

# Identifying Helpful Learnwares Without Examining the Whole Market

Yi Xie<sup>a</sup>, Zhi-Hao Tan<sup>a</sup>, Yuan Jiang<sup>a</sup> and Zhi-Hua Zhou<sup>a</sup>

<sup>a</sup>National Key Laboratory for Novel Software Technology,  
Nanjing University, Nanjing 210023, China  
{xiey, tanzh, jiangy, Zhouzh}@lamda.nju.edu.cn

**Abstract.** The *learnware paradigm* aims to construct a market of numerous well-performing machine learning models, which enables users to leverage these models to accomplish specific tasks without having to build models from scratch. Each learnware in the market is a model associated with a specification, representing the model's utility and enabling it to be identified according to future users' requirements. In the learnware paradigm, due to the vast and ever-increasing number of models in the market, a significant challenge is to identify helpful learnwares efficiently for a specific user task without leaking data privacy. However, existing identification methods require examining the whole market, which is computationally unaffordable in a large market. In this paper, we propose a new framework for identifying helpful learnwares without examining the whole market. Specifically, using the Reduced Kernel Mean Embedding (RKME) specification, we derive a novel learnware scoring criterion for assessing the helpfulness of a learnware, based on which we design an anchor-based framework to identify helpful learnwares by examining only a small portion of learnwares in the market. Theoretical analyses are provided for both the criterion and the anchor-based method. Empirical studies on market containing thousands of learnwares from real-world datasets confirm the effectiveness of our proposed approach.

## 1 Introduction

Nowadays machine learning has achieved great success and become a crucial part of various industries. However, training a well-performing machine learning model from scratch can be a complex, time-consuming, and expensive process; numerous data, massive computing resources, and expertise are usually indispensable. As a result, building high-quality models can be a heavy burden for ordinary users who lack the necessary resources and training skills. Furthermore, data privacy concerns, catastrophic forgetting are serious issues when reusing or adapting a trained model among different users. To deal with these challenges simultaneously, the learnware paradigm [26, 27] proposes to build a learnware market that manages thousands or millions of well-performing models, allowing users to leverage these models to accomplish specific tasks without having to build models from scratch.

In this paradigm, each learnware in the market is a well-trained machine learning model associated with a specification that represents its functionality and enables it to be identified according to the requirements of future users [27]. Expert developers can submit their high-quality models into the learnware market spontaneously, and

the market will assign a specification to each model and incorporate it into the market. When a new user needs to tackle her own learning tasks, the market can identify helpful learnwares based on the specifications, and the user can easily deploy these learnwares to solve her tasks instead of starting from scratch. Note that due to data privacy concerns, the learnware market has no access to either the original training data of developers or the raw task data of users.

A key challenge of the learnware paradigm is how to identify helpful models for a specific user task efficiently without leaking user data privacy. To address this, the specification plays a crucial role. Recently the Reduced Kernel Mean Embedding (RKME) specification [27] is proposed, which calculates a reduced set of Kernel Mean Embeddings (KMEs), preserving the ability to characterize distribution with a concise representation while not exposing the original data. Based on the RKME specification, several methods for identifying helpful learnwares have been proposed. The most basic approach is to match the RKMEs of the user and models, selecting the closest one or finding a weighted combination of models [21]. Furthermore, Zhang *et al.* [22] and Tan *et al.* [18, 19] have extended these methods to handle unseen jobs and heterogeneous feature spaces. However, these algorithms require examining all learnwares in the market, which can be computationally unaffordable due to the vast and ever-increasing number of models in the market. Moreover, these algorithms impose strict restrictions on the market, such as the requirement that all learnwares in a specification space share the same ground-truth labeling function. Therefore, it is highly desirable to design a more efficient and flexible algorithm for identifying helpful learnwares.

In this paper, we focus on identifying helpful learnwares without examining the whole market and leaking user data privacy. To achieve this, we must answer two key questions:

1. *Given a user task, how can we determine whether a specific learnware is potentially helpful?*
2. *To solve a user task, how can we identify helpful learnwares by examining only a small portion of the market?* This involves selecting which learnwares should be examined and identifying which of them are considered helpful.

To address the first question, using the RKME specification, we derive a learnware scoring criterion for instance-recurrent model reuse. For the second question, we further propose an anchor-based framework. Specifically, we organize learnwares structurally by performing learnware clustering using a novel distance metric, and choose

the cluster medoids as anchor learnwares. Then we design an efficient identification process by requesting users to test anchor learnwares and return scores, which gives us an in-depth understanding of user tasks. Thus we can identify helpful learnwares efficiently by only examining a small portion of learnwares in the market. Additionally, we provide theoretical analyses for both the criterion and the anchor-based framework. Empirical studies on real-world data confirm the effectiveness of our proposed method.

The remainder of this paper is organized as follows: Section 2 provides a background review. Section 3 and 4 address the two questions mentioned earlier and introduce a detailed procedure for identifying helpful learnwares without examining the whole market. Section 5 presents experimental results, Section 6 discusses related work, and Section 7 concludes the paper.

## 2 Preliminary

This section reviews the RKME specification and the learnware paradigm, which are closely related to our methods.

**RKME specification** The RKME specification [27] is based on the assumption that each learnware model is well-trained on its own training data. Thus, identifying a suitable model can be approached by identifying a model whose original training data distribution is close to the distribution of the user task. Suppose a developer submits a model trained on her private dataset  $\{(\mathbf{x}_n, y_n)\}_{n=1}^N$ ,  $\mathbf{x}_n \in \mathcal{X}$ ,  $y_n \in \mathcal{Y}$ , then the market will generate the reduced set representation  $(\beta, \mathbf{z})$  by minimizing the distance between the Kernel Mean Embeddings (KMEs) [17] of the reduced set and the raw data measured by the RKHS norm, as

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

where  $k(\cdot, \cdot)$  is the kernel function corresponding to the RKHS  $\mathcal{H}$ ,  $\beta_m \in \mathbb{R}$  is the coefficient,  $\mathbf{z}_m \in \mathcal{X}$  is the newly constructed reduced sample and different from the raw samples, and the size  $M \ll N$ . The RKME  $\tilde{\mu}_P = \sum_{m=1}^M \beta_m k(\mathbf{z}_m, \cdot)$  is a point in the RKHS  $\mathcal{H}$ . It's chosen as the specification because it offers a concise representation of original data distribution  $P$  while protecting data privacy. See the Appendix for a more detailed introduction.

**The learnware paradigm** The learnware paradigm consists of the submitting stage and the deploying stage.

In the submitting stage, developers can spontaneously submit their trained models to the market. Then the market assigns a specification to each model and accommodates it in the market. Formally, suppose there are  $C$  developers in total. The  $i$ -th developer has a trained model  $\hat{f}_i$  to solve her task  $T_i = (P_i, f_i)$ ,  $i \in [C] = \{1, 2, \dots, C\}$ , where  $P_i$  is a distribution on the same input space  $\mathcal{X}$  and  $f_i: \mathcal{X} \rightarrow \mathcal{Y}$  is an optimal labeling function. The developer trained her model on a private dataset  $D_i = \{(\mathbf{x}_{in}, y_{in})\}_{n=1}^{N_i}$  sampled i.i.d from  $P_i$ . We focus on the homogeneous case that all the tasks share the same input and label space. For simplicity, we assume  $N_i = N$ . The developers uploads their models in turn, and for each model  $\hat{f}_i$ , the market computes the RKME  $\tilde{\mu}_i = \sum_{m=1}^M \beta_{im} k(\mathbf{z}_{im}, \cdot)$  as the specification by solving the RKME optimization problem in (1). In practice, the RKME  $\tilde{\mu}_i$  is stored as a reduced set  $\{(\beta_{im}, \mathbf{z}_{im})\}_{m=1}^M$ , with no leakage of raw data in  $D_i$ . Then the market is constructed with learnwares  $\{(\hat{f}_i, \tilde{\mu}_i)\}_{i \in [C]}$ .

In the deploying stage, a user wants to solve her task  $T_t = (P_t, f_t)$ . Specifically, the user attempts to learn a good model  $\hat{f}_t$  which minimize  $\mathcal{L}(P_t, f_t, \hat{f}_t) \triangleq \mathbb{E}_{\mathbf{x} \sim P_t} [L(\hat{f}_t(\mathbf{x}), f_t(\mathbf{x}))]$  with the help of the learnware market. The user can submit her requirements to the market, and the market will identify some helpful learnwares through specifications and return them to the user. Then the user could deploy these learnwares to solve her tasks instead of training a model from scratch.

## 3 Whether is a Learnware Helpful?

In this section, we answer the first question: how to examine whether a specific learnware is potentially helpful for a given user task? This question serves as the basis for identifying helpful learnwares.

### 3.1 Analyses of helpful learnware identification

To begin, we conduct an analysis of helpful learnware identification. Our focus is on identifying learnwares that can help with a specific user task. A learnware that performs well on the user task is undoubtedly helpful. To identify such learnwares, a straightforward and effective approach is to use the user's labeled dataset as a validation set, run the models on it, and choose those models with small validation errors. Then the best model can be directly deployed, or these models can be ensembled through voting, which may perform better than a single model. We refer to this method as `task-recurrent` reuse methods because it typically requires the task of at least one learnware to be similar with that of the user.

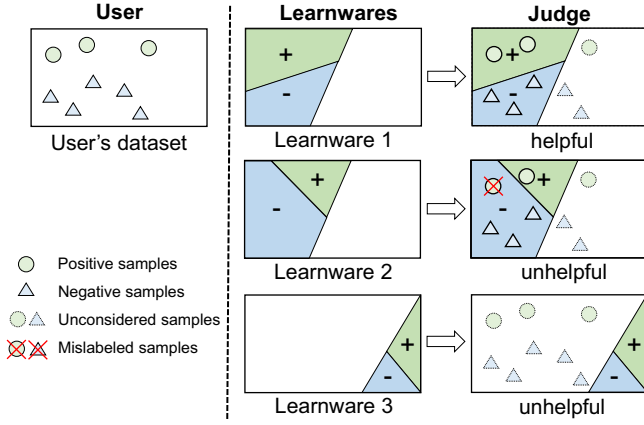
However, there are cases where no single learnware can tackle the user task as a whole, but multiple learnwares can each tackle a part of the user task separately. In such cases, a selector can be trained to assign each instance of the user to one or several chosen models, solving the user task in a divide-and-conquer way [25]. We refer to this approach as `instance-recurrent` reuse methods. For instance-recurrent reuse methods, simply using the user's labeled dataset as a validation set and choosing models with small errors may not work. A learnware that performs poorly on some parts of the user task but excels at others may be overlooked, yet it may be a critical component in instance-recurrent model reuse.

In this paper, we propose a learnware identification framework specifically designed for instance-recurrent reuse. There are several reasons for this. First, task-recurrent methods require some models to perform well on the user's whole task, which may be rare in real-world scenarios. Second, instance-recurrent methods can reuse models for tasks beyond their original purposes, which is one of the core objectives of the learnware paradigm. Third, learnware identification for task-recurrent reuse methods can be viewed as a special case of instance-recurrent methods, since a learnware that can tackle the user task as a whole can certainly solve part of the user task. Therefore, we focus on instance-recurrent model reuse, which is a more challenging, more common, and more meaningful case.

### 3.2 The proposed scoring criterion

In this section, we introduce the problem setup and the assumption, and then propose a scoring criterion to examine whether a learnware is helpful for a specific user task.

**Problem setup** A user wants to solve a classification or regression task  $T_t = (P_t, f_t)$  with the help of learnware market. Suppose the user has a limited labeled dataset  $D_t = \{(\mathbf{x}_{tn}, y_{tn})\}_{n=1}^{N_t}$ , where



**Figure 1.** A simplified example for our scoring criterion.

samples  $\{\mathbf{x}_{tn}\}_{n=1}^{N_t}$  are sampled i.i.d. from distribution  $P_t$  and labels  $\{y_{tn}\}_{n=1}^{N_t}$  satisfy a labeling function  $f_t$ , i.e.  $\forall n \in [N_t], f_t(\mathbf{x}_{tn}) = y_{tn}$ . The dataset is too small to train a good model but can help identify helpful learnwares from the market. Now the user attempts to examine whether a learnware  $(\hat{f}_i, \tilde{\mu}_i)$  is helpful for her task  $T_t$ . It's worth mentioning that we do not place the global labeling function restriction on the market as the previous work. Besides, the models in the market can be any type, e.g., neural networks, forests, etc; we assume these models are all black boxes and the details are unknown.

**Instance-recurrent assumption** Now we give the formal definition of instance-recurrent assumption. Intuitively, we assume that there exist multiple learnwares in the market that each tackles one part of the user task separately. Formally,

- The user's distribution should be a mixture of multiple learnwares' distributions,

$$P_t = \sum_{i \in [C]} w_i P_i.$$

Especially, learnwares with  $w_i > w$  are called key learnwares, where  $w > 0$  is a small constant.

- Each key learnware performs well in the corresponding mixture component, i.e., the difference between the user's ground-truth labeling function and the model's prediction in this mixture component should be small,

$$\forall i \in [C], w_i > w \implies \mathcal{L}(P_i, \hat{f}_i, f_t) < \epsilon.$$

where  $L$  is the loss function,  $\epsilon$  is a small constant and  $\mathcal{L}(P_i, \hat{f}_i, f_t) \triangleq \mathbb{E}_{\mathbf{x} \sim P_i} L(\hat{f}_i(\mathbf{x}), f_t(\mathbf{x}))$ .

These key learnwares are exactly what we aim to find in helpful learnware identification.

**Our approach** Our scoring criterion consists of two parts, exactly corresponding to the instance-recurrent assumption:

- How far is the learnware task's distribution  $P_i$  from a potential mixture component of the current user task's distribution  $P_t$ ?
- How well does the learnware model perform on the current user task in this mixture component?

We use a simplified example to show our main idea in Figure 1. There are several learnwares to be judged according to the user's dataset. Two different colors and shapes correspond to two classes, and each learnware's distribution is a uniform distribution over the colored area. First, we select part of the user's samples to simulate learnware's distribution. The samples with dotted outlines are out of learnware's distribution and thus unconsidered. Then we check the prediction results of the left samples. The first learnware is judged as helpful, the second as unhelpful due to a mislabeled sample, and the third as unhelpful because its distribution is far from a potential mixture component of the user's distribution.

The details of our scoring criterion are shown below, which is a formalized version of what we have done in the example above. First, we seek the closest potential mixture component of the user's distribution  $P_t$  to the learnware's distribution  $P_i$ . Here user's distribution  $P_t$  is portrayed by her dataset  $D_t$ , while learnware's distribution  $P_i$  is represented by RKME  $\tilde{\mu}_i$ . So we reweight the samples in  $D_t$  to simulate  $\tilde{\mu}_i$  in RKHS,

$$\begin{aligned} \min_{\eta_i} & \left\| \sum_{n=1}^{N_t} \eta_{in} k(\mathbf{x}_{tn}, \cdot) - \tilde{\mu}_i \right\|_{\mathcal{H}}^2, \\ \text{s.t.} & \sum_{n=1}^{N_t} \eta_{in} = 1, \\ & \eta_{in} \geq 0, \forall n \in [N_t], \end{aligned} \quad (2)$$

where  $\{\eta_{in}\}_{n=1}^{N_t}$  is the weights of samples in  $D_t$ . This problem can be solved by quadratic programming. Because the size of RKME's reduced set and the user's labeled dataset are both small, the quadratic programming is relatively fast.

Now we get the weighted dataset  $\{\eta_{in}, (\mathbf{x}_{tn}, y_{tn})\}_{n=1}^{N_t}$ , which is the most suitable potential mixture component for the learnware. For simplicity, we represent the distribution of this dataset in RKHS as

$$\hat{\mu}_{t \rightarrow i} \triangleq \sum_{n=1}^{N_t} \eta_{in} k(\mathbf{x}_{tn}, \cdot). \quad (3)$$

Then we calculate the two terms in our scoring criterion: first, the distance between the learnware distribution  $P_i$  and this weighted dataset in RKHS; second, the loss of the learnware model  $\hat{f}_i$  on this weighted dataset. Our scoring criterion adds two terms together:

$$h_i \triangleq U \cdot \|\tilde{\mu}_i - \hat{\mu}_{t \rightarrow i}\|_{\mathcal{H}} + \sum_{n=1}^{N_t} \eta_{in} \cdot L(\hat{f}_i(\mathbf{x}_{tn}), y_{tn}), \quad (4)$$

where  $U$  is a constant that bounds the loss  $L_i \in \mathcal{H}$ . The smaller the score, the more helpful the learnware is. Therefore, we can set a threshold  $\theta$ , and consider learnware with  $h_i < \theta$  helpful while  $h_i \geq \theta$  as unhelpful.

Intuitively, our scoring criterion matches the two points in the instance-recurrent assumption. If the learnware distribution is far from a potential mixture component of the user's distribution, the first term will be large because the reweighting cannot simulate  $P_i$ . If the learnware model doesn't match the user's weighted dataset, the second term will be large. In these cases, the learnware is unhelpful.

### 3.3 The theoretical guarantee

**What are we estimating?** In truth, our scoring criterion is the upper bound of the empirical error

$$\mathcal{L}(\tilde{\mu}_i, \hat{f}_i, f_t) \triangleq \sum_{m=1}^M \beta_{im} L(\hat{f}_i(\mathbf{z}_{im}), f_t(\mathbf{z}_{im})),$$

which is the RKME version of  $\mathcal{L}(P_i, \hat{f}_i, f_t)$  mentioned in instance-recurrent assumption. It estimates the difference between the learnware model's prediction with user's ground-truth labels under the distribution of the learnware task, thus an important measure for helpfulness. Moreover, when the learnware distribution is exactly a potential mixture component, the estimation is tight; otherwise, the first term of the criterion will be large, and then the learnware will be considered unhelpful reasonably.

**Proposition 1** *The scoring criterion  $h_i$  is an upper bound of the loss  $\mathcal{L}(\hat{\mu}_i, \hat{f}_i, f_t)$ .*

**Errors of our scoring criterion** Further, we prove that the errors of our scoring criterion are small.

**Theorem 1** *Let  $k(\mathbf{x}, \mathbf{x}) \leq 1$  for all  $\mathbf{x} \in \mathcal{X}$  and the loss  $L_i \in \mathcal{H}$  be bounded by a constant  $U$ . There exists a good threshold  $\theta$ , with a high probability that:*

- All key learnwares are considered helpful ( $h_i \leq \theta$ ).
- And all learnwares considered helpful can solve the user task over its own distribution  $P_i$ ,

$$\mathcal{L}(P_i, \hat{f}_i, f_t) \leq \epsilon + O\left(\sqrt{\frac{1}{N}} + \sqrt{\frac{1}{M}} + \sqrt{\frac{1}{N_{tw}}}\right)$$

This theorem tells us that the two types of errors for our scoring criterion are both controlled. It seldom treats key learnwares as unhelpful, and all learnwares considered helpful are truly helpful to some extent. Besides, this theorem depicts the relation between the errors and the number of samples, and the most important factor is the size of the user's dataset. Furthermore, we prove that the simple way of traversing and scoring all learnwares in the market can obtain a small generalization error after reuse. See the Appendix for details and proofs.

#### 4 Anchor-based Method for Helpful Learnware Identification

In the previous section, we have presented a scoring criterion for determining the helpfulness of a specific learnware. In this section, we address the most critical problem in this article: how to identify helpful learnwares by examining only a small portion of the learnwares in the market.

The key to solving this problem is to construct anchor learnwares. The market can provide several anchor learnwares, request the user to test them and return some scores, and then identify potentially helpful learnwares based on these scores. The criterion  $h_i$  introduced in Section 3 is a suitable scoring function for this task. It is worth noting that the concept of anchor learnwares was first proposed by Zhou *et al.* [27], and in this paper, we further realize this idea.

**The selection of anchors** The first question is which learnwares should be selected as anchors. To answer this question, we need to explore the relationships between the learnwares, because a good anchor should represent a group of learnwares. In other words, if the anchor is helpful for a specific user task, then all the learnwares it represents are likely to be helpful for that task as well. It is important to note that the selection of anchor learnwares is completed in the submitting stage, independent of the user tasks, as the market has no knowledge of specific user tasks at that time. This requires the

selection of anchor learnwares to be general enough to work for any possible user task.

Our approach to addressing this issue involves utilizing learnware clustering. By implementing clustering, we can identify medoids for each cluster, which serve as effective representatives for a cluster of learnwares. These medoids can be chosen as anchors to facilitate further decision-making.

In order to implement clustering, we should first define a dissimilarity metric for learnwares. Specifically, we define the learnware divergence from learnware  $i$  to learnware  $j$ :

$$d_{ij} \triangleq d((\hat{f}_i, \tilde{\mu}_i), (\hat{f}_j, \tilde{\mu}_j)) \\ = U \|\tilde{\mu}_i - \tilde{\mu}_j\|_{\mathcal{H}} + \min \left\{ \mathcal{L}(\hat{\mu}_i, \hat{f}_i, \hat{f}_j), \mathcal{L}(\hat{\mu}_j, \hat{f}_i, \hat{f}_j) \right\}, \quad (5)$$

where  $\mathcal{L}(\hat{\mu}_i, \hat{f}_i, \hat{f}_j) \triangleq \sum_{m=1}^M \beta_{im} L(\hat{f}_i(z_{im}), \hat{f}_j(z_{im}))$  is the difference between model  $\hat{f}_i$  and  $\hat{f}_j$  over the empirical distribution of  $\hat{\mu}_i$ ;  $\mathcal{L}(\hat{\mu}_j, \hat{f}_i, \hat{f}_j)$  is defined similarly. Intuitively,  $d_{ij}$  is the RKME version of  $U \cdot d(P_i, P_j) + \min\{\mathcal{L}(P_i, \hat{f}_i, \hat{f}_j), \mathcal{L}(P_j, \hat{f}_i, \hat{f}_j)\}$ . Note that it satisfies nonnegativity, identity, and symmetry, but doesn't satisfy the triangle inequality, thus not a strictly-defined distance.

Then we cluster all learnwares using Partitioning Around Medoids (PAM) [14, 15], which is a kind of k-medoids clustering algorithms. Mathematically, we seek for a set of  $k$  medoids  $\mathcal{M} = \{m_1, \dots, m_k\} \subset [C]$  to minimize the overall dissimilarity of points to their closest medoids:

$$\sum_{i=1}^C \min_{m \in \mathcal{M}} d((\hat{f}_m, \tilde{\mu}_m), (\hat{f}_i, \tilde{\mu}_i)) \quad (6)$$

Note that the dissimilarity function in k-medoids doesn't have to be a distance measure [10, 20], so our learnware divergence can be applied here. Besides, there exist many fast PAM implementations; for example, Kuzborskij *et al.* [10] accelerate PAM to  $O(k)$  order.

The cluster structure is maintained in the submitting stage. When the market is built, initial learnwares are clustered using PAM algorithm. As a new learnware comes to the market, it is assigned to the closest existing cluster or added as a new cluster if no close anchor exists. Then the clusters are updated iteratively using the PAM algorithm to minimize the overall dissimilarity, as defined in Eq. 6. For efficiency, cluster updates can be performed periodically instead of in real-time. This approach is scalable and efficient, as the market does not need to recompute all learnware divergences and clustering from scratch. The pseudocode is shown in Appendix.

**The utilization of anchors** Once all the learnwares are represented by several anchors, examining all the anchors gives us a rough knowledge of the helpfulness of all learnwares.

In the deploying stage, the user first downloads all the anchors. The user scores them using the scoring criterion in section 3 and uploads the scores to the market. Then the market finds the helpful anchors according to these scores and marks the corresponding clusters as helpful clusters. The user downloads the candidate learnwares in all helpful clusters and filters them more carefully using the same scoring criterion. The learnwares left are considered helpful. The pseudocode is also shown in Appendix.

After getting helpful learnwares, the user combines them to get a final predictor. The model reuse method here can be any instance-recurrent model reuse method [21, 22], which usually assigns different samples of the users to different learnware models. An example of the new deploying stage is shown in Figure 2.

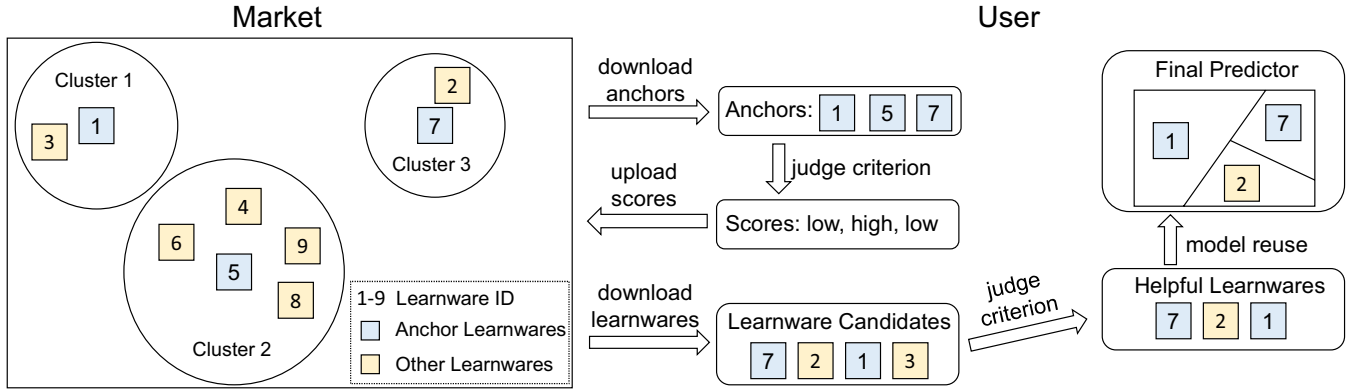


Figure 2. An example of the deploying stage.

**Analysis** First we prove that the chosen anchors are representative.

**Proposition 2** For each learnware  $(\hat{f}_i, \hat{\mu}_i)$ , assume it belongs to the cluster with anchor  $(\hat{f}_m, \hat{\mu}_m)$ , where  $m = \arg \min_{m' \in \mathcal{M}} d((\hat{f}_{m'}, \hat{\mu}_{m'}), (\hat{f}_i, \hat{\mu}_i))$ . Then we have

$$\mathcal{L}(\hat{\mu}_m, \hat{f}_m, f_t) - d_{im} \leq \mathcal{L}(\hat{\mu}_i, \hat{f}_i, f_t) \leq \mathcal{L}(\hat{\mu}_m, \hat{f}_m, f_t) + d_{im}.$$

In section 3, we mentioned that  $\mathcal{L}(\hat{\mu}_i, \hat{f}_i, f_t)$  is an important measure for helpfulness. So this proposition tells us the helpfulness of each learnware is closed to that of its anchor, and the difference is measured by the distance to the anchor  $d_{im}$ . If the learnwares are well-clustered, the summation of  $d_{im}$  is minimized, then the estimation is tight. Besides, we've proved that for each anchor,  $h_m$  is the upper bound of  $\mathcal{L}(\hat{\mu}_m, \hat{f}_m, f_t)$ , so for each learnware,  $h_m + d_{im}$  is an upper bound of  $\mathcal{L}(\hat{\mu}_i, \hat{f}_i, f_t)$ .

Then we illustrate why anchors can improve the efficiency of helpful learnware identification. The core is that by examining the anchors, we exclude many learnwares without access. Each learnware whose corresponding anchor is considered unhelpful will be directly omitted and no further examination is needed. Consider a simplified example, if all the  $C$  learnwares in the market are divided into  $\sqrt{C}$  clusters that each has  $\sqrt{C}$  learnwares. After checking these  $\sqrt{C}$  anchors, we find  $k$  of them helpful; then we carefully check these  $k$  clusters to make the final selection. Here the anchor learnwares have been scored so only  $\sqrt{C} - 1$  learnwares should be checked in each helpful cluster. In a large and open market, usually only a tiny portion of learnwares are helpful to a specific user task, so  $k$  is small w.r.t.  $C$ . In the two phases, we check  $\sqrt{C} + k(\sqrt{C} - 1) \approx O(\sqrt{C})$  learnwares in total, much less than the  $O(C)$  traversal. Besides, using only helpful anchors could also result in a not bad reuse performance, so if the user is very concerned about efficiency, the second stage can be omitted and only  $\sqrt{C}$  learnwares will be examined in total. What's more, If  $C$  is extremely large, multi-level clustering is available for use. Learnwares could be clustered in a tree structure, i.e. in each cluster, if it's still large, a more fine-grained clustering will be conducted. When identifying helpful learnwares, once an anchor at any level is judged as unhelpful, the examination in this subtree breaks up. An example is shown in Appendix.

**Identifying one learnware** Additionally, the anchor-based method we have described can also be adapted to select one most helpful learnware instead of a group of learnwares with a few minor

modifications in the deploying phase. Specifically, we need to change the scoring criterion in the second step to the error rate on the user's labeled data and select the best anchor/learnware instead of filtering by a threshold  $\theta$  in the third and fourth steps. The clustering structure remains the same, allowing our framework to work for both selecting a single model and a group of models. The theoretical analysis presented earlier is still applicable.

## 5 Experiments

In this section, we thoroughly assess our technique against other methods on a toy example and several real-world datasets.

### 5.1 A toy example

We created a toy example to demonstrate the functionality of our method, using the traditional handwritten digit recognition dataset, 'Digits' [3]. To begin, we divided the dataset into 10 parts, each containing the figures of two continuous numbers (e.g., digits 0 and 1), and trained 10 different models on each part (including Logistic Regression and LightGBM models with various hyper-parameters). Then we got a market of 100 learnwares. In the submission stage, the market used our algorithm to group the learnwares into 10 clusters. Our algorithm perfectly placed all models trained for the same task into the same cluster; for example, the first cluster contained all the 10 models trained for recognizing digits '0' and '1'.

**Table 1.** The toy example: results of clustering and the evaluation of anchors. For example, the first column indicates that the first cluster contained all the models for recognizing digits 0 and 1. The score of its anchor (learnware 1-1) was 11.37, the smallest, showing that it was helpful.

Cluster	1	2	3	4	5	6	7	8	9	10
Task	0&1	1&2	2&3	3&4	4&5	5&6	6&7	7&8	8&9	9&0
Anchor	1-1	2-7	3-7	4-1	5-1	6-5	7-2	8-4	9-7	10-6
Score	<b>11.37</b>	<b>12.67</b>	27.03	34.5	40.5	42.09	41.39	36.5	36.51	25.3

Next, we considered a user who needs to identify the numbers 0, 1, and 2. Since the market had no knowledge of the user task, it provided all the anchors to the user. The user then scored these anchors using our criterion and found that the anchor 0 and 1 had significantly better scores than the others (as shown in table 1). The

anchor 1 corresponded to the cluster that recognized the numbers 0 and 1, while the anchor 2 corresponded to recognizing 1 and 2. The market then provided the models in these two clusters, and the user selected them more carefully using our criterion. Finally, the user selected model 1-1 (the first model for task 1) and 2-3. Using the instance-recurrent reuse algorithm, the final predictor achieved an accuracy of 93.06%. It is worth mentioning that none of the models in the market could deal with the numbers 0, 1, and 2 at the same time, yet our method achieved a remarkable result in this task by reusing two of them. Besides, our method only examined 28 models among the 100 ones in the market.

## 5.2 Real-world tasks

Then we demonstrate the effectiveness of our method on large markets containing thousands of learnwares from 4 real-world datasets.

**Datasets** To evaluate our approach, we selected four real-world datasets: M5 [9], PFS [8], PPG-DaLiA [16], and Covtype [2]. M5 and PFS are time-series datasets comprising daily sales data, provided by 1C Company in Russia and Walmart Inc. in the USA, respectively. PPG-DaLiA is a UCI dataset for heart rate estimation during daily life activities. Covtype is a UCI multi-class dataset that contains information about the forest cover type based on cartographic variables. Covtype is a classification task while the others are all regression tasks.

**Table 2.** Information of data sets, including the introduction of raw datasets and corresponding learnware markets.

Dataset	Task	#Instance	Split Criterion	#Models	#Users
M5	Regression	46M	Department	1050	10
PFS	Regression	9M	Shop	795	17
PPG-DaLiA	Regression	517K	Activity	675	22
Covtype	Classification	581K	Soil	450	10

We selected these datasets because of their significant size and natural divisibility. M5 and PFS datasets can be divided into subsets based on Shop/Department, with each subset representing the daily sales data of one Shop/Department. Similarly, PPG-DaLiA and Covtype can be partitioned based on Activity/Soil type.

**Settings** Initially, we constructed learnware markets for these datasets by training 15 models on each sub-dataset, including linear models, LightGBM, and neural networks with different hyperparameters. This diversity of partitions and models ensured that the market contained a broad range of tasks and models. Next, we extracted a portion of data as user tasks, comprising 50-120 labeled data and an unlabeled dataset to predict, which did not appear in the training data of any learnware, and none of the user tasks share the same distribution as any of the learnware tasks. These users approached the market to identify helpful learnwares for their tasks. See more information in table 2 about the datasets and markets.

**Methods** We compared the following methods in our experiments:

- `From-scratch` is a baseline that the user train a new model from scratch without any assistance from the learnware market.
- `Random` randomly selects a learnware from the market. Its expected performance is the average performance of all learnwares.

- `RKME-task` [21] chooses one best learnware using RKME.
- `RKME-instance` [21] utilize all learnwares using weighted estimation by RKME, without an explicit learnware search.
- `Validate` treats the user’s labeled dataset as a validation set and selects several learnwares with small losses.
- `Ours-anchor` is our method which uses anchors to identify helpful learnwares and avoids examining the whole market.
- `Ours-traversal` is a simplified version for comparison, which uses no anchors thus examining the whole market.

**Table 3.** Comparison of different methods.

Methods	#Learnwares examined	#Learnwares selected	Need RKME	Need labels
<code>From-scratch</code>	None	None	No	Yes
<code>Random</code>	All	One	No	No
<code>RKME-task</code>	All	One	Yes	No
<code>RKME-instance</code>	All	“All”	Yes	No
<code>Validate</code>	All	Several	No	Yes
<code>Ours-traversal</code>	All	Several	Yes	Yes
<code>Ours-anchor</code>	<b>Several</b>	Several	Yes	Yes

A comparison of these methods is shown in Figure 3. We unified the model reuse methods to see which identification method above is better: for those methods that select one learnware, the user directly deployed this model; for those selecting multiple learnwares, the same instance-recurrent reuse method is used. It’s worth mentioning that `Ours-anchor` is the only one that does not require examining the whole market.

**Measures** We used the following measures to compare the performance of these methods. The first measure is the loss, which is MSE or RMSE for regression and error rate for classification. However, the loss of model search and reuse is strongly influenced by the quality of the models in the market. Therefore, we used a more informative indicator: the performance improvement percentage, which is calculated as the relative improvement over `Random`. Besides, considering the objective of this paper, the running time and the percentage of learnwares examined is a very important indicator.

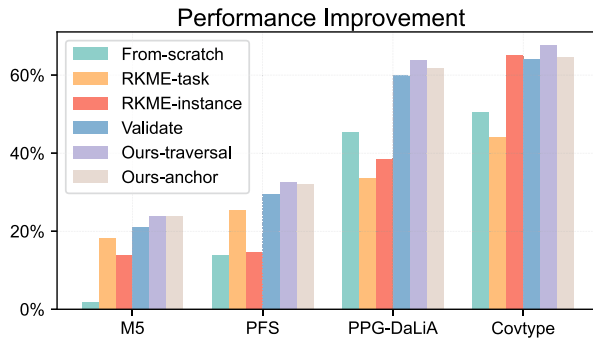
**Results** The results in Table 4 and Figure 3 show the mean performance of these methods. The results indicate that, as the only method that does not require examining the whole market, `Ours-anchor` examined only a small portion of the learnwares (**11.8%**, **14.9%**, **21.42%**, **19.91%**, respectively) in the deployment stage, such has a relatively small running time; and its performance is comparable to its traversal version `Ours-traversal` and is superior to other methods. The methods `RKME-task` that choose only one learnware was also fast but obtained relatively low performance, showing the importance of choosing a group of models. `RKME-instance` select too many learnwares, resulting in the slowest speed and a negative effect in reuse. These findings demonstrate that our approach has successfully achieved the goal of identifying helpful learnwares without examining the whole market. See the Appendix for more details and all codes are provided.

## 6 Related Work

The learnware paradigm [26, 27] aims to systematically reuse small models to do things that may even be beyond their original pur-

**Table 4.** Average performances. For different methods, we calculate the average of three performance measures over all users. Here Imp. is short for the performance improvement percentage, while Time means the time (seconds) of identifying helpful learnwares. Because Random and From-scratch do not examine helpful learnwares, the time is omitted. For RKME-task, we show the time of weight calculation as contrast.

	M5			PFS			PPG-DaLiA			Covtype		
	RMSE	Imp.	Time	RMSE	Imp.	Time	MSE	Imp.	Time	Error	Imp.	Time
From-scratch	4.142	1.85%	-	3.081	13.79%	-	19.83	45.43%	-	0.334	50.60%	-
Random	4.085	0.00%	-	3.297	0.00%	-	36.62	0.00%	-	0.683	0.00%	-
RKME-task	3.389	18.27%	7.77	2.798	25.42%	2.35	24.53	33.64%	<b>0.73</b>	0.380	44.05%	<b>0.30</b>
RKME-instance	3.586	13.72%	137.57	2.931	14.56%	277.10	22.40	38.51%	201.42	0.240	65.00%	21.33
Validate	3.266	21.12%	4.09	2.671	29.46%	3.34	14.70	59.87%	10.07	0.245	64.01%	2.43
Ours-traversal	3.154	23.80%	10.61	<b>2.609</b>	<b>32.48%</b>	8.27	<b>13.29</b>	<b>63.71%</b>	11.35	<b>0.222</b>	<b>67.67%</b>	4.94
Ours-anchor	<b>3.148</b>	<b>23.80%</b>	<b>1.13</b>	2.616	32.07%	<b>1.37</b>	14.03	61.71%	2.57	0.244	64.45%	1.12



**Figure 3.** Average performance improvement percentage of different methods. Random is the baseline thus is always 0%, omitted in this figure.

poses, so that users need not build their machine learning models from scratch. The learnware paradigm assigns a specification to each model, which represents its functionality and enables it to be identified according to future users' requirement. Based on the RKME specification, Wu *et al.* [21] propose an algorithm that matches the distributions of the user and models by matching their RKMEs, and selects the closest one or finds a weighted combination of model. Building upon this work, Zhang *et al.* [22] consider the existence of unseen jobs that any learnware cannot cover, and their method identifies samples from the unseen parts and assigns the rest to proper learnwares. Tan *et al.* [18, 19] extend this framework to handle heterogeneous feature spaces. However, our method differs from these works in that our approach could work in a very large market because we does not require examining the whole market. Also, we relax the restriction of ground-truth labeling function.

The learnware paradigm shows a fundamentally different way compared to model pools that merely collect and store models. For example, the learnware paradigm aims to systematically reuse models for specific tasks beyond their original purposes by associating each model with a specification that describes its utility. Additionally, the learnware paradigm prioritizes data privacy for both developers and users by identifying helpful models without exposing their data. Lastly, the specifications allow for the structural organization of models, enabling the identification of helpful models without examining the whole market.

There are several other fields aiming to reuse other's efforts to solve a new task. Domain adaptation [1] and transfer learning [13] aim to help improve the learning of the target domain using the knowledge in the source domain. However, they usually directly

access the raw data of the source domain to train a new model for the target domain. Besides, domain adaptation with multiple sources [11, 7, 4] and model reuse [23, 10, 6, 5] use the trained models of the source domains to derive a predictor for the target domain. But they require that all the source models are helpful for the target task, so they don't consider the identification of helpful models. In the learnware paradigm, there may be only a tiny portion of learnwares helpful for the current user task, so identifying helpful learnwares is indispensable.

Federated learning [12, 24] focuses on the case that the training data for some specific task is distributed among multiple parties and sharing is forbidden due to privacy. The challenge is to define safe communication protocols for data owners to jointly train a model without leaking data privacy. However, in the learnware paradigm, the market collects and accommodates thousands of well-trained models, and utilizes these models to solve various new tasks. The difficulty is identifying helpful models among the numerous models in the market and assembling them in an appropriate way for different user's tasks. Besides, though federated learning and the learnware paradigm both concern data privacy, their aims are fundamentally different. The learnware paradigm mainly considers the relationship between developers, market and users, while federated learning focus on task originators and collaborators.

## 7 Conclusion

This paper focus on identifying helpful learnwares for a specific task without examining the whole market and leaking user data privacy. Specifically, we propose a novel learnware scoring criterion based on the RKME specification to assess the potential helpfulness of a learnware. Using this criterion, we design an anchor-based framework to achieve efficient learnware identification by examining only a small portion of learnwares in the market. We provide theoretical guarantees for both the criterion and the anchor-based method. Our experimental results demonstrate the effectiveness and efficiency of our proposed approach. It is worth emphasizing the vital importance of organizing and storing learnwares structurally using anchors, particularly in a large market. This can not only accelerate the process of identifying helpful learnwares but also facilitate the extraction of valuable information about the models, which could serve as a basis for future research directions. Moreover, extending our method to efficiently identify helpful learnwares in heterogeneous feature spaces could open up interesting avenues for exploration.

## Acknowledgements

This research was supported by NSFC (62250069). Zhi-Hao Tan is also supported by Postgraduate Research & Practice Innovation Program of Jiangsu Province. The authors would like to thank Jian-Dong Liu and Peng Tan for helpful discussions. We are also grateful for the anonymous reviewers for their valuable comments.

## References

- [1] Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira, 'Analysis of representations for domain adaptation', in *Advances in Neural Information Processing Systems*, pp. 137–144, (2006).
- [2] 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*, **24**(3), 131–151, (1999).
- [3] MGJBG Candela, DDJGP Grother, S Janet, and C Wilson, 'Nist form-based handprint recognition system', Technical report, National Institute of Standards and Technology, (1994).
- [4] Corinna Cortes, Mehryar Mohri, Ananda Theertha Suresh, and Ningshan Zhang, 'A discriminative technique for multiple-source adaptation', in *International Conference on Machine Learning*, volume 139, pp. 2132–2143, (2021).
- [5] 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*, pp. 37389–37400, (2022).
- [6] Yao-Xiang Ding and Zhi-Hua Zhou, 'Boosting-based reliable model reuse', in *Asian Conference on Machine Learning*, volume 129, pp. 145–160, (2020).
- [7] Judy Hoffman, Mehryar Mohri, and Ningshan Zhang, 'Algorithms and theory for multiple-source adaptation', in *Advances in Neural Information Processing Systems*, pp. 8256–8266, (2018).
- [8] Kaggle. Predict future sales. <https://www.kaggle.com/competitions/competitive-data-science-predict-future-sales/overview>, 2018.
- [9] Kaggle. M5 forecasting. <https://www.kaggle.com/competitions/m5-forecasting-accuracy/overview>, 2020.
- [10] Ilja Kuzborskij and Francesco Orabona, 'Fast rates by transferring from auxiliary hypotheses', *Machine Learning*, **106**(2), 171–195, (2017).
- [11] Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh, 'Domain adaptation with multiple sources', in *Advances in Neural Information Processing Systems*, pp. 1041–1048, (2008).
- [12] Mehryar Mohri, Gary Sivek, and Ananda Theertha Suresh, 'Agnostic federated learning', in *International Conference on Machine Learning*, volume 97, pp. 4615–4625, (2019).
- [13] Sinno Jialin Pan and Qiang Yang, 'A survey on transfer learning', *IEEE Transactions on Knowledge and Data Engineering*, **22**(10), 1345–1359, (2010).
- [14] LKPJ Rduseeun and P Kaufman, 'Clustering by means of medoids', in *Statistical Data Analysis Based on the L1-Norm Conference*, volume 31, (1987).
- [15] LKPJ Rduseeun and P Kaufman, 'Partitioning around medoids (program pam)', in *Finding groups in data: an introduction to cluster analysis*, chapter 2, 68–125, John Wiley & Sons, Ltd, (1990).
- [16] Attila Reiss, Ina Indlekofer, Philip Schmidt, and Kristof Van Laerhoven, 'Deep ppg: Large-scale heart rate estimation with convolutional neural networks', *MDPI Sensors*, **19**(14), (2019).
- [17] Alex Smola, Arthur Gretton, Le Song, and Bernhard Schölkopf, 'A hilbert space embedding for distributions', in *International Conference on Algorithmic Learning Theory*, pp. 13–31, (2007).
- [18] Peng Tan, Zhi-Hao Tan, Yuan Jiang, and Zhi-Hua Zhou, 'Towards enabling learnware to handle heterogeneous feature spaces', *Machine Learning*, (2022). doi: <https://doi.org/10.1007/s10994-022-06245-1>.
- [19] Peng Tan, Zhi-Hao Tan, Yuan Jiang, and Zhi-Hua Zhou, 'Handling learnwares developed from heterogeneous feature spaces without auxiliary data', in *International Joint Conference on Artificial Intelligence*, (2023).
- [20] Mo Tiwari, Martin J Zhang, James Mayclin, Sebastian Thrun, Chris Piech, and Ilan Shomorony, 'Banditpam: Almost linear time k-medoids clustering via multi-armed bandits', in *Advances in Neural Information Processing Systems*, pp. 10211–10222, (2020).
- [21] Xi-Zhu Wu, Wen-Kai 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).
- [22] Yu-Jie Zhang, Yu-Hu Yan, Peng Zhao, and Zhi-Hua Zhou, 'Towards enabling learnware to handle unseen jobs', in *AAAI Conference on Artificial Intelligence*, volume 35(12), pp. 10964–10972, (2021).
- [23] Peng Zhao, Le-Wen Cai, and Zhi-Hua Zhou, 'Handling concept drift via model reuse', *Machine Learning*, **109**(3), 533–568, (2020).
- [24] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra, 'Federated learning with non-iid data', *arXiv:1806.00582*, (2018).
- [25] Zhi-Hua Zhou, *Ensemble methods: foundations and algorithms*, 184, CRC press, 2012.
- [26] Zhi-Hua Zhou, 'Learnware: on the future of machine learning', *Frontiers of Computer Science*, **10**(4), 589–590, (2016).
- [27] Zhi-Hua Zhou and Zhi-Hao Tan, 'Learnware: Small models do big', *Science China Information Sciences*, (2023). doi:<https://doi.org/10.1007/s11432-023-3823-6>.