortality after radical cystectomy for bladder cancer by machine learning techniques, Computers in Biology and Medicine 63 (2015) 124–132 (2015).

[32] E. Chan, S. Yip, S. Hou, H. Cheung, W. Lee, C. Ng, Age, tumour stage, and preoperative serum albumin level are independent predictors of mortality after radical cystectomy for treatment of bladder cancer in Hong Kong Chinese, Hong Kong Medical Journal 19 (5) (2013) 400–406 (2013).

[33] Nomograms, Nomogram predicting the probability of mortality due to bladder cancer versus other causes.
URL http://labs.fccc.edu/nomograms/nomogram.php?id=48&audience=1.

[34] G. Lughezzani, M. Sun, S. F. Shariat, L. Budäus, R. Thuret, C. Jeldres, D. Liberman, F. Montorsi, P. Perrotte, P. I. Karakiewicz, A population-based competing-risks analysis of the survival of patients treated with radical cystectomy for bladder cancer, Cancer 117 (1) (2011) 103–109 (2011).

[35] B. Quanz, J. Huan, Large margin transductive transfer learning, in: Proceedings of the 18th ACM conference on Information and knowledge management, ACM, 2009, pp. 1327–1336 (2009).

[36] C.-C. Chang, C.-J. Lin, LIBSVM: a library for Support Vector Machines, ACM Transactions on Intelligent Systems and Technology (TIST) 2 (3) (2011) 27 (2011).

[37] M. Buscema, Back propagation neural networks, Substance Use & Misuse 33 (2) (1998) 233–270 (1998).

[38] L. E. Peterson, K-nearest neighbor, Scholarpedia 4 (2) (2009) 1883 (2009).

[39] K. Fernandes, J. S. Cardoso, Hypothesis transfer learning based on structural model similarity, Neural Computing and Applications 3 (2017) 1–14 (2017).

[40] T. Kato, M. Kobayashi, D. Sano, Sign-constrained regularized loss minimization, arXiv preprint arXiv:1710.04380 (2017).