Handling Learnwares Developed from Heterogeneous Feature Spaces without Auxiliary Data

Peng Tan, **Zhi-Hao Tan**, **Yuan Jiang** and **Zhi-Hua Zhou**

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

{tanp, tanzh, jiangy, zhouzh}@lamda.nju.edu.cn

Abstract

The *learnware paradigm* proposed by Zhou [2016] devotes to constructing a market of numerous well-performed models, enabling users to solve problems by reusing existing efforts rather than starting from scratch. A learnware comprises a trained model and the specification which enables the model to be adequately identified according to the user's requirement. Previous studies concentrated on the homogeneous case where models share the same feature space based on Reduced Kernel Mean Embedding (RKME) specification. However, in real-world scenarios, models are typically constructed from different feature spaces. If such a scenario can be handled by the market, all models built for a particular task even with different feature spaces can be identified and reused for a new user task. Generally, this problem would be easier if there were additional auxiliary data connecting different feature spaces, however, obtaining such data in reality is challenging. In this paper, we present a general framework for accommodating heterogeneous learnwares without requiring additional auxiliary data. The key idea is to utilize the submitted RKME specifications to establish the relationship between different feature spaces. Additionally, we give a matrix factorization-based implementation and propose the overall procedure for constructing and exploiting the heterogeneous learnware market. Experiments on real-world tasks validate the efficacy of our method.

1 Introduction

The current machine learning paradigm has achieved great success in many scenarios, such as medicine, finance, and ecology. However, achieving a well-performing model requires several essential conditions, such as access to abundant high-quality labeled data, strong computational resources, and expertise in feature engineering and algorithms. Consequently, ordinary individuals face a significant burden when aiming to build a high-quality model. Moreover, the challenges of data privacy and catastrophic forgetting arise when reusing or adapting a trained model among different users. In order to address these challenges simultaneously, the learnware paradigm [Zhou, 2016; Zhou and Tan, 2022] aims to establish a model market that assists users in solving their tasks by leveraging existing efforts instead of starting from scratch. The design of learnware market is to allow well-performed models submitted by developers can be used "beyond-what-was-submitted" instead of "as-what-was-submitted"; in other words, the models can be reused to help tasks that were not planned/targeted by their developers.

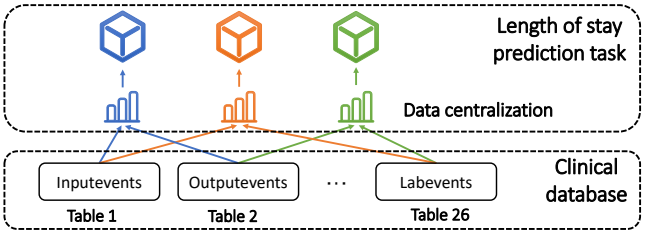


Figure 1: Heterogeneous models in real-world scenarios: trained models usually have different feature spaces even for the same medical task due to different data centralization.

A *learnware* is comprised of a well-performed trained model and a *specification* that describes its capabilities, allowing the model to be adequately identified according to the requirements of future tasks [Zhou and Tan, 2022]. The learnware market accommodates various learnwares. Machine learning developers can submit their well-performing models spontaneously to the market, and the market assigns specifications to accepted models. Once the learnware market is established, the user can exploit it by specifying her task requirement to the market and reusing recommended learnwares from the market. It is important to note that the learnware market have no access to the original task data of developers and users. Therefore, the specification plays a central role when the market identifies useful models based on user requirements. Recently the Reduced Kernel Mean Embedding (RKME) specification [Zhou and Tan, 2022] was proposed, describing the model as an element in the reproducing kernel Hilbert space (RKHS), which is also called the *specification space*. Based on the RKME specification, different learnware identification and reuse algorithms have been proposed [Wu *et al.*, 2023; Zhang *et al.*, 2021].

Current studies on the learnware paradigm require that all

learnwares in a specification space share the same feature space. However, in real-world scenarios, models are usually from different feature spaces even if they solve the same machine learning task. For instance, in the extensively utilized clinical database of critical care units [Johnson *et al.*, 2016], data is organized across 26 tables such as inputevents, outputevents, labevents etc. One crucial medical task is to predict the length of stay (LOS) for patients in the intensive care unit (ICU), aiming at improving scheduling and hospital resource management [Purushotham *et al.*, 2018; Harutyunyan *et al.*, 2019]. When solving this task, models are built on the centralized data aggregated from selected tables. Due to varying prior medical knowledge, different models developed for the same task may possess different feature spaces since training data are centralized from different subsets of tables. This situation is illustrated in Figure 1. For such a realistic scenario, an intriguing question arises: *Can the market accommodate all these models with heterogeneous feature spaces and identify helpful models for new user tasks?* In such a medical scenario, the user could be a newly established hospital lacking access to precious and sensitive raw data concerning critically ill patients.

In this problem, exploring the relationship between different feature spaces is an essential step. However, temporary solutions often require a significant amount of co-occurrence data across all feature spaces to reveal this relationship [Yang *et al.*, 2015; Wang *et al.*, 2016; Tan *et al.*, 2022], which is challenging to obtain or even absent in real-world scenarios. Moreover, due to privacy concerns within the learnware paradigm, access to the raw data of submitted models is prohibited. Consequently, the market needs to *establish the relationship between different feature spaces without relying on the raw data of models or additional auxiliary co-occurrence data*. To tackle this challenge, our proposed solution is that the market constructs the relationship of heterogeneous feature spaces based on the specifications provided with the submitted models. Based on this idea, with an increasing number of models accommodated by the market, the established relationship can become more accurate.

In this paper, we realize a heterogeneous learnware market. The market accommodates heterogeneous models from developers without requiring additional auxiliary data while ensuring the privacy of the submitted models. When the user exploits the market, the market can recommend helpful models whose feature spaces even differs from the user task. In summary, the contributions are threefold.

- We present a general framework for addressing the heterogeneous learnware problem without requiring additional auxiliary data. RKME specifications are utilized to establish connections between different feature spaces of models and generate corresponding mapping functions. These mapping functions are employed to adjust the specifications of models during market construction, align task requirements, and address missing features when the user exploit the market and reuse helpful models.
- Our general framework can be integrated with various existing subspace learning methods, and we provide an implementation based on matrix factorization. Additionally, we present a two-stage procedure for constructing and utilizing the heterogeneous learnware market.
- Experiments on both synthetic datasets and real-world tasks validate the efficacy of our methods.

The paper proceeds as follows. Section 2 provides the preliminaries, followed by Section 3 where the problem is formulated. Our general framework is introduced in Section 4, and a matrix factorization-based implementation is presented in Section 5. The experimental results are presented in Section 6. Section 7 discusses related topics, and finally, Section 8 summarizes the work.

2 Preliminary

In this section, we give a brief introduction to the pioneer work realizing the specification which can accommodate homogeneous learnwares via Reduced Kernel Mean Embedding (RKME) [Zhou and Tan, 2022] based on Kernel Mean Embedding (KME) [Smola *et al.*, 2007].

Kernel Mean Embedding. KME gives a new presentation to the distribution which supports convenient operations like mean calculation. More specifically, KME maps a distribution $\mathcal{P}$ defined over $\mathcal{X}$ to an element in a reproducing kernel Hilbert space (RKHS) $\mathcal{H}$ 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 and Smola, 2002] with associated RKHS $\mathcal{H}$. When equipped with the characteristic kernel such as Gaussian kernel, no information about the distribution $\mathcal{P}$ will be lost [Sriperumbudur *et al.*, 2011]. When provided with a data set $\{\boldsymbol{x}_i\}_{i=1}^n$ sampled from $\mathcal{P}$, the empirical estimation of KME is $\hat{\mu}_k(\mathcal{P}) := \frac{1}{n}\sum_{i=1}^n k(\boldsymbol{x}_i, \cdot)$.

Reduced Kernel Mean Embedding. KME is a potential specification due to several favorable properties. However, the access to the raw data violates the privacy concern which the specification needs. Based on KME, RKME specification [Zhou and Tan, 2022] was recently proposed, with the main idea of using the reduced set containing minor weighted samples $\{(\beta_j, \boldsymbol{z}_j)\}_{j=1}^m$ to approximate the empirical KME of the original data set $\{\boldsymbol{x}_i\}_{i=1}^n$. The reduced set is generated by

$$\min_{\boldsymbol{\beta}, \mathbf{Z}} \left\| \frac{1}{n}\sum_{i=1}^{n} k(\boldsymbol{x}_i, \cdot) - \sum_{j=1}^{m} \beta_j k(\boldsymbol{z}_j, \cdot) \right\|_{\mathcal{H}}^2, \tag{1}$$

with non-negative constraints of coefficients $\{\beta_j\}_{j=1}^m$. The RKME $\Phi(\cdot) = \sum_{j=1}^m \beta_j k(\boldsymbol{z}_j, \cdot) \in \mathcal{H}$ serves as the specification and the RKHS $\mathcal{H}$ is called specification space. This specification captures the major information of the distribution $\mathcal{P}$ and the raw data is not exposed which satisfies the data privacy property which the specification needs.

3 Formulation

We consider the entire feature space $\mathcal{X}_{\text{all}}$ as a composition of Q components: $\mathcal{X}_{\text{all}} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_Q$. The feature space of the developer $\mathcal{X}^{\text{up}}$ and the user $\mathcal{X}^{\text{us}}$ are the Cartesian product of several components $\times_{i \in C}\mathcal{X}_i$ where C is the set of component indices. We assume the market encounters T kinds of feature space for submitted models $\mathcal{X}_1^{\text{up}}, \cdots, \mathcal{X}_T^{\text{up}}$.

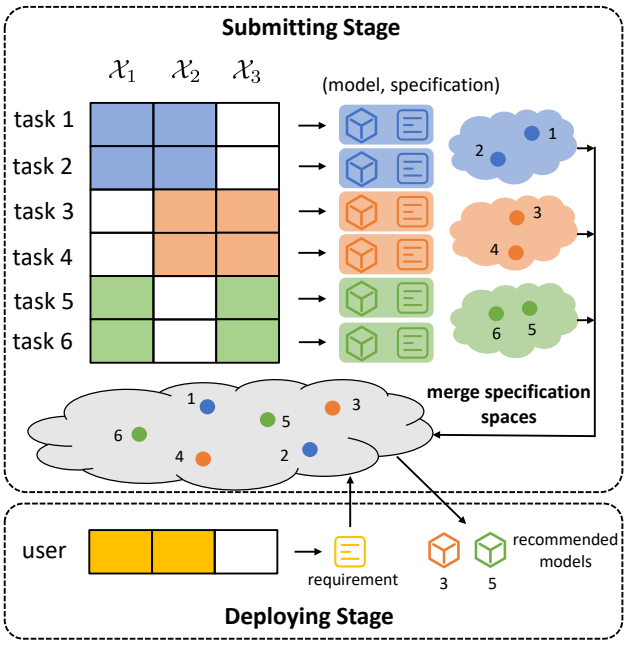


Figure 2: Two-staged formulation of the heterogeneous learnware problem. In the submitting stage, the market receives heterogeneous models with raw specifications located in different specification spaces, the market then merges different specification spaces to a unified one, which enhances its ability for learnware recommendation. In the deploying stage, the user submits her task requirement and reuses recommended learnwares from the market.

The learnware paradigm consists of the submitting stage and the deploying stage. In the submitting stage, there are N developers submit their models. Each model f_i is trained on the labeled data set $D_i := \{(\boldsymbol{x}_{ij}, y_{ij})\}_{j=1}^{n_i}$ defined over $\mathcal{X}_{\phi_i}^{\text{up}} \times \mathcal{Y}$ where $\phi_i \in \{1, \cdots, T\}$. In addition to the well-trained model f_i, the developer also provides the raw RKME specification $\boldsymbol{s}_i^{\text{raw}}$ to the market. The market receives a total of N models along with their raw specifications $\{f_i, \boldsymbol{s}_i^{\text{raw}}\}_{i=1}^N$. Temporarily, raw specifications are located in T different specification spaces, and thus, the learnware recommendation can only be conducted in each isolated specification space, which restricts the scope of learnware recommendation. In order to recommend and reuse all heterogeneous learnwares, the market will combine all these specification spaces and generate a global specification space, and then, reassign the specification $\boldsymbol{s}_i$ for the model f_i. The heterogeneous learnware market is built as $\{f_i, \boldsymbol{s}_i\}_{i=1}^N$.

In the deploying stage, the user has a unlabeled dataset $D_0 := \{\boldsymbol{x}_{0j}\}_{j=1}^{n_0}$ defined over $\mathcal{X}^{\text{us}}$. The user generates the raw requirement $\boldsymbol{s}_0^{\text{raw}}$ and passes it to the market. The market recommends several helpful learnwares $\{f_i | i \in I\}$ based on specifications $\{s_i\}_{i=1}^N$ and the requirement $\boldsymbol{s}_0^{\text{raw}}$ accordingly where I is the indices of recommended learnwares. The user solves her problem by reusing learnwares $\{f_i | i \in I\}$ to construct a model f_0 and tries to minimizing its expected error $E_{(\boldsymbol{x},y)\sim\mathcal{D}_0}(l(f_0(\boldsymbol{x}), y))$ where $\mathcal{D}_0$ is the data distribution of the user. The overall procedure is illustrated in Figure 2.

4 General Framework

In this section, we provide a general framework for handling the heterogeneous learnware problem based on the RKME specification. The problem involves two main challenges:

- How to align raw specifications of models $\{\boldsymbol{s}_i^{\text{raw}}\}_{i=1}^N$ and the raw user's requirement $\boldsymbol{s}_0^{\text{raw}}$ located in different spaces?
- How does the user reuse recommended models whose feature space may be different from the user's task?

The first problem is fundamental because the learnware recommendation relies on model specifications and user requirements being in the same space, and the recommendation procedure significantly impacts the second reuse problem. To address these challenges simultaneously, our general framework proposes a key idea: connecting different feature spaces associated with models and future user tasks to a shared subspace using only *raw specifications* of submitted models.

One approach is connecting T feature spaces of models $\{\mathcal{X}_j^{\text{up}}\}_{j=1}^T$ to a common subspace and generate $2T$ mapping functions, which consists of T projection functions mapping the data from original feature space to an identical subspace $\{h_j : \mathcal{X}_j^{\text{up}} \mapsto \mathcal{X}_{\text{sub}}\}_{j=1}^T$ and T reconstruction functions mapping the data in the subspace to the original feature space $\{g_j : \mathcal{X}_{\text{sub}} \mapsto \mathcal{X}_j^{\text{up}}\}_{j=1}^T$. However, this approach has a significant limitation. When the user's task feature space $\mathcal{X}^{\text{us}}$, is not included in $\{\mathcal{X}_j^{\text{up}}\}_{j=1}^T$, the market is unable to recommend models or assist the user in reusing models. Noticing that the feature spaces of submitted models $\{\mathcal{X}_j^{\text{up}}\}_{j=1}^T$ and user's task $\mathcal{X}^{\text{us}}$ are actually the Cartesian product of a subset of component feature spaces $\{\mathcal{X}_k\}_{k=1}^Q$, it is more effective to connect component feature spaces $\{\mathcal{X}_k\}_{k=1}^Q$ to a subspace and generate $2Q(Q \ll T)$ functions $\{h_k : \mathcal{X}_k \mapsto \mathcal{X}_{\text{sub}}\}_{k=1}^Q, \{g_k : \mathcal{X}_{\text{sub}} \mapsto \mathcal{X}_k\}_{k=1}^Q$ instead. This approach heavily reduces the number of mapping functions required and can handle future user tasks defined on *any* combinations of component feature spaces. The generated mapping functions provide a unified solution to address the aforementioned challenges.

- The raw specifications $\{\boldsymbol{s}_i^{\text{raw}}\}_{i=1}^N$ and raw requirement $\boldsymbol{s}_0^{\text{raw}}$ are generated by RKME and they are based on the reduced set containing minor samples and their weights. By utilizing projection functions $\{h_k : \mathcal{X}_k \mapsto \mathcal{X}_{\text{sub}}\}_{k=1}^Q$, samples defined on different feature spaces $\{\mathcal{X}_j^{\text{up}}\}_{j=1}^T, \mathcal{X}^{\text{us}}$ can be aligned to the common subspace $\mathcal{X}_{\text{sub}}$.
- When the user wants to reuse a model with different feature space, i.e., $\mathcal{X}_j^{\text{up}} \neq \mathcal{X}^{\text{us}}$, missing features that model prediction needs must be filled up, the user can fist project the data to the subspace via projection functions $\{h_k : \mathcal{X}_k \mapsto \mathcal{X}_{\text{sub}}\}_{k=1}^Q$ and then reconstruct them in corresponding missing component feature spaces via reconstruction functions $\{g_k : \mathcal{X}_{\text{sub}} \mapsto \mathcal{X}_k\}_{k=1}^Q$.

In the following, we describe the general framework about how to learn a common subspace and generate mapping functions using raw specifications of models. The market receives N models from T feature spaces with their raw specifications implemented by RKME and the corresponding reduced sets are $\{\boldsymbol{s}_i^{\text{raw}} := \{(\beta_{ij}, \boldsymbol{z}_{ij})\}_{j=1}^{m_i}\}_{i=1}^N$ where m_i is the size of the i-th reduced set. For simplicity, we define the union set of all reduced sets as $U := \{\beta_l, \boldsymbol{z}_l\}_{l=1}^m$ where $m = \sum_{i=1}^N m_i$.

The procedure of subspace learning uses whole reduced set U to optimize mapping functions between component feature spaces and subspace $\{h_k, g_k\}_{k=1}^{Q}$, it also generate sample projections $\{v_l\}_{l=1}^{m}$. The objective accordingly is

$$\min_{\{h_k,g_k\},\{\boldsymbol{v}_l\}} \sum_{l=1}^{m} \beta_l L(\boldsymbol{z}_l, \boldsymbol{v}_l) + S(\{\boldsymbol{z}_l\}_{l=1}^{m}, \{\boldsymbol{v}_l\}_{l=1}^{m}). \quad (2)$$

The objective comprises two parts: the weighted subspace learning loss and the similarity loss. Specifically, the first term calculates the sum of the losses of each sample $\boldsymbol{z}_l$ in the reduced set, weighted by β_l, during subspace learning. Since each sample's feature space is a combination of multiple components $\mathcal{X}_k$, the subspace learning loss can be decomposed as $L(\boldsymbol{z}_l, \boldsymbol{v}_l) = \sum_{k\in C} L(\boldsymbol{z}_l^{(k)}, \boldsymbol{v}_l^{(k)})$, where C represents the set of component indices, $\boldsymbol{z}_l^{(k)}$ and $\boldsymbol{v}_l^{(k)}$ are the slices of $\boldsymbol{z}_l, \boldsymbol{v}_l$ on $\mathcal{X}_k$. The loss $L(\boldsymbol{z}_l^{(k)}, \boldsymbol{v}_l^{(k)})$ can be defined in three ways:

- Defined on the original component feature space $\mathcal{X}_k$. For example, $L_o = \|\boldsymbol{z}_l^{(k)} - g_k(\boldsymbol{v}_l^{(k)})\|$. After obtaining g_k, h_k can be optimized by $\min_{\boldsymbol{v}} \sum_{l\in I_k} \|\boldsymbol{z}_l^{(k)} - g_k(\boldsymbol{v})\|$ where I_k is indices of samples whose feature space contains $\mathcal{X}_k$.
- Defined on the identical subspace $\mathcal{X}_{\text{sub}}$. For example, $L_s = \|h_k(\boldsymbol{z}_l^{(k)}) - \boldsymbol{v}_l^{(k)}\|$. After obtaining h_k, g_k can be optimized by $\min_{\boldsymbol{z}} \sum_{l\in I_k} \|h_k(\boldsymbol{z}) - \boldsymbol{v}_l^{(k)}\|$.
- Defined on both component feature space and subspace like $L_o + L_s$. $\{h_k, g_k\}_{k=1}^{Q}$ can be obtained simultaneously.

The second item indicates the similarity loss. This loss aims to ensure that when two samples $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are similar, their respective projections $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ exhibit similarity as well. To be noticed that $\boldsymbol{z}_i, \boldsymbol{z}_j$ may defined on *different* feature space. In such a case, the similarity can be calculated on the intersection of two feature spaces, which is still the Cartesian product of some components within $\{\mathcal{X}_k\}_{k=1}^{Q}$.

The paper presents a general framework that supports three types of subspace learning loss, and can be incorporated with various existing subspace learning methods to obtain different mapping functions $\{h_k\}_{k=1}^{Q}, \{g_k\}_{k=1}^{Q}$. The subsequent section describes a matrix factorization-based implementation that satisfies the first type of loss.

5 Matrix Factorization Implementation

In this section, we present our matrix factorization-based implementation of the general framework, which serves as the foundation for the overall procedure of constructing and utilizing the heterogeneous learnware market.

5.1 Construct the Learnware Market

The market receives N models with raw specifications from developers and submitted models totally come from T different feature spaces. The raw specification implemented by RKME transforms a data set defined on $\mathcal{X}_t^{\text{up}}$ $(t \in 1, \cdots, T)$ to an element in a particular specification space (RKHS) $\mathcal{H}_t$. Therefore, total N raw specifications $\{\boldsymbol{s}_i^{\text{raw}}\}_{i=1}^{N}$ locate in T different specification spaces. In order to better organize heterogeneous models to support service of model recommendation and reuse, the market constructs the learnware market by merging T specification spaces to a global one and adjusting raw specifications $\{\boldsymbol{s}_i^{\text{raw}}\}_{i=1}^{N}$ to specifications $\{\boldsymbol{s}_i\}_{i=1}^{N}$.

Concept factorization. As described in the general framework, merging different specification spaces (align different raw specifications) is achieved by subspace learning, we first present the preliminary of a classical matrix factorization-based subspace learning method called Concept Factorization (CF) [Xu and Gong, 2004; Cai *et al.*, 2010; Wang *et al.*, 2016]. Given a data matrix $\mathbf{Z} = [\boldsymbol{z}_1, \cdots, \boldsymbol{z}_N] \in \mathbb{R}^{d\times n}$, CF first generates k concepts which are the linear combination of the original data $\mathbf{C} := \mathbf{ZW} = [\boldsymbol{c}_1, \cdots, \boldsymbol{c}_k] \in \mathbb{R}^{d\times k}$, then CF uses concepts to reconstruct the original data by linear combination with the coefficient matrix $\mathbf{V}$. The two coefficient matrices $\mathbf{W}, \mathbf{V}$ is learned by minimizing the reconstruction error $\|\mathbf{Z} - \mathbf{ZWV}^\top\|_{\text{F}}^2$ and the reconstruction coefficient matrix $\mathbf{V}$ is projection of $\mathbf{Z}$ in the subspace with dimension k.

Subspace learning. Based on CF, we proposed the following objective function to learn a subspace by only using raw specifications $\{\boldsymbol{s}_i^{\text{raw}} := \{(\beta_{ij}, \boldsymbol{z}_{ij})\}_{j=1}^{m_i}\}_{i=1}^{N}$ located in different spaces instead of accessing to raw data of model or collecting extra data across the entire feature space. The objective is optimized over $\{\mathbf{W}^{(k)}\}, \{\mathbf{V}^{(k)}\}, \{(\mathbf{V}^*)^{(k)}\}$.

$$\begin{aligned}\min \sum_{k=1}^{Q} \Big(& \|[\mathbf{Z}^{(k)} - \mathbf{Z}^{(k)}\mathbf{W}^{(k)}(\mathbf{V}^{(k)})^\top](\mathbf{\Gamma}^{(k)})^{1/2}\|_{\text{F}}^2 \\ & + \alpha \operatorname{Tr}((\mathbf{V}^{(k)})^\top \mathbf{L}^{(k)} \mathbf{V}^{(k)}) \\ & + \gamma \|(\mathbf{\Gamma}^{(k)})^{1/2}[\mathbf{V}^{(k)} - (\mathbf{V}^*)^{(k)}]\|_{\text{F}}^2 \Big) \\ \text{s.t. } & \mathbf{W}^{(k)} \geq 0. \end{aligned} \quad (3)$$

For the sample $\boldsymbol{z}_{ij}$ defined on $\mathcal{X}_{\phi_i}^{\text{up}} = \times_{i\in C_\phi} \mathcal{X}_i$ with related component feature space index set C_{ϕ_i}, it can be split into several slices $\{\boldsymbol{z}_{ij}^{(k)}\}_{k\in C_{\phi_i}}$. $\mathbf{Z}^{(k)}$ is the concatenation of all slices on $\mathcal{X}_k$, $\mathbf{\Gamma}^{(k)}$ is a diagonal matrix constructed from weights associated with $\mathbf{Z}^{(k)}$, $\mathbf{L}^{(k)}$ is the Laplacian matrix calculated on $\mathbf{Z}^{(k)}$. $\mathbf{V}^{(k)}$ is the projection of $\mathbf{Z}^{(k)}$ in the subspace and $(\mathbf{V}^*)^{(k)}$ is the intermediate optimization results of $\mathbf{V}^{(k)}$. α, γ are regularizer coefficients. The objective sums the loss of Q component feature spaces, each consists of three items, the first is the reconstruction error of subspace learning, the second is a manifold regularizer which keeps the local structure and the third enforces similarity among the projections of $\{\boldsymbol{z}_{ij}^{(k)}\}_{k\in C_{\phi_i}}$. The iterative optimization using gradient descent and multiplicative updated rule [Févotte and Idier, 2011] is described in the Appendix.

Subspace projection and reconstruction. During the step of subspace learning, the base matrix $\mathbf{B}^{(k)} = \mathbf{Z}^{(k)}\mathbf{W}^{(k)}$ of the component feature space $\mathcal{X}_k$ is obtained, resulting in the reconstruction function $g_k : \mathcal{X}_{\text{sub}} \mapsto \mathcal{X}_k$ given by

$$g_k(\boldsymbol{v}) = \mathbf{B}^{(k)}\boldsymbol{v}. \quad (4)$$

Subsequently, the subspace projection functions $h_k : \mathcal{X}_k \mapsto \mathcal{X}_{\text{sub}}$ can be obtained by $\min_{\boldsymbol{v}} \|\boldsymbol{z} - \mathbf{B}^{(k)}\boldsymbol{v}\|_{\text{F}}^2$, which has a closed-form solution given by

$$h_k(\boldsymbol{z}) = ((\mathbf{B}^{(k)})^\top \mathbf{B}^{(k)})^{-1}(\mathbf{B}^{(k)})^\top \boldsymbol{z}. \quad (5)$$

Heterogeneous learnware market construction. Based on the aforementioned matrix factorization-based implementation for the general framework, the learnware market can generate mapping functions $\{h_k, g_k\}_{k=1}^{Q}$ from submitted raw specifications $\boldsymbol{s}_i^{\text{raw}}$ to connect component feature spaces $\{\mathcal{X}_k\}_{k=1}^{Q}$ to the unified subspace $\mathcal{X}_{\text{sub}}$. For the raw specification of i-th model $\boldsymbol{s}_i^{\text{raw}} := \{(\beta_{ij}, \boldsymbol{z}_{ij})\}_{j=1}^{m_i}$ whose component feature space index set is C_{ϕ_i}, the market transforms it to the specification $\boldsymbol{s}_i = \{(\beta_{ij}, \boldsymbol{v}_{ij})\}_{j=1}^{m_i}$ by

$$\boldsymbol{v}_{ij} = \frac{1}{|C_{\phi_i}|} \sum_{k \in C_{\phi_i}} h_k(\boldsymbol{z}_{ij}^{(k)}). \tag{6}$$

The raw specifications $\{\boldsymbol{s}_i^{\text{raw}}\}_{i=1}^{k}$ are located in T different specification spaces initially, but with market merging different specification spaces, all adjusted specifications $\{\boldsymbol{s}_i\}_{i=1}^{N}$ are located in an identical specification space.

5.2 Exploit the Learnware Market

Learnware recommendation. When the user submits her task requirement $\boldsymbol{s}_0^{\text{raw}} := \{\beta_{0j}, \boldsymbol{z}_{0j}\}_{j=1}^{m_0}$ generated by RKME, the market uses subspace projection functions $\{h_k\}_{k=1}^{Q}$ to project the requirement to the unified specification space which contains all specifications $\{\boldsymbol{s}_i\}_{i=1}^{N}$ via Eq. (6) and the projected requirement is $\boldsymbol{s}_0 := \{\beta_{0j}, \boldsymbol{v}_{0j}\}_{j=1}^{m_0}$. The market then calculate the relevance for each learnware by

$$\min_{\boldsymbol{w}} \left\| \Phi_0(\cdot) - \sum_{i=1}^{N} \omega_i \Phi_i(\cdot) \right\|_{\mathcal{H}}^2, \text{ s.t. } \omega_i \geq 0, \sum_{i=1}^{N} \omega_i = 1, \tag{7}$$

where $\Phi_0(\cdot) = \sum_{j=1}^{m_0} \beta_{0j} k(\boldsymbol{v}_{0j}, \cdot)$ is KME of the adjusted requirement $\boldsymbol{s}_0$ and $\Phi_i(\cdot) = \sum_{j=1}^{m_i} \beta_{ij} k(\boldsymbol{v}_{ij}, \cdot)$ is KME of adjusted specification $\boldsymbol{s}_i$. The relevance estimation problem Eq. (7) can be solved by quadratic programming [Smola *et al.*, 2007]. With pre-defined threshold th, the learnware market recommends learnwares whose relevance is above the threshold. Furthermore, the market will pass a learnware selector [Wu *et al.*, 2023] and mapping functions $\{h_k, g_k\}_{k=1}^{Q}$ to the user. The learnware selector is used to predict which learnware should each sample use and the selector $F(\cdot)$ is trained by the samples in $\{\boldsymbol{s}_i | \omega_i \geq th\}$ and the labels are corresponding learnware indices.

Learnware reuse. When the user receives recommended models $\{f_i | \omega_i \geq th\}$, the learnware selector $F(\cdot)$ and mapping functions $\{h_k, g_k\}_{k=1}^{Q}$, the user can make a prediction via dynamic classifier selection [Zhou, 2012] and feature space transformation. More specifically, the user first projects her task data $D_0 := \{\boldsymbol{x}_{0j}\}_{j=1}^{n_0}$ to the subspace via Eq. (6) and get $D_0^{\text{proj}} := \{\boldsymbol{v}_{0j}\}_{j=1}^{n_0}$. The learnware selector predicts on D_0^{proj} to decide for each example which model should be used. For example, the sample $\boldsymbol{x}_{0j}$ is predicted to use the learnware $f_{F(\boldsymbol{v}_{0j})}$, when the model shares the same feature space with $\boldsymbol{x}_{0j}$, the model can make a prediction directly, otherwise, the user needs to fill up the corresponding missing part by $g_k(\boldsymbol{v}_{0j})$. For example, the model is defined on $\mathcal{X}_1 \times \mathcal{X}_2$ and the instance $\boldsymbol{x}_{0j}$ is defined on $\mathcal{X}_2 \times \mathcal{X}_3$, then the user needs to fill up the data on $\mathcal{X}_1$ by $g_1(\boldsymbol{v}_{0j})$.

Algorithm 1 Submitting stage (market construction)

1: Each developer trains a model f_i and generates the raw specification $\boldsymbol{s}_i^{\text{raw}}$ on her dataset D_i.
2: Each developer uploads both model and the raw specification $(f_i, \boldsymbol{s}_i^{\text{raw}})$ to the learnware market.
3: The learnware market generates $2Q$ mapping functions $\{h_k, g_k\}_{k=1}^{Q}$ on all feature space components $\{\mathcal{X}_i\}_{i=1}^{Q}$ based on submitted raw specifications $\{\boldsymbol{s}_i^{\text{raw}}\}_{i=1}^{N}$.
4: The learnware market uses mapping functions $\{h_k\}_{k=1}^{Q}$ to generate the specification $\boldsymbol{s}_i$ for each uploaded model f_i based on the raw specification $\boldsymbol{s}_i^{\text{raw}}$.
5: The heterogeneous learnware market is established as $\{(f_i, \boldsymbol{s}_i)\}_{i=1}^{N}$.

Algorithm 2 Deploying stage (market exploitation)

1: The user generates her requirement of the task $\boldsymbol{s}_0^{\text{raw}}$ and passes it to the market.
2: The market uses mapping functions $\{h_k\}_{k=1}^{Q}$ to post-process the user requirement and get $\boldsymbol{s}_0$.
3: The market recommends models based on the requirement $\boldsymbol{s}_0$ and specifications $\{\boldsymbol{s}_i\}_{i=1}^{N}$, it also passes mapping functions $\{h_k, g_k\}_{k=1}^{Q}$ and the model selector F in the subspace.
4: The user reuses recommended models on her task via dynamic classifier selection.

5.3 Overall Procedure

In the submitting stage, the market manager receives models from different feature spaces and builds a heterogeneous learnware market by aligning specifications to a common space. In the deploying stage, the user describes her requirement of the task and get help from the market by reusing recommended learnwares. The overall procedure is sketched in Algorithms 1 and 2.

6 Experiments

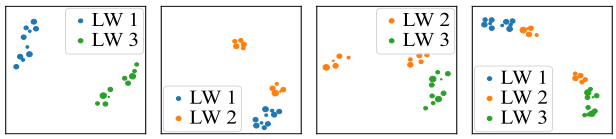
The experiments[1] consists of two parts, we first illustrate the overall procedure through a toy example for visualizing the superiority of our solution, and then we present performance on several real-world tasks to showcase the effectiveness.

6.1 Toy Example

In the toy example, we assume the overall feature space is split into three components $\mathcal{X}_{\text{all}} = \mathcal{X}_1 \times \mathcal{X}_2 \times \mathcal{X}_3$. The overall feature space $\mathcal{X}_{\text{all}}$ is $\mathbb{R}^6$ and each component $\mathcal{X}_i$ is $\mathbb{R}^2$.

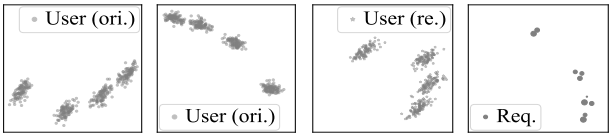
In the submitting stage, there are three developers submit their models defined over $\mathcal{X}_1 \times \mathcal{X}_2, \mathcal{X}_2 \times \mathcal{X}_3, \mathcal{X}_1 \times \mathcal{X}_3$ separately and each model is a 2-class support vector machine using RBF kernel. Raw specifications are illustrated in Figures 3(a), 3(b) and 3(c) by the reduced set (minor weighted samples) and based on which the market generates mapping functions. Then, the market can adjust three specifications located in three spaces to a unified one as Figure 3(d) shows.

[1] https://github.com/LAMDA-TP/Heterogeneous-learnware-without-auxiliary-data



(a) $\mathcal{X}_1$ (b) $\mathcal{X}_2$ (c) $\mathcal{X}_3$ (d) $\mathcal{X}_{\text{sub}}$

Figure 3: Submitting stage: (a), (b) and (c) present raw specifications of three learnwares in three spaces, (d) illustrates the learned unified space for all specifications.



(a) $\mathcal{X}_1$ (b) $\mathcal{X}_2$ (c) $\mathcal{X}_3$ (d) $\mathcal{X}_{\text{sub}}$

Figure 4: Deploying stage: (a) and (b) present the user data defined on the union of two component feature spaces, (d) is the projection of the requirement in the subspace. (c) shows the reconstructed user data in $\mathcal{X}_3$ for reusing recommended models.

In the deploying stage, the user possesses the unlabeled data defined over $\mathcal{X}_1 \times \mathcal{X}_2$ showed in Figures 4(a) and 4(b) and hopes to receive service from the market. The distribution of user's task is a mixture of that of learnware $\#2$ and $\#3$ with equal weights (the mixture is defined over $\mathcal{X}$). The user first generates the task requirement and submit it to the market and then the market projects the requirement in the unified space which contains all specifications as Figure 4(d) shows. After requirement projection, the market is able to calculate the relevance of each learnwares as $[0, 0.457, 0.543]$. With preset threshold $L = 0.1$, the learnware market recommends models $\#2, \#3$ to the user. To be noticed that the feature space of user's task $\mathcal{X}_1 \times \mathcal{X}_2$ doesn't match that of learnwares $\#2, \#3$ ($\mathcal{X}_2 \times \mathcal{X}_3, \mathcal{X}_1 \times \mathcal{X}_3$), the user can reconstruct data on $\mathcal{X}_3$ based on the original data and mapping functions as Figure 4(c) shows. After features are complemented, the user reuses recommended models via dynamic classifier selection and the final accuracy is 0.995.

If the market only recommends learnwares which precisely match user's feature space, then the totally irrelevant model $\#1$ is recommended to the user and the final accuracy on user's task is 0, which shows the necessity of exploring learnwares whose feature spaces are not matched with user's task.

6.2 Real-world Tasks

Data set. We conduct empirical experiments on six heterogeneous learnware scenarios involving five real-world tasks: Mfeat [van Breukelen *et al.*, 1998], Anuran [Colonna *et al.*, 2012], Digits [Garris *et al.*, 1997], Kddcup99 [Lippmann *et al.*, 2000] and Covtype [Blackard and Dean, 1999]. Mfeat is a digit recognition data set with six feature spaces, we split it into two parts, generating two separate scenarios. Anuran is used for classifying Anuran sounds. Digits is a collection of handwritten numbers, Kddcup99 is a dataset for network intrusion detection, and Covtype is used for classifying the cover type (the dominant species of trees) in forest patches across the United States. The original datasets of Anuran, Digits, Kddcup99, and Covtype each have a single feature space. We randomly split them into three parts. The information of five processed data set is shown in Table 1.

Dataset	#samples	#classes	#dim
Mfeat (far, kar, pix)	2000	10	[76, 216, 64]
Mfeat (zer, fou, mor)	2000	10	[240, 47, 6]
Anuran	7195	10	[7,7,8]
Digits	1797	10	[21, 21, 22]
Kddcup99	3200	6	[29, 29, 30]
Covtype	6000	6	[18, 18, 18]

Table 1: Information of data sets.

Contenders. As the heterogeneous learnware problem is a new problem, we first compare with two basic contenders.

- `Random`: The market randomly selects a model whose feature space is identical with the user's task and the user uses it directly to make a prediction.
- `Ensemble`: The market selects all models whose feature space are identical with the user's task and the user uses them via average ensemble.

Then we consider other three additional contenders in which model recommendation and reuse rely on specifications.

- `MMD`: Each model is equipped with the raw specification and the market recommends the learnware with minimum maximum mean discrepancy within models sharing the same feature space with the user's task. The user reuses the model directly [Wu *et al.*, 2023].
- `Auxiliary`: Each model is equipped with specification generated based on the raw data of the model and the extra *auxiliary* data across the entire feature space. The user reuses recommended models via dynamic classifier selection [Tan *et al.*, 2022].
- `Projection`: This method is slightly different from our method. The market recommends learnwares instead of models to the user and the user samples a mimic data set by kernel herding [Chen *et al.*, 2012] for each learnware and uses the learnware to make a prediction, then the user project multiple pseudo-labeled mimic data sets to the subspace and trains a classifier. Finally, the prediction is made on the projected task data with newly trained model.

Experiment setup. The overall feature space is split into three parts $\mathcal{X}_{\text{all}} = \mathcal{X}_1 \times \mathcal{X}_2 \times \mathcal{X}_3$, we generate 6 learnwares on three feature spaces ($\mathcal{X}_1 \times \mathcal{X}_2, \mathcal{X}_2 \times \mathcal{X}_3, \mathcal{X}_1 \times \mathcal{X}_3$) and each learnware only classifies a subset of entire classes. The distribution of the user's task is a mixture of several learnwares with equal weights. In our experiment, parameters are set as follows: the reduced set size m_i is 10, which is much smaller than the size of original data set n_i ($m_i = O(\ln n_i)$). For subspace learning, the trade-off parameters is set as $\alpha = 10^{-5}, \gamma = 1$, the max iteration is $t = 500$ and the learning rate is $\eta = 10^{-2}$. The dimension of subspace is chosen by cross validation. We test several model types like SVM and random forest. All experiments are repeated 50 times.

User's task performance. Table 2 shows the accuracy of different methods on the user's task, with our method

Task name	#Mix	Random	Ensemble	MMD	Auxiliary	Projection	Ours
Mfeat (far, kar, pix)	1	0.284 ± 0.358	0.599 ± 0.357	0.568 ± 0.352	0.537 ± 0.306	0.629 ± 0.256	$\mathbf{0.770 \pm 0.223}$
	2	0.299 ± 0.205	0.640 ± 0.229	0.483 ± 0.176	0.576 ± 0.194	0.623 ± 0.135	$\mathbf{0.735 \pm 0.115}$
	3	0.284 ± 0.146	0.612 ± 0.173	0.405 ± 0.113	0.549 ± 0.149	0.579 ± 0.114	$\mathbf{0.665 \pm 0.108}$
Mfeat (zer, fou, mor)	1	0.279 ± 0.353	0.522 ± 0.323	0.555 ± 0.343	0.536 ± 0.317	0.531 ± 0.338	$\mathbf{0.590 \pm 0.357}$
	2	0.292 ± 0.201	0.548 ± 0.208	0.436 ± 0.197	0.572 ± 0.205	0.546 ± 0.198	$\mathbf{0.584 \pm 0.207}$
	3	0.277 ± 0.142	0.526 ± 0.153	0.357 ± 0.129	0.547 ± 0.153	0.519 ± 0.148	$\mathbf{0.555 \pm 0.157}$
Anuran	1	0.235 ± 0.310	0.409 ± 0.334	0.462 ± 0.365	0.418 ± 0.293	0.543 ± 0.268	$\mathbf{0.597 \pm 0.254}$
	2	0.192 ± 0.180	0.417 ± 0.259	0.364 ± 0.245	0.407 ± 0.224	0.437 ± 0.176	$\mathbf{0.468 \pm 0.156}$
	3	0.163 ± 0.126	0.356 ± 0.203	0.294 ± 0.183	0.357 ± 0.180	0.400 ± 0.163	$\mathbf{0.420 \pm 0.115}$
Digits	1	0.284 ± 0.355	0.513 ± 0.310	0.569 ± 0.350	0.527 ± 0.320	0.626 ± 0.096	$\mathbf{0.729 \pm 0.246}$
	2	0.299 ± 0.204	0.551 ± 0.197	0.480 ± 0.182	0.549 ± 0.205	0.500 ± 0.087	$\mathbf{0.581 \pm 0.117}$
	3	0.283 ± 0.145	0.526 ± 0.152	0.411 ± 0.121	0.527 ± 0.152	0.479 ± 0.051	$\mathbf{0.550 \pm 0.107}$
Kddcup99	1	0.320 ± 0.370	0.550 ± 0.247	0.638 ± 0.316	0.609 ± 0.276	0.719 ± 0.205	$\mathbf{0.735 \pm 0.239}$
	2	0.324 ± 0.219	0.571 ± 0.146	0.503 ± 0.195	0.642 ± 0.166	0.592 ± 0.126	$\mathbf{0.687 \pm 0.164}$
	3	0.309 ± 0.148	0.550 ± 0.118	0.418 ± 0.137	0.616 ± 0.133	0.576 ± 0.117	$\mathbf{0.652 \pm 0.123}$
Covtype	1	0.293 ± 0.349	0.391 ± 0.372	0.491 ± 0.375	0.352 ± 0.208	0.534 ± 0.268	$\mathbf{0.575 \pm 0.300}$
	2	0.240 ± 0.197	0.297 ± 0.214	0.335 ± 0.217	0.334 ± 0.143	0.383 ± 0.129	$\mathbf{0.427 \pm 0.126}$
	3	0.251 ± 0.154	0.322 ± 0.161	0.321 ± 0.166	0.338 ± 0.091	0.368 ± 0.080	$\mathbf{0.378 \pm 0.082}$
Ours: win/tie/loss		18/0/0	18/0/0	18/0/0	18/0/0	18/0/0	Rank first 18/18

Table 2: Accuracy (mean $\pm$ std.) on true labels of the user data. The best method is emphasized in bold.

consistently outperforming the others. However, `Random`, `Ensemble`, and `MMD` don't fully leverage the potential of the learnware market as they only recommend learnwares within the same feature space. Among them, `Random` performs the worst, while `MMD` achieves better accuracy using distribution information. Compared to these methods, `Auxiliary` identifies all heterogeneous models but can't reuse models with partially intersected feature spaces. In contrast, both `Projection` and our method can identify and reuse all heterogeneous models. `Projection` uses pseudo-labels to reuse models, while we directly use models for predictions. Our method offers a superior approach to reusing models and provides better data privacy protection than `Projection`.

7 Related Work

The learnware paradigm [Zhou, 2016; Zhou and Tan, 2022] devotes to build a model market to help users identify and reuse helpful models for their tasks instead of starting from scratch. When all models share the same feature space, the learnware identification can be conducted by RKME matching and job selector [Wu *et al.*, 2023]. Based on RKME specification, the market can also identify unseen part within user's data and handle the remaining part [Zhang *et al.*, 2021]. When models have different feature spaces, existing study can only initialize the market with the help of extra auxiliary data [Tan *et al.*, 2022].

Domain adaptation [Ben-David *et al.*, 2006] and transfer learning [Pan and Yang, 2009; Ding *et al.*, 2022] transfer the knowledge from the source domain to the target domain. This presupposes a similarity between target and source tasks to avoid negative transfer [Wang *et al.*, 2019]. However, within the learnware paradigm, the user's task may only correlate with a few learnwares in the market, making model identification vital. Model reuse focuses on reusing existing models for a current task without raw data access [Zhao *et al.*, 2020; Ding and Zhou, 2020]. However, it assumes all models are beneficial to the task, which differs from the learnware paradigm where only a few models are useful.

Existing studies on heterogeneous feature spaces including heterogeneous domain adaptation [Duan *et al.*, 2012; Wang and Mahadevan, 2011], heterogeneous transfer learning [Day and Khoshgoftaar, 2017], heterogeneous model reuse [Ye *et al.*, 2018; Ye *et al.*, 2020], etc., generally map different feature spaces to an intermediate subspace. In this process, original data from both domains or co-occurrence data are always necessary for constructing the relationship between different spaces. However, in the learnware paradigm, handling models developed from different feature spaces without auxiliary data becomes feasible due to the existence of RKME specifications associated with each model. Based on the learnware paradigm, we realize to solve the challenge of accommodating, identifying, and reusing heterogeneous models of any type while having no access to original data or extra co-occurrence auxiliary data.

8 Conclusion

This paper presents a general framework for constructing and utilizing the heterogeneous learnware market without relying on additional auxiliary data, while ensuring the privacy of the developers and users. The key idea is to utilize the raw RKME specifications to generate mapping functions that connect different feature spaces to a shared subspace. The generated mapping functions allow for the adjustment of raw specifications during market construction and the handling of missing features when users exploit the market. A compelling issue for future research is how to further adapt mapping functions on user's task for better learnware reuse.

Acknowledgements

This research was supported by NSFC (62250069) and the Collaborative Innovation Center of Novel Software Technology and Industrialization. Authors want to thank Jin-Hui Wu for helpful discussions, and thank reviewers for helpful comments.

References

[Ben-David *et al.*, 2006] Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. Analysis of representations for domain adaptation. In *Advances in Neural Information Processing Systems 19*, 2006.

[Blackard and Dean, 1999] Jock A Blackard and Denis J Dean. Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. *Computers and Electronics in Agriculture*, pages 131–151, 1999.

[Cai *et al.*, 2010] Deng Cai, Xiaofei He, and Jiawei Han. Locally consistent concept factorization for document clustering. *IEEE Transactions on Knowledge & Data Engineering*, 23(6):902–913, 2010.

[Chen *et al.*, 2012] Yutian Chen, Max Welling, and Alex Smola. Super-samples from kernel herding. *arXiv:1203.3472*, 2012.

[Colonna *et al.*, 2012] Juan Gabriel Colonna, Afonso D. Ribas, Eulanda Miranda dos Santos, and Eduardo Freire Nakamura. Feature subset selection for automatically classifying anuran calls using sensor networks. In *Proceedings of the 2012 International Joint Conference on Neural Networks*, pages 1–8, 2012.

[Day and Khoshgoftaar, 2017] Oscar Day and Taghi M Khoshgoftaar. A survey on heterogeneous transfer learning. *Journal of Big Data*, 4:1–42, 2017.

[Ding and Zhou, 2020] Yao-Xiang Ding and Zhi-Hua Zhou. Boosting-based reliable model reuse. In *Proceedings of the 12th Asian Conference on Machine Learning*, pages 145–160, 2020.

[Ding *et al.*, 2022] Yao-Xiang Ding, Xi-Zhu Wu, Kun Zhou, and Zhi-Hua Zhou. Pre-trained model reusability evaluation for small-data transfer learning. In *Advances in Neural Information Processing Systems 35*, pages 37389–37400, 2022.

[Duan *et al.*, 2012] Lixin Duan, Dong Xu, and Ivor Tsang. Learning with augmented features for heterogeneous domain adaptation. *arXiv:1206.4660*, 2012.

[Févotte and Idier, 2011] Cédric Févotte and Jérôme Idier. Algorithms for nonnegative matrix factorization with the β-divergence. *Neural Computation*, 23(9):2421–2456, 2011.

[Garris *et al.*, 1997] Michael Garris, J Blue, Gerald Candela, Patrick Grother, Stanley Janet, and Charles Wilson. Nist form-based handprint recognition system. In *NIST Interagency/Internal Report*, 1997.

[Harutyunyan *et al.*, 2019] Hrayr Harutyunyan, Hrant Khachatrian, David C Kale, Greg Ver Steeg, and Aram Galstyan. Multitask learning and benchmarking with clinical time series data. *Scientific Data*, 6(1):1–18, 2019.

[Johnson *et al.*, 2016] Alistair EW Johnson, Tom J Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. MIMIC-III, a freely accessible critical care database. *Scientific Data*, 3(1):1–9, 2016.

[Lippmann *et al.*, 2000] Richard Lippmann, Joshua W Haines, David J Fried, Jonathan Korba, and Kumar Das. Analysis and results of the 1999 darpa off-line intrusion detection evaluation. In *International Workshop on Recent Advances in Intrusion Detection*, pages 162–182, 2000.

[Pan and Yang, 2009] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge & Data Engineering*, 22(10):1345–1359, 2009.

[Purushotham *et al.*, 2018] Sanjay Purushotham, Chuizheng Meng, Zhengping Che, and Yan Liu. Benchmarking deep learning models on large healthcare datasets. *Journal of Biomedical Informatics*, 83:112–134, 2018.

[Schölkopf and Smola, 2002] Bernhard Schölkopf and Alexander Johannes Smola. *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[Sha *et al.*, 2007] Fei Sha, Yuanqing Lin, Lawrence K. Saul, and Daniel D. Lee. Multiplicative updates for nonnegative quadratic programming. *Neural Computation*, 19(8):2004–2031, 2007.

[Smola *et al.*, 2007] Alex Smola, Arthur Gretton, Le Song, and Bernhard Schölkopf. A hilbert space embedding for distributions. In *Proceedings of the 18th International Conference on Algorithmic Learning Theory*, pages 13–31, 2007.

[Sriperumbudur *et al.*, 2011] Bharath K. Sriperumbudur, Kenji Fukumizu, and Gert R. G. Lanckriet. Universality, characteristic kernels and RKHS embedding of measures. *Journal of Machine Learning Research*, 12(7):2389–2410, 2011.

[Tan *et al.*, 2022] Peng Tan, Zhi-Hao Tan, Yuan Jiang, and Zhi-Hua Zhou. Towards enabling learnware to handle heterogeneous feature spaces. *Machine Learning*, 2022.

[van Breukelen *et al.*, 1998] Martijn van Breukelen, Robert PW Duin, David MJ Tax, and JE Den Hartog. Handwritten digit recognition by combined classifiers. *Kybernetika*, 34(4):381–386, 1998.

[Wang and Mahadevan, 2011] Chang Wang and Sridhar Mahadevan. Heterogeneous domain adaptation using manifold alignment. In *Proceedings of the 22th International Joint Conference on Artificial Intelligence*, 2011.

[Wang *et al.*, 2016] Hao Wang, Yan Yang, and Tianrui Li. Multi-view clustering via concept factorization with local

manifold regularization. In *Proceedings of the 16th International Conference on Data Mining*, pages 1245–1250, 2016.

[Wang *et al.*, 2019] Zirui Wang, Zihang Dai, Barnabás Póczos, and Jaime Carbonell. Characterizing and avoiding negative transfer. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pages 11293–11302, 2019.

[Wu *et al.*, 2023] Xi-Zhu Wu, Wenkai Xu, Song Liu, and Zhi-Hua Zhou. Model reuse with reduced kernel mean embedding specification. *IEEE Transactions on Knowledge and Data Engineering*, 35(1):699–710, 2023.

[Xu and Gong, 2004] Wei Xu and Yihong Gong. Document clustering by concept factorization. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 202–209, 2004.

[Yang *et al.*, 2015] Liu Yang, Liping Jing, Jian Yu, and Michael K Ng. Learning transferred weights from co-occurrence data for heterogeneous transfer learning. *IEEE Transactions on Neural Networks and Learning Systems*, 27(11):2187–2200, 2015.

[Ye *et al.*, 2018] Han-Jia Ye, De-Chuan Zhan, Yuan Jiang, and Zhi-Hua Zhou. Rectify heterogeneous models with semantic mapping. In *Proceedings of the 37th International Conference on Machine Learning*, pages 5630–5639, 2018.

[Ye *et al.*, 2020] Han-Jia Ye, De-Chuan Zhan, Yuan Jiang, and Zhi-Hua Zhou. Heterogeneous few-shot model rectification with semantic mapping. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(11):3878–3891, 2020.

[Zhang *et al.*, 2021] Yu-Jie Zhang, Yu-Hu Yan, Peng Zhao, and Zhi-Hua Zhou. Towards enabling learnware to handle unseen jobs. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence*, pages 10964–10972, 2021.

[Zhao *et al.*, 2020] Peng Zhao, Le-Wen Cai, and Zhi-Hua Zhou. Handling concept drift via model reuse. *Machine Learning*, 109(3):533–568, 2020.

[Zhou and Tan, 2022] Zhi-Hua Zhou and Zhi-Hao Tan. Learnware: Small models do big. *arXiv:2210.03647*, 2022.

[Zhou, 2012] Zhi-Hua Zhou. *Ensemble methods: foundations and algorithms*. CRC press, 2012.

[Zhou, 2016] Zhi-Hua Zhou. Learnware: On the future of machine learning. *Frontiers of Computer Science*, 10(4):589–590, 2016.

Supplementary Materials for "Handling Learnwares Developed from Heterogeneous Feature Spaces without Auxiliary Data"

This is the supplemental material for the paper "Handling Learnwares Developed from Heterogeneous Feature Spaces without Auxiliary Data".

A Notations

The major notations of this paper are summarized in Table 3.

Category	Notations	Description
basic	$\mathcal{X}_{\text{all}} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_Q$	the overall feature space and its Q components, the corresponding dimensions are $d_{\text{all}}, d_1, \cdots, d_Q$.
	$\mathcal{X}_1^{\text{up}}, \cdots, \mathcal{X}_T^{\text{up}}$	T kinds of feature spaces for developers' task, each of them is the Cartesian product of several component feature spaces. The index set of components that $\mathcal{X}_k^{\text{up}}$ has is C_k, i.e., $\mathcal{X}_k^{\text{up}} = \times_{i \in C_k} \mathcal{X}_i$.
	$\mathcal{X}^{\text{us}}$	the feature space of the user's task, it is the Cartesian product of several component feature spaces, the corresponding index set of components is C_0, i.e., $\mathcal{X}^{\text{us}} = \times_{i \in C_0} \mathcal{X}_i$.
	$\mathcal{Y}$	the label space.
developer	$D_i := \{(\boldsymbol{x}_{ij}, y_{ij})\}_{j=1}^{n_i}$	the labeled dataset of the i-th developer defined on $\mathcal{X}_{\phi_i}^{\text{up}} \times \mathcal{Y}$ where $k_i \in [1, \cdots, T]$.
	$f_i : \mathcal{X}_{\phi_i}^{\text{up}} \mapsto \mathcal{Y}$	the model of the i-th developer trained on D_i.
	$\boldsymbol{s}_i^{\text{raw}} := \{(\beta_{ij}, \boldsymbol{z}_{ij})\}_{j=1}^{m_i}$	the *raw* specification of the i-th model generated from D_i via RKME.
user	$D_0 := \{\boldsymbol{x}_{0j}\}_{j=1}^{n_0}$	the unlabeled dataset of the user.
	$\boldsymbol{s}_0^{\text{raw}} := \{(\beta_{0j}, \boldsymbol{z}_{0j})\}_{j=1}^{m_0}$	the *raw* requirement of the user generated from D_0 via RKME.
market	$\boldsymbol{s}_i := \{(\beta_{ij}, \boldsymbol{v}_{ij})\}_{j=1}^{m_i}$	the specification of the i-th model given by the learnware market, which is generated by adjusting the raw specification $\boldsymbol{s}_i^{\text{raw}}$.
	$\boldsymbol{l}_i := (f_i, \boldsymbol{s}_i)$	the i-th learnware accommodated by the learnware market.
	$\{\boldsymbol{l}_i\}_{i=1}^{N}$	the heterogeneous learnware market.
	$\boldsymbol{s}_0 := \{(\beta_{0j}, \boldsymbol{v}_{0j})\}_{j=1}^{m_0}$	the requirement of the user generated by the market.
subspace	$\mathcal{X}_{\text{sub}}$	the learned subspace with dimension d_{sub}.
	$h_k : \mathcal{X}_k \mapsto \mathcal{X}_{\text{sub}}$	the mapping function which transform the data on the k-th component feature space $\mathcal{X}_k$ to the subspace $\mathcal{X}_{\text{sub}}$.
	$g_k : \mathcal{X}_{\text{sub}} \mapsto \mathcal{X}_k$	the mapping function which transform the data on the subspace $\mathcal{X}_{\text{sub}}$ to the k-th component feature space $\mathcal{X}_k$.
	$\mathbf{Z}_i$	the concatenation of all specification without weights in the i-th feature space $\mathcal{X}_i^{\text{up}}$ whose domain is $\mathbb{R}^{N_i \times (\sum_{k \in C_i} d_k)}$ where N_i is total number of samples in $\mathcal{X}_i^{\text{up}}$. $\mathbf{Z}_i$ can be split into $\lvert C_i \rvert$ parts: $\{\mathbf{Z}_i^{(j)} \mid j \in C_i\}$.
	$\mathbf{\Gamma}_i$	the i-th weight matrix whose domain is $\mathbb{R}^{N_i \times N_i}$, it is a diagonal matrix constructed from weights of all specification in the i-th feature space $\mathcal{X}_i^{\text{up}}$.
	$\mathbf{V}_i$	the projection of i-th data matrix in the subspace whose domain is $\mathbb{R}^{n_i \times d_{\text{sub}}}$.
	C_i	the index set of component feature space that $\mathbf{Z}_i$ relates to.
	R_k	the index set of data matrix whose feature space contains $\mathcal{X}_k$.
	$\mathbf{Z}^{(k)}$	the concatenation of all data matrix slices in $\mathcal{X}_k$ for all data matrix $\{\mathbf{Z}_i\}$: $\{\mathbf{Z}_i^{(k)} \mid i \in R_k\}$ with the shape $(\sum_{i \in R_k} N_i) \times d_k$.
	$\mathbf{V}^{(k)}$	the concatenation of some mapped data matrices: $\{\mathbf{V}_i^{(k)} \mid i \in R_k\}$ with the shape is $(\sum_{i \in R_k} N_i) \times d_{\text{sub}}$.
	$(\mathbf{V}^*)^{(k)}$	intermediate optimization results of $\mathbf{V}^{(k)}$.
	$\mathbf{L}^{(k)}$	the Laplacian matrix calculated on $\mathbf{Z}^{(k)}$ with the shape $(\sum_{i \in R_k} N_i) \times (\sum_{i \in R_k} N_i)$.
	$\mathbf{\Gamma}^{(k)}$	the diagonal matrix with weights associated with $\mathbf{Z}^{(k)}$, the shape is $(\sum_{i \in R_k} N_i) \times (\sum_{i \in R_k} N_i)$.

Table 3: Notations of this work.

B Optimization

This section provides a detailed description of the optimization methods employed in our study. Initially, we introduce the optimization of RKME with non-negative constraints, followed by the optimization of subspace learning and the associated procedure for generating mapping functions as defined by Eq. (3). Since the subspace learning objective, as denoted by Eq. (3), is not convex across all variables, achieving the global minimum becomes impossible. We propose an iterative optimization solution to attain the local minimum. During each iteration, we employ gradient descent and the multiplicative updated rule to

optimize the variables. We provide a brief introduction of the multiplicative updated rule for nonnegative quadratic programming, followed by a detailed description of our optimization approach.

B.1 Optimization of RKME

The optimization problem of RKME with non-negative constraints is

$$\begin{aligned} \min_{\boldsymbol{\beta},\mathbf{Z}} \quad & \left\| \frac{1}{n}\sum_{i=1}^{n} k\left(\boldsymbol{x}_i,\cdot\right) - \sum_{j=1}^{m}\beta_j k\left(\boldsymbol{z}_j,\cdot\right) \right\|_{\mathcal{H}}^2 \\ \text{s.t.} \quad & \beta_j \geq 0, \end{aligned} \tag{8}$$

which can be solved by alternative optimization. The objective function can be rewritten as

$$\begin{aligned} F(\boldsymbol{\beta},\mathbf{Z}) &= \sum_{i,j=1}^{n}\frac{1}{n^2}k\left(x_i,x_j\right) + \sum_{i,j=1}^{m}\beta_i\beta_j k\left(z_i,z_j\right) - 2\sum_{i=1}^{n}\sum_{j=1}^{m}\frac{\beta_j}{N}k\left(x_i,z_j\right) \\ &= \boldsymbol{\beta}^\top\mathbf{K}_{zz}\boldsymbol{\beta} - 2\frac{\mathbf{1}^\top}{N}\mathbf{K}_{xz}\boldsymbol{\beta} + \frac{1}{N^2}\mathbf{1}^\top\mathbf{K}_{xx}\mathbf{1}, \end{aligned} \tag{9}$$

where $\boldsymbol{\beta} = (\beta_1;\cdots;\beta_m)$, $[\mathbf{K}_{xz}]_{ij} = k(\boldsymbol{x}_i,\boldsymbol{z}_j)$.

Update coefficients β_j. Coefficients β_j can be updated by quadratic programming with non-negative constraint.

Update samples $\boldsymbol{z}_j$. Samples $\boldsymbol{z}_j$ can be updated by gradient descent [Wu *et al.*, 2023].

$$\boldsymbol{z}_j^{(t)} = \boldsymbol{z}_j^{(t-1)} - \eta\frac{\partial F(\boldsymbol{\beta},\mathbf{Z})}{\partial \boldsymbol{z}_j}. \tag{10}$$

B.2 Multiplicative Updated Rule

One efficient way to solve the *nonnegative* quadratic programming is multiplicative update rule [Sha *et al.*, 2007]. We first review the multiplicative update rule as follows,

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 $\mathbf{b}$ is an arbitrary d dimensional vector. Let $\mathbf{A}^+$and $\mathbf{A}^-$denote the nonnegative matrices with elements:

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

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

$$\boldsymbol{x}_i \leftarrow \boldsymbol{x}_i\left[\frac{-\mathbf{b}_i + \sqrt{\mathbf{b}_i^2 + 4\left(\mathbf{A}^+\boldsymbol{x}\right)_i\left(\mathbf{A}^-\boldsymbol{x}\right)_i}}{2\left(\mathbf{A}^+\boldsymbol{x}\right)_i}\right]. \tag{11}$$

In this proposition, the crucial elements used for update are $\boldsymbol{b}$, $\mathbf{A}^+\boldsymbol{x}$ and $\mathbf{A}^-\boldsymbol{x}$. The elements can be calculated by $\mathbf{b} = \nabla f(\boldsymbol{x})|_{\boldsymbol{x}=0}$, $\mathbf{A}\boldsymbol{x} = \nabla f(\boldsymbol{x})|_{\boldsymbol{x}=0} - \mathbf{b}$ and decomposing $\mathbf{A}\boldsymbol{x}$ as $\mathbf{A}^+\boldsymbol{x}$, $\mathbf{A}^-\boldsymbol{x}$ [Tan *et al.*, 2022].

For the nonnegative quadratic optimization when the optimized variable is a matrix, this paper *first* gives the explicit general updated rules. We use the following notation $\mathrm{MUR}(\mathbf{X},\mathbf{B},\mathbf{P},\mathbf{N})$ to describe the update.

$$\mathbf{X}_{ij} \leftarrow \mathbf{X}_{ij}\left(\frac{\mathbf{B}_{ij} + \sqrt{\mathbf{B}_{ij}^2 + 4\mathbf{P}_{ij}\mathbf{N}_{ij}}}{2\mathbf{P}_{ij}}\right), \tag{12}$$

where $\mathbf{B} = -\nabla f(\mathbf{X})|_{\mathbf{X}=0}$. $\mathbf{P}$ and $\mathbf{N}$ are generated from $\nabla f(\mathbf{X}) - \nabla f(\mathbf{X})|_{\mathbf{X}=0}$. $\nabla f(\mathbf{X}) - \nabla f(\mathbf{X})|_{\mathbf{X}=0}$ has the form of $\sum_i \mathbf{C}_i\mathbf{X}\mathbf{D}_i$, which generates $\mathbf{P} = \sum_i(\mathbf{C}_i^+\mathbf{X}\mathbf{D}_i^+ + \mathbf{C}_i^-\mathbf{X}\mathbf{D}_i^-)$ and $\mathbf{N} = \sum_i(\mathbf{C}_i^+\mathbf{X}\mathbf{D}_i^- + \mathbf{C}_i^-\mathbf{X}\mathbf{D}_i^+)$. If all elements of $\mathbf{C}_i$ are non-negative, then $\mathbf{C}_i^- = 0$, $\mathbf{C}_i^+ = \mathbf{C}_i$, and the decomposition $\mathbf{C}_i\mathbf{X}\mathbf{D}_i = (\mathbf{C}_i^+\mathbf{X}\mathbf{D}_i^+ + \mathbf{C}_i^-\mathbf{X}\mathbf{D}_i^-) - \sum_i(\mathbf{C}_i^+\mathbf{X}\mathbf{D}_i^- + \mathbf{C}_i^-\mathbf{X}\mathbf{D}_i^+)$ degenerates to $\mathbf{C}_i\mathbf{X}\mathbf{D}_i = \mathbf{C}_i\mathbf{X}\mathbf{D}_i^+ - \mathbf{C}_i\mathbf{X}\mathbf{D}_i^-$. To be noticed that *the update result won't be changed if $\mathbf{B},\mathbf{P},\mathbf{N}$ are scaled with a common positive real number*. When $\mathbf{N} = \mathbf{0}$ and elements of $\mathbf{B}$ are all *non-negative*, the update rule degenerates to

$$\mathbf{X}_{ij} \leftarrow \mathbf{X}_{ij}\left(\frac{\mathbf{B}_{ij}}{\mathbf{P}_{ij}}\right), \tag{13}$$

B.3 Detailed Optimization

The objective function of subspace learning is reviewed as

$$\begin{aligned}\min_{\{\mathbf{W}^{(k)}\},\{\mathbf{V}^{(k)}\},\{(\mathbf{V}^*)^{(k)}\}} O = \sum_{k=1}^{Q} \Big(&\|[\mathbf{Z}^{(k)} - \mathbf{Z}^{(k)}\mathbf{W}^{(k)}(\mathbf{V}^{(k)})^\top](\mathbf{\Gamma}^{(k)})^{1/2}\|_{\mathrm{F}}^2 \\ &+ \alpha \operatorname{Tr}((\mathbf{V}^{(k)})^\top \mathbf{L}^{(k)}\mathbf{V}^{(k)}) + \gamma\|(\mathbf{\Gamma}^{(k)})^{1/2}[\mathbf{V}^{(k)} - (\mathbf{V}^*)^{(k)}]\|_{\mathrm{F}}^2\Big) \\ \text{s.t.} \quad & \mathbf{W}^{(k)} \geq 0.\end{aligned} \tag{14}$$

Initialization. The parameters $\{\mathbf{W}^{(k)}, \mathbf{V}^{(k)}\}_{k=1}^{Q}$ are initialized with the clustering method. More specifically,we use $\mathbf{C}^{(k)} \in \{0,1\}^{(\sum_{i\in R_k} N_i)\times d_k}$ to denote the indicator matrix of weighted k-means clustering results of $\mathbf{Z}^{(k)} \in \mathbb{R}^{(\sum_{i\in R_k} N_i)\times d_k}$, i.e., if $\boldsymbol{x}_i$ belongs to the j-th cluster, then $\mathbf{C}_{ij}^{(k)} = 1$ and $\mathbf{C}_{il}^{(k)} = 0$ for $l \neq j$. Then, we initialize $\mathbf{W}^{(k)}$ as $\mathbf{W}^{(k)} = (\mathbf{C}^{(k)} + 0.1\mathbf{E}^{(k)})(\mathbf{D}^{(k)})^{-1}$ where $\mathbf{D}^{(k)} = \operatorname{diag}(n_1, \cdots, n_k)$, n_k is the cardinality of the k-th cluster and $\mathbf{E}^{(k)}$ is a matrix with all elements equal to 1. $\{\mathbf{V}^{(k)}\}$ is initialized by $(\mathbf{V}^{(k)})^\top = ((\mathbf{W}^{(k)})^\top\mathbf{K}^{(k)}\mathbf{W}^{(k)})^{-1}(\mathbf{W}^{(k)})^\top\mathbf{K}^{(k)}$. $(\mathbf{V}^*)^{(k)}$ is initialized by concatenating $\{\mathbf{V}_i^*|i \in R_k\}$ where $\mathbf{V}_i^* = \frac{1}{|C_i|}\sum_{k\in C_i}\mathbf{V}_i^{(k)}$.

Optimizing $\mathbf{W}^{(k)}$ with Fixed Other Variables. The subproblem is

$$\begin{aligned}\min_{\mathbf{W}^{(k)}} O &= \|[\mathbf{Z}^{(k)} - \mathbf{Z}^{(k)}\mathbf{W}^{(k)}(\mathbf{V}^{(k)})^\top](\mathbf{\Gamma}^{(k)})^{1/2}\|_{\mathrm{F}}^2 \\ \text{s.t.}\ & \mathbf{W}^{(k)} \geq 0.\end{aligned} \tag{15}$$

For brevity, we ignore the superscript and mark $\mathbf{Z}^{(k)}, \mathbf{W}^{(k)}, \mathbf{V}^{(k)}, \mathbf{\Gamma}^{(k)}, \mathbf{L}^{(k)}$ as $\mathbf{Z}, \mathbf{W}, \mathbf{V}, \mathbf{\Gamma}, \mathbf{L}$. The subproblem is restated as

$$\begin{aligned}\min_{\mathbf{W}} O &= \|(\mathbf{Z} - \mathbf{Z}\mathbf{W}\mathbf{V}^\top)\mathbf{\Gamma}^{1/2}\|_{\mathrm{F}}^2 \\ \text{s.t.}\ & \mathbf{W} \geq 0.\end{aligned} \tag{16}$$

The objective function can be rewritten as

$$O(\mathbf{W}) = \operatorname{Tr}(\mathbf{\Gamma}\mathbf{K}) - 2\operatorname{Tr}(\mathbf{\Gamma}\mathbf{K}\mathbf{W}\mathbf{V}^\top) + \operatorname{Tr}(\mathbf{\Gamma}\mathbf{V}\mathbf{W}^\top\mathbf{K}\mathbf{W}\mathbf{V}^\top), \tag{17}$$

where $\mathbf{K}_{ij} = \boldsymbol{z}_i^\top\boldsymbol{z}_j$. The derivative of $O(\mathbf{W})$ is

$$\frac{\partial O}{\partial \mathbf{W}} = -2\mathbf{K}\mathbf{\Gamma}\mathbf{V} + 2\mathbf{K}\mathbf{W}\mathbf{V}^\top\mathbf{\Gamma}\mathbf{V}, \tag{18}$$

which results in the multiplicative update rule:

$$\mathbf{W} \leftarrow \mathrm{MUR}(\mathbf{W}, \mathbf{K}\mathbf{\Gamma}\mathbf{V}, \mathbf{P}_{\mathbf{W}}, \mathbf{N}_{\mathbf{W}}), \tag{19}$$

$$\mathbf{P}_{\mathbf{W}} = \mathbf{K}^+\mathbf{W}(\mathbf{V}^\top\mathbf{\Gamma}\mathbf{V})^+ + \mathbf{K}^-\mathbf{W}(\mathbf{V}^\top\mathbf{\Gamma}\mathbf{V})^-, \tag{20}$$

$$\mathbf{N}_{\mathbf{W}} = \mathbf{K}^+\mathbf{W}(\mathbf{V}^\top\mathbf{\Gamma}\mathbf{V})^- + \mathbf{K}^-\mathbf{W}(\mathbf{V}^\top\mathbf{\Gamma}\mathbf{V})^+, \tag{21}$$

Optimizing $\mathbf{V}^{(k)}$ with Fixed Other Variables. The subproblem is

$$\begin{aligned}\min_{\mathbf{V}^{(k)}} O = &\|[\mathbf{Z}^{(k)} - \mathbf{Z}^{(k)}\mathbf{W}^{(k)}(\mathbf{V}^{(k)})^\top](\mathbf{\Gamma}^{(k)})^{1/2}\|_{\mathrm{F}}^2 + \alpha\operatorname{Tr}((\mathbf{V}^{(k)})^\top\mathbf{L}^{(k)}\mathbf{V}^{(k)}) \\ &+ \gamma\|(\mathbf{\Gamma}^{(k)})^{1/2}[\mathbf{V}^{(k)} - (\mathbf{V}^*)^{(k)}]\|_{\mathrm{F}}^2\end{aligned} \tag{22}$$

We ignore the superscript and mark $\mathbf{Z}^{(k)}, \mathbf{W}^{(k)}, \mathbf{V}^{(k)}, (\mathbf{V}^*)^{(k)}, \mathbf{\Gamma}^{(k)}, \mathbf{L}^{(k)}$ as $\mathbf{Z}, \mathbf{W}, \mathbf{V}, \mathbf{V}^*, \mathbf{\Gamma}, \mathbf{L}$ for brevity, and the subproblem is restated as

$$\min_{\mathbf{V}} O = \|(\mathbf{Z} - \mathbf{Z}\mathbf{W}\mathbf{V}^\top)\mathbf{\Gamma}^{1/2}\|_{\mathrm{F}}^2 + \alpha\operatorname{Tr}(\mathbf{V}^\top\mathbf{L}\mathbf{V}) + \gamma\|\mathbf{\Gamma}^{1/2}(\mathbf{V} - \mathbf{V}^*)\|_{\mathrm{F}}^2 \tag{23}$$

The objective function can be rewritten as

$$\begin{aligned}O(\mathbf{V}) = &\operatorname{Tr}(\mathbf{\Gamma}\mathbf{K}) - 2\operatorname{Tr}(\mathbf{\Gamma}\mathbf{K}\mathbf{W}\mathbf{V}^\top) + \operatorname{Tr}(\mathbf{\Gamma}\mathbf{V}\mathbf{W}^\top\mathbf{K}\mathbf{W}\mathbf{V}^\top) + \alpha\operatorname{Tr}(\mathbf{V}^\top\mathbf{L}\mathbf{V}) \\ &+ \gamma\operatorname{Tr}(\mathbf{V}^\top\mathbf{\Gamma}\mathbf{V}) - 2\gamma\operatorname{Tr}((\mathbf{V}^*)^\top\mathbf{\Gamma}\mathbf{V}) + \gamma\operatorname{Tr}((\mathbf{V}^*)^\top\mathbf{\Gamma}(\mathbf{V}^*))\end{aligned} \tag{24}$$

where $\mathbf{K}_{ij} = k(\boldsymbol{z}_i, \boldsymbol{z}_j)$. The derivative of $O(\mathbf{V})$ is

$$\frac{\partial O}{\partial \mathbf{V}} = -2\mathbf{\Gamma}\mathbf{K}\mathbf{W} + 2\mathbf{\Gamma}\mathbf{V}\mathbf{W}^\top\mathbf{K}\mathbf{W} + 2\alpha\mathbf{L}\mathbf{V} + 2\gamma\mathbf{\Gamma}\mathbf{V} - 2\gamma\mathbf{\Gamma}\mathbf{V}^*, \tag{25}$$

which results in the gradient decent update rule:

$$\mathbf{V} \leftarrow \mathbf{V} - \eta\frac{\partial O}{\partial \mathbf{V}}. \tag{26}$$

Algorithm 3 Detailed optimization of subspace learning

Input: data matrices $\{\mathbf{Z}_i\}_{i=1}^T$ and coefficient matrices $\{\mathbf{\Gamma}_i\}_{i=1}^T$ generated by concatenating raw specifications of submitted models $\{\boldsymbol{s}_i^{\text{raw}}\}_{i=1}^N$, C_i describing the indices of component feature spaces $\mathbf{Z}_i$ contains, R_k describing the indices of data matrix whose feature space contains $\mathcal{X}_k$.
Hyper-Parameters: the dimension of subspace d, trade-off parameters $\{\alpha, \gamma\}$; learning rate η, max iteration t; number of nearest neighbors p for manifold regularizer.
Output: mapping functions $\{h_k\}_{k=1}^Q, \{g_k\}_{k=1}^Q$ used for connecting component feature spaces $\{\mathcal{X}_k\}_{k=1}^Q$ with the unified subspace $\mathcal{X}_{\text{sub}}$.

```
Use raw specifications {(Z_i, Γ_i)}_{i=1}^T to generate Z^(k), Γ^(k) according to R_k.
initialize W^(k), V^(k), (V*)^(k) with the weighted k-means clustering method.
while max iteration t is not achieved do
    for k = 1 to Q do
        update W^(k) according to Eq. (19).
        update V^(k) according to Eq. (26).
        cooperative normalized W^(k), V^(k) according to Eq. (31).
    end for
    update (V*)^(k) according to Eq. (30).
end while
return mapping functions {h_k(z) = ((B^(k))^⊤ B^(k))^{-1} (B^(k))^⊤ z}_{k=1}^Q, {g_k(v) = B^(k) v}_{k=1}^Q.
```

Optimizing $\{(\mathbf{V}^*)^{(k)}\}_{k=1}^Q$ with Fixed Other Variables. The subproblem is

$$\min_{\{(\mathbf{V}^*)^{(k)}\}} O = \|(\mathbf{\Gamma}^{(k)})^{1/2}[\mathbf{V}^{(k)} - (\mathbf{V}^*)^{(k)}]\|_{\text{F}}^2 \tag{27}$$

Each element in $\{(\mathbf{V}^*)^{(k)}\}_{k=1}^Q$ is the concatenation of $\{\mathbf{V}_i^*\}_{i=1}^T$, we rewrite the subproblem as

$$\min_{\{\mathbf{V}_i^*\}_{i=1}^T} \sum_{i=1}^T \sum_{k \in C_i} \|\mathbf{\Gamma}_i^{1/2}(\mathbf{V}_i^{(k)} - \mathbf{V}_i^*)\|_{\text{F}}^2 \tag{28}$$

The derivative of $O(\mathbf{V}_i^*)$ is

$$\frac{\partial O}{\partial \mathbf{V}_i^*} = \mathbf{\Gamma}_i \sum_{k \in C_i} (\mathbf{V}_i^* - \mathbf{V}_i^{(k)}), \tag{29}$$

which results in the closed-form solution

$$\mathbf{V}_i^* = \frac{1}{|C_i|} \sum_{k \in C_i} \mathbf{V}_i^{(k)} \tag{30}$$

Cooperative Normalization. To ensure uniqueness of the solution and comparability of mapping results across different component feature spaces, we impose a restriction that limits the maximum absolute value of each column in $\mathbf{V}$ to 1. This leads to the cooperative normalization described below:

$$\begin{aligned} \mathbf{W}^{(k)} &= \mathbf{W}^{(k)} \mathbf{\Lambda}^{(k)}, \\ \mathbf{V}^{(k)} &= \mathbf{V}^{(k)} (\mathbf{\Lambda}^{(k)})^{-1}, \end{aligned} \tag{31}$$

where $\mathbf{\Lambda}^{(k)} = \text{diag}(\|\mathbf{V}_{1\cdot}\|, \cdots, \|\mathbf{V}_{k\cdot}\|)$ and the norm used is the maximum norm $\|\boldsymbol{x}\|_\infty = \max_j |x_j|$. This step is followed by updating $\mathbf{W}^{(k)}, \mathbf{V}^{(k)}, (\mathbf{V}^*)^{(k)}$ in each updated iteration.

Overall Algorithm. The overall procedure is summarized in Algorithm 3.

C More Discussion for the General Framework

Weighted extensions. In Equation (3), all component feature spaces are assigned equal weights. However, incorporating prior knowledge for specific tasks would entail assigning different weights to different components. We can easily generalize our objective function by multiplying the weights $c_1, \cdots, c_Q$ with their corresponding component losses. Additionally, by including the norm of the weights as a regularizer, the weights can be automatically adjusted rather than being assigned manually (Wang et al., 2016).

Time complexity analysis. Subspace learning is the procedure that dominates the time complexity of the overall process, with a time complexity of $O(QN^2M^2d_{\text{sub}}t)$. In this context, Q represents the number of feature space blocks, N denotes the number of learnwares available in the market, and $M = O(\log(n))$ indicates the size of the RKME specification, which is significantly smaller than the original dataset size n. d_{sub} represents the dimension of the subspace, and t indicates the maximum iteration of optimization. In the following section, we provide a detailed explanation of the time complexity calculation. The primary computation in subspace learning involves calculating the product of five matrices: the different variants of $\mathbf{KWV}^\top\mathbf{\Gamma V}$ in the optimization step for $\mathbf{W}^{(k)}$ and $\mathbf{\Gamma V W}^\top\mathbf{KW}$ in the optimization step for $\mathbf{V}^{(k)}$. Each of these calculations has a time complexity of $n^2 d_{\text{sub}}$, where n represents the number of samples with data on $\mathcal{X}_k$ and can be bounded by NM. By traversing all M component feature spaces and performing t iterations, the overall time complexity of subspace learning can be expressed as $O(QN^2M^2d_{\text{sub}}t)$.

D More Experiments

This section presents additional results of the heterogeneous learnware market with various configurations. The Mfeat dataset naturally comprises multiple feature spaces, whereas Anuran, Digits, Kddcup99, and Covtype datasets contain only one feature space. In the experiments described in Section 6.2, the heterogeneous learnware scenario is generated by randomly splitting the feature space of each of the four datasets into three parts. However, in this section, we conduct experiments where the feature space of the four datasets is multiplied by three random Gaussian matrices to generate component feature spaces for preprocessing. The processed Anuran, Digits, Kddcup99, and Covtype datasets have component dimensions of [22, 22, 22], [64, 64, 64], [88, 88, 88], and [18, 18, 18], respectively. The number of mixed components for the user's task ranges from 1 to 3, and the other configurations are similar to those described in Section 6.2.

Performance. The performance of various methods is summarized in Table 4. Our method continues to outperform other methods in this scenario. Out of the total of 12 cases, our method achieves the best performance in 11 cases. The results presented in Table 2 and Table 4 demonstrate the strong performance of our method across various situations, thereby validating the effectiveness of our proposed approach.

Task name	#Mix	Random	Ensemble	MMD	MLJ	Projection	Ours
Anuran	1	0.272 ± 0.358	0.464 ± 0.336	0.511 ± 0.407	0.333 ± 0.310	0.639 ± 0.150	$\mathbf{0.723 \pm 0.196}$
	2	0.219 ± 0.199	0.458 ± 0.265	0.372 ± 0.247	0.273 ± 0.194	0.534 ± 0.146	$\mathbf{0.554 \pm 0.187}$
	3	0.184 ± 0.135	0.394 ± 0.209	0.310 ± 0.193	0.227 ± 0.119	0.528 ± 0.089	$\mathbf{0.529 \pm 0.118}$
Digits	1	0.285 ± 0.358	0.560 ± 0.341	0.568 ± 0.351	0.527 ± 0.304	0.536 ± 0.258	$\mathbf{0.760 \pm 0.237}$
	2	0.299 ± 0.205	0.600 ± 0.218	0.482 ± 0.175	0.549 ± 0.193	0.520 ± 0.138	$\mathbf{0.677 \pm 0.126}$
	3	0.283 ± 0.145	0.573 ± 0.167	0.416 ± 0.111	0.526 ± 0.147	0.497 ± 0.110	$\mathbf{0.665 \pm 0.078}$
Kddcup99	1	0.289 ± 0.359	0.379 ± 0.380	0.430 ± 0.358	0.381 ± 0.377	0.386 ± 0.190	$\mathbf{0.451 \pm 0.336}$
	2	0.260 ± 0.215	0.324 ± 0.224	0.324 ± 0.187	0.328 ± 0.222	0.210 ± 0.154	$\mathbf{0.402 \pm 0.194}$
	3	0.240 ± 0.140	0.325 ± 0.148	0.256 ± 0.155	0.329 ± 0.147	0.178 ± 0.111	$\mathbf{0.377 \pm 0.167}$
Covtype	1	0.313 ± 0.366	0.329 ± 0.186	$\mathbf{0.547 \pm 0.382}$	0.411 ± 0.255	0.369 ± 0.253	0.493 ± 0.382
	2	0.315 ± 0.215	0.283 ± 0.114	0.454 ± 0.231	0.414 ± 0.158	0.321 ± 0.112	$\mathbf{0.472 \pm 0.197}$
	3	0.301 ± 0.144	0.283 ± 0.084	0.326 ± 0.149	0.401 ± 0.112	0.279 ± 0.095	$\mathbf{0.419 \pm 0.145}$

Table 4: Accuracy (mean $\pm$ std.) on true labels of the user data. The best method is emphasized in bold.