Identifying Useful Learnwares via Reduced Neural Conditional Mean Embedding

Anonymous submission

Abstract

The learnware paradigm aims to establish a market of learnwares, each of which is a well-trained model combined with a specification that describes its functionality without leaking data privacy. The market enables users to efficiently reuse relevant models based on specifications on their own tasks instead of training models from scratch. Recent works have attempted to generate specifications using Reduced Kernel Mean Embedding (RKME), which maps input distributions into Reproducing Kernel Hilbert Space (RKHS) while ignoring the output space, causing models trained on similar input spaces to yield similar specifications, even when their functionalities differ. Many labeled-RKME improvements attempt to address this by indirectly modeling the input-output conditional distributions, but they remain limited to classification tasks and lack clear theoretical explanations. In this work, we propose Reduced Neural Conditional Mean Embedding (RNCME), a novel specification generation method that directly models input-output conditional distributions via Conditional Mean Embedding (CME). Our RNCME method has a clear theoretical understanding based on CME and is applicable to both regression and classification tasks. Empirical experiments demonstrate the effectiveness of our RNCME method for learnware recommendation.

1 Introduction

The learnware paradigm (Zhou 2016; Zhou and Tan 2024) was proposed to establish a shared market of learnwares for machine learning model sharing and reuse. In this paradigm, each learnware consists of a well-trained model accompanied by a specification that describes its functionality. This shared learnware market enables users to efficiently identify and reuse relevant learnwares based on specifications, eliminating the need for expensive model training from scratch, which requires abundant training data, substantial computational resources, and specialized expertise. It is worth noting that the learnware market has no access to either the data of developers or users: developers only need to submit models and their specifications, while users are required to provide only task specifications and will receive suitable learnware(s) from the market based on specifications, addressing privacy concerns in traditional model reuse approaches.

The specification plays a crucial role in the learnware paradigm. Wu et al. (2023) proposed to generate the specification based on the Reduced Kernel Mean Embedding (RKME), which maps input distributions into a Reproducing Kernel Hilbert Space (RKHS). Specifically, for a model f well-trained on a dataset (X, Y), RKME first estimates the empirical Kernel Mean Embedding (KME) for the marginal distribution P_X on dataset X, and then generates a reduced set as the specification, which approximates the empirical KME. Moreover, Lei, Tan, and Zhou (2024) provided a theoretical analysis of RKME specification about its preservation ability for training data, ensuring data privacy preservation in RKME specifications.

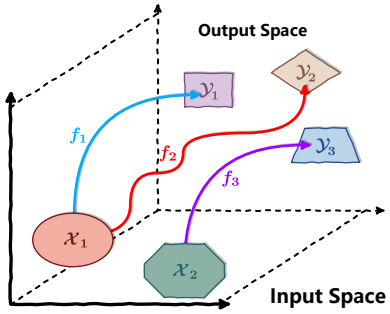


Figure 1: Three models on different input and output spaces. Curve shape represents corresponding model functionality. Models sharing the same input space may not have similar functionalities, and vice versa.

The basic RKME method mentioned above only consider the input space while ignoring the output space, causing models trained on similar input spaces to be assigned similar specifications, even when their functionalities differ, and vice versa. For example, consider three models in Figure 1. Both f_1 and f_2 take news titles as input: f_1 classifies them into sports or technology categories, while f_2 analyzes their sentiment as positive or negative. And f_3 takes paper abstracts as input, classifying them into AI or history categories. Although f_1 and f_2 share input space, f_3 would actually be more suitable for news title classification task than f_2, which is represented visually by the similar shapes of f_1 and f_3 in Figure 1. But the basic RKME method would assign similar specifications to f_1 and f_2 because they share input space, while giving f_3 a different specification. Several recent improvements to it have incorporated output space information: Guo et al. (2023) distill models into Linear Prox-

ies (LP) using label information and recommend models via bipartite graph matching; Tan et al. (2024a) evolve heterogeneous feature spaces into a unified subspace with less entangled class representations and more coherent embeddings by using model outputs; and LANE (Chen, Mao, and Zhang 2025) incorporates label supervision and neural embedding spaces for specification generation. However, these methods (1) lack clear theoretical explanations; (2) only indirectly model the conditional distributions, and (3) are inherently constrained to classification tasks.

In this work, we propose a novel specification generation method called Reduced Neural Conditional Mean Embedding (RNCME) that directly models input-output conditional distributions using Conditional Mean Embedding (CME) (Song et al. 2009). RNCME has a clear theoretical understanding based on CME, and handles both regression and classification problems. However, standard CME faces well-known computational challenges, requiring $O(n^3)$ computational complexity for empirical estimation from a dataset of size n. The root cause lies in the fact that we cannot explicitly compute the feature vector of the infinite-dimensional RKHS. To address this, we design a data-adaptive neural feature map for the input space that induces a finite-dimensional neural feature space, achieving $O(nd^2 + d^3)$ complexity for empirical CME estimation, where $d \ll n$. While finite-dimensional RKHS typically lacks universal approximation capabilities and characteristic properties, we overcome this through a specialized training stage that learns data-adaptive feature maps capturing essential input-output mapping patterns. Moreover, our neural feature map avoids the manual kernel selection problem, including both the choice of kernel functions and the tuning of regularization parameter. To preserve data privacy, we propose a variant of the Frank-Wolfe algorithm (Wolfe 1976) to generate a reduced set that approximates the empirical CME, and provide the theoretical upper bound of the approximation error. Empirical experiments on both regression and classification tasks demonstrate the effectiveness of our method in learnware recommendation. We summarize the main contributions of our work as follows:

- We propose RNCME for specification generation, which directly models input-output conditional distributions with a clear theoretical understanding based on CME, extending existing labeled-RKME improvements to both regression and classification;
- We design a data-adaptive neural feature map, which reduces the high computational complexity of empirical CME estimation while avoiding the manual kernel selection problem;
- We propose a variant of the Frank-Wolfe algorithm to generate a privacy-preserving reduced set that approximates the empirical CME.

The remainder of this paper is structured as follows. Section 2 provides essential background and preliminaries. Section 3 introduces the proposed RNCME method. The experimental evaluation and results are detailed in Section 4, while Section 5 discusses connections to prior research. Finally, concluding remarks are presented in Section 6.

2 Preliminaries

In this section, we briefly introduce some preliminaries, beginning with the definition of notations.

2.1 Notations

Let X be a random variable taking values in a measurable space $\mathcal{X}$, with realizations x. Consider a measurable positive definite kernel $k_{\mathcal{X}} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ that induces a Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}$, endowed with inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ and norm $\|\cdot\|_{\mathcal{H}}$. The canonical feature map $\phi : \mathcal{X} \to \mathcal{H}$ satisfies $\phi(x) = k_{\mathcal{X}}(x, \cdot)$ for all $x \in \mathcal{X}$. And the reproducing property of $k_{\mathcal{X}}$ yields $h(x) = \langle h, \phi(x) \rangle_{\mathcal{H}}$ holds for all $h \in \mathcal{H}$ and $x \in \mathcal{X}$.

Analogously, for random variable Y in measurable space $\mathcal{Y}$ with kernel $k_{\mathcal{Y}}$, we denote its induced RKHS as $\mathcal{G}$ with feature map $\psi : \mathcal{Y} \to \mathcal{G}$, inner product $\langle \cdot, \cdot \rangle_{\mathcal{G}}$, and norm $\|\cdot\|_{\mathcal{G}}$, satisfying the corresponding reproducing property.

2.2 The Learnware Paradigm

The learnware paradigm can be divided into two stages: submitting and deploying stages.

The Submitting Stage In this stage, all developers can submit their well-performing models to the learnware market. Suppose there are N developers, and the c-th developer is accessible to a local dataset $D_c = (X_c, Y_c) = \{(x_{ci}, y_{ci})\}_{i=1}^{n_c}$ sampled i.i.d. from the joint distribution P_{XY}^c over $\mathcal{X}_c \times \mathcal{Y}_c$, where $\mathcal{X}_c \subseteq \mathcal{X}$ and $\mathcal{Y}_c \subseteq \mathcal{Y}$. Based on D_c, the developer can locally train a well-performing model $f_c : \mathcal{X}_c \to \mathcal{Y}_c$ and generate its corresponding specification $\mathcal{S}_c$. Then, the developer can provide f_c along with its specification $\mathcal{S}_c$ to the learnware market. With N such submissions, the learnware market becomes $\mathcal{M} = \{(f_c, \mathcal{S}_c)\}_{c=1}^{N}$.

The Deploying Stage When a user wants to solve a new task t, with a small labeled dataset $D_l = (X_l, Y_l) = \{(x_{li}, y_{li})\}_{i=1}^{n_l}$ and a large unlabeled dataset $D_u = X_u = \{x_{ui}\}_{i=1}^{n_u}$, he or she can generate a specification $\mathcal{S}_t$ describing the requirement of the task based on D_l and D_u, and then submit $\mathcal{S}_t$ to the market. The market will identify relevant learnwares based on the specifications, and recommend the most suitable learnware(s) for the user to solve task t.

2.3 Kernel Mean Embedding (KME)

In the absence of ambiguity, we use $\mathbb{E}_X, \mathbb{E}_{XY}$ instead of $\mathbb{E}_{X \sim P_X}, \mathbb{E}_{(X,Y) \sim P_{XY}}$, and so on. For the marginal distribution P_X, the KME (Smola et al. 2007) is defined as:

$$\mu_X = \mathbb{E}_X[\phi(X)] \in \mathcal{H}.$$

If kernel $k_{\mathcal{X}}$ is characteristic, then μ_X is unique for P_X. For a dataset $\{x_i\}_{i=1}^n$ sampled i.i.d. from P_X, the μ_X can be estimated empirically as:

$$\widehat{\mu}_X = \frac{1}{n} \sum_{i=1}^{n} \phi(x_i).$$

The KME of the joint distribution P_{XY}, also known as the cross-covariance operator (Baker 1973), is defined as:

$$C_{XY} = \mathbb{E}_{XY}[\psi(Y) \otimes \phi(X)] \in \mathcal{G} \otimes \mathcal{H},$$

where $\otimes$ is the tensor product operator, and space $\mathcal{G} \otimes \mathcal{H}$ is isomorphic to the Hilbert-Schmidt space $\mathrm{HS}(\mathcal{H}, \mathcal{G})$.

2.4 Reduced Kernel Mean Embedding (RKME)

KME is a potential specification for the learnware market, but it depends on the raw data violates the data privacy requirement. To address this issue, Wu et al. (2023) proposed the RKME by using a reduced set $(\boldsymbol{\beta}, \boldsymbol{Z}) = \{(\beta_j, z_j)\}_{j=1}^m$ to approximate the empirical KME of the raw data, where $\beta_j \in \mathbb{R}$ and $z_j \in \mathcal{X}$, which can be generated by the following optimization problem:

$$\min_{\boldsymbol{\beta}, \boldsymbol{Z}} \left\| \frac{1}{n} \sum_{i=1}^{n} \phi(x_i) - \sum_{j=1}^{m} \beta_j \phi(z_j) \right\|_{\mathcal{H}}^2 .$$

The reduced set $(\boldsymbol{\beta}, \boldsymbol{Z})$ will be submitted to the market, and the corresponding RKME $\widetilde{\mu} = \sum_{j=1}^m \beta_j \phi(z_j)$ typically serves as the specification. RKME only considers the input space while ignoring the output space. A possible solution is to use Condition Mean Embedding (CME).

2.5 Conditional Mean Embedding (CME)

The KME of the conditional distributions $P_{Y|X}$, usually called Conditional Mean Embeddings (CME) (Song et al. 2009), is defined as:

$$\mathcal{U}_{Y|X} = \mathbb{E}_{Y|X}[\psi(Y)] = C_{YX} C_{XX}^{-1},$$

which is an operator from $\mathcal{H}$ to $\mathcal{G}$. For a fixed $x \in \mathcal{X}$, the CME of $P_{Y|X=x}$ is defined as:

$$\mathcal{U}_{Y|X=x} = \mathbb{E}_{Y|X=x}[\psi(Y)] = \mathcal{U}_{Y|X} \phi(x) \in \mathcal{G}.$$

For a dataset $\{(x_i, y_i)\}_{i=1}^n$ sampled i.i.d. from P_{XY}, let $\Phi = (\phi(x_1) \quad \cdots \quad \phi(x_n))$ and $\Psi = (\psi(y_1) \quad \cdots \quad \psi(y_n))$ denote the implicit feature matrices; $K_X = \Phi^{\mathrm{T}} \Phi \in \mathbb{R}^{n \times n}$ denotes the explicitly computable Gram matrix with entries $(K_X)_{ij} = k_{\mathcal{X}}(x_i, x_j)$. Given the regularization parameter $\lambda > 0$ and the $n \times n$ identity matrix I, the empirical CME $\widehat{\mathcal{U}}_{Y|X}$ (simply denoted as $\widehat{\mathcal{U}}$) can be estimated as:

$$\widehat{\mathcal{U}} = \Psi \left(K_X + n\lambda I \right)^{-1} \Phi^{\mathrm{T}}. \tag{1}$$

The majority of kernel methods use infinite-dimensional kernel functions (e.g., RBF kernel), especially on the input space, due to their favorable theoretical properties. Specifically, most infinite-dimensional kernels enjoy the universal approximation property, enabling CME to capture arbitrarily complex dependencies between variables. And characteristic kernels ensure injective embeddings that preserve all distributional information in RKHS representations, enabling different conditional distributions under distinct inputs to be discriminated.

3 Reduced Neural CME

Our goal is to design a specification, which fully captures input-output mapping patterns of the model or task. We propose to directly model input-output conditional distributions via CME, which has a clear theoretical understanding, and handles both regression and classification tasks.

3.1 Address High Computational Complexity

Estimating $\widehat{\mathcal{U}}$ requires $O(n^3)$ computational complexity, which is infeasible for large n. The root cause lies in the fact that $\widehat{\mathcal{U}}$ was theoretically meant to take the form:

$$\widehat{\mathcal{U}} = \widehat{C}_{YX} \left(\widehat{C}_{XX} + \lambda \mathcal{I} \right)^{-1}, \tag{2}$$

where $\mathcal{I}$ is the identity operator; $\widehat{C}_{YX} = \frac{1}{n} \sum_{i=1}^n \psi(y_i) \otimes \phi(x_i)$ and $\widehat{C}_{XX}$ admits a similar expansion.

However, for the infinite-dimensional RKHS $\mathcal{H}$, each $\phi(x) \in \mathcal{H}$ can be viewed as an infinite-dimensional vector, which prohibits us from explicitly computing $\widehat{C}_{YX}$ and $\widehat{C}_{XX}$. This intrinsic limitation compels us to leverage the reproducing property of $\mathcal{H}$, through which we transform Equation (2) into the computationally feasible formulation Equation (1). The resulting expression, while implementable, necessitates inversion of an $n \times n$ Gram matrix, thereby introducing the computational complexity of $O(n^3)$.

To overcome the computational complexity challenge, we propose a pragmatic compromise by replacing the infinite-dimensional feature map ϕ with a finite-dimensional version $\phi_d : \mathcal{X} \to \mathbb{R}^d$, inducing a finite-dimensional RKHS $\mathcal{H}_d$, where $d \ll n$. This dimensional reduction enables direct computation of (2) as $\Psi \Phi^{\mathrm{T}} \left(\Phi \Phi^{\mathrm{T}} + n\lambda I \right)^{-1}$ while significantly reducing the computational complexity from $O(n^3)$ to $O(nd^2 + d^3)$.

However, this replacement causes $\mathcal{H}_d$ to lose the favorable theoretical properties mentioned in Section 2.5. Specifically: (1) the universal approximation capability (limiting its ability to model complex dependencies between input and output), and (2) the characteristic property (leading to entangled conditional distributions given distinct inputs).

3.2 A Data-Adaptive Neural Feature Map

To alleviate the shortcomings mentioned above, we design a data-adaptive neural feature map $\phi_\theta : \mathcal{X} \to \mathbb{R}^d$ for the input space, which is inspired by Xu et al. (2021), originally developed for instrumental variable regression.

Specifically, it has been proven that estimating CME is equivalent to solving the following function-valued ridge regression problem (Grünewälder et al. 2012):

$$\arg\min_{U: \mathcal{H} \to \mathcal{G}} \frac{1}{n} \sum_{i=1}^{n} \| \psi(y_i) - U \phi(x_i) \|_{\mathcal{G}}^2 + \lambda \| U \|_{\mathrm{HS}}^2 . \tag{3}$$

Substituting ϕ with ϕ_θ in Equation (3) yields a joint optimization over θ and U. With U admitting a closed-form solution, we can optimize θ directly via minimizing the following loss function:

$$\mathcal{L}(\theta) = -\mathrm{tr} \left(K_Y \Phi_\theta^{\mathrm{T}} \left(\Phi_\theta \Phi_\theta^{\mathrm{T}} + n\lambda I \right)^{-1} \Phi_\theta \right),$$

where $K_Y = \Psi^{\mathrm{T}} \Psi \in \mathbb{R}^{n \times n}$ is the output Gram matrix.

With a well-trained ϕ_θ, we can explicitly compute the feature matrix of X as $\Phi_\theta \in \mathbb{R}^{d \times n}$, and the empirical CME can be estimated as:

$$\widehat{\mathcal{U}} = \Psi \Phi_\theta^{\mathrm{T}} \left(\Phi_\theta \Phi_\theta^{\mathrm{T}} + n\lambda I \right)^{-1} = \Psi Q, \tag{4}$$

where $Q \in \mathbb{R}^{n \times d}$ can be viewed as the weight matrix of Ψ.

3.3 The Effectiveness of the Neural Feature Map

Through training, ϕ_θ maps inputs into a data-adaptive finite-dimensional RKHS $\mathcal{H}_\theta$ that is maximally correlated with the output RKHS $\mathcal{G}$. As a result, complex dependencies between inputs and outputs can be effectively captured, mitigating the lack of the universal approximation property. Simultaneously, ϕ_θ mines discriminative input-output mapping patterns from data, allowing conditional distributions given different inputs to be distinguished, thereby alleviating the absence of the characteristic property.

Additionally, when estimating $\widehat{\mathcal{U}}$ on finite samples, compared to the original $\mathcal{H}$, on the one hand $\mathcal{H}_\theta$ enjoys a lower risk of overfitting due to its bounded representational capacity, while $\mathcal{H}$ tends to fit noise; on the other hand, the parameterized nature of ϕ_θ enables flexible feature learning that adapts to data-specific patterns, reducing underfitting potential. This dual regularization mechanism gives $\mathcal{H}_\theta$ better generalization performance than $\mathcal{H}$.

Moreover, such a data-adaptive feature map ϕ_θ eliminates the need for choosing specific kernel functions, after all, a certain kernel is not always applicable to all tasks, even when they possess universal approximation capabilities. Experimentally, infinite-dimensional kernels show remarkable sensitivity to the regularization parameter λ, and different tasks can exhibit varying sensitivities, making hyperparameter tuning extremely challenging, while the neural kernels maintain stable performance across parameter variations.

3.4 Find the Reduced Set

To preserve data privacy, we generate a reduced set approximating the original $\widehat{\mathcal{U}}$. This requires first establishing an assumption and deriving a corollary.

Assumption 1. *We assume that feature maps ϕ_θ and ψ are bounded, i.e., there exist constants $B_X, B_Y > 0$, such that:*

$$\sup_{x\in\mathcal{X}} \|\phi_\theta(x)\| \leqslant B_X, \quad \sup_{y\in\mathcal{Y}} \|\psi(y)\|_{\mathcal{G}} \leqslant B_Y.$$

Corollary 1. *Under assumption 1, for all $i \in [n]$, weight vector $q_i \in Q$ in Equation* (4) *satisfies $\|q_i\| \leqslant \frac{B_X}{n\lambda}$.*

Corollary 1 indicates that we require a reduced set $V = \{v_j\}_{j=1}^m \subseteq \mathcal{Y}$ with bounded-weight matrix $R \in \mathbb{R}^{m\times d}$, where $m \ll n$, to approximate $\widehat{\mathcal{U}} = \Psi Q$. This leads to the following constrained vector-weighted low-rank kernel approximation problem:

$$(V, R) = \mathop{\arg\min}_{V\subseteq\mathcal{Y}, R\in\mathbb{R}^{m\times d}} \|\Psi Q - \Psi_V R\|_{\mathrm{HS}}^2,$$
$$\text{subject to } \|r_j\| \leqslant \frac{B_X}{m\lambda}, \forall j \in [m].$$

To solve this problem, we propose a variant of Frank-Wolfe algorithm (Wolfe 1976). First, we construct a compact convex set $\mathcal{D}$ defined as:

$$\mathcal{D} = \operatorname{conv}\left\{ r\psi(v) : v \in \mathcal{Y}, r \in \mathbb{R}^{1\times d}, \|r\| \leqslant \frac{B_X}{\lambda} \right\}. \quad (5)$$

Notably, $\|r\| \leqslant B_X/\lambda$ holds rather than $B_X/(m\lambda)$, as will be justified subsequently. We then define a convex differentiable function $f : \mathcal{D} \to \mathbb{R}$ as:

$$f(U) = \frac{1}{2}\left\|\widehat{\mathcal{U}} - U\right\|_{\mathrm{HS}}^2. \quad (6)$$

Obviously, we aim to solve $\min_{U\in\mathcal{D}} f(U)$. Following the Frank-Wolfe algorithm, we initialize U_0 as the zero vector in $\mathcal{D}$ and perform iterative updates through:

$$(r_{t+1}, v_{t+1}) \in \mathop{\arg\max}_{r\psi(v)\in\mathcal{D}} \langle W_t, r\psi(v)\rangle_{\mathrm{HS}}, \quad (7)$$

$$U_{t+1} = \frac{t}{t+1}U_t + \frac{r_{t+1}\psi(v_{t+1})}{t+1}, \quad (8)$$

where $W_t = \widehat{\mathcal{U}} - U_t$ can be viewed as the deviation. In order to solve Equation (7), we need to find $s = r\psi(v)$ that has the maximum inner product with W_t over the convex set $\mathcal{D}$:

$$\langle W_t, r\psi(v)\rangle_{\mathrm{HS}} = \langle r, W_t\psi(v)\rangle \leqslant \|r\| \cdot \|W_t\psi(v)\|.$$

The maximum is achieved when r and $W_t\psi(v)$ are collinear, since $\|r\| \leqslant B_X/\lambda$, we can select:

$$v_{t+1} = \mathop{\arg\max}_{v\in\mathcal{Y}} \|W_t\psi(v)\|, \; r_{t+1} = \frac{B_X}{\lambda}\cdot\frac{W_t\psi(v_{t+1})}{\|W_t\psi(v_{t+1})\|}.$$

If $\mathcal{Y}$ is a discrete space, we can solve v_{t+1} by traversing the whole space directly. If $\mathcal{Y}$ is continuous, we can optimize it by using gradient ascent, as long as $k_{\mathcal{Y}}$ is smooth. It is equivalent to solving $v_{t+1} = \arg\max_{v\in\mathcal{Y}} \|W_t\psi(v)\|^2$, where $\|W_t\psi(v)\|^2$ equals:

$$\sum_{l=1}^{d}\left(\sum_{i=1}^{n} q_{il}k_{\mathcal{Y}}(y_i, v) - \frac{1}{t}\sum_{j=1}^{t} r_{jl}k_{\mathcal{Y}}(v_j, v)\right)^2.$$

Theorem 1. *The above procedure is in a standard Frank-Wolfe form, which can minimize the objective function $f(U)$ over the convex set $\mathcal{D}$.*

According to Equation (8), after m iterations, we actually obtain V and R such that $\Psi_V R$ well approximates $m\cdot\widehat{\mathcal{U}}$. This justifies our construction of $\mathcal{D}$ with the constraint $\|r\| \leqslant B_X/\lambda$ rather than $B_X/(m\lambda)$. Finally, the RNCME is expressed as $\widetilde{\mathcal{U}} = \Psi_V R/m = U_m$. This procedure can be viewed as a vector-weighted version of kernel herding algorithm (Chen, Welling, and Smola 2010).

Theorem 2. *By applying the above procedure, we have the convergence rate:*

$$\left\|\mathcal{U} - \widetilde{\mathcal{U}}\right\|_{\mathrm{HS}} \leqslant O\left(\frac{1}{\sqrt{n\lambda}} + \sqrt{\lambda} + \frac{1}{\lambda}\sqrt{\frac{\ln m}{m}}\right).$$

Theorem 2 provides an upper bound of the discrepancy between the real CME $\mathcal{U}$ and the RNCME specification $\widetilde{\mathcal{U}}$. The details, including the specifics about the Frank-Wolfe algorithm, and the proofs of corollary and theorems, can be found in Appendix.

3.5 Learnware with RNCME

The RNCME $\widetilde{\mathcal{U}}$ effectively captures input-output mapping patterns while preserving data privacy, making it a promising specification. We demonstrate how RNCME enhances the learnware framework introduced in Section 2.2.

The Submitting Stage Following Guo et al. (2023), we assume a public feature extractor $\varphi : \mathcal{X} \to \mathcal{X}_{\text{uni}}$ (such as a pre-trained representation learning model) that maps input features into a unified feature space $\mathcal{X}_{\text{uni}}$. Consider a developer c with model $f_c : \mathcal{X}_c \to \mathcal{Y}_c$ trained on local dataset $D_c = (X_c, Y_c)$, he or she should first generate unified features $X_c' = \varphi(X_c)$. Using (X_c', Y_c), the developer locally trains a data-adaptive neural feature map $\phi_{\theta_c} : \mathcal{X}_{\text{uni}} \to \mathbb{R}^d$ and generates the RNCME specification $\widetilde{\mathcal{U}}_c$. We consider RKME specification $\widetilde{\mu}_c$ as a supplement, forming the final specification as $\mathcal{S}_c = (\widetilde{\mathcal{U}}_c, \widetilde{\mu}_c)$. For market submission, the developer should provide f_c and $\mathcal{S}_c$.

The Deploying Stage A user with task t possesses small labeled data $D_l = (X_l, Y_l)$ and large unlabeled data $D_u = X_u$. Similarly, after feature unification, the user trains the neural feature map ϕ_{θ_t}, then generates and submits the specification $\mathcal{S}_t = \left(\widetilde{\mathcal{U}}_t, \widetilde{\mu}_t\right)$. When the market receives $\mathcal{S}_t$, it compares the similarity between $\mathcal{S}_t$ and all $\mathcal{S}_c$, returning the most relevant learnware f_{c^*} based on:

$$c^* = \underset{c\in[N]}{\arg\min}\ \log\left\|\widetilde{\mathcal{U}}_c - \widetilde{\mathcal{U}}_t\right\|_{\text{HS}}^2 + \gamma \log\|\widetilde{\mu}_c - \widetilde{\mu}_t\|_{\mathcal{H}}^2, \quad (9)$$

where $\gamma = 0.1$ in our experiments. Here $\widetilde{\mathcal{U}}_c$ and $\widetilde{\mathcal{U}}_t$ remain comparable even when ϕ_{θ_c} and ϕ_{θ_t} differ, because $\mathcal{H}_{\theta_c} \otimes \mathcal{G}$ and $\mathcal{H}_{\theta_t} \otimes \mathcal{G}$ share the same measure — each ϕ_θ maps elements from $\mathcal{X}_{\text{uni}}$ to the same Euclidean space.

4 Experiments

We conduct experiments to evaluate the effectiveness of our proposed RNCME method. We organize our experiments into three parts: synthetic tasks, real-world regression tasks, and real-world classification tasks.

4.1 Synthetic Experiments

To provide intuitive visualization of the distinction between KME and CME methods, we construct a controlled synthetic environment where we can manipulate both the input distributions and the input-output conditional mappings.

Marginal Distributions We define five synthetic input domains, each characterized by a distinct marginal distribution P_{X_i}. Specifically, we consider a Gaussian distribution $X_1 \sim \mathcal{N}(0, 1)$; uniform distributions $X_2 \sim \text{Unif}(0, 1)$ and $X_3 \sim \text{Unif}(-1, 0.5)$; an exponential distributions $X_4 \sim \mathcal{E}(1)$; and a beta distribution $X_5 \sim \text{Beta}(2, 5)$. For each distribution P_{X_i}, we draw n i.i.d. samples $X_i = \{x_i\}_{i=1}^n$ with additive distribution-adaptive noises, where each $x_i \in \mathbb{R}^d$. In our experiments, we set $n = 2048$ and $d = 128$.

Conditional Mappings We define four complex conditional mappings $\{\pi_j\}_{j=1}^4$ from input space to scalar outputs with additive mapping-adaptive noises, representing different model behaviors, for each $x \in \mathbb{R}^d$:

- $\pi_1(x) = \text{sum}\left(x\text{e}^{-x^2/2}\right)$;
- $\pi_2(x) = \sin(\text{sum}(x)) + \cos(\text{mean}(x))$;
- $\pi_3(x) = \text{sum}(\pi'(x))$, $\pi'(x) = \begin{cases}\log(1+\text{e}^x) & x > 0\\ |x| & x \leqslant 0\end{cases}$;
- $\pi_4(x) = \text{mean}^2(\sin(\pi x/2)) + \text{mean}^2(\cos(\pi x/2))$.

Task Construction For each (X_i, π_j), we construct a synthetic task $T_{ij} = (X_i, Y_{ij})$, where $Y_{ij} = \{\pi_j(x) : x \in X_i\}$. This yields a total of 20 synthetic datasets, each corresponding to a model f_{ij} trained on a specific domain and mapping.

Kernel Selection We use the RBF kernel for the output space while considering both neural kernel and the RBF kernel for the input space. Specifically, the neural kernel is defined as $k_\theta(x, x') = \phi_\theta(x)^{\text{T}}\phi_\theta(x')$, where ϕ_θ is a 3-layer MLP with a 64-dimensional hidden layer and a 16-dimensional output layer. And we set the width parameter of the RBF kernel to $1/d$.

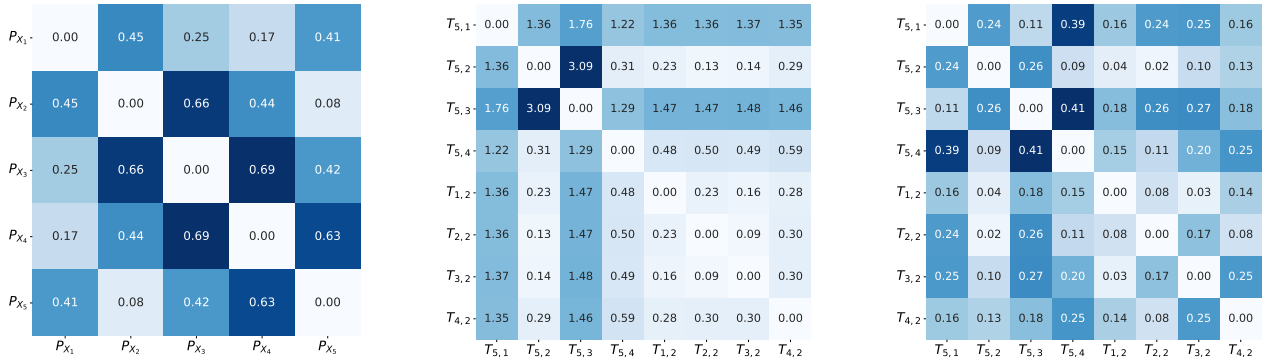
KME Discrepancy For each pair of marginal distributions, we consider their KME discrepancy, i.e., Maximum Mean Discrepancy (MMD) (Gretton et al. 2012). And we can draw a heatmap of pairwise MMD across all marginal distributions, as shown in Figure 2a.

CME Discrepancy For all tasks $\{T_{ij}\}$, we simultaneously show three scenarios: (1) tasks employing different π demonstrate larger CME discrepancies despite sharing identical P_X, and (2) tasks employing the same π exhibit smaller CME discrepancies even when sampled from different P_X, and (3) neural kernel is more stable than RBF kernel. To avoid overcrowding the 20×20 heatmap, we focus on representative cases using beta distribution P_{X_5} and mapping π_2, and compare all tasks sampled from P_{X_5} against all tasks employing π_2. We use a shared untrained neural kernel with task-agnostic hyperparameters and task-specific RBF kernels with carefully tuned hyperparameters. The comparative results are presented in Figures 2b and 2c.

Observations From Figure 2, we can see that: (1) different pairs of marginal distributions have different KME discrepancy; (2) tasks have different mappings demonstrate larger CME discrepancies even when sampled from the same P_X; (3) tasks have the same mapping π_2 exhibit smaller CME discrepancies even when sampled from different P_X; and (4) neural kernel is more stable than RBF kernel, even when neural kernel is untrained.

4.2 Real-World Regression Experiments

Synthetic experiments show that CME with neural feature map can effectively capture the input-output mapping patterns on regression tasks. To further validate the effectiveness of our RNCME method on real-world learnware recommendation scenarios that cannot leak data privacy, we extend our experiments on a real-world regression dataset.



(a) (b) (c)

Figure 2: Comparison of distribution discrepancies measured by different methods: (a) KME discrepancies across all five marginal distributions; (b) CME discrepancies between the beta distribution P_{X_5} with all mappings versus other distributions with mapping π_2 using a shared task-agnostic neural kernel; (c) CME discrepancies under identical conditions as (b) but with task-specific RBF kernels.

Dataset Preparation We design regression tasks on real-world California Housing Prices dataset (Nugent 2017), which contains $20,640$ samples with 10 features.

Experimental Details We consider three features as potential labels and the remaining seven features as potential inputs. From these, we randomly select five input features and one label feature to generate 63 distinct tasks. These tasks are divided into 42 tasks for constructing the learnware market and 21 tasks for simulating user scenarios. For each learnware task, we train a linear model and generate its specification as the learnware. Then, each user dataset will be split into two disjoint part: the testing part, which contains 90% of the samples and is used to evaluate the model performance; and remaining 10% is used to generate the specification. In our RNCME implementation, we employ an untrained linear model as the neural feature map ϕ_θ due to the simplicity of the input-output relationships in this setup. We compare our RNCME method with the basic RKME method (Wu et al. 2023), which only considers the input space.

Evaluation Metrics For each user task, we recommend an ordered set of k learnwares $\{f_{\mathcal{I}_1}, f_{\mathcal{I}_2}, \cdots, f_{\mathcal{I}_k}\}$ based on their specifications, where $f_{\mathcal{I}_i}$ represents the i-th nearest learnware to the user task. Suppose $\mathcal{J}_i$ denotes the rank of learnware $f_{\mathcal{I}_i}$ based on its performance on the user task, we compute the following metric:

$$\tau_k = \frac{\sum_{i=1}^{k} |i - \mathcal{J}_i|}{\sum_{i=1}^{k} |i - (n+1-i)|} \in [0, 1], \tag{10}$$

where the denominator represents the worst case when the top k learnwares are ranked in reverse order of their actual performance. Lower τ_k indicates better agreement between the specification-based recommendation and the actual performance. Finally, we compute the average $\bar{\tau}_k$ over all user tasks to evaluate the overall performance of each method.

Results and Analysis We present the experimental regression results in Table 1a. The results show that our RNCME method achieves superior performance compared to the basic RKME method for every k value. Particularly, for $k = 1$, there are $13/21$ tasks received the best-performing learnware, and $18/21$ received top-3 learnwares. When k increases, τ_k also increases. This is because the learnwares with lower rankings have very limited reusability for user tasks, and the differences between their input-output mappings show almost no relationship to how well the learnware actually performs on the task.

4.3 Real-World Classification Experiments

To further validate the effectiveness of our RNCME method, we conduct experiments on a larger and more complex real-world image classification dataset.

Dataset Preparation We use the NICO^{++} dataset (Zhang et al. 2023) for our classification experiments, which is designed for OOD image classification. The dataset contains over $200,000$ images across 80 categories, with each image annotated by both category label (e.g., cat or dog) and domain label (e.g., lying or running). The dataset provides 10 aligned common domains shared across all categories and 10 unique domains specific to each category, effectively simulating real-world scenarios with potential distribution shifts between training and testing data. This characteristic makes it suitable for evaluating the learnware paradigm.

Experimental Details We construct the learnware market using the common domain portion of the NICO^{++} dataset. First, we create 30 tasks by randomly selecting $30 \sim 40$ categories per task and sampling 90% of images from each category (averaging over $50,000$ images per task). For each task, we train a corresponding model and subsequently generate its specification to form a complete learnware. Specifically, the model comprises a frozen DenseNet201 (Huang et al. 2017) backbone (i.e., the common feature extractor), coupled with a two-layer MLP classifier. After thorough training on their respective tasks, each model achieves an accuracy of approximately 85%.

Method	$\bar{\tau}_1$	$\bar{\tau}_3$	$\bar{\tau}_5$	$\bar{\tau}_{10}$	$\bar{\tau}_{42}$
RNCME	$\mathbf{0.038}$	$\mathbf{0.074}$	$\mathbf{0.120}$	$\mathbf{0.331}$	$\mathbf{0.570}$
RKME	0.304	0.491	0.426	0.517	0.630

(a) Comparison in regression tasks.

Method	$\bar{\tau}_1$	$\bar{\tau}_3$	$\bar{\tau}_5$	$\bar{\tau}_{10}$	$\bar{\tau}_{30}$
RNCME	$\mathbf{0.033}$	$\mathbf{0.044}$	$\mathbf{0.064}$	$\mathbf{0.107}$	$\mathbf{0.190}$
LP	0.288	0.320	0.338	0.395	0.507
RKME	0.619	0.536	0.498	0.513	0.570

(b) Comparison in classification tasks.

RNCME	$\gamma = 0$	$\gamma = 0.1$	$\gamma = 0.3$	$\gamma = 0.5$
Trained	0.036	$\mathbf{0.033}$	0.036	0.079
Untrained	0.050	0.050	0.050	0.071

(c) Ablation Studies: the metric is $\bar{\tau}_1$.

Table 1: Real-world experiments results. The best performance for each k is highlighted in bold.

For user task simulation, we use the unique domain portion to create 20 tasks (about $15,000$ images per task) with inherent distribution shifts from the market. Then, similar to the regression tasks, user datasets will be split into testing and specification subsets. Critically, for each user task, we guarantee that no learnware in the market shares its exact label space configuration.

To investigate whether training ϕ_θ is effective and examine the impact of the hyperparameter γ defined in Equation (9), we conduct ablation studies.

Compared Methods Except for the basic RKME method, we also compare our RNCME method with the LP method (Guo et al. 2023), which additionally considers the output space, especially the heterogeneous output spaces. In our RNCME method, the neural feature map ϕ_θ takes features extracted by the shared DenseNet201 as input, and processes them through a 3-layer MLP with a 128-dimensional hidden layer and a 64-dimensional output layer. It is trained for only one epoch with a learning rate of 0.001, and we use the RBF kernel for the output space. For the basic RKME and LP, we use their official settings.

Results and Analysis We use the same metric $\bar{\tau}_k$ defined in Equation (9) to evaluate the performance of the compared methods. The results of comparative evaluation and ablation studies are shown in Tables 1b and 1c, respectively. From Table 1b, our RNCME method achieves superior performance compared to any other method for every k value. Particularly, for $k = 1$, there are $13/20$ tasks received the best-performing learnware, and $17/20$ received top-3 learnwares. And according to Table 1c, training ϕ_θ improves the capture of input-output mapping patterns, while appropriate RKME supplementation further enhances the performance.

5 Related Works

The learnware paradigm (Zhou 2016; Zhou and Tan 2024) aims to establish a market of learnwares that help users identify and reuse helpful machine learning model(s) instead of training from scratch. The specification plays a crucial role in the learnware paradigm, and there have been some efforts in this direction. Wu et al. (2023) proposed the RKME specification, which uses a reduced set to approximate the empirical KME of the training data, while Lei, Tan, and Zhou (2024) demonstrated its capability to preserve data privacy. Zhang et al. (2021) extended it to handle user tasks that contain unseen parts in the market. Tan et al. (2024a) adapted it for heterogeneous feature spaces. Guo et al. (2023); Tan et al. (2024a); Chen, Mao, and Zhang (2025) leveraged label information to model the input-output conditional distributions. Liu, Tan, and Zhou (2024) proposed an evolvable learnware specification to address the challenge of evaluating a model's capacity to exceed its original training task. Based on these works, Beimingwu (Tan et al. 2024b) has been released as the first learnware dock system.

Kernel Mean Embedding (KME) (Smola et al. 2007) provides a powerful framework for representing probability distributions as elements in RKHS. This framework was further advanced by Gretton et al. (2012), who established rigorous theoretical guarantees and demonstrated its connection to Maximum Mean Discrepancy (MMD) for hypothesis testing. The key advantage of KME lies in its nonparametric representation of distributions through RKHS embeddings, enabling efficient computation of distributional distances and expectations. The framework was extended to conditional distributions by Song et al. (2009), leading to the development of Conditional Mean Embedding (CME). Grünewälder et al. (2012) showed that empirical CME estimation corresponds to solving a vector-valued ridge regression problem. More recently, Shimizu, Fukumizu, and Sejdinovic (2024) enhanced CME by incorporating deep neural networks, combining the flexibility of deep learning with the theoretical guarantees of kernel methods. For a comprehensive overview of these developments and their applications, Muandet et al. (2017) offers a detailed survey.

6 Conclusion

Learnware paradigm aims to establish a market of learnwares, enabling users to effectively identify and reuse relevant models based on specifications instead of training from scratch. To address the limitation of existing labeled-RKME improvements, which lack clear theoretical explanations and are constrained to classification tasks, we propose a novel specification generation method called RNCME. Our method directly maps input-output conditional distributions into RKHS through CME, which maintains CME's theoretical foundations and handles both regression and classification tasks. We design a data-adaptive neural feature map for the input space, which reduces the high computational complexity of empirical CME estimation while avoiding the manual kernel selection problem, including both the choice of kernel functions and the tuning of regularization parameter. To preserve data privacy, we propose a variant of the Frank-Wolfe algorithm to generate a reduced set that approximates the empirical CME. The experimental results demonstrate the effectiveness of RNCME in generating specifications that accurately capture input-output mapping patterns in both regression and classification learnware recommendation tasks.

References

Baker, C. R. 1973. Joint measures and cross-covariance operators. *Transactions of the American Mathematical Society*, 186: 273–289.

Chen, W.; Mao, J.-X.; and Zhang, M.-L. 2025. Learnware Specification via Label-Aware Neural Embedding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39 of *AAAI'25*, 15857–15865.

Chen, Y.; Welling, M.; and Smola, A. J. 2010. Super-Samples from Kernel Herding. In Grünwald, P.; and Spirtes, P., eds., *UAI 2010, Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*. AUAI Press.

Gretton, A.; Borgwardt, K. M.; Rasch, M. J.; Schölkopf, B.; and Smola, A. 2012. A kernel two-sample test. *Journal of Machine Learning Research*, 13: 723–773.

Grünewälder, S.; Lever, G.; Gretton, A.; Baldassarre, L.; Patterson, S.; and Pontil, M. 2012. Conditional mean embeddings as regressors. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*. icml.cc / Omnipress.

Guo, L.-Z.; Zhou, Z.; Li, Y.-F.; and Zhou, Z.-H. 2023. Identifying useful learnwares for heterogeneous label spaces. In *International Conference on Machine Learning*, 12122–12131. PMLR.

Huang, G.; Liu, Z.; Van Der Maaten, L.; and Weinberger, K. Q. 2017. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Jaggi, M. 2013. Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In *International conference on machine learning*, 427–435. PMLR.

Lei, H.-Y.; Tan, Z.-H.; and Zhou, Z.-H. 2024. On the ability of developers' training data preservation of learnware. In *Advances in Neural Information Processing Systems*, volume 37, 36471–36513.

Liu, J.-D.; Tan, Z.-H.; and Zhou, Z.-H. 2024. Towards making learnware specification and market evolvable. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 13909–13917.

Muandet, K.; Fukumizu, K.; Sriperumbudur, B.; and Schölkopf, B. 2017. Kernel mean embedding of distributions: A review and beyond. *Foundations and Trends® in Machine Learning*, 10(1-2): 1–141.

Nugent, C. 2017. The California Housing Prices Dataset. https://www.kaggle.com/datasets/camnugent/california-housing-prices. Accessed: 2025-07-01.

Shimizu, E.; Fukumizu, K.; and Sejdinovic, D. 2024. Neural-kernel conditional mean embeddings. In *International conference on machine learning*.

Smola, A.; Gretton, A.; Song, L.; and Schölkopf, B. 2007. A Hilbert space embedding for distributions. In *International Conference on Algorithmic Learning Theory*, 13–31. Springer.

Song, L.; Huang, J.; Smola, A.; and Fukumizu, K. 2009. Hilbert space embeddings of conditional distributions with applications to dynamical systems. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML'09, 961–968.

Tan, P.; Liu, H.-T.; Tan, Z.-H.; and Zhou, Z.-H. 2024a. Handling learnwares from heterogeneous feature spaces with explicit label exploitation. In *Advances in Neural Information Processing Systems*, volume 37, 12767–12795.

Tan, Z.-H.; Liu, J.-D.; Bi, X.-D.; Tan, P.; Zheng, Q.-C.; Liu, H.-T.; Xie, Y.; Zou, X.-C.; Yu, Y.; and Zhou, Z.-H. 2024b. Beimingwu: A Learnware Dock System. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery*, 5773–5782. ACM.

Wolfe, P. 1976. Finding the nearest point in a polytope. *Mathematical Programming*, 11: 128–149.

Wu, X.-Z.; Xu, W.; Liu, S.; and Zhou, Z.-H. 2023. Model reuse with reduced kernel mean embedding specification. *IEEE Transactions on Knowledge and Data Engineering*, 35(1): 699–710.

Xu, L.; Chen, Y.; Srinivasan, S.; de Freitas, N.; Doucet, A.; and Gretton, A. 2021. Learning Deep Features in Instrumental Variable Regression. In *9th International Conference on Learning Representations*. OpenReview.net.

Zhang, X.; He, Y.; Xu, R.; Yu, H.; Shen, Z.; and Cui, P. 2023. Nico++: Towards better benchmarking for domain generalization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*.

Zhang, Y.-J.; Yan, Y.-H.; Zhao, P.; and Zhou, Z.-H. 2021. Towards enabling learnware to handle unseen jobs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35.

Zhou, Z.-H. 2016. Learnware: on the future of machine learning. *Frontiers Computer Science*, 10(4): 589–590.

Zhou, Z.-H.; and Tan, Z.-H. 2024. Learnware: Small models do big. *Science China Information Sciences*, 67(1): 112102.

A The Details of the Reduced Set Generation

Given the estimation of empirical CME $\widehat{\mathcal{U}} = \Psi Q$ in Equation (4), we seek a reduced set $V = \{v_j\}_{j=1}^m \subseteq \mathcal{Y}$ with bounded-weight matrix $R \in \mathbb{R}^{m \times d}$ to approximate $\widehat{\mathcal{U}}$. We first prove corollary 1, then introduce the Frank-Wolfe algorithm, and finally prove all theorems.

A.1 The Proof of Corollary 1

Proof. First, for a symmetric and positive definite matrix A, we have $\left\|A^{-1}\right\| = 1/\lambda_{\min}(A)$, where $\lambda_{\min}(A)$ denotes the minimal eigenvalue of A. Therefore, we need to compute the minimal eigenvalue of the symmetric and positive definite matrix $\Phi_\theta \Phi_\theta^{\mathrm{T}} + n\lambda I$. Suppose the minimal eigenvalue of $\Phi_\theta \Phi_\theta^{\mathrm{T}}$ is λ_d, and since $\Phi_\theta \Phi_\theta^{\mathrm{T}}$ is postive semi-definite, we have:

$$\lambda_{\min}\left(\Phi_\theta \Phi_\theta^{\mathrm{T}} + n\lambda I\right) = \lambda_d + n\lambda \geqslant n\lambda,$$

Then, under assumption 1, by Cauchy-Schiwarz inequality, we have:

$$\|q_i\| \leqslant \left\|\phi_\theta(x_i)^{\mathrm{T}}\right\| \cdot \left\|\left(\Phi_\theta \Phi_\theta^{\mathrm{T}} + n\lambda I\right)^{-1}\right\| \leqslant \frac{B_X}{n\lambda}.$$

□

A.2 Frank-Wolfe Algorithm

Frank-Wolfe algorithm (Wolfe 1976) is an iterative first-order optimization algorithm for constrained convex optimization. Suppose $\mathcal{D}$ is a compact convex set in a vector space, and $f : \mathcal{D} \to \mathbb{R}$ is a convex, differentiable function. After initializing $x_0 \in \mathcal{D}$, Frank-Wolfe algorithm solves the optimization problem:

$$\min_{x \in \mathcal{D}} f(x),$$

by performing an iterative procedure, where in each iteration t, the following steps are taken:

$$s_{t+1} \in \arg\min_{s \in \mathcal{D}} \langle \nabla f(x_t), s \rangle, \tag{11}$$

$$x_{t+1} = (1 - \alpha_t) x_t + \alpha_t s_{t+1}, \tag{12}$$

where, $\alpha_t \in [0, 1]$ can either be simply set to $\alpha_t = 1/(t+1)$ or optimized via line search to find the point in the segment with optimal value. Equation (11) minimizes the linear approximation of the problem given by the first-order Taylor approximation of f around x_t constrained to stay within $\mathcal{D}$.

A.3 The Proof of Theorem 1

Theorem 1 establishes that the iterative application of Equations (7) and (8) following initialization minimizes the objective function $f(U)$ from Equation (6) over the convex set $\mathcal{D}$ defined in Equation (5). To prove this result, it suffices to demonstrate that the entire procedure follows the standard Frank-Wolfe framework.

Proof. Firstly, for $f(U)$ defined in Equation (6), we have:

$$\nabla f(U) = \widehat{\mathcal{U}} - U = -W_t,$$

Denote s as the atom of $\mathcal{D}$ in Equation (7), i.e., $s = r\psi(v)$, we can show that Equation (7) is equivalent to Equation (11). Secondly, Equation (8) can be rewritten as:

$$U_{t+1} = \frac{t}{t+1} U_t + \frac{1}{t+1} s_{t+1},$$

which is equivalent to selecting $\alpha_t = 1/(t+1)$ in Equation (12). In addition, since $U_0 \in \mathcal{D}$ and $\mathcal{D}$ is a convex set, we can inductively show that $U_t \in \mathcal{D}$ for all $t \geqslant 0$. Therefore, the entire procedure follows the standard Frank-Wolfe framework, and it can minimize $f(U)$ over $\mathcal{D}$. □

A.4 Lemmas for Proving Theorem 2

In order to prove theorem 2, we first provide lemmas 1 to 4.

Lemma 1. *The optimization problem* $\min_{U \in \mathcal{D}} f(U)$ *has a trivial solution* $U^* = \widehat{\mathcal{U}}$.

Proof. It is obvious that $\min_{U\in\mathcal{G}} f(U) = 0$ when $U = \widehat{\mathcal{U}}$, but it may not hold when the domain changes to $\mathcal{D}$, therefore we need to prove that $\widehat{\mathcal{U}} \in \mathcal{D}$. From the definition of $\mathcal{D}$ in Equation (5), we know that $\mathcal{D}$ contains all elements expressible as finite linear combinations:

$$\sum_{j=1}^{t} \gamma_j r_j \psi(v_j),$$

for some $t \in \mathbb{N}$, where $\gamma \in \Delta^t$, $\|r_j\| \leqslant \frac{B_X}{\lambda}, \forall j \in [t]$. We have $\widehat{\mathcal{U}}$ in the following form:

$$\widehat{\mathcal{U}} = \Psi Q = \sum_{i=1}^{n} q_i \psi(y_i),$$

where $\|q_i\| \leqslant \frac{B_X}{n\lambda}, \forall i \in [n]$. Now we set $q_i = \gamma_i r_i$, where $\gamma_i = 1/n$ satisfies $\gamma \in \Delta^n$, and for each i, we have:

$$\|q_i\| = \frac{1}{n} \cdot \|r_i\| \leqslant \frac{B_X}{n\lambda} \implies \|r_i\| \leqslant \frac{B_X}{\lambda}.$$

Therefore, we can rewrite $\widehat{\mathcal{U}}$ as:

$$\widehat{\mathcal{U}} = \sum_{i=1}^{n} \frac{1}{n} r_i \psi(y_i),$$

and each $r_i\psi(y_i)$ is the atom of $\mathcal{D}$, which means $\widehat{\mathcal{U}} \in \mathcal{D}$. □

Lemma 2. *Suppose $\mathcal{D}$ is the convex hull of set S, defined in Equation* (5)*, and denote $d_{\mathcal{D}}$ and d_S are the diameters of $\mathcal{D}$ and S respectively, then $d_{\mathcal{D}} = d_S = \frac{2B_X B_Y}{\lambda}$.*

Proof. On the one hand, $\mathcal{D}$ contains all elements expressible as finite linear combinations, i.e., $U = \sum_{j=1}^{t} \gamma_j s_j$, where $t \in \mathbb{N}, \gamma \in \Delta^t$, and $s_j \in S$ is the atom of $\mathcal{D}$. According to the triangle inequality and the convexity of $\mathcal{D}$, $\forall U, U' \in \mathcal{D}$, we have:

$$\|U - U'\| \leqslant \max_{i,j} \left\|s_i - s'_j\right\| \leqslant d_S,$$

which means:

$$d_{\mathcal{D}} = \sup_{U,U'\in\mathcal{D}} \|U - U'\| \leqslant \sup_{s_i,s'_j\in S} \left\|s_i - s'_j\right\| = d_S.$$

On the other hand, $\mathcal{D} = \operatorname{conv}(S)$ means $S \subseteq \mathcal{D}$, then $d_S \leqslant d_{\mathcal{D}}$. Therefore:

$$d_{\mathcal{D}} = d_S = \sup_{s,s'\in S} \|s - s'\| \leqslant \sup_{s\in S} 2\|s\| = 2\|r\| \cdot \|\psi(v)\| = \frac{2B_X B_Y}{\lambda}.$$

□

Lemma 3. *The curvature constant of function f defined in Equation* (6) *is $C_f = \left(\frac{2B_X B_Y}{\lambda}\right)^2$.*

Proof. The curvature constant of a function f is defined as:

$$C_f = \sup_{\substack{U,U'\in\mathcal{D},\alpha\in[0,1] \\ y=U+\alpha(U'-U)}} \frac{2}{\alpha^2}\Big(f(y) - f(U) - \langle \nabla f(U), y - U\rangle\Big).$$

The Hessian matrix of f is $\nabla^2 f(U) = I$, therefore, according to the second-order Taylor expansion, we have:

$$\begin{aligned} f(y) &= f(U) + \langle \nabla f(U), y - U\rangle + \frac{1}{2}\left\langle y - U, \nabla^2 f(U)(y - U)\right\rangle \\ &= f(U) + \langle \nabla f(U), y - U\rangle + \frac{1}{2}\|y - U\|^2 \\ &= f(U) + \langle \nabla f(U), y - U\rangle + \frac{\alpha^2}{2}\|U' - U\|^2. \end{aligned}$$

Substituting this into the curvature constant definition, and according to lemma 2, we have:

$$\begin{aligned} C_f &= \sup_{\substack{U,U'\in\mathcal{D} \\ \alpha\in[0,1]}} \frac{2}{\alpha^2} \cdot \frac{\alpha^2}{2}\|U' - U\|^2 \\ &= \sup_{U,U'\in\mathcal{D}} \|U' - U\|^2 = d_{\mathcal{D}}^2 = \left(\frac{2B_X B_Y}{\lambda}\right)^2. \end{aligned}$$

□

Lemma 4. *For a step $U_{t+1} = U_t + \alpha_t(s_{t+1} - U_t)$ with $\alpha_t = 1/(t+1)$, it holds that:*

$$f(U_{t+1}) \leqslant f(U_t) - \alpha_t g(U_t) + \frac{\alpha_t^2}{2} C_f,$$

where $g(U) = \max_{U' \in \mathcal{D}} \langle \nabla f(U), U - U' \rangle$ is the current duality gap, and C_f is the curvature constant of f.

Proof. Lemma 4 is a specicial case of lemma 5 in Jaggi (2013), Appendix A. □

A.5 The proof of Theorem 2

Proof. Due to the convexity of function f, we have $\forall U \in \mathcal{D}$:

$$f(U) - f(U^*) \leqslant \langle \nabla f(U), U - U^* \rangle,$$

where $U^* = \widehat{\mathcal{U}}$. Then:

$$g(U) = \max_{U' \in \mathcal{D}} \langle \nabla f(U), U - U' \rangle \geqslant \langle \nabla f(U), U - U^* \rangle \geqslant f(U) - f(U^*).$$

According to lemma 1, we have $f(U^*) = 0$, thus $g(U) \geqslant f(U)$. Then according to lemma 4, we have:

$$\begin{aligned} f(U_{t+1}) &\leqslant f(U_t) - \alpha_t g(U_t) + \frac{\alpha_t^2}{2} C_f \\ &\leqslant f(U_t) - \alpha_t f(U_t) + \frac{\alpha_t^2}{2} C_f \\ &= (1 - \alpha_t) f(U_t) + \frac{\alpha_t^2}{2} C_f. \end{aligned}$$

Substituting $\alpha_t = 1/(t+1)$, we have:

$$f(U_{t+1}) \leqslant \frac{t}{t+1} f(U_t) + \frac{C_f}{2(t+1)^2} \implies (t+1) f(U_{t+1}) \leqslant t f(U_t) + \frac{C_f}{2(t+1)}.$$

Replace $t+1$ with m, we have:

$$m f(U_m) \leqslant (m-1) f(U_{m-1}) + \frac{C_f}{2m}.$$

By induction, we can show that:

$$m f(U_m) \leqslant \frac{C_f}{2} \sum_{i=1}^{m} \frac{1}{i} \leqslant \frac{C_f (\ln m + 1)}{2}.$$

According to lemmas 2 and 3, we have $C_f = d_{\mathcal{D}}^2 = \left(\frac{2B_X B_Y}{\lambda}\right)^2$, therefore:

$$\begin{aligned} f(U_m) = \frac{1}{2} \left\| U_m - \widehat{\mathcal{U}} \right\|^2 &\leqslant \frac{\left(\frac{2B_X B_Y}{\lambda}\right)^2 (\ln m + 1)}{2m} \\ \implies \left\| U_m - \widehat{\mathcal{U}} \right\| &\leqslant \frac{2B_X B_Y}{\lambda} \sqrt{\frac{\ln m + 1}{m}}. \end{aligned}$$

According to Song et al. (2009), $\widehat{\mathcal{U}}$ converges to $\mathcal{U}$ in the RKHS norm at a rate of $O\left(\frac{1}{\sqrt{n\lambda}} + \sqrt{\lambda}\right)$. Since $\widetilde{\mathcal{U}} = U_m$, we have:

$$\left\| \mathcal{U} - \widetilde{\mathcal{U}} \right\|_{\mathcal{G}} \leqslant O\left(\frac{1}{\sqrt{n\lambda}} + \sqrt{\lambda} + \frac{1}{\lambda} \sqrt{\frac{\ln m}{m}} \right).$$

□