Personalized Online Federated Learning with Multiple Kernels

Pouya M. Ghari
University of California Irvine
pmollaeb@uci.edu

Yanning Shen *
University of California Irvine
yannings@uci.edu

Abstract

Multi-kernel learning (MKL) exhibits well-documented performance in online non-linear function approximation. Federated learning enables a group of learners (called clients) to train an MKL model on the data distributed among clients to perform online non-linear function approximation. There are some challenges in online federated MKL that need to be addressed: i) Communication efficiency especially when a large number of kernels are considered ii) Heterogeneous data distribution among clients. The present paper develops an algorithmic framework to enable clients to communicate with the server to send their updates with affordable communication cost while clients employ a large dictionary of kernels. Utilizing random feature (RF) approximation, the present paper proposes scalable online federated MKL algorithm. We prove that using the proposed online federated MKL algorithm, each client enjoys sub-linear regret with respect to the RF approximation of its best kernel in hindsight, which indicates that the proposed algorithm can effectively deal with heterogeneity of the data distributed among clients. Experimental results on real datasets showcase the advantages of the proposed algorithm compared with other online federated kernel learning ones.

1 Introduction

Kernel learning exhibits well-documented performance in function approximation tasks, while providing theoretical guarantees associated with different performance metrics, see e.g. [48, 21, 38]. In some cases, a group of learners aims at collaborating to perform function approximation without revealing their data. To this end, federated learning has been emerged as a crucial learning paradigm by enabling a group of learners called clients to collaborate with each other by communicating with a central server to train a centralized model [33, 27, 12, 23]. Through this process, clients send model parameters and updates to the server without revealing their data. Upon receiving updates from clients, the server updates the model. Therefore, federated learning enables clients to perform kernel learning for function approximation. In this context, a server and clients collaborate with each other to learn the optimal kernel. Furthermore, in some practical cases, clients may need to perform the function approximation in an online fashion while they are collaborating with the server to learn the kernel. For example, consider the case where clients may not have enough memory to store data in batch. In addition, data samples may arrive in a sequential manner such that clients are not able to perform the function approximation in batch form. There are major challenges in performing online kernel learning in federated fashion that need to be addressed:

Communication Efficiency: Communication efficiency arises as a bottleneck in federated learning (see e.g. [25, 37, 19, 16]). Specifically, limited clients-to-server communication bandwidth restricts the number of parameters that can be sent from clients to the server.

Heterogeneous Data: The distribution of data observed by a client might be different from others

*Corresponding author

Thus, the optimal kernel that is aimed to be learned is different across clients. **Computational Complexity:** Clients should be able to perform function approximation fast enough in order to make a decision in real-time. Therefore, the computational complexity of kernel learning methods should be affordable for clients.

Conventional online kernel learning approaches (see e.g. [21, 39]) suffer from ‘curse of dimensionality’ [3] in the sense that the number of parameters that should be learned increases with the number of observed data. This can make employing conventional online kernel learning approaches infeasible to perform online federated kernel learning since clients may be required to send a large number of parameter updates to the server while the available clients-to-server communication bandwidth is not enough for sending such information. Approximating kernels by finite-dimensional feature representations (e.g. Nyström method [49] and random feature method of [36]) makes online kernel learning approaches scalable in the sense that the learner can choose the number of parameters that should be learned, independent of the number of observed data samples (see e.g. [30, 4, 52]).

Employing finite-dimensional feature representations of kernels to perform online federated kernel learning, clients can choose the number of parameters that they should send to the server. Therefore, finite-dimensional kernel approximation can better cope with limited clients-to-server communication bandwidth compared to conventional kernel learning approaches. Random feature (RF) approximation [56] has been exploited to perform online federated kernel learning when a single pre-selected kernel function is employed [20, 17]. The choice of the kernel function greatly affects the performance of function approximation when it comes to exploiting only a single kernel function. Employing multiple kernels instead of a single pre-selected one, can lead to obtaining more accurate function approximation since multi-kernel learning (MKL) can learn combination of kernels [24]. Online federated MKL algorithms with theoretical guarantees called vM-KOFL and eM-KOFL have been proposed in [22]. However, eM-KOFL and vM-KOFL do not provide personalized MKL models for clients since they learn the same combination of kernels for all clients.

The present paper proposes a novel personalized online federated MKL algorithm called POF-MKL that provides a personalized MKL model for each client while it is ensured that the available clients-to-server communication bandwidth can afford communication cost of sending clients’ updates to the server. In order to alleviate the communication cost of MKL, the proposed POF-MKL employs RF approximation of kernels and at each time instant, each client chooses a subset of kernels to send their updates to the server instead of sending the updates of all kernels. The number of kernels in the chosen subset is selected such that the required bandwidth to send all clients’ updates does not exceed the available clients-to-server communication bandwidth. Therefore, clients can send their updates to the server independent of the number of kernels in the dictionary and as a result a comparatively large dictionary of kernels can be considered to perform function approximation. Contributions of the present paper can be summarized as follows:

c1. Leveraging the proposed POF-MKL, clients can update a subset of kernels’ parameters which alleviates computational complexity and communication cost of sending updates to the server;
c2. Through theoretical analysis, it is proved that using the proposed POF-MKL, each client achieves sub-linear regret with respect to RF approximation of the best kernel in hindsight associated with the corresponding client data samples (c.f. Theorem 1). Moreover, it is guaranteed that the server achieves sub-linear regret with respect to the best function approximator (c.f. Theorem 2);
c3. Experiments on real datasets showcase the effectiveness of the proposed POF-MKL compared to existing online federated kernel learning algorithms.

### 2 Problem Statement and Preliminaries

Let there be a set of $K$ clients performing function approximation task in an online fashion. The $k$-th client’s goal is to learn the function $f$ using the stream of data samples $\{(x_{k,t}, y_{k,t})\}_{t=1}^{T}$ such that $x_{k,t} \in \mathbb{R}^d$ is the data sample observed by the $k$-th client at time $t$ and $y_{k,t}$ is the label associated with $x_{k,t}$. In the kernel learning context, the function $f$ is assumed to belong to a reproducing kernel Hilbert space (RKHS). The present paper studies the personalized federated supervised function approximation problem

$$\min_{f \in \mathcal{H}} \sum_{t=1}^{T} \sum_{k=1}^{K} \mathcal{L}(f(x_{k,t}), y_{k,t})$$  \hfill (1)
where $\mathbb{H}$ represents the RKHS the function $f$ belongs to and $\mathcal{L}(\cdot, \cdot)$ denotes the loss function which can be defined as

$$\mathcal{L}(f(x), y) = \mathcal{L}(f(x), y) + \lambda \Omega(||f||^2)$$

where $\mathcal{L}(\cdot, \cdot)$ is the cost function (e.g., least squares function for regression task), $\lambda$ denotes the regularization coefficient and $\Omega(\cdot)$ represents the model complexity. Let $\Theta$ be the global parameters of the function $f$ which are learned through collaboration of clients with the server while $w_k$ be the personalized parameter of the function $f$ learned locally by the $k$-th client. Thus, the goal is that the cumulative difference between $f(x_{k,t}; \Theta, w_k)$ and $y_{k,t}$ over time is minimized. At each time instant $t$, upon observing the data sample $x_{k,t}$, the $k$-th client make the prediction $f(x_{k,t}; \Theta, w_k)$ and then observes the true label $y_{k,t}$. Therefore, the function approximation problem that the server aims at solving can be expressed as $\min_\Theta \sum_{t=1}^T \sum_{k=1}^K \mathcal{L}(f(x_{k,t}; \Theta, w_k), y_{k,t})$. Moreover, finding the local parameters $w_k$ by the $k$-th client can be expressed as the optimization problem $\min_{w_k} \sum_{t=1}^T \mathcal{L}(f(x_{k,t}; \Theta, w_k), y_{k,t})$. In order to perform the function approximation task in an online fashion, the $k$-th client needs to perform the task with the values of $\Theta$ and $w_k$ at time $t$ denoted by $\Theta_t$ and $w_{k,t}$, respectively. Thus, the values of function parameters $\Theta$ and $w_k$ should be updated ‘on the fly’. Since the function $f(\cdot, \cdot)$ belongs to a reproducing kernel Hilbert space (RKHS), based on the representer theorem [43], given data samples, the optimal solution for (1) can be obtained as

$$\hat{f}(x) = \sum_{t=1}^T \sum_{k=1}^K \alpha_{k,t} \kappa(x, x_{k,t})$$

where $\kappa(\cdot, \cdot)$ denotes symmetric positive definite kernel function such that $\kappa(x, x')$ measures the similarity between $x$ and $x'$. And $\alpha_{k,t}$ is an unknown coefficient associated with $\kappa(x, x_{k,t})$ which is required to be estimated. In this case, $\hat{f}(\cdot)$ in (3) belongs to the RKHS $H := \{f(\cdot) \in \mathbb{H} \mid \text{RKHS norm is defined as } ||f||_H^2 := \sum_{x} \sum_{x'} \alpha_{k,t} \kappa(x, x_{k,t})\}$. Furthermore, from (3), it can be inferred that $\Theta_t = [\alpha_{1,1}, \ldots, \alpha_{K,1}, \ldots, \alpha_{1,K}, \ldots, \alpha_{K,K}]$. Therefore, the number of coefficients $\{\alpha_{k,t}\}_{t=1}^T, \forall k$ that should be estimated to obtain $\hat{f}(\cdot)$ increases over time. This is known as curse of dimensionality [43] since the computational complexity of function approximation increases with time. This brings challenges for federated implementation of function approximation since dimension of updates that should be sent to the server by each client grows over time and when $T$ is large, the available communication bandwidth may not be enough for clients to send their updates.

In order to deal with the increasing number of unknown coefficients, one can employ random Fourier approximation [36]. Assume that $\kappa(\cdot)$ is a shift-invariant kernel meaning that $\kappa(x, x') = \kappa(x - x')$. Let $\pi_n(\rho)$ denotes the Fourier transform of $\kappa(\cdot)$. If the kernel function $\kappa(\cdot)$ is normalized such that $\kappa(0) = 1$, then $\pi_n(\rho)$ can be viewed as a probability density function (PDF) (see e.g. [36]). Let $\rho_1, \ldots, \rho_D$ be a set of $D$ independent and identically distributed (i.i.d) vectors drawn from $\pi_n(\cdot)$. Let the vector $z(x)$ be defined as

$$z(x) = \frac{1}{\sqrt{D}}[\sin(\rho_1^T x), \ldots, \sin(\rho_D^T x), \cos(\rho_1^T x), \ldots, \cos(\rho_D^T x)].$$

Then, $\hat{\kappa}_n(x - x') = z(x)\mathsf{T}z(x')$ constitutes an unbiased estimator of $\kappa(x - x')$ and the random feature (RF) approximation of $\hat{f}(x)$ in (3) can be obtained as

$$\hat{f}_{RF}(x) = \sum_{t=1}^T \sum_{k=1}^K \alpha_{k,t} z(x_{k,t})\mathsf{T}z(x) := \Theta^{\mathsf{T}}z(x)$$

where in this case $\Theta = \sum_{t=1}^T \sum_{k=1}^K \alpha_{k,t} z(x_{k,t})$. According to (4), $z(x_{k,t})$ is a 2D vector and as a result it can be concluded that $\Theta$ is a 2D vector as well. Therefore, using RF approximation, the vector $\Theta$ should be estimated whose dimension does not grow over time.

The performance of a kernel learning algorithm depends on the choice of the kernel. Thus, performing the function approximation using a pre-selected kernel requires prior information which may not be available. To cope with this, employing a dictionary of kernels in lieu of a pre-selected single kernel has been proposed in the literature (see e.g. [45, 24, 31]). Specifically, the kernel is learned as a combination of kernels in the dictionary. Let $\kappa_1(\cdot), \ldots, \kappa_N(\cdot)$ be a set of
The present paper proposes an algorithmic framework to enable clients to perform online function approximation with sufficiently large dictionary of kernels while the available clients-to-server communication bandwidth can afford sending updates from clients to the server when a desirable value for the number of random features $D$ is chosen. To this end, at each time instant, each client
Algorithm 1 The $k$-th client kernel subset selection at time $t$.

**Input:** Weights $w_{ik,t}$, $i \in [N]$, parameter $M$ and exploration rate $0 < \xi_k \leq 1$.

Sort the kernels in descending order with respect to weights $\{w_{ik,t}\}_{k=1}^N$.

Obtain the index sequence $s_1, \ldots, s_N$ such that $w_{s_ik,t} \leq w_{s_jk,t}$ if $b > a$, $\forall a, b \in [N]$.

Open bin $B_1$ and initialize $j = 1$.

for all $i \in [N]$, the $k$-th client do

if the bin $B_j$ includes less than $M$ kernels then

    Adds the $s_i$-th kernel to $B_j$.

else

    Opens new bin $B_{j+1}$, adds the $s_i$-th kernel to $B_{j+1}$ and updates $j \leftarrow j + 1$.

end if

end for

Draw an index $I_{k,t}$ via PMF $q_{k,t}$ in (3).

**Output:** $S_{k,t}$: indices set of kernels in the selected bin $B_{I_{k,t}}$.

randomly chooses a subset of $M$ kernels among all $N$ kernels in the dictionary. Then, each client updates and sends the global parameters of the chosen $M$ kernels to the server instead of updating and sending global parameters of all kernels. To choose a subset of $M$ kernels, each client splits kernels into some bins and draws randomly one of the bins at each time instant. Each bin contains at most $M$ kernels and each client updates and sends global parameters associated with kernels in the chosen bin. In order to distribute kernels among bins, at first the $k$-th client sorts kernels in descending order according to kernels’ weights $\{w_{ik,t}\}_{k=1}^N$. Let $B_j$ represents the $j$-th bin of kernels.

The $k$-th client adds kernels from sorted list one by one to $B_j$ until either all kernels are assigned to a bin or the number of kernels in $B_j$ reaches $M$. When there are some kernels that are not assigned to any bins while there are $M$ kernels in $B_j$, the $k$-th client opens the bin $B_{j+1}$ and adds kernels to this bin. This continues until all kernels are assigned to a bin. As it can be inferred from the procedure of distributing kernels into bins, the number of bins at every client is $m = \lceil \frac{N}{M} \rceil$. Furthermore, it can be concluded that $B_1$ includes $M$ kernels with the largest weights while the bin $B_m$ includes $N - (m - 1)M$ kernels with lowest weights. The $k$-th client assigns the weight $w_{ik,t}$ at time $t$ to $B_j$ defined as $u_{jk,t} = \sum_{i \in B_j} w_{ik,t}$. The $k$-th client draws one of the bins according to the probability mass function (PMF) $q_{k,t}$ defined as

$$q_{jk,t} = (1 - \xi_k) \frac{u_{jk,t}}{U_{k,t}} + \frac{\xi_k}{m}, \forall j \in [m],$$

(8)

where $U_{k,t} = \sum_{j=1}^m u_{jk,t}$ and $0 < \xi_k \leq 1$ is an exploration rate determined by the $k$-th client. Let $I_{k,t}$ be the index of the chosen bin by the $k$-th client at time $t$. The PMF in (3) constitutes trade-off between exploitation and exploration. According to the first term in the right hand side of (8), it is more probable that the $k$-th client draws a bin which includes kernels with larger weights $w_{ik,t}$. Hence, it is more probable that the $k$-th client collaborates in updating the global parameters of a kernel with larger weight $w_{ik,t}$. Let $S_{k,t}$ denotes the set which includes the indices of kernels in the chosen bin at time $t$. The Algorithm 1 summarizes the procedure that the $k$-th client determines the set $S_{k,t}$. According to Algorithm 1, kernel subset selection is personalized since each client chooses its own subset of kernels to update their parameters.

Let $p_{ik,t}$ denotes the probability that $i \in S_{k,t}$. Then $p_{ik,t} = q_{b_i,k,t}$ where $b_i$ is the index of the bin which includes the $i$-th kernel. The $k$-th client updates global parameters locally as follows

$$\theta_{ik,t+1} = \theta_{i,t} - \eta \frac{\nabla L(\theta^T i, z_i(x_{ik,t}, y_{ik,t}))}{p_{ik,t}} 1_{i \in S_{k,t}},$$

(9)

where $1_{i \in S_{k,t}}$ denotes an indicator function and it is 1 when $i \in S_{k,t}$. The update rule in (9) implies that when $i \notin S_{k,t}$, the $k$-th client does not update $\theta_{i,t}$ (i.e. $\theta_{ik,t+1} = \theta_{ik,t}$). Therefore, the $k$-th client sends $\theta_{ik,t+1}$ to the server only if $i \in S_{k,t}$. Therefore, at each time instant, each client needs to send at most $2MD$ parameters to the server. Let $C_{i,t}$ be a set of client indices such that $k \in C_{i,t}$ if the $k$-th client sends $\theta_{ik,t+1}$ to the server. Upon aggregating updates from clients, the server updates $\theta_{i,t}$ as

$$\theta_{i,t+1} = \theta_{i,t} - \frac{1}{K} \sum_{k \in C_{i,t}} (\theta_{i,t} - \theta_{ik,t+1}) = \theta_{i,t} - \frac{\eta}{K} \sum_{k=1}^K \frac{\nabla L(\theta_{ik,t}^T z_i(x_{ik,t}, y_{ik,t}))}{p_{ik,t}} 1_{i \in S_{k,t}},$$

(10)
Algorithm 2 Personalized Online Federated Multi-Kernel Learning (POF-MKL)

**Input:** Kernels $\kappa_i$, $i = 1, \ldots, N$, learning rate $\eta > 0$ and the number of random features $D$.

**Initialize:** $\theta_{1,1} = 0$, $w_{ik,1} = 1$, $\forall i \in [N]$, $\forall k \in [K]$.

**for** $t = 1,\ldots,T$ **do**

The server transmits the global parameters $\Theta_t = [\theta_{1,t}, \ldots, \theta_{N,t}]$ to all clients.

**for all** $k \in [K]$, the $k$-th client **do**

Receive one datum $x_{k,t}$.

Predicts $\hat{f}(x_{k,t}; \Theta_t, w_{k,t})$ via (6).

Calculates losses $\mathcal{L}(\hat{f}_{RF,i}(x_{k,t}; \theta_{i,t}), y_{k,t})$, $\forall i \in [N]$.

Updates $w_{ik,t+1}$, $\forall i \in [N]$ via (7).

Selects a subset of kernel indices $S_{k,t}$ using Algorithm 1.

Updates $\theta_{ik,t+1}$, $\forall i \in S_{k,t}$ and sends $\theta_{ik,t+1}$ to the server.

**end for**

The server updates $\theta_{i,t+1}$, $\forall i \in [N]$ via (10).

**end for**

Algorithm 2 summarizes the proposed personalized online federated multi-kernel learning algorithm called POF-MKL. It is useful to note that using our proposed POF-MKL, the server cannot find the gradients $\nabla \mathcal{L}(\theta_{i,t}, z_i(x_{k,t}), y_{k,t})$ from updates received from clients. Instead, the server can find $\nabla \mathcal{L}(\hat{f}_{RF,i}(x_{k,t}; \theta_{i,t}), y_{k,t})/p_{ik,t}$ where $p_{ik,t}$ is a time-varying value determined locally by the $k$-th client. This can promote the privacy of the proposed POF-MKL since exchanging the gradients can be hazardous to the privacy of federated learning (see e.g. [53, 14]).

**Complexity.** Each client needs to store $d$-dimensional $D$ random feature vectors for each kernel. Therefore, the memory requirement of each client to implement function approximation using POF-MKL is $\mathcal{O}(dND)$. Using POF-MKL, at each time instant, each client needs to perform $\mathcal{O}(dND)$ operations including inner products and summations. Furthermore, when $\xi_k < 1$, in order to choose one subset of kernels, the $k$-th client needs to sort kernels which imposes worst case computational complexity of $\mathcal{O}(N \log N)$. However, when $\xi_k = 1$, according to PMF in [6], the $k$-th client chooses one bin uniformly at random and as a result in this case the $k$-th client does not need to sort kernels. Therefore, setting $\xi_k < 1$, the computational complexity for the $k$-th client is $\mathcal{O}(dND + N \log N)$ while setting $\xi_k = 1$, the computational complexity of the $k$-th client at each time instant is $\mathcal{O}(dND)$.

### 3.2 Regret Analysis

The present section analyzes the regret of the proposed POF-MKL. Specifically, two types of regret $R_{k,T}$ and $R_{s,T}$ are considered for the $k$-th client and the server, respectively. The performance of the $k$-th client utilizing POF-MKL is analyzed in terms of regret defined as

$$R_{k,T} = \sum_{t=1}^{T} \mathcal{L}(\hat{f}(x_{k,t}; \Theta_t, w_{k,t}), y_{k,t}) - \min_{\theta \in \Theta_t} \sum_{t=1}^{T} \mathcal{L}(\hat{f}_{RF,i}(x_{k,t}; \theta_{i,t}), y_{k,t})$$

(11)

where $R_{k,T}$ measures the cumulative difference between the loss of the $k$-th client and the loss of the RF approximation of the kernel with minimum loss among all kernels’ RF approximations. Let $\alpha_{ik,t}^*$, $\forall t \in [T], \forall k \in [K]$ represents the optimal coefficients associated with the $i$-th kernel such that $f^*_i(x) = \sum_{t=1}^{T} \sum_{k=1}^{K} \alpha_{ik,t}^* \kappa_i(x, x_{k,t})$. Then the best function approximator is defined as $f^*(\cdot) = \arg \min_{f^*_i, i \in [N]} \sum_{t=1}^{T} \sum_{k=1}^{K} \mathcal{L}(f^*_i(x_{k,t}), y_{k,t})$. Furthermore, the regret of the server is defined as the cumulative difference between the loss of POF-MKL and that of the best function approximator over all data samples distributed among clients which can be expressed as

$$R_{s,T} = \sum_{t=1}^{T} \sum_{k=1}^{K} \mathcal{L}(\hat{f}(x_{k,t}; \Theta_t, w_{k,t}), y_{k,t}) - \sum_{t=1}^{T} \sum_{k=1}^{K} \mathcal{L}(f^*(x_{k,t}), y_{k,t})$$

(12)

In order to analyze the regret of POF-MKL, suppose that the following assumptions hold true:

**(as1)** $\mathcal{L}(\theta_{i,t}, z_i(x_{k,t}), y_{k,t}), \forall k \in [K]$ is convex with respect to $\theta_{i,t}$ at each time instant $t$.

**(as2)** For $\theta$ in a bounded set satisfying $\|\theta\| \leq C$, the loss function and its gradient are bounded.
as \(0 \leq \mathcal{L}(\theta^T z_i(x_{k,t}), y_{k,t}) \leq 1\) and \(\| \nabla \mathcal{L}(\theta^T z_i(x_{k,t}), y_{k,t}) \| \leq L\). Moreover, each data sample is bounded as \(\|x_{k,t}\| \leq 1, \forall k \in [K], \forall t \in [T]\).

(as3) Kernels \(\kappa_i(\cdot), \forall i \in [N]\) are shift-invariant with \(\kappa_i(0) = 1, \forall i \in [N]\).

The following theorem investigates the regret of the \(k\)-th client according to the \(k\)-th client data. The proof of the following Theorem can be found in Appendix A.

**Theorem 1.** Under (as1)–(as3), the regret of the \(k\)-th client with respect to the best function approximator satisfies

\[
\begin{align*}
\sum_{t=1}^{T} \mathcal{L}(\hat{f}(x_{k,t}; \hat{\Theta}_t, w_{k,t}), y_{k,t}) - \min_{i \in [N]} \sum_{t=1}^{T} \mathcal{L}((f^*_i(x_{k,t}; \theta^*_i), y_{k,t})) \leq \frac{\ln N}{\eta_k} + \frac{\eta k}{2}T. 
\end{align*}
\]

Theorem 1 shows that by setting \(\eta_k = O\left(\sqrt{\frac{T}{N}}\right)\), the \(k\)-th client achieves sub-linear regret of \(O(\sqrt{T})\).

Furthermore, Theorem 1 shows that POF-MKL can deal with heterogeneous data among clients since the regret of each client defined in (11) is calculated with respect to the corresponding client data. The following theorem studies the regret of the server with respect to the best function approximator.

The proof can be found in Appendix B.

**Theorem 2.** Let \(i^* := \arg\min_{i \in [N]} \sum_{t=1}^{T} \sum_{k=1}^{K} \mathcal{L}(f^*_i(x_{k,t}), y_{k,t})\) and \(\sigma_i\) be the second Fourier moment of the \(i\)-th kernel. Under (as1)–(as3), the regret of the server with respect to the best function approximator satisfies

\[
\begin{align*}
\sum_{t=1}^{T} \sum_{k=1}^{K} \mathcal{L}(\hat{f}(x_{k,t}; \hat{\Theta}_t, w_{k,t}), y_{k,t}) - \sum_{t=1}^{T} \sum_{k=1}^{K} \mathcal{L}(f^*_i(x_{k,t}), y_{k,t}) \\
\leq \frac{K C^2}{2\eta} + \frac{\eta}{2} \sum_{t=1}^{T} \sum_{k=1}^{K} \frac{L^2}{\eta_t} + \sum_{k=1}^{K} \left(\ln N + \frac{\eta k}{2}T\right) + \epsilon L K T C
\end{align*}
\]

with probability at least \(1 - 2^8 \left(\frac{2\pi^2}{\epsilon}\right)^2 \exp\left(-\frac{D \epsilon^2}{4(\epsilon + 2)}\right)\) where \(C := \max_{i \in [N]} \sum_{t=1}^{T} \sum_{k=1}^{K} \sigma_{i,k,t}^2\).

As it can be inferred from (14), the regret of the server with respect to the best function approximator depends on \(\frac{1}{\eta_t}\) and the fact that \(p_{i,k,t} = q_{b,k,t}\), it can be concluded that \(p_{i,k,t} > \frac{\xi_k}{m}\). Thus, setting \(\xi_k = O(1)\), then \(p_{i,k,t} > O\left(\frac{M}{N}\right)\). The regret bound in (14) shows that setting \(\eta = O\left(\sqrt{\frac{M}{NT}}\right)\) and \(\epsilon = \eta_k = \frac{1}{\sqrt{T}}, \forall k \in [K]\), the server obtains regret of \(O\left(\sqrt{\frac{N}{MT}}\right)\) with probability at least \(1 - 2^8 \left(\frac{2\pi^2}{\epsilon}\right)^2 \exp\left(-\frac{D \epsilon^2}{4(\epsilon + 2)}\right)\). This shows that increasing \(M\) tightens the regret bound and increasing \(D\) increases the probability that the regret bound in (14) holds true. However, using POF-MKL, each client needs to transmit \(MD\) parameters at each time instant. Since both \(M\) and \(D\) are determined by the algorithm POF-MKL, this shows that POF-MKL can provide flexibility to tighten regret bound while the available clients-to-server communication bandwidth can afford transmission of clients’ updates to the server. It is useful to mention that choosing larger value for \(\xi_k\) increases the lower bound of \(p_{i,k,t}\) and as a result the optimal choice for \(\xi_k\) in terms of regret is \(\xi_k = 1\). However, choosing smaller values for \(\xi_k\) makes the value of \(p_{i,k,t}\) more dependent on weights \(\{w_{i,k,t}\}_{k=1}^{K}\) (c.f. (8)). Therefore, choosing smaller values for \(\xi_k\) makes \(p_{i,k,t}\) more predictable. This makes estimating \(\nabla \mathcal{L}(\theta^*_i z_i(x_{k,t}), y_{k,t})\) given \(\nabla \mathcal{L}(\theta^*_i z_i(x_{k,t}), y_{k,t})/p_{i,k,t}\) more difficult which leads to better protection of privacy.

**Comparison with personalized federated learning.** In order to deal with heterogeneous data among clients, personalized federated learning has been studied extensively in the literature (see [44][8][9][12][20][10][29][28][42][6][1][3][5][2][46][5]). Utilizing model-agnostic meta-learning [13], personalized federated learning algorithms have been proposed in [12][11]. In [42][5], personalized federated learning algorithms have been designed by learning hyper-networks [18]. In [32], a personalized model is a linear combination of a set of shared component models such that each client constructs its personalized mixture of models. However, in aforementioned personalized federated learning works, clients are assumed to store a dataset to perform local updates with. Therefore, when clients are not able to store data in batch and they have to make a decision upon receiving a new data sample, aforementioned works in personalized federated learning cannot guarantee sub-linear
We tested the performance of the proposed POF-MKL for online regression task through a set of which enables clients to perform their learning task in an online and federated fashion while it is proved that Fed-OMD enjoys sub-linear regret when the loss function is convex with respect to parameters required to be learnt at each time instant. The proposed POF-MKL differs from Fed-OMD in the sense that Fed-OMD cannot guarantee sub-linear regret when it comes to performing the online learning task with RF approximations of multiple kernels since the loss function \( \mathcal{L}(\sum_{i=1}^{N} w_i \theta_i^T z_i(x), y) \) is not convex with respect to both \( \theta_i \) and \( w_i \). However, according to Theorems 1 and 2, the proposed POF-MKL guarantees sub-linear regret.

Comparison with online federated learning. Fed-OMD algorithm has been proposed in [34] that Fed-OMD cannot guarantee sub-linear regret when it comes to performing the online learning task with RF approximations of multiple kernels since the loss function \( \mathcal{L}(\sum_{i=1}^{N} w_i \theta_i^T z_i(x), y) \) is not convex with respect to both \( \theta_i \) and \( w_i \). Therefore, according to Theorems 1 and 2, POF-MKL provides sub-linear regret for clients when clients cannot store data in batch and make decision in an online fashion.

Comparison with [22]. Online federated MKL algorithms called vM-KOFL and eM-KOFL have been presented in [22]. Both POF-MKL and algorithms in [22] exploit random feature approximation to alleviate computational complexity of online kernel learning. Furthermore, both POF-MKL and algorithms in [22] learn a linear combination of kernels. The proposed POF-MKL has the following advantages and innovations compared to vM-KOFL and eM-KOFL: i) The proposed POF-MKL allows clients to learn their own personalized combination of kernels (c.f. [6]). As it is proved in Theorem 1 the proposed POF-MKL can deal with heterogeneous data among clients in the sense that using POF-MKL each client guarantees sub-linear regret with respect to the best kernel RF approximation according to the corresponding client data. However, both vM-KOFL and eM-KOFL are not able to provide such guarantee. ii) Using vM-KOFL, each client needs to send \((N+1)D\) parameters to the server. However, using the proposed POF-MKL, each client needs to send \(MD\) parameters to the server such that \(M \leq N\) is determined by POF-MKL and can be chosen to be much smaller than \(N\). iii) In eM-KOFL, the server chooses a kernel at each time instant and clients send their local updates associated with the chosen kernel by the server. The proposed POF-MKL provides more flexibility compared to eM-KOFL in the sense that using POF-MKL each client can send local updates of \(M \geq 1\) kernels to the server. And each client chooses its own subset of kernels to send their updates to the server. Therefore, even though POF-MKL sets \(M\) to 1, it is possible that at a time instant the server receives updates associated with all kernels in the dictionary. It is useful to mention that using eM-KOFL the cumulative regret of all clients is sub-linear with respect to the best kernel RF approximation with probability \(1 - \delta\) where \(0 < \delta \leq 1\). However, utilizing the update rule in [9], using the proposed POF-MKL, each client obtains sub-linear regret with respect to RF approximation of its best kernel with probability 1.

4 Experiments

We tested the performance of the proposed POF-MKL for online regression task through a set of experiments. The performance of POF-MKL is compared with the baselines PerFedAvg [12], OFSKL [34], OFMKL-Avg [34], vM-KOFL [22] and eM-KOFL [22]. PerFedAvg refers to the personalized federated averaging algorithm in [12]. In the experiments, PerFedAvg employs a fully connected feedforward neural network model. More information about the implementation of PerFedAvg can be found in Appendix C. OFSKL and OFMKL-Avg are two variations of Fed-OMD [34]. OFSKL leverages Fed-OMD [34] when a single radial basis function (RBF) with bandwidth of 10 is employed to perform the learning task. In OFMKL-Avg, kernels are learned independently from each other using Fed-OMD [34] and the prediction is the average of approximations given by kernels. Moreover, vM-KOFL and eM-KOFL are online federated MKL algorithms of [22] such that vM-KOFL requires transmission of all kernel updates at every time instant while eM-KOFL requires transmission of a kernel update at each time instant. In the experiments, each client observes \(500\) samples until the end of the learning task meaning that \(T = 500\). The performance of the proposed POF-MKL and other baselines are tested on the following real datasets downloaded from UCI machine learning repository [11]: Naval [7], UJI [47], Air [51] and WEC [35]. More detailed information about datasets can be found in Appendix C. Data samples of Naval and UJI datasets are distributed i.i.d among clients. Data samples in Air and WEC datasets are distributed non-i.i.d among clients. More information about distributing data samples among clients can be found in Appendix C. The number of clients for Naval, UJI, Air and WEC datasets are 23, 42, 240 and 560, respectively. The dictionary of kernels consists of 51 RBFs with different bandwidth such that the bandwidth of the \(i\)-th kernel is \(\sigma_i = 10\left(\frac{2i-1}{50}\right)\). We consider the case where the clients-to-server communication bandwidth is limited such that at each time instant, the maximum number of parameters that a client is allowed to transmit to the server
is 1000. Furthermore, the memory and computational capability of clients are limited such that the maximum value can be picked for the number of random features $D$ is 100. The experiments were carried for 20 different sets of random feature vectors. The performance of algorithms is measured using average of mean squared error (MSE) defined as

$$\text{MSE} = \frac{1}{20} \sum_{j=1}^{20} \frac{1}{KT} \sum_{t=1}^{T} \sum_{k=1}^{K} (\hat{y}_{jk,t} - y_{k,t})^2$$

where $\hat{y}_{jk,t}$ denotes the prediction of the $k$-th client at time instant $t$ corresponding to the $j$-th set of random feature vectors. Learning rates are set to $\eta = \eta_k = \frac{20}{\sqrt{D}}$, $\forall k$. Also, exploration rates are set to $\xi_k = 1$, $\forall k$. The performance of POF-MKL with different $\xi_k$ is studied in Appendix C. Codes are available at https://github.com/pouyamghari/POF-MKL.

Table 1 presents the MSE and run time performance of online federated kernel learning algorithms on real datasets. Run time refers to average total run time of clients to perform online learning task on the entire data samples that they observe. In Table 1, $M$ refers to the number of kernels whose updates are sent to the server after prediction at each time instant. And, $D$ is the number of random features. Comparing MSE of POF-MKL with that of OFMKL-Avg, it can be concluded that learning the weights to combine kernels provides higher accuracy than averaging kernels’ predictions. Table 1 shows that POF-MKL with $M = 1$ provides lower MSE than eM-KOFL. Using eM-KOFL, at each time instant, the server receives updates belong to only one kernel. However, using POF-MKL with $M = 1$, each client sends an update belongs to a kernel which is selected by the client. Therefore, the server receives updates associated with different kernels even when $M = 1$. Therefore, experimental results show the effectiveness of the personalized kernel selection provided by POF-MKL. It can be observed that POF-MKL with $M = 25$ obtains lower MSE than those of POF-MKL with $M = 51$ and vM-KOFL. Since each client is allowed to send at most 1000 parameters per time instant, if clients send updates of all kernels at every time instant as this is the case in vM-KOFL, $D$ cannot be chosen to be greater than 9. However, setting $M = 25$, POF-MKL can set $D = 20$ which can improve the accuracy of online regression task compared to the case where $D = 9$. Note that according to Theorem 1, increase in $D$ increases the probability that the server achieves sub-linear regret with respect to the best function approximator. Furthermore, POF-MKL with $M = 51$ achieves lower MSE than vM-KOFL even if data samples are distributed i.i.d among clients. This shows that the proposed POF-MKL can better cope with heterogeneous data among clients which is in agreement with theoretical results in Theorem 1. In fact, the optimal combination of kernels can be different across clients. Using POF-MKL, each client constructs its own personalized combination of kernels which results in lower MSE compared to vM-KOFL. The proposed POF-MKL with $M = 1$ and $M = 25$ runs faster than eM-KOFL. In fact, using POF-MKL, clients only need to update parameters associated with $M$ kernels while employing vM-KOFL and eM-KOFL, clients have to update parameters of all kernels. Moreover, POF-MKL obtains lower MSE than PerFedAvg. Note that since clients are not able to store data in batch, at each time instant clients update PerFedAvg’s model using only the newly observed data sample. Therefore, convergence of PerFedAvg is not guaranteed. Experimental results show that POF-MKL achieves higher accuracy than PerFedAvg in online regression task when it is not possible for clients to store data in batch. Since OFSKL employs only a pre-selected single kernel, OFSKL runs faster than POF-MKL. However, utilizing multiple kernels enables POF-MKL to obtain lower MSE than that of OFSKL. In fact, using POF-MKL clients learn a linear combination of kernels which is proved to enjoy sub-linear regret with respect to the best kernel in hindsight while employing OFSKL clients have to make predictions using a pre-selected kernel. Furthermore, Figure 1 illustrates the average regret of clients when clients employ vM-KOFL and the proposed POF-MKL with different $M$ parameters. From Figure 1 it can be observed that the proposed POF-MKL achieves sub-linear regret.

5 Conclusions

The present paper proposed a personalized online federated MKL algorithm called POF-MKL based on RF approximation. Employing the proposed POF-MKL, each client updates the parameters of a subset of kernels which alleviates the computational complexity of the client as well as communication cost of sending updated parameters of kernels. Theoretical analysis proved that using POF-MKL, each client achieves sub-linear regret with respect to the RF approximation of its best kernel in hindsight which indicates that POF-MKL can deal heterogeneous data among clients. While each
Table 1: MSE($\times 10^{-3}$) and run time of online federated learning algorithms on real datasets.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>M</th>
<th>D</th>
<th>Naval</th>
<th>UJI</th>
<th>Air</th>
<th>WEC</th>
<th>Naval</th>
<th>UJI</th>
<th>Air</th>
<th>WEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>PerFedAvg</td>
<td>-</td>
<td>-</td>
<td>118.60</td>
<td>63.03</td>
<td>13.68</td>
<td>77.33</td>
<td>44.59</td>
<td>41.67</td>
<td>37.40</td>
<td>33.56</td>
</tr>
<tr>
<td>OFSKL</td>
<td>1</td>
<td>100</td>
<td>77.77</td>
<td>61.82</td>
<td>13.65</td>
<td>87.87</td>
<td>0.07</td>
<td>0.06</td>
<td>0.08</td>
<td>0.06</td>
</tr>
<tr>
<td>OFMKL-Avg</td>
<td>51</td>
<td>9</td>
<td>33.25</td>
<td>55.44</td>
<td>10.63</td>
<td>34.01</td>
<td>1.51</td>
<td>1.73</td>
<td>0.91</td>
<td>0.47</td>
</tr>
<tr>
<td>vM-KOFL</td>
<td>51</td>
<td>9</td>
<td>26.42</td>
<td>51.50</td>
<td>10.58</td>
<td>25.17</td>
<td>2.01</td>
<td>2.22</td>
<td>1.37</td>
<td>0.67</td>
</tr>
<tr>
<td>eM-KOFL</td>
<td>1</td>
<td>100</td>
<td>28.64</td>
<td>61.08</td>
<td>21.94</td>
<td>20.14</td>
<td>2.27</td>
<td>10.13</td>
<td>1.45</td>
<td>1.70</td>
</tr>
<tr>
<td>POF-MKL</td>
<td>1</td>
<td>100</td>
<td><strong>16.16</strong></td>
<td><strong>33.02</strong></td>
<td><strong>9.27</strong></td>
<td><strong>11.44</strong></td>
<td>1.22</td>
<td>9.02</td>
<td>1.25</td>
<td>1.10</td>
</tr>
<tr>
<td>POF-MKL</td>
<td>25</td>
<td>20</td>
<td>16.82</td>
<td>37.34</td>
<td>9.34</td>
<td>11.58</td>
<td>0.69</td>
<td>2.29</td>
<td>0.63</td>
<td>0.52</td>
</tr>
<tr>
<td>POF-MKL</td>
<td>51</td>
<td>9</td>
<td>16.65</td>
<td>41.00</td>
<td>9.38</td>
<td>11.97</td>
<td>0.82</td>
<td>1.07</td>
<td>0.81</td>
<td>0.65</td>
</tr>
</tbody>
</table>

(client updates a subset of kernels, it was proved that the server achieves sub-linear regret with respect to the best function approximator. Experiments on real datasets showcased the advantages of POF-MKL compared with other online federated kernel learning algorithms.

Acknowledgement

This work is supported by NSF ECCS 2207457. PI Yanning Shen also receives support from Microsoft Academic Grant Award for AI Research. Contact: Yanning Shen (yannings@uci.edu).

References


[40] Doyen Sahoo, Steven C. H. Hoi, and Bin Li. Large scale online multiple kernel regression with application to time-series prediction. ACM Transactions on Knowledge Discovery from Data, 13(1), Jan 2019.


Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper’s contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] In section 4, we mention that the proposed multi-kernel learning algorithm POF-MKL runs slower than single kernel learning algorithm OFSKL.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] We do not foresee any direct societal impact.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] Please see section 3.2 where we explicitly explain the assumptions.
   (b) Did you include complete proofs of all theoretical results? [Yes] Complete proofs of theoretical results are presented in Appendices A and B.

3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We provide code, data, and instructions to reproduce the main experimental results in the supplementary material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] In section 4 and Appendix C, detailed information about experimental setup is provided.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We did the experiments for 20 sets of random features. In Appendix C, the standard deviation of MSE is reported.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] In Table 1 we report run time of algorithms. Also, in Appendix C we explain that all experiments were carried out using Intel(R) Core(TM) i7-10510U CPU @ 1.80 GHz 2.30 GHz processor with a 64-bit Windows operating system.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] In section 4 we cited all datasets used for experiments.
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you’re using/curating? [Yes] In section 4 we point out that datasets are downloaded from UCI Machine Learning Repository.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]
A Proof of Theorem 1

Since \( W_{k,t} = \sum_{i=1}^{N} u_{i,k,t} \) and according to (7), we can write

\[
\frac{W_{k,t+1}}{W_{k,t}} = \sum_{i=1}^{N} \frac{u_{i,k,t+1}}{W_{k,t}} = \sum_{i=1}^{N} \frac{u_{i,k,t}}{W_{k,t}} \exp \left( -\eta_k \mathcal{L}(\hat{f}_{RF,it}(x_{k,t}; \theta_{i,t}), y_{k,t}) \right). \tag{15}
\]

Using the inequality \( e^{-x} \leq 1 - x + \frac{1}{2} x^2, \forall x \geq 0 \), the upper bound of (15) can be obtained as

\[
\frac{W_{k,t+1}}{W_{k,t}} \leq \sum_{i=1}^{N} \frac{u_{i,k,t}}{W_{k,t}} \left( 1 - \eta_k \mathcal{L}(\hat{f}_{RF,it}(x_{k,t}; \theta_{i,t}), y_{k,t}) + \frac{\eta_k^2}{2} \mathcal{L}^2(\hat{f}_{RF,it}(x_{k,t}; \theta_{i,t}), y_{k,t}) \right). \tag{16}
\]

Using the inequality \( 1 + x \leq e^x \) and taking the logarithm from both sides of (16), we get

\[
\ln \frac{W_{k,t+1}}{W_{k,t}} \leq \sum_{i=1}^{N} \frac{u_{i,k,t}}{W_{k,t}} \left( -\eta_k \mathcal{L}(\hat{f}_{RF,it}(x_{k,t}; \theta_{i,t}), y_{k,t}) + \frac{\eta_k^2}{2} \mathcal{L}^2(\hat{f}_{RF,it}(x_{k,t}; \theta_{i,t}), y_{k,t}) \right). \tag{17}
\]

According to (as2), \( \mathcal{L}^2(\hat{f}_{RF,it}(x_{k,t}; \theta_{i,t}), y_{k,t}) \leq 1 \). Therefore, from (17), we can conclude that

\[
\ln \frac{W_{k,t+1}}{W_{k,t}} \leq \sum_{i=1}^{N} \frac{u_{i,k,t}}{W_{k,t}} \left( \frac{\eta_k^2}{2} - \eta_k \mathcal{L}(\hat{f}_{RF,it}(x_{k,t}; \theta_{i,t}), y_{k,t}) \right). \tag{18}
\]

Summing (18) over time, we arrive at

\[
\ln \frac{W_{k,T+1}}{W_{k,1}} \leq \sum_{t=1}^{T} \sum_{i=1}^{N} \frac{u_{i,k,t}}{W_{k,t}} \left( \frac{\eta_k^2}{2} - \eta_k \mathcal{L}(\hat{f}_{RF,it}(x_{k,t}; \theta_{i,t}), y_{k,t}) \right). \tag{19}
\]

Moreover, for any \( i \in [N] \), \( \ln \frac{W_{k,T+1}}{W_{k,1}} \) can be lower bounded as

\[
\ln \frac{W_{k,T+1}}{W_{k,1}} = \ln \frac{\sum_{i=1}^{N} u_{i,k,t}}{W_{k,1}} = -\eta_k \sum_{t=1}^{T} \mathcal{L}(\hat{f}_{RF,it}(x_{k,t}; \theta_{i,t}), y_{k,t}) - \ln N. \tag{20}
\]

Combining (19) with (20), we obtain

\[
\sum_{t=1}^{T} \sum_{i=1}^{N} \frac{u_{i,k,t}}{W_{k,t}} \mathcal{L}(\hat{f}_{RF,it}(x_{k,t}; \theta_{i,t}), y_{k,t}) - \sum_{i=1}^{N} \mathcal{L}(\hat{f}_{RF,it}(x_{k,t}; \theta_{i,t}), y_{k,t}) \leq \ln N \eta_k + \frac{\eta_k^2}{2} T. \tag{21}
\]

Since the loss function \( \mathcal{L}(\cdot, \cdot) \) is convex, using (21) and Jensen inequality we can write

\[
\sum_{t=1}^{T} \mathcal{L}(\hat{f}_{RF,it}(x_{k,t}; \theta_{i,t}), y_{k,t}) - \sum_{i=1}^{N} \mathcal{L}(\hat{f}_{RF,it}(x_{k,t}; \theta_{i,t}), y_{k,t}) \leq \ln N \eta_k + \frac{\eta_k^2}{2} T \tag{22}
\]

which proves the Theorem 1.

B Proof of Theorem 2

In order to prove Theorem 2, the following Lemma is used as a stepstone.

Lemma 3. Let \( \alpha_{x_{k,t}}, \forall t \in [T], \forall k \in [K] \) represents the optimal coefficients associated with the \( i \)-th kernel such that \( f_i^*(x) = \sum_{t=1}^{T} \sum_{k=1}^{K} \alpha_{x_{k,t}} x_{k,t}(x, x_{k,t}) \). And \( \hat{f}_i^*(x) = (\theta_i^*)^T z_i(x) \) denotes the best RF-based estimator associated with the \( i \)-th kernel such that \( \theta_i^* = \sum_{t=1}^{T} \sum_{k=1}^{K} \alpha_{x_{k,t}}^* z_i(x_{k,t}) \). Under assumptions (as1)–(as3), using POF-MKL, the RF approximation of the \( i \)-th kernel satisfies

\[
\frac{1}{\eta} \| \theta_i^* \|^2 + \sum_{t=1}^{T} \sum_{k=1}^{K} \frac{L^2}{p_{i,k,t}}. \tag{23}
\]
Proof. Let $\ell_{ik,t}$ be the importance sampling loss estimate defined as

$$
\ell_{ik,t} = \frac{\mathcal{L}(\theta_t^\top z_i(x_{k,t}), y_{k,t})}{p_{ik,t}} 1_{i \in S_{k,t}}.
$$

Then according to (25), for any fixed $\theta$, it can be written that

$$
\left\| \frac{1}{K} \sum_{k=1}^{K} \theta_{i,k+1} - \theta \right\|^2 = \left\| \frac{1}{K} \sum_{k=1}^{K} (\theta_{i,t} - \eta \nabla \ell_{ik,t}) - \theta \right\|^2 = \left\| \theta_{i,t} - \theta - \eta \frac{1}{K} \sum_{k=1}^{K} \nabla \ell_{ik,t} \right\|^2
$$

$$= \left\| \theta_{i,t} - \theta \right\|^2 - \frac{2\eta}{K} \left( \sum_{k=1}^{K} \nabla^\top \ell_{ik,t} \right) \left( \theta_{i,t} - \theta \right) + \left\| \eta \frac{1}{K} \sum_{k=1}^{K} \nabla \ell_{ik,t} \right\|^2
$$

According to the convexity of the loss function $\mathcal{L}(\theta^\top x, y)$ with respect to $\theta$ as stated in (as1), we find

$$
\mathcal{L}(\theta_{i,t}^\top z_i(x_{k,t}), y_{k,t}) - \mathcal{L}(\theta^\top z_i(x_{k,t}), y_{k,t}) \leq \nabla^\top \mathcal{L}(\theta_{i,t}^\top z_i(x_{k,t}), y_{k,t}) (\theta_{i,t} - \theta).
$$

Multiplying both sides of (26) by $\frac{1_{i \in S_{k,t}}}{p_{ik,t}}$, we get

$$
\left( \frac{\mathcal{L}(\theta_{i,t}^\top z_i(x_{k,t}), y_{k,t})}{p_{ik,t}} - \frac{\mathcal{L}(\theta^\top z_i(x_{k,t}), y_{k,t})}{p_{ik,t}} \right) 1_{i \in S_{k,t}} \leq \nabla^\top \mathcal{L}(\theta_{i,t}^\top z_i(x_{k,t}), y_{k,t}) (\theta_{i,t} - \theta) 1_{i \in S_{k,t}}.
$$

Summing (27) over $k, \forall k \in [K]$, we arrive at

$$
\sum_{k=1}^{K} \left( \frac{\mathcal{L}(\theta_{i,t}^\top z_i(x_{k,t}), y_{k,t})}{p_{ik,t}} - \frac{\mathcal{L}(\theta^\top z_i(x_{k,t}), y_{k,t})}{p_{ik,t}} \right) 1_{i \in S_{k,t}} \leq \sum_{k=1}^{K} \nabla^\top \mathcal{L}(\theta_{i,t}^\top z_i(x_{k,t}), y_{k,t}) (\theta_{i,t} - \theta) 1_{i \in S_{k,t}}.
$$

Based on the definition of $\ell_{ik,t}$, (28) can be rewritten as

$$
\sum_{k=1}^{K} \ell_{ik,t} - \frac{1}{K} \sum_{k=1}^{K} \mathcal{L}(\theta_{i,t}^\top z_i(x_{k,t}), y_{k,t}) 1_{i \in S_{k,t}} \leq \left( \sum_{k=1}^{K} \nabla^\top \ell_{ik,t} \right) (\theta_{i,t} - \theta).
$$

According to (25), (29) is equivalent to

$$
\sum_{k=1}^{K} \ell_{ik,t} - \frac{1}{K} \sum_{k=1}^{K} \mathcal{L}(\theta_{i,t}^\top z_i(x_{k,t}), y_{k,t}) 1_{i \in S_{k,t}} \leq \left( \sum_{k=1}^{K} \nabla^\top \ell_{ik,t} \right) (\theta_{i,t} - \theta).
$$

Expectations of $\ell_{ik,t}$ and $\|\nabla \ell_{ik,t}\|^2$ with respect to $1_{i \in S_{k,t}}$, can be calculated as

$$
\mathbb{E}_t[\ell_{ik,t}] = \frac{\mathcal{L}(\theta_{i,t}^\top z_i(x_{k,t}), y_{k,t})}{p_{ik,t}} = \mathcal{L}(\theta_{i,t}^\top z_i(x_{k,t}), y_{k,t})
$$

$$
\mathbb{E}_t[\|\nabla \ell_{ik,t}\|^2] = \frac{\|\nabla \mathcal{L}(\theta_{i,t}^\top z_i(x_{k,t}), y_{k,t})\|^2}{p_{ik,t}^2} = \frac{\|\nabla \mathcal{L}(\theta_{i,t}^\top z_i(x_{k,t}), y_{k,t})\|^2}{p_{ik,t}^2}.
$$

Furthermore, using AM-GM inequality and (31b), it can be concluded that

$$
\mathbb{E}_t \left[ \sum_{k=1}^{K} \|\nabla \ell_{ik,t}\|^2 \right] \leq \mathbb{E}_t \left[ \sum_{k=1}^{K} \|\nabla \ell_{ik,t}\|^2 \right] \leq K \sum_{k=1}^{K} \frac{\|\nabla \mathcal{L}(\theta_{i,t}^\top z_i(x_{k,t}), y_{k,t})\|^2}{p_{ik,t}}.
$$
According to (33), considering the fact that

\[ \| \theta_{i,t+1} - \theta \|^2 = \left\| \frac{1}{K} \sum_{k=1}^{K} \theta_{k,t+1} - \theta \right\|^2 \]

taking the expectation from (30) with respect to \( 1_{i \in S_{i,t}} \) for all \( k \in [K] \)

\[ \leq \frac{K}{2\eta} (\| \theta_{i,t} - \theta \|^2 - \| \theta_{i,t+1} - \theta \|^2) + \eta \sum_{k=1}^{K} \frac{\| \nabla \mathcal{L}(\theta_{i,t}, z_i(x_k,t), y_k,t) \|^2}{p_{ik,t}}. \]  (33)

According to (as2), we can conclude that \( \| \nabla \mathcal{L}(\theta_{i,t}, z_i(x_k,t), y_k,t) \|^2 \leq L^2 \). Hence, summing (33) over time, given the fact that \( \theta_{i,t} = 0, \forall i \in [N] \), we get

\[ \sum_{t=1}^{T} \sum_{k=1}^{K} \mathcal{L}(\theta_{i,t}, z_i(x_k,t), y_k,t) - \sum_{t=1}^{T} \sum_{k=1}^{K} \mathcal{L}(\theta^*_i, z_i(x_k,t), y_k,t) \leq \frac{K\| \theta \|^2}{2\eta} + \eta \sum_{t=1}^{T} \sum_{k=1}^{K} \frac{L^2}{p_{ik,t}}. \]  (34)

Replacing \( \theta \) with \( \theta^*_i \) and considering the fact that \( \| \theta_{i,t+1} - \theta \|^2 \geq 0 \), we obtain

\[ \sum_{t=1}^{T} \sum_{k=1}^{K} \mathcal{L}(\theta^*_i, z_i(x_k,t), y_k,t) - \sum_{t=1}^{T} \sum_{k=1}^{K} \mathcal{L}(\hat{f}_{RF,i}(x_k,t; \theta^*_i), y_k,t) \leq \sum_{k=1}^{K} \left( \ln \frac{N}{\eta_k} + \frac{\eta_k}{2} T \right). \]  (35)

Combining (35) with (23), we arrive at

\[ \sum_{t=1}^{T} \sum_{k=1}^{K} \mathcal{L}(\hat{f}(x_k,t; \Theta_t, w_{k,t}), y_k,t) - \sum_{t=1}^{T} \sum_{k=1}^{K} \mathcal{L}(\hat{f}_{RF,i}(x_k,t; \theta^*_i), y_k,t) \leq \frac{K\| \theta \|^2}{2\eta} + \eta \sum_{t=1}^{T} \sum_{k=1}^{K} \frac{L^2}{p_{ik,t}} + \sum_{k=1}^{K} \left( \ln \frac{N}{\eta_k} + \frac{\eta_k}{2} T \right). \]  (36)

According to claim 1 in (36), it can be written that \( \sup_{x,x'} |z_i^\top(x)z_i(x') - \kappa_i(x, x')| \leq \epsilon \) holds true with probability greater than \( 1 - 2^8 (\frac{\alpha}{\epsilon})^2 \exp \left( - \frac{D_2^2}{4(\epsilon + 2)} \right) \) where \( \sigma_i \) is the second Fourier moment of the \( i \)-th kernel \( \kappa_i(\cdot) \). Furthermore, according to (as3), the loss function is \( L \)-Lipschitz continuous and as a result it can be written that

\[ \sum_{t=1}^{T} \sum_{k=1}^{K} |\mathcal{L}(\hat{f}_i(x_k,t), y_k,t) - \mathcal{L}(\hat{f}_i^*(x_k,t), y_k,t)| \]

\[ \leq \sum_{t=1}^{T} \sum_{k=1}^{K} \left( \sum_{\tau=1}^{K} \sum_{j=1}^{K} \alpha_{ij,\tau}^* z_i^\top(x_j,\tau)z_i(x_k,t) - \sum_{\tau=1}^{K} \sum_{j=1}^{K} \alpha_{ij,\tau}^* \kappa_i(x_j,\tau, x_k,t) \right). \]  (37)
Applying Cauchy-Schwartz inequality to the right hand side of (37), the left hand side of (37) can be bounded from above as
\[
\sum_{t=1}^{T} \sum_{k=1}^{K} |\mathcal{L}(\hat{f}_i^*(x_{k,t}), y_{k,t}) - \mathcal{L}(f_i^*(x_{k,t}), y_{k,t})| 
\leq \sum_{t=1}^{T} \sum_{k=1}^{K} L \sum_{\tau=1}^{K} \sum_{j=1}^{K} |\alpha_{i,k,\tau}^*| |z_i^T(x_{j,\tau})z_i(x_{k,t}) - \kappa_i(x_{j,\tau}, x_{k,t})|.
\] (38)

Let \( C := \max_{i \in [N]} \sum_{t=1}^{T} \sum_{k=1}^{K} \alpha_{i,k,t}^* \). Therefore, we can conclude that
\[
\sum_{t=1}^{T} \sum_{k=1}^{K} |\mathcal{L}(\hat{f}_i^*(x_{k,t}), y_{k,t}) - \mathcal{L}(f_i^*(x_{k,t}), y_{k,t})| \leq \epsilon L K T C
\] (39)
with probability at least \( 1 - 2^8 \left( \frac{C}{\epsilon} \right)^2 \exp \left( - \frac{D \epsilon^2}{4(d+2)} \right) \). Moreover, using Triangle inequality, we can write
\[
\sum_{t=1}^{T} \sum_{k=1}^{K} \mathcal{L}(\hat{f}_i^*(x_{k,t}), y_{k,t}) - \sum_{t=1}^{T} \sum_{k=1}^{K} \mathcal{L}(f_i^*(x_{k,t}), y_{k,t}) 
\leq \sum_{t=1}^{T} \sum_{k=1}^{K} |\mathcal{L}(\hat{f}_i^*(x_{k,t}), y_{k,t}) - \mathcal{L}(f_i^*(x_{k,t}), y_{k,t})|
\leq \sum_{t=1}^{T} \sum_{k=1}^{K} |\mathcal{L}(\hat{f}_i^*(x_{k,t}), y_{k,t}) - \mathcal{L}(f_i^*(x_{k,t}), y_{k,t})| \leq \epsilon L K T C
\] (40)
which holds true with probability at least \( 1 - 2^8 \left( \frac{C}{\epsilon} \right)^2 \exp \left( - \frac{D \epsilon^2}{4(d+2)} \right) \). Moreover, for \( z_i^T(x)z_i(x') \) we can write
\[
z_i^T(x)z_i(x') = \frac{1}{D} \sum_{j=1}^{D} (\sin(\rho_{i,j}^T x) \sin(\rho_{i,j}^T x') + \cos(\rho_{i,j}^T x) \cos(\rho_{i,j}^T x')).
\] (41)

Based on arithmetic-mean geometric-mean, (41) can be relaxed to
\[
z_i^T(x)z_i(x') \leq \frac{1}{D} \sum_{j=1}^{D} \frac{1}{2} (\sin^2(\rho_{i,j}^T x) + \sin^2(\rho_{i,j}^T x') + \cos^2(\rho_{i,j}^T x) + \cos^2(\rho_{i,j}^T x')) = 1.
\] (42)
Thus, given the fact that \(|z_i^T(x)z_i(x')| \leq 1\), \( \|\theta_i^*\|^2 \) can be bounded as
\[
\|\theta_i^*\|^2 \leq \sum_{t=1}^{T} \sum_{k=1}^{K} \sum_{\tau=1}^{K} \sum_{j=1}^{K} |\alpha_{i,k,\tau}^*| \alpha_{i,j,\tau}^* z_i^T(x_{j,\tau})z_i(x_{k,t}) \leq C^2.
\] (43)

Combining (40) and (43) with (36) yields
\[
\sum_{t=1}^{T} \sum_{k=1}^{K} \mathcal{L}(\hat{f}(x_{k,t}; \hat{\Theta}_t, w_{k,t}), y_{k,t}) - \sum_{t=1}^{T} \sum_{k=1}^{K} \mathcal{L}(f_i^*(x_{k,t}), y_{k,t}) 
\leq \frac{KC^2}{2 \eta} + \frac{\eta}{2} \sum_{t=1}^{T} \sum_{k=1}^{K} \frac{L^2}{p_{k,t}} + \sum_{k=1}^{K} \left( \frac{\ln N}{\eta_k} + \frac{\eta_k}{2} T \right) + \epsilon L K T C
\] (44)
which holds true for any \( i \in [N] \) with probability at least \( 1 - 2^8 \left( \frac{C}{\epsilon} \right)^2 \exp \left( - \frac{D \epsilon^2}{4(d+2)} \right) \). Therefore, this proves the Theorem 2.
Table 2: MSE($\times 10^{-3}$) and standard deviation($\times 10^{-3}$) of online federated learning algorithms on real datasets.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>$M$</th>
<th>$D$</th>
<th>Naval</th>
<th>UJI</th>
<th>Air</th>
<th>WEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFSKL</td>
<td>1</td>
<td>100</td>
<td>77.77 ± 1.04</td>
<td>61.82 ± 2.76</td>
<td>13.65 ± 0.61</td>
<td>87.87 ± 3.93</td>
</tr>
<tr>
<td>OFMKL-Avg</td>
<td>51</td>
<td>9</td>
<td>33.25 ± 1.46</td>
<td>55.44 ± 2.48</td>
<td>10.63 ± 0.47</td>
<td>34.01 ± 1.52</td>
</tr>
<tr>
<td>vM-KOFL</td>
<td>51</td>
<td>9</td>
<td>26.42 ± 1.16</td>
<td>51.50 ± 2.30</td>
<td>10.58 ± 0.47</td>
<td>25.17 ± 1.12</td>
</tr>
<tr>
<td>eM-KOFL</td>
<td>1</td>
<td>100</td>
<td>28.64 ± 1.32</td>
<td>61.08 ± 2.73</td>
<td>21.94 ± 1.16</td>
<td>20.14 ± 0.93</td>
</tr>
<tr>
<td>POF-MKL</td>
<td>1</td>
<td>100</td>
<td>16.16 ± 0.72</td>
<td>33.02 ± 1.48</td>
<td>9.27 ± 0.41</td>
<td>11.44 ± 0.52</td>
</tr>
<tr>
<td>POF-MKL</td>
<td>25</td>
<td>20</td>
<td>16.82 ± 0.74</td>
<td>37.34 ± 1.67</td>
<td>9.34 ± 0.42</td>
<td>11.58 ± 0.53</td>
</tr>
<tr>
<td>POF-MKL</td>
<td>51</td>
<td>9</td>
<td>16.65 ± 0.74</td>
<td>41.00 ± 1.83</td>
<td>9.38 ± 0.42</td>
<td>11.97 ± 0.55</td>
</tr>
</tbody>
</table>

C Supplementary Experimental Results and Details

This section presents further experimental results testing different aspects of the proposed algorithm POF-MKL. Moreover, this section provides more detailed information about experimental setup associated with results in section 3. The performance of federated kernel learning algorithms are tested on the following datasets:

- **Naval**: The dataset consists of 11,500 samples. Each sample has 15 features of a naval vessel. The goal is to predict lever position [7].
- **UJI**: The dataset consists of 21,000 data samples. Each data sample has 520 features which are WiFi fingerprints. The goal is to predict the geographical longitude associated with each data sample.
- **Air**: The dataset consists of 120,000 samples with 14 features including information related to air quality such as concentration of some chemicals in the air. Data samples are collected from 4 different geographical sites. The goal is to predict the concentration of CO in the air [51]. For each site there are 30,000 samples in the dataset.
- **WEC**: The dataset consists of 280,000 samples with 48 features of wave energy converters. Data samples are collected from 4 different geographical sites. The goal is to predict total power output [35]. For each site, there are 70,000 samples.

Data samples of Naval and UJI datasets are distributed i.i.d among clients. The number of clients for Naval and UJI datasets are 23 and 42, respectively. Data samples in Air and WEC datasets are distributed non-i.i.d among clients. The number of clients for Air and WEC datasets are 240 and 560, respectively. For both Air and WEC datasets, there are 4 different geographical sites that each sample belongs to one of them. Each client observes 350 samples from one site and 50 samples from each of the rest of 3 sites. Moreover, PerFedAvg uses a feedforward neural network model. Each layer is a fully-connected dense layer with at most 20 neurons. Neurons in hidden layers exploit ReLU activation functions. Since each client cannot transmit more than 1000 parameters to the server, the number of hidden layers is determined in a way that the number of the neural network’s parameters to be less than 1000. The number of parameters depends on the number of features in data samples. Therefore, the number of hidden layers varies across different datasets. For each dataset, given the number of features, the maximum number of hidden layers with 20 neurons is chosen. All experiments were carried out using Intel(R) Core(TM) i7-10510U CPU @ 1.80 GHz 2.30 GHz processor with a 64-bit Windows operating system.

Table 2 presents average MSE along with MSE standard deviation calculated over 20 different sets of random feature vectors. As it can be seen from Table 2 the proposed POF-MKL provides lower standard deviation compared to all other baselines. This shows that the proposed POF-MKL is less sensitive to the choice of random features. Furthermore, Table 2 reports the average cumulative regret of clients along with the standard deviation of regret among clients. As it can be inferred from Table 3, the proposed POF-MKL obtains lower regret than other online federated MKL algorithms. Moreover, for Air and WEC datasets, the standard deviation of regret among clients associated with POF-MKL is considerably lower than those of other online federated MKL algorithms. Note that data samples in Air and WEC datasets are distributed non-i.i.d among clients. Therefore, the results in Table 3 confirm that the proposed POF-MKL can better deal with heterogeneous data among clients.
Table 3: Average regret and its standard deviation across clients for online federated MKL learning algorithms.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>$M$</th>
<th>$D$</th>
<th>Naval</th>
<th>UJI</th>
<th>Air</th>
<th>WEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFMKL-Avg</td>
<td>51</td>
<td>9</td>
<td>16.95 ± 0.39</td>
<td>24.23 ± 20.33</td>
<td>3.05 ± 2.83</td>
<td>17.07 ± 14.52</td>
</tr>
<tr>
<td>vM-KOFL</td>
<td>51</td>
<td>9</td>
<td>13.40 ± 0.39</td>
<td>22.28 ± 15.56</td>
<td>3.02 ± 2.79</td>
<td>12.65 ± 11.53</td>
</tr>
<tr>
<td>eM-KOFL</td>
<td>1</td>
<td>100</td>
<td>14.92 ± 0.65</td>
<td>27.26 ± 19.79</td>
<td>8.53 ± 2.51</td>
<td>10.43 ± 8.28</td>
</tr>
<tr>
<td>POF-MKL</td>
<td>1</td>
<td>100</td>
<td>8.33 ± 0.39</td>
<td>13.23 ± 8.80</td>
<td>2.95 ± 2.08</td>
<td>6.37 ± 5.28</td>
</tr>
<tr>
<td>POF-MKL</td>
<td>25</td>
<td>20</td>
<td>8.67 ± 0.39</td>
<td>15.41 ± 9.58</td>
<td>2.98 ± 2.12</td>
<td>6.41 ± 5.39</td>
</tr>
<tr>
<td>POF-MKL</td>
<td>51</td>
<td>9</td>
<td>8.55 ± 0.39</td>
<td>17.23 ± 10.07</td>
<td>3.01 ± 2.15</td>
<td>6.51 ± 5.57</td>
</tr>
</tbody>
</table>

Figure 2: Average cumulative regret of clients with the change in the value of exploration rate ($\xi_k$).

Figure 2 illustrates the average regret of clients employing POF-MKL with the change in the value of exploration rate $\xi_k$ when the exploration rate of all clients are the same. In particular, Figure 2 depicts the performance of POF-MKL for $M = 1$ and $M = 25$ with the change in $\xi_k$. According to the PMF $q_{k,t}$ defined in (8), the increase in $\xi_k$ leads to an increase in exploration such that if $\xi_k = 1$, the $k$-th client chooses a subset of kernels uniformly at random. Figure 2 indicates that the optimal choice of $\xi_k$ in terms of regret depends on the dataset distributed among clients as well as the number of chosen kernels $M$. Moreover, the choice of $\xi_k$ is related to the computational complexity of executing POF-MKL by clients. Specifically, when $\xi_k < 1$, in order to choose a subset of kernels, the $k$-th client needs to sort kernels which imposes worst case computational complexity of $\mathcal{O}(N \log N)$. However, when $\xi_k = 1$, according to PMF in (8), the $k$-th client chooses one bin uniformly at random and as a result in this case the $k$-th client does not need to sort kernels. Also, it is useful to note that as it can be inferred from (9), clients can leverage the exploration rate $\xi_k$ to send their updates to the server without revealing both the gradient of loss and the loss of kernels which can promote the privacy of the proposed POF-MKL.