Machine Learning (2024) 113:1839–1860
https://doi.org/10.1007/s10994-022-06245-1

Towards enabling learnware to handle heterogeneous feature spaces

Peng Tan[1] · Zhi-Hao Tan[1] · Yuan Jiang[1] · Zhi-Hua Zhou[1]

Received: 9 June 2022 / Revised: 12 August 2022 / Accepted: 12 September 2022 /
Published online: 28 November 2022

Abstract
The learnware paradigm was recently proposed by Zhou (2016) with the wish of developing the learnware market to help users build models more efficiently by reusing existing well-performed models rather than starting from scratch. Specifically, a learnware in the learnware market is a well-performed pre-trained model with a specification describing its specialty and utility, and the market identifies helpful learnware(s) for the user's task based on the specification. Recent studies have attempted to realize a homogeneous prototype learnware market initially through Reduced Kernel Mean Embedding (RKME) specification, which requires all models in the market to share the same feature space. However, this limits the application scope of the learnware paradigm because various pre-trained models are often obtained from different feature spaces in real-world scenarios. In this paper, we make the first attempt to enable the learnware to handle heterogeneous feature spaces. We propose a more powerful specification to manage heterogeneous learnwares by integrating subspace learning in the specification design, along with a practical approach for identifying and reusing helpful learnwares for the user's task. Empirical studies on both synthetic data and real-world tasks validate the efficacy of our approach.

Keywords Learnware · Heterogeneous feature spaces · Model reuse · Subspace learning

Editors: Yu-Feng Li and Prateek Jain.

✉ Zhi-Hua Zhou
zhouzh@lamda.nju.edu.cn; zhouzh@nju.edu.cn

Peng Tan
tanp@lamda.nju.edu.cn

Zhi-Hao Tan
tanzh@lamda.nju.edu.cn

Yuan Jiang
jiangy@lamda.nju.edu.cn

[1] National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China

1 Introduction

Machine learning techniques have achieved great success in many real-world applications (Butler et al., 2018; Jumper et al., 2021; LeCun et al., 2015). However, except for experienced machine learning experts, most ordinary people can hardly produce well-performed models if starting from scratch, due to a lack of proficient skills or abundant high-quality training data. In addition, although there are numerous powerful pre-trained models, due to data privacy concerns, it is generally difficult for ordinary people to identify beneficial models and apply them to their tasks.

To address these issues simultaneously, Zhou (2016) proposed to develop a *learnware* market, in which a learnware is a pre-trained model combined with a *specification* used to describe the specialty and utility of the model. As described in Zhou and Tan (2022), all developers could submit their trained models from various tasks into the market spontaneously. Once the submitted model is accepted, the market will assign it a specification. When the user wants to tackle her learning tasks, instead of starting from scratch, she can figure out her *requirement* to the market, and the market will identify and recommend helpful learnwares whose specifications match the user's requirement. Then the user can apply these recommended learnwares to her task directly or polish them with minor labeled data.

In the learnware paradigm, the specification plays a pivotal role in identifying which models are helpful for the user's current task, which leads to specification design as a fundamental problem. Recently, Wu et al. (2021) proposed the Reduced Kernel Mean Embedding (RKME) as the specification, based on which several efforts have initially attempted to realize a homogeneous prototype learnware market. The RKME specification makes a good approximation for the distribution of training data used by the model without revealing the raw data. This specification maps the data set to an element in the reproducing kernel Hilbert space (RKHS) which is also called *specification space*. The learnware market can then search helpful learnwares in this space upon the user's request. However, the requirement that all pre-trained models in the market are obtained from the same feature space limits the scope of the learnware market for helping users identify and reuse models on their tasks.

In real-world scenarios, it is more common that the learnware market comprises models from different feature spaces. We take a medical scenario illustrated by Fig. 1 for example. The blue models and orange models are provided by hospitals, using the blood routine features and computed tomography (CT) features respectively for the diagnosis of common diseases. Carcinoembryonic antigen (CEA) and alpha-fetoprotein (AFP) features are used by green models provided by laboratories for cancer-related tasks. For described more practical learnware market, one appealing question comes that *is it possible to identify and reuse helpful models across all models from different feature spaces*, not only models sharing totally the same feature space with the user's task. For example, it is eagerly hoped that the market could recommend helpful learnwares from all heterogeneous learnwares (#1-#6) and that the clinic could receive and reuse recommended learnwares (#1, #5) from two feature spaces for its task.

This paper makes the first attempt to handle learnwares from heterogeneous feature spaces, making the learnware paradigm viable in broader applications. The most notable difference compared with the homogeneous learnware studied before (Wu et al., 2021) is the mismatch of feature spaces between different learnwares and between learnwares and the user's task, which results in harder learnware search and reuse. In this paper,

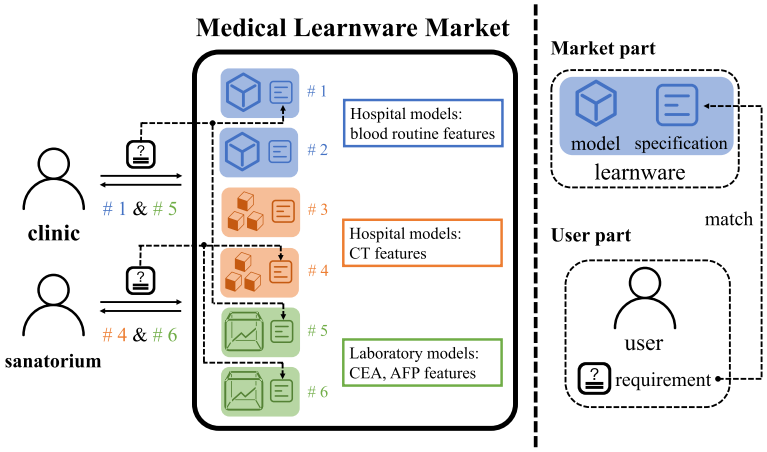


Fig. 1 An example of the learnware market in a medical scenario. A learnware consists of a well-performed pre-trained model and a specification describing its ability. The market is naturally composed of learnwares from *different feature spaces* and different tasks. The market can help the user identify and reuse helpful learnwares upon the user's requirement

we consider a fundamental heterogeneous scenario that *the overall feature space can be divided into several disjoint parts*. For example, the newly-built clinic as the medical learnware market user may collect multiple groups of features (e.g. both blood routine features and CT features) of patients with different machines as illustrated in Fig. 1. In order to realize the heterogeneous learnware search and reuse, it is essential to design a more powerful specification to manage learnwares from different feature spaces. This paper provides a solution by generating the RKME specification on a subspace learned from heterogeneous feature spaces, so as to provide a unified specification space for identifying learnwares matching the user's requirement.

The main contributions of this work can be summarized as follows:

- We give the first formulation for the heterogeneous learnware problem where the overall feature space can be divided into several disjoint parts.
- We propose a more powerful specification that provides a unified specification space for learnwares from heterogeneous feature spaces, where the market identifies helpful learnwares matching user's requirement effectively.
- We provide a detailed procedure for the construction and the usage of the heterogeneous prototype learnware market based on the new specification. Promising experimental results reported on both synthetic and real-world tasks validate the efficacy of the proposed specification and procedure.

The remaining part of the paper proceeds as follows. We briefly review preliminary techniques in Sect. 2. Then, the heterogeneous learnware problem is formulated in Sect. 3, followed by a novel specification design with corresponding procedure for learnware usage in Sect. 4. Next, Sect. 5 provides empirical studies on both synthetic and real-world tasks and Sect. 6 is concerned with related works. Finally, we conclude in Sect. 7.

2 Preliminary

In this section, we first review a specification designed for homogeneous learnwares, which is based on the kernel mean embedding (Smola et al., 2007). Then we review a subspace learning method based on the matrix factorization, which can be extended to finding the unified subspace of heterogeneous feature spaces.

Kernel mean embedding (KME) KME describes the probability distribution in a concise way without information loss and supports convenient operations like mean calculation. It maps a probability distribution $\mathcal{P}$ defined over $\mathcal{X}$ to an element in a reproducing kernel Hilbert space (RKHS) as $\mu_k(\mathcal{P}) := \int_{\mathcal{X}} k(\boldsymbol{x}, \cdot) \mathrm{d}\mathcal{P}(\boldsymbol{x})$, where $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a symmetric and positive definite kernel function (Schölkopf & Smola, 2002) with associated RKHS $\mathcal{H}$ and feature map $\phi : \mathcal{X} \to \mathcal{H}$. The embedding $\mu_k(\mathcal{P})$ exists and belongs to $\mathcal{H}$ when $\mathbb{E}_{x \sim \mathcal{P}}[\sqrt{k(\boldsymbol{x}, \boldsymbol{x})}] < \infty$. When equipped with characteristic kernels such as Gaussian kernel, no information about the distribution $\mathcal{P}$ will be lost (Sriperumbudur et al., 2011). In reality, we can only access to a data set $\{\boldsymbol{x}_n\}_{n=1}^N$ sampled from $\mathcal{P}$, the empirical approximation for KME $\mu_k(\mathcal{P})$ is $\hat{\mu}_k(\mathcal{P}) := \frac{1}{N} \sum_{n=1}^N k(\boldsymbol{x}_n, \cdot)$ with $O(1/\sqrt{N})$ convergence (Smola et al., 2007).

Reduced kernel mean embedding (RKME) The favorable properties of KME make it a potential specification, however, the dependence on the raw data violates the privacy protection that the learnware paradigm needs. To tackle this issue, Wu et al. (2021) proposed the RKME by using a reduced set $\{(\beta_m, \boldsymbol{z}_m)\}_{m=1}^M$ to approximate the empirical KME of the original data set $\{\boldsymbol{x}_n\}_{n=1}^N$ via the following minimization problem:

$$\min_{\boldsymbol{\beta}, \mathbf{z}} \left\| \frac{1}{N} \sum_{n=1}^{N} k(\boldsymbol{x}_n, \cdot) - \sum_{m=1}^{M} \beta_m k(\boldsymbol{z}_m, \cdot) \right\|_{\mathcal{H}}^2, \tag{1}$$

where $\boldsymbol{z}_m$ is the element in the reduced set and β_m is the corresponding coefficient. The RKME $\tilde{\mu}_k(\mathcal{P})$ satisfies a linear convergence rate $O(e^{-M})$ to the empirical KME of original data set $\hat{\mu}_k(\mathcal{P})$.

Subspace learning Subspace learning aims to find a subspace which can better describe the inherent structure of the data compared with the original feature space, and the methods (Lee & Seung, 2001; Zhu et al., 2022) based on the matrix factorization are commonly used. In this part, we review a wildly used technique called concept factorization (CF) (Xu & Gong, 2004). The idea of CF is to represent each concept as a linear combination of instances and reconstruct each instance as a linear combination of concepts. Given a data matrix $\mathbf{X} = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N] \in \mathbb{R}^{d \times N}$ and the number of concepts k, CF aims to find a new representation in the subspace of $\mathbf{X}$ as $\mathbf{V}^\top \in \mathbb{R}^{k \times N}$ by minimizing the reconstruction error via

$$\min_{\mathbf{W}, \mathbf{V}} \|\mathbf{X} - \mathbf{X}\mathbf{W}\mathbf{V}^\top\|_{\mathrm{F}}^2, \quad \text{s.t. } \mathbf{W}, \mathbf{V} \geq 0.$$

where $\mathbf{C} := \mathbf{X}\mathbf{W} = [\boldsymbol{c}_1, \cdots, \boldsymbol{c}_k] \in \mathbb{R}^{d \times k}$ is the concept matrix, $\mathbf{V}$ is the reconstruction coefficient matrix and can also be explained as the projection of the input matrix in the subspace, $\mathbf{W}, \mathbf{V} \geq 0$ means all elements of $\mathbf{W}, \mathbf{V}$ is non-negative. The performance of CF can be improved by local structure maintenance (Cai et al., 2010; Wang et al., 2016).

3 Problem formulation

The learnware paradigm generally consists of the submitting stage and the deploying stage. In the submitting stage, the developers submit their models to the market, and the market will assign the specification for accepted high-quality learnwares and accommodate them in the market. In the deploying stage, the market identifies and returns learnwares matching the user's requirement and the user reuses them directly or polishes them with the user's data. For the heterogeneous learnware problem, since the relationship of various feature spaces needs to be mined first by exploiting some aligned data, we simplify the submitting stage as an establishing stage where the market uses several owned models for initialization, and thus, the market is accessible to the raw data of the models which help connect different feature spaces better. Note that the raw data of the models in establishing stage is still invisible to users, matching the data privacy concern of the learnware paradigm. The two-staged formulation is illustrated in Fig. 2 and described as follows.

In the *establishing stage*, the market accommodating well-trained models generated from different feature spaces will assign specifications in a *shared* space for models. This shared specification space makes it possible and effective to match the user's requirements in the next deploying stage. We first assume that the overall feature space $\mathcal{X}$ can be split into k disjoint parts, i.e., $\mathcal{X}_1, \cdots, \mathcal{X}_k$. Supposing that the market has R models $\{f_i\}_{i=1}^{R}$ and the i-th model is trained on the local data set $D_i := \{(\boldsymbol{x}_{i,n}, y_{i,n})\}_{n=1}^{N_i}$ whose feature space is $\mathcal{X}_{v_i}, v_i \in [k]$. Temporarily, each local data set only provides information of single feature space and no relationship between different feature spaces can be uncovered, which makes it impossible to find a shared specification space. To tackle this troublesome difficulty, a few data across different feature spaces is necessary. In reality, auxiliary data crossing different feature spaces is often accessible. For example, there are abundant multi-modal data (Chen et al., 2015) on the web to connect the figures with texts. Another more detailed

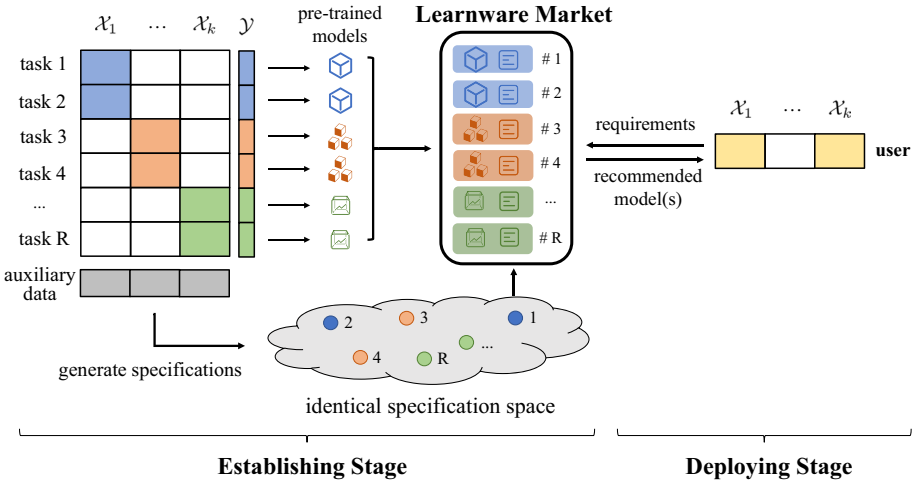


Fig. 2 An illustration for two-stage heterogeneous learnware problem formulation. To initialize the learnware market, the market assigns specifications in the *same* space for models constructed from *different* feature spaces. In the establishing stage, auxiliary data across entire feature spaces is necessary to reveal the relationship between different feature spaces. In the deploying stage, the user's task is defined over the *Cartesian product* of some feature spaces and the user can reuse helpful learnwares from the market

example is medical data from different organizations (hospital, clinic laboratory) serving as different feature spaces which is generated from the same patients (Johnson et al., 2016). After the market collects such *unlabeled* auxiliary data defined on the entire feature space $\mathcal{X}$ from existing satisfied web data, the heterogeneous learnware market is constructed as $\{(f_i, \boldsymbol{s}_i)\}_{i=1}^{R}$, where $\boldsymbol{s}_i$ is the specification of the model f_i.

In the *deploying stage*, the user hopes to exploit heterogeneous market $\{(f_i, \boldsymbol{s}_i)\}_{i=1}^{R}$ to handle her own task. Specifically, we consider the scenario that the user hopes to make a prediction on her unlabeled data set $D_u := \{\boldsymbol{x}_{u,n}\}_{n=1}^{N_u}$ defined over a Cartesian product of several feature spaces $\mathcal{X}_{v_u} := \times_{i \in v_u \subseteq [k]} \mathcal{X}_i$. For example, the feature space of the user in Fig. 2 composed of the 1st and k-th feature space is $\mathcal{X}_{\{1,k\}}$.

4 Our approach

We first sketch the overall procedure. The specification design is discussed in the *establishing stage* and the *deploying stage* shows how to use the specification to meet the requirements of the user. In the establishing stage, the market assigns specifications in a union space based on the subspace learning and RKME for models from different feature spaces to construct the learnware market. In the deploying stage, the user generates her requirement in the subspace with the projection tool provided by the market and the market identifies highly-relevant learnwares. After that, the user reuses learnwares via dynamic classifier selection. The main idea of our approach is to find a subspace to bridge different feature spaces with the shared specification space (RKHS).

4.1 Establishing stage

In this stage, in order to build a union specification space, the market first constructs a union subspace for local tasks from different feature spaces. When the i-th task data set is mapped to the subspace, we generate RKME for such mapped data set as the i-th model's specification $\mathbf{s}_i$. Meanwhile, we hope the procedure of subspace generation can provide applicable tools to map the user data in the following deploying stage.

Subspace generation In this step, the market finds a common subspace for different feature spaces based on the local data sets each defined on a single feature space and extra auxiliary data across the entire feature space. We denote by $\mathbf{X}_i \in \mathbb{R}^{d_i \times N_i}$ the feature of the i-th local data set $\boldsymbol{D}_i (i \in [R])$, and further denote by $\hat{\mathbf{X}}^{(i)} \in \mathbb{R}^{d_i \times N^{(i)}}$ the concatenation of all data sets in the i-th feature space, in above, d_i is the dimension of i-th feature space $\mathcal{X}_i$ and $N^{(i)}$ is the total number of samples for all data sets in the i-th feature space. The extra auxiliary data is denoted as $\mathbf{X}_c = [\mathbf{X}_c^{(1)}; \cdots; \mathbf{X}_c^{(k)}] \in \mathbb{R}^{(d_1 + \cdots + d_k) \times N_c}$ where N_c is the size of auxiliary data. Then, the overall data containing the local tasks and auxiliary data in the i-th feature space is $\mathbf{X}^{(i)} = [\hat{\mathbf{X}}^{(i)}, \mathbf{X}_c^{(i)}] \in \mathbb{R}^{d_i \times (N^{(i)} + N_c)}$. The problem of subspace generation using local task data sets and extra auxiliary data is defined as

Table 1 Main notations and corresponding definitions of the subspace generation

Notation	Definition
$\hat{\mathbf{X}}^{(i)} \in \mathbb{R}^{d_i \times N^{(i)}}$	The concatenation of all the locak task data sets in the i-th feature space $\mathcal{X}_i$
$\mathbf{X}_c^{(i)} \in \mathbb{R}^{d_i \times N_c}$	The sliced auxiliary data in the i-th feature space
$\mathbf{X}^{(i)} = [\hat{\mathbf{X}}^{(i)}, \mathbf{X}_c^{(i)}]$	The overall data in the i-th feature space $\mathcal{X}_i$
$(\hat{\mathbf{V}}^{(i)})^\top \in \mathbb{R}^{d \times N^{(i)}}$	The projection of local task data $\hat{\mathbf{X}}^{(i)}$ in the subspace
$(\mathbf{V}_c^{(i)})^\top \in \mathbb{R}^{d \times N_c}$	The projection of sliced auxiliary data $\mathbf{X}_c^{(i)}$ in the subspace
$(\mathbf{V}^{(i)})^\top = [(\hat{\mathbf{V}}^{(i)})^\top, (\mathbf{V}_c^{(i)})^\top]$	The projection of $\mathbf{X}^{(i)}$ in the subspace
$(\mathbf{V}_c^*)^\top \in \mathbb{R}^{d \times N_c}$	The projection of the entire auxiliary data $\mathbf{X}_c = [\mathbf{X}_c^{(1)}; \cdots; \mathbf{X}_c^{(k)}]$ in the subspace

$$\begin{aligned} \min_{\mathbf{W}^{(i)}, \mathbf{V}^{(i)}, \mathbf{V}_c^*} O = & \sum_{i=1}^{k} \left\{ \left\| \mathbf{X}^{(i)} - \mathbf{X}^{(i)} \mathbf{W}^{(i)} \left(\mathbf{V}^{(i)} \right)^\top \right\|_{\mathrm{F}}^2 \right. \\ & \left. + \alpha \mathrm{Tr}\left(\left(\mathbf{V}^{(i)} \right)^\top \mathbf{L}^{(i)} \mathbf{V}^{(i)} \right) + \beta \left\| \mathbf{V}_c^{(i)} - \mathbf{V}_c^* \right\|_{\mathrm{F}}^2 \right\} \\ & \text{s.t. } \mathbf{W}^{(i)} \geq 0, \hat{\mathbf{V}}^{(i)} \geq 0, \mathbf{V}_c^{(i)} \geq 0, \mathbf{V}_c^* \geq 0, \end{aligned} \tag{2}$$

and the major notations are summarized in Table 1. The first item $\|\mathbf{X}^{(i)} - \mathbf{X}^{(i)}\mathbf{W}^{(i)}(\mathbf{V}^{(i)})^\top\|_{\mathrm{F}}^2$ presents the reconstruction loss of concept factorization loss, where $(\mathbf{V}^{(i)})^\top \in \mathbb{R}^{d \times (N^{(i)}+N_c)}$ is the new representation of $\mathbf{X}^{(i)}$ in the subspace whose dimension is d. The second item $\mathrm{Tr}((\mathbf{V}^{(i)})^\top \mathbf{L}^{(i)} \mathbf{V}^{(i)})$ is a manifold regularizer used to maintain the local structure during the mapping, forcing similar outputs when inputs are closed. In which, $\mathbf{L}^{(i)} \in \mathbb{R}^{(N^{(i)}+N_c) \times (N^{(i)}+N_c)}$ is the Laplacian matrix induced by $\mathbf{X}^{(i)}$. The last item $\|\mathbf{V}_c^{(i)} - \mathbf{V}_c^*\|_{\mathrm{F}}^2$ reveals the internal consensus between different feature spaces, punishing the inconsistency of different mapped auxiliary data, where $\mathbf{V}_c^{(i)}$, a part of $(\mathbf{V}^{(i)})^\top = [(\hat{\mathbf{V}}^{(i)})^\top, (\mathbf{V}_c^{(i)})^\top]$, is the mapped auxiliary data of i-th feature space, $\mathbf{V}_c^*$ is the final representation of the auxiliary data $\mathbf{X}_c$ in the subspace.

Algorithm 1 Sketched optimization of subspace generation

```
Initialize W^(i), V^(i) and V_c^* with results of k-means.
while max iteration is not achieved do
    for i = 1 to k do
        Fix V^(i), V_c^*, update W^(i).
        Fix W^(i), V_c^*, update V^(i) = [V_c^(i), V̂^(i)] with decomposed tasks.
        Cooperatively normalize W^(i) and V^(i).
    end for
    Fix W^(i), V^(i), i ∈ [k] and update V_c^* with closed-form solution.
end while
```

The objective function Eq. (2) of subspace generation is not convex over all variables $\mathbf{W}^{(i)}, \mathbf{V}^{(i)}, \mathbf{V}_c^*$, which makes it unrealistic to find its global minimum. Therefore, we propose an alternative optimization algorithm based on the multiplicative updated rule similar to (Févotte & Idier, 2011; Xu & Gong, 2004) with local minimum achieved. We provide the sketched optimization as Algorithm 1 shows and details are presented in Appendix A.1.

Specification assignment Based on the subspace generation, we can eventually design more powerful specifications. With the help of auxiliary data, we connect different feature

spaces with a common subspace, and further develop specifications on such space. After subspace generation, the local data sets $\{\mathbf{X}_i\}_{i=1}^R$ used by models $\{f_i\}_{i=1}^R$ are mapped to $\{\mathbf{V}_i\}_{i=1}^R$, which are achieved by splitting $\{\mathbf{V}^{(i)}\}_{i=1}^k$. We generate RKME $\mathbf{s}_i = \{(\gamma_m, \boldsymbol{w}_m)\}_{m=1}^M$ for each $\mathbf{V}_i := \{\boldsymbol{v}_n\}_{n=1}^{N_i}$ as the specification for i-th model via

$$\min_{\boldsymbol{\gamma},\mathbf{w}} \left\| \frac{1}{N} \sum_{n=1}^{N_i} k(\boldsymbol{v}_n, \cdot) - \sum_{m=1}^{M} \gamma_m k(\boldsymbol{w}_m, \cdot) \right\|_{\mathcal{H}}^2, \tag{3}$$

where M is the size of reduced set. The minimization problem Eq. (3) can be solved with stochastic gradient descent (Wu et al., 2021).

The proposed specification provides models from various feature spaces with a shared specification space (RKHS), which makes the learnwares identification upon the user's requirements concerning several feature spaces possible and effective. Furthermore, the specification can protect the original data set efficaciously. More specifically, the size of the specification M can be much smaller than that of original data set N_i and it is impossible to utilize the specification $\mathbf{s}_i = \{(\gamma_m, \boldsymbol{w}_m)\}_{m=1}^M$ to recover the mapped data set $\mathbf{V}_i$ and the original data set $\mathbf{X}_i$.

4.2 Deploying stage

In this stage, the user tries to exploit the market to tackle her prediction task while preserving data privacy. The user maps her data via the projection tool provided by the market and generates a reduced set accordingly served as user's *requirements*. After receiving requirements, the market recommends highly-reusable learnwares to the user and the user reuses them on her task.

User data mapping In this step, the user produces requirements to the market for learnware recommendation and protects privacy in the meanwhile. The market first passes the projection tool to help the user map her data and the projection tool $\{\mathbf{B}^{(i)}\}_{i=1}^k$ consists of base matrices of all feature space, which is generated from the byproduct $\mathbf{W}^{(i)}$ of subspace learning via $\mathbf{B}^{(i)} = \mathbf{X}^{(i)}\mathbf{W}^{(i)}$.

After receiving the projection tool $\{\mathbf{B}^{(i)}\}_{i=1}^k$, the user can map her data into the same subspace. Without loss of generality, we assume that the user has data over top t feature spaces $\mathcal{X}_u = \mathcal{X}_1 \times \cdots \times \mathcal{X}_t$ $(t \leq k)$, we denote by $\mathbf{X}_u = [\mathbf{X}_u^{(1)}; \cdots; \mathbf{X}_u^{(t)}] \in \mathbb{R}^{(d_1+\cdots+d_t)\times N_u}$ the user data, where N_u is the size of user data. The projection of user data is formulated as

$$\begin{aligned} \min_{\mathbf{V}^{(i)},\mathbf{V}^*} O_u = & \sum_{i=1}^{t} \left\{ \left\| \mathbf{X}_u^{(i)} - \mathbf{B}^{(i)} \left(\mathbf{V}^{(i)}\right)^\top \right\|_{\mathrm{F}}^2 \right. \\ & \left. + \alpha \mathrm{Tr}\left(\left(\mathbf{V}^{(i)}\right)^\top \mathbf{L}^{(i)} \mathbf{V}^{(i)} \right) + \beta \left\| \mathbf{V}^{(i)} - \mathbf{V}^* \right\|_{\mathrm{F}}^2 \right\} \\ \text{s.t. } & \mathbf{V}^{(i)} \geq 0, \mathbf{V}^* \geq 0, \end{aligned} \tag{4}$$

where trade-off parameters $\alpha, \beta > 0$ are used to control the contribution of the manifold regularizer and the consensus loss between different feature spaces, $\mathbf{L}^{(i)}$ is the Laplacian matrix induced by $\mathbf{X}_u^{(i)}$ and $\mathbf{V}^*$ is the final representation of user data in the subspace. This problem has a similar structure to Eq. (2). One of major differences is that Eq. (4) possesses the fixed base matrix $\mathbf{B}^{(i)}$ while Eq. (2) contains the learned base matrix $\mathbf{X}^{(i)}\mathbf{W}^{(i)}$. The optimization using the multiplicative updated rule is described in Appendix A.2.

In order to keep the data privacy, the user only passes the reduced set $r_u := \{\beta_{u,m}, \mathbf{z}_{u,m}\}_{m=1}^{M_u}$ constructed from mapped data set $V^* := \{\mathbf{v}_n\}_{n=1}^{N_u}$ via Eq. (1) as *requirements* to the market,

where M_u is the size of user's reduced set and can be much smaller than the original data set size N_u. Meanwhile, such minor information can still guarantee the performance of learnware recommendation.

Learnware recommendation In this step, the market identifies useful learnwares for the user. After receiving the user's requirement $\mathbf{r}_u = \{\beta_{u,m}, \boldsymbol{z}_{u,m}\}_{m=1}^{M_u}$, the learnware market estimates the reusability score w_i of each learnware via according specifications $\{\mathbf{s}_i := \{\beta_{i,m}, \boldsymbol{z}_{i,m}\}_{m=1}^{M_i}\}_{i=1}^{R}$. The reusability score is estimated by the following problem:

$$\min_{\boldsymbol{w}} \left\| \Phi_u(\cdot) - \sum_{i=1}^{R} \omega_i \Phi_i(\cdot) \right\|_{\mathcal{H}}^2, \text{ s.t. } \omega_i \geq 0, \sum_{i=1}^{R} \omega_i = 1,$$

where $\Phi_u(\cdot) = \sum_{m=1}^{M_u} \beta_{u,m} k(\boldsymbol{z}_{u,m}, \cdot)$ is the KME of user's requirements, $\Phi_i(\cdot) = \sum_{m=1}^{M_u} \beta_{i,m} k(\boldsymbol{z}_{i,m}, \cdot)$ represents the specification of the i-th model. This problem can be solved by quadratic programming (Smola et al., 2007). After reusability score estimation, the market delivers highly reusable learnwares whose reusability score ω_i is no less than the pre-defined threshold L to the user. This step makes the user only access to highly relevant learnwares, heavily reducing the exchanged information between the market and the user when the market possesses plentiful learnwares.

User data prediction After the learnware market returns highly-related learnwares $\{(f_i, \mathbf{s}_i) | w_i \geq L\}$, the user can take full advantage of these well-performed models for her problem. The specification based on the subspace can help the user to identify which model should be used for each instance. More specifically, the *kernel herding* technique (Chen et al., 2012) can be employed to sample mimic data set $\bar{\boldsymbol{D}}_i = \{\mathbf{x}_n\}_{n=1}^{\bar{N}_i}$ from the specification $\mathbf{s}_i$ on the subspace, and then, a selector can be trained based on these data to predict which learnware should each unlabeled instance use. For example, the projection of one instance is classified by using the #1-learnware, then the user can reuse such learnware defined on the original feature space to predict on the instance. Above all, the user predicts unlabeled data via returned learnwares and trained selector.

4.3 Overall procedure and discussion

This part first summarizes the establishing stage for the heterogeneous learnware market construction and deploying stage for learnwares search and reuse. In the establishing stage, the market needs to specify the specification in an identical space for heterogeneous learnwares, which is implemented by using a subspace to bridge the heterogeneous feature spaces and the identical specification space. More specifically, specifications are generated via RKME on the mapped local task data obtained by the CF-based method. In the deploying stage, the user generates the subspace-based requirement, which is a reduced set built on the mapped user data generated by the projection tool (base matrices of different feature spaces) provided by the market. The overall procedure is illustrated by Fig. 3.

Afterwards, we give a preliminary discussion on the performance of the two-stage procedure. When the user's task is covered by the market, the proposed method can make the user well assisted by the learnware market on her task. More specifically, considering learnwares whose feature space $\mathcal{X}_{v_i}$ is a subset of the user's feature space $\mathcal{X}_{v_u}$, if the distribution of the user's task $\mathcal{P}_u$ is a mixture of task distribution of aforementioned learnwares, i.e., $\mathcal{P}_u = \sum_{i:\mathcal{X}_{v_i} \subseteq \mathcal{X}_{v_u}} \xi_i \mathcal{P}_i$, then our procedure can make the market select out learnwares with large ξ_i and accurately tell the user for each instance which learnware should be used. Besides this basic scenario, our methods can be extended to solve more complicated

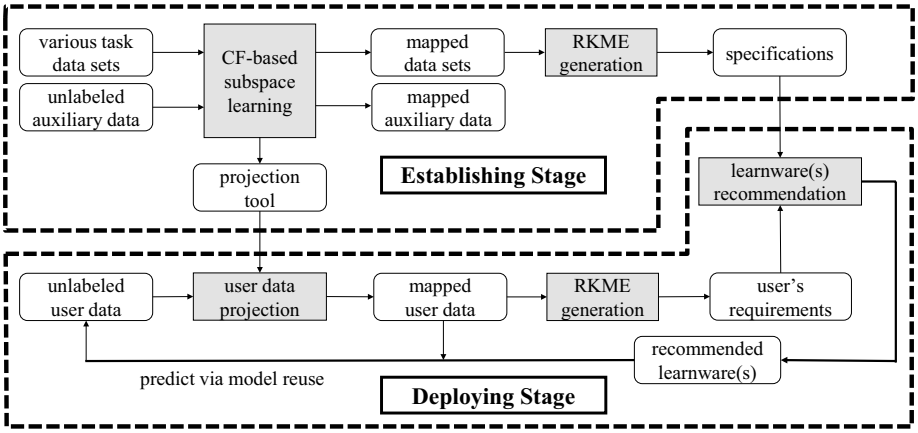


Fig. 3 An illustration of the proposed heterogeneous learnware procedure. In the establishing stage, various task data sets from heterogeneous feature spaces are mapped to a common subspace and specifications are built accordingly. In the deploying stage, the user passes requirements with the help of the projection tool to the market and gets helpful learnwares

scenarios. For example, the user's task may have a part that isn't covered by the market. For the homogeneous case where all learnwares and the user's task share the same feature space, (Zhang et al., 2021) uses the mixture proportion estimation (MPE) technique (Zhang et al., 2020; Ramaswamy et al., 2016) to help identify such a unseen part and this technique can be further adapted to the heterogeneous case.

5 Experiments

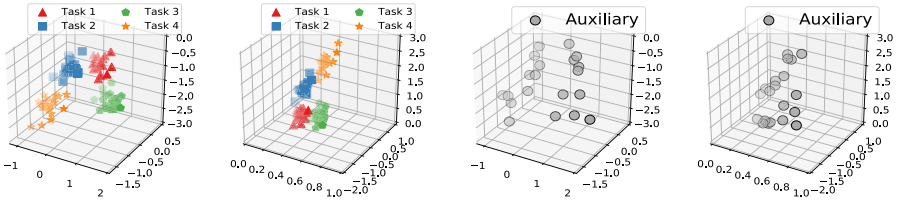
This section demonstrates the effectiveness of our methods. We illustrate our methods by a synthetic task and show the performance on real-world tasks compared with contenders.

5.1 Synthetic task

We first illustrate the two stages of the heterogeneous learnware market workflow through a synthetic task.

In the beginning, we develop four tasks defined on the Cartesian product of two feature spaces plotted by four colors (red, blue, green and orange) illustrated by Fig. 4a and b, each task is a binary classification problem and is generated from the Gaussian distribution. These tasks will be used to generate four local task data sets (Fig. 5a and b), the auxiliary data (Fig. 4c and d) and the test user data (Fig. 6a and b) of our heterogeneous learnware problem.

The *unlabeled* auxiliary data is sampled from four tasks with a size of 20 as shown in Fig. 4c and d. Each local task is sampled from one task and reserves the data of a *single feature space*. Each task has 200 samples, and positive samples are plotted in yellow while negative samples are plotted in blue. All tasks are shown in Fig. 5a and b. Figure 6a and b

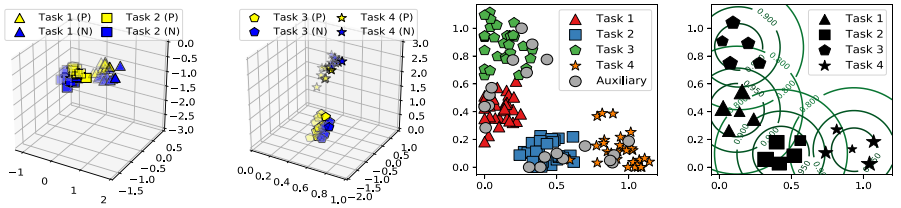


(a) Feature space 1 **(b)** Feature space 2 **(c)** Feature space 1 **(d)** Feature space 2

Fig. 4 Basic data source: **a** and **b** present four tasks on two feature spaces, **c** and **d** present the sampled auxiliary data implying the connection of two feature spaces

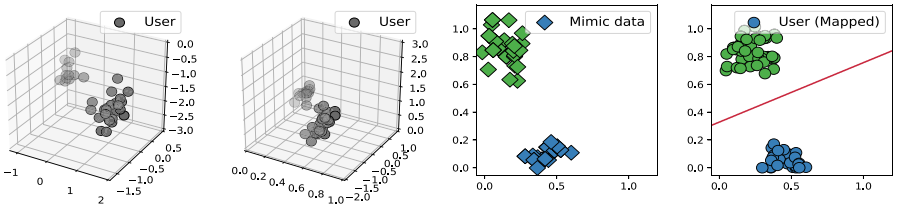
illustrate the test user data containing 100 samples, which is a mixture of task 2 and task 3 with sampling ratios of 0.3 and 0.7 accordingly.

Figure 5 illustrates the *establishing stage*. The market accommodates SVMs trained on four heterogeneous data sets from two feature spaces (Fig. 5a and b), the accuracy of SVMs is 0.967 ± 0.015. To assign the specification for each model, the market collects minor unlabeled auxiliary data across the entire feature space, which can build the connection of different feature spaces. With the help of the auxiliary data, four local data sets from two feature spaces can be mapped to a common subspace as illustrated in Fig. 5(c). Figure 5(d) shows the specification assigned for each SVM via RKME using mapped task data. The size of specification is $M = 5 < \lfloor \ln 200 \rfloor$, which is much smaller than the size



(a) Feature space 1 **(b)** Feature space 2 **(c)** Joint subspace **(d)** Joint subspace

Fig. 5 Establishing stage: **a** and **b** present four tasks generated from aforementioned four tasks, which are used for building learnwares (#1-#4), **c** illustrates the subspace learned by four task data sets and auxiliary data, **d** illustrates specifications in the subspace



(a) Feature space 1 **(b)** Feature space 2 **(c)** Joint subspace **(d)** Joint subspace

Fig. 6 Deploying stage: **a** and **b** present the user data generated from task 2 and 3, **c** illustrates the mimic data generated from two learnwares (#2 and #3) delivered from the market, **d** illustrates the model selector (red line shows the classified boundary.) trained on the mimic data, providing which learnware should each sample use (#2 or #3)

of the original data set. The specifications can effectively protect the privacy of each local task.

Figure 6 illustrates the *deploying stage*. After the learnware market was constructed, the user can exploit it for her own task (Fig. 6a and b). Although the user only possesses the *unlabeled* data, she can still make a valid and satisfying prediction with the help of learnware markets under some mild assumptions. Using projection tools provided by the market, the user gets her mapped data (Fig. 6d) and generates the reduced set with a size of $M_u = 5$ to the market. The reduced set can keep user data privacy and provide helpful information for market identifying helpful learnwares in the meanwhile. Based on specifications, the market calculates reusability of four learnwares: [0.001, 0.399, 0.559, 0.042]. With the preset threshold $L = 0.1$, the market delivers learnware #2 and #3 to the user. Upon receiving the helpful learnwares, the user takes advantage of specifications of two learnwares to generate some mimic data (Fig. 6(c)) and trains a model selector that tells for each sample what learnware should be used (Fig. 6(d)), More specifically, learnware #2 is chosen for the lower samples and learnware #3 is chosen for the upper samples. With prediction on the original feature spaces using learnware #2 and #3, the accuracy on user data is 0.987.

5.2 Real-world tasks

We further verify the superior performance of the proposed specification and the procedure designed for the heterogeneous learnware problem on practical tasks.

Data sets The evaluation is conducted on four real world tasks: MFEAT (van Breukelen et al., 1998), AWA (Lampert et al., 2009), KDDCUP99 (Lippmann et al., 2000) and COVTYPE (Blackard & Dean, 1999). MFEAT is a digit data set containing 10 classes over six feature spaces: Fourier coefficients (fou.), profile correlations (fac.), Karhunen-Love coefficients (kar.), pixel averages (pix.), Zernike moments (zer.) and morphological features (mor.). Each class contains 200 samples and the dimensions of feature spaces are 76, 216, 64, 240, 47, 6. AwA is a large-scale animal data set, we randomly select 10 classes with totally 2000 samples over six feature spaces: color histogram features (cq.), local self-similarity features (lss.), PyramidHOG features (phog.), SIFT features (sift.), colorSIFT features (rgsift.) and SURF features (surf.). The dimensions of feature spaces are 2688, 2000, 252, 2000, 2000, 2000. KDDCUP99 is a unbalanced network intrusion detection dataset, we sample a balanced subset containing 8 classes and each class contains 1000 samples. COVTYPE is used for classifying the cover type (the dominant species of trees) of the patches of forest in the United States. We sample a balanced dataset containing 6 classes and each class has 1500 samples.

Data set configuration For MFEAT and AWA defined over six feature spaces, we choose two feature spaces to simulate the learnware scenario and generate totally six heterogeneous learnware tasks: MFEAT_fac_kar, MFEAT_pix_zer, MFEAT_fou_mor, AWA_cq_lss, AWA_phog_rgsift and AWA_sift_surf. For KDDCUP99 and COVTYPE, we randomly divide the entire feature space into two parts to simulate the heterogeneous learnware problem. For each task, we randomly split instances into three or four tasks based on their labels, each task has two or three classes. The mechanism of generating local data sets, the auxiliary data, and the test user data is similar to the synthetic task. The test user data is a mixture of several tasks. The number of mixed tasks ranges from 2 to 4.

Contenders As the heterogeneous learnware problem is new, we first compare it with two naive baselines. In these two methods, heterogeneous models are assigned with *blank* specifications.

- `Random`: The market randomly selects a learnware for the user and the user reuses it to make a prediction directly.
- `Ensemble`: The market returns *all* learnwares to the user without filtering and the user reuses them via ensemble, i.e., using all learnwares to predict one test instance and taking out the most confident predicted class.

Observing that both two methods don't consider the relevance between learnwares and the user's task. We equip each heterogeneous model with specification via *RKME using raw features*, and followed by delivering the learnwares with minimum MMD distance under the same feature space case proposed by Wu et al. (2021), we propose two variants for the heterogeneous case. For both methods, the user will also generate the reduced set on her original feature space to the market.

- `MMD`: The market calculates the MMD distance of each learnwares and returns the learnware with the minimum MMD to the user. The user reuses the single learnware to make a prediction.
- `MMD+Ens`: The market calculates the MMD distance of each learnwares and returns learnwares with the minimum MMD for each feature space. The user reuses learnwares via ensemble.

Experiment setup For all RKME-based methods, we use the Gaussian kernel $k(x, y) = \exp(-\gamma\|x - y\|_2^2)$ with $\gamma \in [0.1, 0.01, 0.001]$ for different tasks. The size of reduced set is 10 for specifications ($M = 10$) and user's requirements ($M_u = 10$). We set the dimension of subspace as 10 for MFEAT-based tasks, KDDCUP99, COVTYPE and 50 for AWA-based tasks, which are much smaller than the dimension of original feature spaces. The auxiliary data consists of 160 samples, less than the size of local tasks. The threshold is set as $L = 0.1$. We use the linear SVM for the MFEAT-based tasks, KDDCUP99 and COVTYPE. We use random forest for the AWA-based tasks. All experiments are repeated 10 times.

Performance on user data Table 2 presents the prediction accuracy over the *true labels* on the user data. Our method outperforms other contenders, it achieves the best on the 22 over 23 cases, and it behaves significantly better than others in most cases, especially for MFEAT-based tasks and KDDCUP99 task. The `Random` performs poorly with low mean accuracy and large variance mainly due to selecting models aimlessly. `Ensemble` performs better than other three contenders because of making the user access all learnwares, however, this leaks information of irrelevant learnwares. Furthermore, when the market has abundant learnwares, it causes heavy burden on passing the learnware information and heavily expands the complexity of reusing learnwares. Compared with `Ensemble`, our method can make the user only access to the highly irrelevant learnwares. `MMD` and `MMD+Ens` performs well than `Random` in more than half of cases. With the help of distribution matching via RKME, `MMD` and `MMD+Ens` identify more reliable learnwares for the user. However, due to a lack of considering the relationship of different feature spaces, it's still hard to identify truly helpful learnwares and performs much poorly than our methods except for one case.

Convergence analysis Figure 7 presents the convergence curve for the major optimization steps of our procedure, i.e., subspace learning and user data projection. The objective loss is

Table 2 Accuracy (mean ± std.) on true labels of the user data.

Task name	#Mix[1]	Random	Ensemble	MMD	MMD+Ens	Ours
MFEAT_fac_kar	2	29.91 ±24.43	58.72 ±11.75	49.60 ± 1.68	34.22 ± 22.47	**78.20 ± 10.41**
	3	22.27 ± 14.58	62.53 ± 7.33	32.17 ± 0.39	41.77 ± 13.66	**71.57 ± 6.67**
	4	24.00 ± 0.00	63.50 ± 0.00	24.00 ± 0.00	24.50 ± 0.00	**75.50 ± 0.00**
MFEAT_pix_zer	2	30.53 ± 24.93	62.40 ± 9.45	37.19 ± 18.78	34.84 ± 22.85	**84.73 ± 6.01**
	3	22.98 ± 15.04	63.48 ± 5.58	32.22 ± 0.74	42.22 ± 13.69	**86.41 ± 1.71**
	4	24.50 ± 0.00	60.00 ± 0.00	24.50 ± 0.00	25.00 ± 0.00	**84.00 ± 0.00**
MFEAT_fou_mor	2	29.60 ± 24.18	33.40 ± 14.54	28.96 ± 23.65	25.63 ± 24.48	**47.21 ± 11.09**
	3	22.63 ± 14.81	38.74 ± 9.22	20.00 ± 16.33	36.72 ± 13.01	**54.39 ± 4.48**
	4	24.00 ± 0.00	38.50 ± 0.00	25.00 ± 0.00	35.00 ± 0.00	**49.50 ± 0.00**
AWA_cq_lss	2	23.43 ± 19.14	24.96 ± 10.78	**36.25 ± 3.50**	26.30 ± 17.49	26.25 ± 4.98
	3	22.27 ± 7.42	26.62 ± 4.75	23.84 ± 1.39	24.09 ± 1.97	**28.84 ± 2.54**
	4	17.50 ± 0.00	23.00 ± 0.00	17.50 ± 0.00	17.50 ± 0.00	**27.50 ± 0.00**
AWA_phog_rgsift	2	16.40 ± 16.41	18.30 ± 5.93	9.53 ± 14.56	13.51 ± 12.75	**18.66 ± 5.57**
	3	18.18 ± 6.06	21.62 ± 4.16	17.37 ± 8.69	19.24 ± 6.63	**23.03 ± 4.17**
	4	16.00 ± 0.00	23.00 ± 0.00	17.00 ± 0.00	19.50 ± 0.00	**23.50 ± 0.00**
AWA_sift_surf	2	22.82 ± 18.64	24.13 ± 8.42	21.67 ± 11.24	23.44 ± 17.76	**27.25 ± 9.07**
	3	21.36 ± 7.12	28.94 ± 5.52	18.99 ± 2.78	27.68 ± 7.96	**30.81 ± 3.96**
	4	17.50 ± 0.00	28.00 ± 0.00	12.00 ± 0.00	25.50 ± 0.00	**30.00 ± 0.00**
KDDCUP99	2	25.00 ± 25.00	43.42 ± 20.87	48.70 ± 1.06	43.70 ± 34.20	**95.00 ± 1.72**
	3	20.00 ± 16.33	49.80 ± 14.09	31.97 ± 0.45	41.97 ± 15.18	**86.36 ± 8.15**
	4	25.00 ± 0.00	49.50 ± 0.00	25.00 ± 0.00	49.00 ± 0.00	**82.00 ± 0.00**
COVTYPE	2	33.60 ± 22.00	38.85 ± 11.55	34.60 ± 10.53	34.80 ± 10.78	**41.90 ± 5.71**
	3	31.82 ± 0.00	33.84 ± 0.00	31.31 ± 0.00	34.34 ± 0.00	**41.92 ± 0.00**
Ours: win/tie/loss		23/0/0	23/0/0	22/1/0	23/0/0	Rank first 22/23

[1] #Mix: The number of task's distribution in the market used for generate the user's task

The best method is emphasized in bold.

normalized to [0, 1] for different tasks. As shown in Fig. 7(a), the subspace generation of all tasks except KDDCUP99 can be converged within 300 iterations and the value of the objective function decreases remarkably in the first 100 iterations. For user data projection, all tasks except KDDCUP99 converge within 50 iterations and go down rapidly in the first 20 iterations, which are shown in Fig. 7(b). For KDDCUP99, the subspace generation convergences at around 1000 iterations and the user task projection convergences within 50 iterations.

6 Related work

The *learnware* paradigm (Zhou, 2016) aims to build a learnware market to help users solve their machine learning tasks more efficiently rather than starting from scratch. By helping users identify and reuse helpful well-performed models in the market for their tasks, this paradigm exploits the potential value of existing trained models and significantly reduces

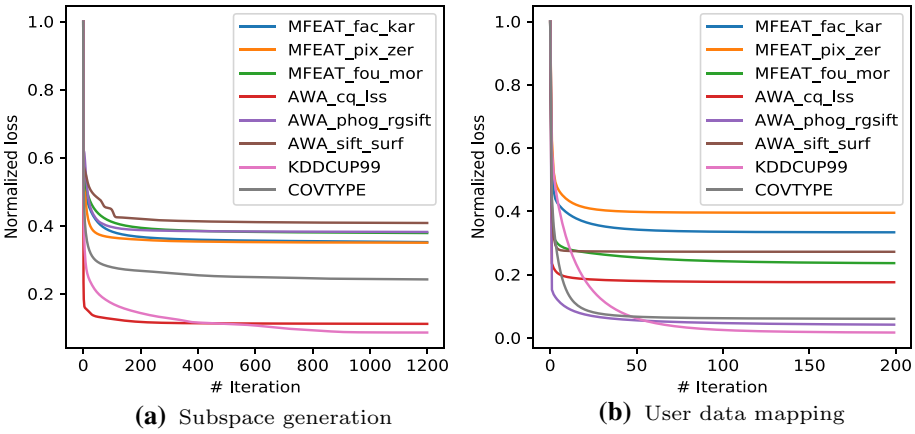


Fig. 7 Convergence curves of subspace generation and user data mapping

the needed resources for users, like computing resources, expert knowledge and labeled data.

As a novel branch of machine learning research, the learnware paradigm considers a general and realistic framework where a huge amount of models in the market are submitted spontaneously by developers from various tasks, and neither the original training data of developers nor the original data of users can be accessed. These bring grand challenges for users to identify and reuse helpful models in the market, and the specification is the original core component of the learnware paradigm to achieve this goal. Recently, there have been some efforts in this branch attempting to realize a simplified prototype framework. For instance, Wu et al. (2021) proposed the reduced kernel mean embedding (RKME) as the specification, which constructs the specification space by mapping the training data of models to an element of the reproducing kernel Hilbert space (RKHS). When the user's task involves certain unseen parts not covered by the learnware market, based on RKME specification, Zhang et al. (2021) used the mixture proportion estimation (MPE) technique (Ramaswamy et al., 2016; Zhang et al., 2020) to identify samples from the unseen parts while assigning the rest to proper models returned from the market. This paper provides a solution for learnwares from heterogeneous feature spaces by generating the RKME specification on a unified subspace.

Note that the techniques in transfer learning (Pan and Yang, 2009) and domain adaptation (Ben-David et al., 2007; Wang et al., 2022), which hope to transfer the knowledge in the source domain to the target domain, typically assume the accessibility of raw data (Dai et al., 2007; Fernando et al., 2013; Huang et al., 2006; Pan et al., 2010), and thus do not satisfy the privacy concerns in learnware paradigm. Besides, hypothesis transfer learning (Kuzborskij and Orabona, 2013) and model reuse (Ding and Zhou, 2020; Zhao et al., 2020) only apply to specific scenarios where the model to be adapted is helpful to the user task, and do not consider how to identify helpful models from a market without leaking raw data. There is limited study to reuse the model from different feature spaces without accessing raw data (Ye et al., 2018, 2020), but they also assume models are helpful for the current task. In this paper, we focus on a more comprehensive process comprising how to accommodate heterogeneous models in the market with appropriate specifications and how to identify and reuse helpful learnwares for the user's current task.

Besides, since the learnwares in the market are submitted spontaneously by developers from various tasks and are identified for arbitrary user tasks, the learnware paradigm are studied in the open environment (Zhou, 2022), and techniques for open-environment machine learning (Zhao et al., 2021; Zhao and Zhou, 2021) may also bring some inspiration.

Recently, Zhou and Tan (2022) provided a brief overview of progress on learnware, which clarified the process of the learnware market and the design of the specification. It describes the prospects of the learnware paradigm and sheds light on future exploration.

7 Conclusion

In this paper, we have proposed the first practical approach to handling learnwares from heterogeneous feature spaces, which makes the learnware paradigm viable in broader applications. We give a basic formulation for the heterogeneous learnware problem and propose a novel specification design strategy via integrating the subspace learning, along with a detailed procedure for establishing and reusing the heterogeneous learnware market. Empirical studies on both synthetic data and real-world tasks substantiate the effectiveness of our methods. Although our method is designed for the basic scenario that each learnware only comes from one of the disjoint feature spaces, it can be naturally extended to the more general scenario that the learnware comes from the Cartesian product of several disjoint feature spaces. To summarize, for the basic heterogeneous learnware scenario where the overall feature space can be divided into disjoint parts and the feature space of the user's task and learnwares can be any combination of different parts, the learnware market can be well established and used. For future research, formalizing the heterogeneous learnware problem in a more general way and proposing an effective solution are interesting subjects.

Appendix A Detailed optimization procedure

In this section, we provide the omitted details for the optimization procedures.

Detailed optimization of subspace generation

Firstly, we introduce the following proposition of the multiplicative update rule for *nonnegative quadratic programming* (Sha et al., 2007).

Proposition 1 *The general nonnegative quadratic form is defined as*

$$f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^\top \mathbf{A}\boldsymbol{x} + \boldsymbol{b}^\top \boldsymbol{x},$$

where $\boldsymbol{x}$ *is an d-dimensional nonnegative vector,* $\mathbf{A}$ *is a symmetric positive definite matrix and* $\boldsymbol{b}$ *is an arbitrary d dimensional vector. Let* $\mathbf{A}^+$*and* $\mathbf{A}^-$*denote the nonnegative matrices with elements*:

$$\mathbf{A}_{ij}^{+} = \begin{cases} \mathbf{A}_{ij} & \text{if } \mathbf{A}_{ij} > 0, \\ 0 & \text{otherwise,} \end{cases} \quad \mathbf{A}_{ij}^{-} = \begin{cases} \left|\mathbf{A}_{ij}\right| & \text{if } \mathbf{A}_{ij} < 0, \\ 0 & \text{otherwise.} \end{cases}$$

It is easily to observe that $\mathbf{A} = \mathbf{A}^{+} - \mathbf{A}^{-}$. *Then, the solution* $\boldsymbol{x}$ *that minimizes* $f(\boldsymbol{x})$ *can be obtained through the iterative update as*

$$\boldsymbol{x}_i \leftarrow \boldsymbol{x}_i \left[\frac{-\mathbf{b}_i + \sqrt{\mathbf{b}_i^2 + 4(\mathbf{A}^{+}\boldsymbol{x})_i(\mathbf{A}^{-}\boldsymbol{x})_i}}{2(\mathbf{A}^{+}\boldsymbol{x})_i} \right].$$

In this proposition, the crucial variables used for update are $\boldsymbol{b}$, $\mathbf{A}^{+}\boldsymbol{x}$ and $\mathbf{A}^{-}\boldsymbol{x}$. Instead of calculating the complicated second derivative of $f(\boldsymbol{x})$ to get $\mathbf{A}^{-}$ and $\mathbf{A}^{-}$ (Cai et al., 2010; Xu & Gong, 2004), we can calculate $\mathbf{b} = \nabla f(\boldsymbol{x})|_{\boldsymbol{x}=0}, \mathbf{A}\boldsymbol{x} = \nabla f(\boldsymbol{x})|_{\boldsymbol{x}=0} - \mathbf{b}$ and decompose $\mathbf{A}\boldsymbol{x}$ as $\mathbf{A}^{+}\boldsymbol{x}$, $\mathbf{A}^{-}\boldsymbol{x}$, which gives more concise computation and easier extension for matrix variable case.

The subspace generation problem is reviewed as

$$\begin{aligned} \min_{\mathbf{W}^{(i)},\mathbf{V}^{(i)},\mathbf{V}_c^{*}} O = \sum_{i=1}^{k} \Bigg\{ & \left\| \mathbf{X}^{(i)} - \mathbf{X}^{(i)}\mathbf{W}^{(i)}\left(\mathbf{V}^{(i)}\right)^{\top} \right\|_{\mathrm{F}}^{2} \\ & + \alpha \mathrm{Tr}\left(\left(\mathbf{V}^{(i)}\right)^{\top} \mathbf{L}^{(i)} \mathbf{V}^{(i)} \right) + \beta \left\| \mathbf{V}_c^{(i)} - \mathbf{V}_c^{*} \right\|_{\mathrm{F}}^{2} \Bigg\} \\ \text{s.t. } & \mathbf{W}^{(i)} \geq 0, \hat{\mathbf{V}}^{(i)} \geq 0, \mathbf{V}_c^{(i)} \geq 0, \mathbf{V}_c^{*} \geq 0, \end{aligned} \tag{A1}$$

The objective function Eq. (A1) of subspace generation is not convex over all variables $\mathbf{W}^{(i)}, \mathbf{V}^{(i)}, \mathbf{V}_c^{*}$ which makes it unrealistic to find its global minimum. We propose an alternative optimization algorithm based on the multiplicative updated rule like (Févotte & Idier, 2011; Xu & Gong, 2004) with local minimum achieved. For the optimization of Eq. (A1), we initialize the variables with k-means, and then, we alternately optimize the variables.

(1) Initialization We initialize $\mathbf{W}^{(i)}$ and $\mathbf{V}^{(i)} = [\hat{\mathbf{V}}^{(i)}; \mathbf{V}_c^{(i)}]$ with k-means clustering. Let $\mathbf{V}^{(i)} = \mathbf{B}^{(i)} + 0.1\mathbf{E}^{(i)}$, where $\mathbf{B}^{(i)}$ denotes the cluster outcome of $\mathbf{X}^{(i)}$ and $\mathbf{E}^{(i)}$ denotes the matrix whose elements are all 1. We add $\mathbf{E}^{(i)}$ in $\mathbf{B}^{(i)}$ to avoid ineffective multiplicative update for those zero elements. We set $\mathbf{W}^{(i)} = \mathbf{V}^{(i)}\left(\mathbf{D}^{(i)}\right)^{-1}$, where $\mathbf{D}^{(i)} = \mathrm{diag}(n_1, \cdots, n_k)$ and n_k denotes the cardinality of the k-th cluster of $\mathbf{X}^{(i)}$. $\mathbf{V}_c^{*}$ is set as $\sum_{i=1}^{k} \mathbf{V}_c^{(i)}/k$. The Laplacian matrix is calculated as $\mathbf{L}^{(i)} = \mathbf{D}^{(i)} - \mathbf{S}^{(i)}$, where $\mathbf{S}^{(i)}$ is the similarity matrix calculated with cosine similarity $[\mathbf{S}^{(i)}]_{kj} = \cos(\boldsymbol{x}_k, \boldsymbol{x}_j) = \langle \boldsymbol{x}_k, \boldsymbol{x}_j \rangle / \left(\|\boldsymbol{x}_k\| \|\boldsymbol{x}_j\| \right)$ on $\mathbf{X}^{(i)}$. $\mathbf{D}^{(i)}$ is a diagonal matrix with $[\mathbf{D}^{(i)}]_{kk} = \sum_j [\mathbf{S}^{(i)}]_{kj}$.

(2) Minimizing O over $\mathbf{W}^{(i)}$ *with* $\mathbf{V}^{(i)}, \mathbf{V}_c^{*}$ *Fixed* For the brevity, we ignore the superscript of $\mathbf{K}^{(i)}, \mathbf{V}^{(i)}, \mathbf{W}^{(i)}$ and abbreviate them as $\mathbf{K}, \mathbf{V}, \mathbf{W}$ where $\mathbf{K}^{(i)} = (\mathbf{X}^{(i)})^{\top}\mathbf{X}^{(i)}$. After fixing irrelevant variables, the subproblem is

$$\min O(\mathbf{W}) = -2\mathrm{tr}\left(\mathbf{W}^{\top}\mathbf{K}\mathbf{H}\right) + \mathrm{tr}\left(\mathbf{W}^{\top}\mathbf{K}\mathbf{W}\mathbf{H}^{\top}\mathbf{H}\right)$$

with $\mathbf{W} \geq 0$. Using multiplicative updated rule (Xu & Gong, 2004), we get

$$\mathbf{W}_{ij} \leftarrow \mathbf{W}_{ij}\left[\frac{(\mathbf{KV})_{ij}+\sqrt{(\mathbf{KV})_{ij}^2+4\mathbf{P}_{ij}^+\mathbf{P}_{ij}^-}}{2\mathbf{P}_{ij}^+}\right], \tag{A2}$$

where $\mathbf{P}^+ = \mathbf{K}^+\mathbf{W}\mathbf{V}^\top\mathbf{V}$, $\mathbf{P}^- = \mathbf{K}^-\mathbf{W}\mathbf{V}^\top\mathbf{V}$.

(3) Minimizing O over $\mathbf{V}^{(i)}$ with $\mathbf{W}^{(i)}$, $\mathbf{V}_c^*$ Fixed For simplicity of the notation, we abbreviate $\mathbf{X}^{(i)} = [\hat{\mathbf{X}}^{(i)}, \mathbf{X}_c^{(i)}]$, $\mathbf{V}^{(i)} = [\hat{\mathbf{V}}^{(i)};\mathbf{V}_c^{(i)}]$, $\mathbf{L}^{(i)}$ as $\mathbf{X} = [\hat{\mathbf{X}}, \mathbf{X}_c]$, $\mathbf{V} = [\hat{\mathbf{V}};\mathbf{V}_c]$ and $\mathbf{L}$ respectively. We present the Laplacian matrix as $\mathbf{L} = [\mathbf{L}_{11},\mathbf{L}_{12};\mathbf{L}_{21},\mathbf{L}_{22}] = \mathbf{D} - \mathbf{S} = [\mathbf{D}_{11},\mathbf{D}_{12};\mathbf{D}_{21},\mathbf{D}_{22}] - [\mathbf{S}_{11},\mathbf{S}_{12};\mathbf{S}_{21},\mathbf{S}_{22}]$ and the kernel matrix as $\mathbf{K} = [\mathbf{K}_0;\mathbf{K}_1]$ based on the shape of $\mathbf{V} = [\hat{\mathbf{V}};\mathbf{V}_c]$.

The subproblem is optimizing $O(\mathbf{V}) = \|\mathbf{X} - \mathbf{X}\mathbf{W}\mathbf{V}^\top\|_F^2 + \alpha\mathrm{Tr}(\mathbf{V}^\top\mathbf{L}\mathbf{V}) + \beta\|\mathbf{V}_c - \mathbf{V}_c^*\|_F^2$ with $\mathbf{V} = [\hat{\mathbf{V}};\mathbf{V}_c] \geq 0$. We rewrite the objective function as

$$\begin{aligned} O(\mathbf{V}) = &\left\|\hat{\mathbf{X}} - \mathbf{X}\mathbf{W}\hat{\mathbf{V}}^\top\right\|_F^2 + \left\|\mathbf{X}_c - \mathbf{X}\mathbf{W}\mathbf{V}_c^\top\right\|_F^2 \\ &+ \alpha\Big(\mathrm{Tr}(\hat{\mathbf{V}}^\top\mathbf{L}_{11}\hat{\mathbf{V}}) + 2\mathrm{Tr}(\mathbf{V}_c^\top\mathbf{L}_{21}\hat{\mathbf{V}}) \\ &+ \mathrm{Tr}(\mathbf{V}_c^\top\mathbf{L}_{22}\mathbf{V}_c)\Big) + \beta\|\mathbf{V}_c - \mathbf{V}_c^*\|_F^2. \end{aligned} \tag{A3}$$

We optimize $\mathbf{V} = [\hat{\mathbf{V}};\mathbf{V}_c]$ in two steps, we first optimize $\hat{\mathbf{V}}$ with fixed $\mathbf{V}_c$ and then optimize $\mathbf{V}_c$ with fixed $\hat{\mathbf{V}}$.

The subproblem for $\hat{\mathbf{V}}$ is optimizing $O(\hat{\mathbf{V}}) = \|\hat{\mathbf{X}} - \mathbf{X}\mathbf{W}\hat{\mathbf{V}}^\top\|_{\mathrm{F}}^2 + \alpha\mathrm{Tr}(\hat{\mathbf{V}}^\top\mathbf{L}_{11}\hat{\mathbf{V}}) + 2\alpha\mathrm{Tr}(\mathbf{V}_c^\top\mathbf{L}_{21}\hat{\mathbf{V}})$ with $\hat{\mathbf{V}} \geq 0$. By taking the first order derivative, we get $\nabla O(\hat{\mathbf{V}}) = -2\mathbf{K}_0\mathbf{W} + 2\hat{\mathbf{V}}\mathbf{W}^\top\mathbf{K}\mathbf{W} + 2\alpha\mathbf{L}_{11}\hat{\mathbf{V}} + 2\alpha\mathbf{L}_{12}\mathbf{V}_c$ and thus $\nabla O(\hat{\mathbf{V}})|_{\hat{\mathbf{V}}=0} = -2\mathbf{K}_0\mathbf{W} + 2\alpha\mathbf{L}_{12}\mathbf{V}_c$, $\nabla O(\hat{\mathbf{V}}) - \nabla O(\hat{\mathbf{V}})|_{\hat{\mathbf{V}}=0} = 2\hat{\mathbf{V}}\mathbf{W}^\top\mathbf{K}\mathbf{W} + 2\alpha\mathbf{L}_{11}\hat{\mathbf{V}} = (2\hat{\mathbf{V}}\mathbf{W}^\top\mathbf{K}^+\mathbf{W} + 2\alpha\mathbf{D}_{11}\hat{\mathbf{V}}) - (2\hat{\mathbf{V}}\mathbf{W}^\top\mathbf{K}^-\mathbf{W} + 2\alpha\mathbf{S}_{11}\hat{\mathbf{V}})$. According to the proposition (1), we get the updated rules as

$$(\hat{\mathbf{V}})_{ij} \leftarrow (\hat{\mathbf{V}})_{ij}\left(\frac{\hat{\mathbf{A}}_{ij}+\sqrt{\hat{\mathbf{A}}_{ij}^2+4\hat{\mathbf{Q}}_{ij}^+\hat{\mathbf{Q}}_{ij}^-}}{2\hat{\mathbf{Q}}_{ij}^+}\right), \tag{A4}$$

where $\hat{\mathbf{A}} = \mathbf{K}_0\mathbf{W} - \alpha\mathbf{L}_{12}\mathbf{V}_c$, $\hat{\mathbf{Q}}^+ = \hat{\mathbf{V}}\mathbf{W}^\top\mathbf{K}^+\mathbf{W} + \alpha\mathbf{D}_{11}\hat{\mathbf{V}}$, $\hat{\mathbf{Q}}^- = \hat{\mathbf{V}}\mathbf{W}^\top\mathbf{K}^-\mathbf{W} + \alpha\mathbf{S}_{11}\hat{\mathbf{V}}$.

The subproblem for $\mathbf{V}_c$ is optimizing $O(\mathbf{V}_c) = \|\mathbf{X}_c - \mathbf{X}\mathbf{W}\mathbf{V}_c^\top\|_{\mathrm{F}}^2 + \beta\|\mathbf{V}_c - \mathbf{V}_c^*\|_{\mathrm{F}}^2 + \alpha\mathrm{Tr}(\mathbf{V}_c^\top\mathbf{L}_{22}\mathbf{V}_c) + 2\alpha\mathrm{Tr}(\mathbf{V}_c^\top\mathbf{L}_{21}\hat{\mathbf{V}})$ with $\mathbf{V}_c \geq 0$. By taking the first order derivative, we get $\nabla O(\mathbf{V}_c) = -2\mathbf{K}_1\mathbf{W} - 2\beta\mathbf{V}_c^* + 2\mathbf{V}_c\mathbf{W}^\top\mathbf{K}\mathbf{W} + 2\beta\mathbf{V}_c + 2\alpha\mathbf{L}_{22}\mathbf{V}_c + 2\alpha\mathbf{L}_{21}\hat{\mathbf{V}}$. By decompose the derivative similarly and use the proposition (1), we get the updated rule as

$$(\mathbf{V}_c)_{ij} \leftarrow (\mathbf{V}_c)_{ij}\left[\frac{(\mathbf{A}_c)_{ij}+\sqrt{(\mathbf{A}_c)_{ij}^2+4(\mathbf{Q}_c^+)_{ij}(\mathbf{Q}_c^-)_{ij}}}{2(\mathbf{Q}_c^+)_{ij}}\right], \tag{A5}$$

where $\mathbf{A}_c = \mathbf{K}_1\mathbf{W} + \beta\mathbf{V}_c - \alpha\mathbf{L}_{21}\hat{\mathbf{V}}$, $\mathbf{Q}_c^+ = \mathbf{V}_c\mathbf{W}^\top\mathbf{K}^+\mathbf{W} + \beta\mathbf{V}_c + \alpha\mathbf{D}_{22}\mathbf{V}_c$, $\mathbf{Q}_c^- = \mathbf{V}_c\mathbf{W}^\top\mathbf{K}^-\mathbf{W} + \alpha\mathbf{S}_{22}\mathbf{V}_c$.

(4) Cooperative Normalization This step imposed on $\mathbf{V}^{(i)} = [\hat{\mathbf{V}}^{(i)};\mathbf{V}_c^{(i)}]$ and $\mathbf{W}^{(i)}$ aims to make the consistency loss expressed by $\|\mathbf{V}_c^{(i)} - \mathbf{V}_c^*\|_{\mathrm{F}}^2$ reasonable. It is obvious to check that

when $\mathbf{W}^{(i)}$ and $\mathbf{V}^{(i)}$ are solutions of concept factorization, $\mathbf{W}^{(i)}\mathbf{D}^{(i)}$ and $\mathbf{V}^{(i)}(\mathbf{D}^{(i)})^{-1}$ also form another solutions for any diagonal matrix $\mathbf{D}^{(i)}$ with positive diagonal elements. In order to keep the uniqueness of solution and suitable comparison of different $\mathbf{V}_c^{(i)}$, we restrict the ℓ_1 -norm of each column of $\mathbf{V}_c^{(i)}$ as 1 by

$$\begin{aligned} \mathbf{V}^{(i)} &\leftarrow \mathbf{V}^{(i)}\left(\mathbf{Q}^{(i)}\right)^{-1} \\ \mathbf{W}^{(i)} &\leftarrow \mathbf{W}^{(i)}\mathbf{Q}^{(i)}, \end{aligned} \tag{A6}$$

where $\mathbf{Q}^{(i)} = \mathrm{diag}\left(\sum_j [\mathbf{V}_c^{(i)}]_{j,1}, \cdots, \sum_j [\mathbf{V}_c^{(i)}]_{j,d}\right)$ and d is the dimension of subspace.

(5) Minimizing O over $\mathbf{V}_c^*$ with $\mathbf{W}^{(i)}, \mathbf{V}^{(i)}$ Fixed The sub problem is given as $\min_{\mathbf{V}_c^*} \sum_{i=1}^k \|\mathbf{V}_c^{(i)} - \mathbf{V}_c^*\|_{\mathrm{F}}^2$ with $\mathbf{V}_c^* \geq 0$, which gives the closed-form solution as

$$\mathbf{V}_c^* = \frac{1}{k}\sum_{i=1}^{k} \mathbf{V}_c^{(i)}. \tag{A7}$$

The proposed optimization updates the variables $\mathbf{W}^{(i)}, \mathbf{V}^{(i)} = [\mathbf{V}_c^{(i)}, \hat{\mathbf{V}}^{(i)}]$ and $\mathbf{V}_c^*$ alternately. The whole algorithm is showed in Algorithm 2.

Algorithm 2 Optimization of subspace generation

Input: local data sets $\{\mathbf{X}_1, \mathbf{X}_2, \cdots, \mathbf{X}_R\}$ from different feature spaces, feature space indicators $\{v_1, \cdots, v_R\}$, auxiliary data across entire feature space $\mathbf{X}_c = [\mathbf{X}_c^{(1)}; \cdots; \mathbf{X}_c^{(k)}]$.

Parameter: Trade-off parameters $\{\alpha, \beta\}$, number of nearest neighbors p for manifold regularizer, the dimension of subspace d, max iteration T.

Output: New representation of data matrix in subspace $\{\mathbf{V}_1, \mathbf{V}_2, \cdots, \mathbf{V}_R\}$, projection tools $\{\mathbf{B}^{(1)}, \mathbf{B}^{(2)}, \cdots, \mathbf{B}^{(k)}\}$ used for the user data projection.

For each feature space $\mathcal{X}_i, i \in [k]$, concatenate the corresponding data matrix $\mathbf{X}_j$ satisfying $v_j = i$ and get $\{\mathbf{X}^{(1)}, \cdots, \mathbf{X}^{(k)}\}$.
Initialize the $\mathbf{W}^{(i)}, \mathbf{V}^{(i)}$ and $\mathbf{V}_c^*$ with the results of k-means.
while max iteration is not achieved **do**
  for $i = 1$ to k **do**
    Fix $\mathbf{V}^{(i)}, \mathbf{V}_c^*$, update $\mathbf{W}^{(i)}$ by the rule (A2).
    Fix $\mathbf{W}^{(i)}, \mathbf{V}_c^*$, update $\mathbf{V}^{(i)} = [\mathbf{V}_c^{(i)}, \hat{\mathbf{V}}^{(i)}]$ by the rules (A4) and (A5).
    Cooperatively normalize $\mathbf{W}^{(i)}$ and $\mathbf{V}^{(i)}$ by Eq. (A6).
  end for
  Fix $\mathbf{W}^{(i)}, \mathbf{V}^{(i)}, i \in [k]$ and update $\mathbf{V}_c^*$ by Eq. (A7).
end while
Decompose each concatenated mapped data matrix $\{\mathbf{V}^{(1)}, \cdots, \mathbf{V}^{(k)}\}$ as single elements $\{\mathbf{V}_1, \mathbf{V}_2, \cdots, \mathbf{V}_R\}$ and $\mathbf{V}_c^*$.
Calculate the base matrices of all feature space via $\mathbf{B}^{(i)} = \mathbf{X}^{(i)}\mathbf{W}^{(i)}, i \in [k]$.
return mapped data matrices $\{\mathbf{V}_1, \mathbf{V}_2, \cdots, \mathbf{V}_R\}$ and projection tools of overall feature space $\{\mathbf{B}^{(1)}, \mathbf{B}^{(2)}, \cdots, \mathbf{B}^{(k)}\}$.

Detailed optimization for user data mapping

The problem is reviewed as Eq. (A8) and can be optimized similarly as Appendix A.1 via alternately optimizing $\mathbf{V}^{(i)}$ and $\mathbf{V}^*$.

$$\begin{aligned}\min_{\mathbf{V}^{(i)},\mathbf{V}^*} O_u = \sum_{i=1}^{t} \Bigg\{ & \left\| \mathbf{X}_u^{(i)} - \mathbf{B}^{(i)} \left(\mathbf{V}^{(i)}\right)^\top \right\|_{\mathrm{F}}^2 \\ & + \alpha \mathrm{Tr}\left(\left(\mathbf{V}^{(i)}\right)^\top \mathbf{L}^{(i)} \mathbf{V}^{(i)} \right) + \beta \left\| \mathbf{V}^{(i)} - \mathbf{V}^* \right\|_{\mathrm{F}}^2 \Bigg\} \\ \text{s.t. } & \mathbf{V}^{(i)} \geq 0, \mathbf{V}^* \geq 0,\end{aligned} \tag{A8}$$

First, we discuss the subproblem of $\mathbf{V}^{(i)}$. We mark $\mathbf{X}_u^{(i)}, \mathbf{B}^{(i)}, \mathbf{V}^{(i)}, \mathbf{L}^{(i)}$ as $\mathbf{X}, \mathbf{B}, \mathbf{V}, \mathbf{L}$ in short. The subproblem is optimizing $O(\mathbf{V}) = \|\mathbf{X} - \mathbf{B}\mathbf{V}^\top\|_{\mathrm{F}}^2 + \alpha \mathrm{Tr}(\mathbf{V}^\top \mathbf{L}\mathbf{V}) + \beta \|\mathbf{V} - \mathbf{V}^*\|_{\mathrm{F}}^2$ with $\mathbf{V} \geq 0$. Thus, $\nabla O(\mathbf{V}) = -2\mathbf{X}^\top \mathbf{B} + 2\mathbf{V}\mathbf{K} + 2\alpha \mathbf{L}\mathbf{V} + 2\beta(\mathbf{V} - \mathbf{V}^*)$ where $\mathbf{K} = \mathbf{B}^\top \mathbf{B}$. The updated rule goes similarly as

$$\mathbf{V}_{ij} \leftarrow \mathbf{V}_{ij} \left(\frac{\mathbf{A}_{ij} + \sqrt{\mathbf{A}_{ij}^2 + 4\mathbf{Q}_{ij}^+ 4\mathbf{Q}_{ij}^-}}{2\mathbf{Q}_{ij}^+} \right), \tag{A9}$$

where $\mathbf{A} = \mathbf{X}^\top \mathbf{B} + \beta \mathbf{V}^*, \mathbf{Q}^+ = \mathbf{V}\mathbf{K}^+ + \alpha \mathbf{D}\mathbf{V} + \beta \mathbf{V}$ and $\mathbf{Q}^- = \mathbf{V}\mathbf{K}^- + \alpha \mathbf{S}\mathbf{V}$.

Second, we discuss the update of $\mathbf{V}^*$, the subproblem of $\mathbf{V}^*$ is $\min_{\mathbf{V}^*} \sum_{i=1}^t \|\mathbf{V}^{(i)} - \mathbf{V}^*\|_{\mathrm{F}}^2$ with $\mathbf{V}^* \geq 0$, which results in $\mathbf{V}^* = \frac{1}{t} \sum_{i=1}^t \mathbf{V}^{(i)}$.

Acknowledgements This research was supported by the National Key Research and Development Program of China (2020AAA0109401), the National Science Foundation of China (62176116, 61921006), the Collaborative Innovation Center of Novel Software Technology and Industrialization, and Nanjing University-Huawei Joint Research Program. The authors would like to thank Peng Zhao for valuable suggestions. We are also grateful to the anonymous reviewers for their constructive comments.

Author contributions Peng Tan conceived and developed the procedure, performed the experiments and wrote the draft manuscript. Zhi-Hao Tan helped shape the procedure and analysis, and revised the manuscript. Zhi-Hua Zhou and Yuan Jiang conceived the study and were in charge of overall direction and planning. All authors discussed the results and contributed to the final manuscript. All authors approved the final version of the manuscript.

Code availability The code is available at https://www.lamda.nju.edu.cn/code_RKME_heterogeneous.ashx.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Ethical approval Not applicable.

Consent to participate Not applicable.

Consent for publication Not applicable.

References

Ben-David, S., Blitzer, J., Crammer, K. et al. (2007) Analysis of representations for domain adaptation. In Advances in Neural Information Processing Systems 19.

Blackard, J. A., & Dean, D. J. (1999). Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. *Computers and Electronics in Agriculture, 24,* 131–151.

Butler, K. T., Davies, D. W., Cartwright, H., et al. (2018). Machine learning for molecular and materials science. *Nature, 559*(7715), 547–555.

Cai, D., He, X., & Han, J. (2010). Locally consistent concept factorization for document clustering. *IEEE Transactions on Knowledge & Data Engineering, 23*(6), 902–913.

Chen, X., Fang, H., Lin, TY., et al. (2015). Microsoft coco captions: Data collection and evaluation server. arXiv:1504.00325.

Chen, Y., Welling, M., & Smola, A. (2012). Super-samples from kernel herding. arXiv:1203.3472.

Dai, W., Yang, Q., Xue, G., et al. (2007). Boosting for transfer learning. In Proceedings of the 24th International Conference on Machine Learning, pp 193–200.

Ding, Y. X., & Zhou, Z. H. (2020). Boosting-based reliable model reuse. In Proceedings of the 12th Asian Conference on Machine Learning, pp 145–160.

Fernando, B., Habrard, A., Sebban, M. et al. (2013). Unsupervised visual domain adaptation using subspace alignment. In Proceedings of the IEEE International Conference on Computer Vision, pp 2960–2967.

Févotte, C., & Idier, J. (2011). Algorithms for nonnegative matrix factorization with the β-divergence. *Neural Computation, 23*(9), 2421–2456.

Huang, J., Gretton, A., Borgwardt, K. et al. (2006). Correcting sample selection bias by unlabeled data. In Advances in Neural Information Processing Systems 19.

Johnson, A. E., Pollard, T. J., Shen, L., et al. (2016). MIMIC-III, a freely accessible critical care database. *Scientific Data, 3*(1), 1–9.

Jumper, J., Evans, R., Pritzel, A., et al. (2021). Highly accurate protein structure prediction with AlphaFold. *Nature, 596*(7873), 583–589.

Kuzborskij, I., & Orabona, F. (2013). Stability and hypothesis transfer learning. In Proceedings of the 30th International Conference on Machine Learning, pp 942–950.

Lampert, CH., Nickisch, H., & Harmeling, S. (2009). Learning to detect unseen object classes by between-class attribute transfer. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 951–958.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature, 521*(7553), 436–444.

Lee, D. D., & Seung, H. S. (2001). Algorithms for non-negative matrix factorization. In Advances in Neural Information Processing Systems 13. pp 556–562.

Lippmann, R., Haines, JW., Fried, DJ. et al. (2000). Analysis and results of the 1999 darpa off-line intrusion detection evaluation. In International Workshop on Recent Advances in Intrusion Detection, pp 162–182.

Pan, S. J., & Yang, Q. (2009). A survey on transfer learning. *IEEE Transactions on Knowledge & Data Engineering, 22*(10), 1345–1359.

Pan, S. J., Tsang, I. W., Kwok, J. T., et al. (2010). Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks, 22*(2), 199–210.

Ramaswamy, H., Scott, C., & Tewari, A. (2016). Mixture proportion estimation via kernel embeddings of distributions. In Proceedings of the 33rd International Conference on Machine Learning, pp 2052–2060.

Schölkopf, B., & Smola, AJ. (2002). Learning with kernels: Support vector machines, regularization, optimization, and beyond. MIT press.

Sha, F., Lin, Y., Saul, L. K., et al. (2007). Multiplicative updates for nonnegative quadratic programming. *Neural Computation, 19*(8), 2004–2031.

Smola, A., Gretton, A., Song, L. et al. (2007). A Hilbert space embedding for distributions. In Proceedings of the 18th International Conference on Algorithmic Learning Theory, pp 13–31.

Sriperumbudur, B. K., Fukumizu, K., & Lanckriet, G. R. G. (2011). Universality, characteristic kernels and RKHS embedding of measures. *Journal of Machine Learning Research, 12*(7), 2389–2410.

van Breukelen, M., Duin, R. P., Tax, D. M., et al. (1998). Handwritten digit recognition by combined classifiers. *Kybernetika, 34*(4), 381–386.

Wang, H., Yang, Y., & Li, T. (2016). Multi-view clustering via concept factorization with local manifold regularization. In Proceedings of the 16th International Conference on Data Mining, pp 1245–1250.

Wang, Y., Wang, C., Xue, H., et al. (2022). Self-corrected unsupervised domain adaptation. *Frontiers of Computer Science, 16*(5), 1–9.
Wu, X. Z., Xu, W., Liu, S., et al. (2021). Model reuse with reduced kernel mean embedding specification. *IEEE Transactions on Knowledge & Data Engineering*. https://doi.org/10.1109/TKDE.2021.3086619.
Xu, W., & Gong, Y. (2004). Document clustering by concept factorization. In Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp 202–209.
Ye, H. J., Zhan, D. C., Jiang, Y. et al. (2018). Rectify heterogeneous models with semantic mapping. In Proceedings of the 37th International Conference on Machine Learning, pp 5630–5639.
Ye, H. J., Zhan, D. C., Jiang, Y., et al. (2020). Heterogeneous few-shot model rectification with semantic mapping. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 43*(11), 3878–3891.
Zhang, Y. J., Yan, Y. H., Zhao, P. et al. (2021). Towards enabling learnware to handle unseen jobs. In Proceedings of the 35th AAAI Conference on Artificial Intelligence, pp 10,964–10,972.
Zhang, Y. J., Zhao, P., Ma, L., et al. (2020). An unbiased risk estimator for learning with augmented classes. In Advances in Neural Information Processing Systems 33. pp 10,247–10,258.
Zhao, P., & Zhou, Z. H. (2021). Learning from distribution-changing data streams via decision tree model reuse. *Scientia Sinica Informationis, 51*(1), 1–12.
Zhao, P., Cai, L. W., & Zhou, Z. H. (2020). Handling concept drift via model reuse. *Machine Learning, 109*(3), 533–568.
Zhao, P., Zhang, Y. J., & Zhou, Z. H. (2021). Exploratory machine learning with unknown unknowns. In Proceedings of the 35th AAAI Conference on Artificial Intelligence, pp 10,999–11,006.
Zhou, Z. H. (2016). Learnware: On the future of machine learning. *Frontiers of Computer Science, 10*(4), 589–590.
Zhou, Z. H. (2022). Open-environment machine learning. *National Science Review, 9*(8), nwac123.
Zhou, Z. H., & Tan, Z. H. (2022). Learnware: Small models do big. arXiv:2210.03647.
Zhu, G. W. Z., Fan, R. D., Luo, Y. J., et al. (2022). Incomplete multi-view clustering via independent self-representation learning. *Scientia Sinica Informationis, 52*(7), 1186–1203.