An Online Multiple Kernel Parallelizable Learning Scheme

Emilio Ruiz-Moreno and Baltasar Beferull-Lozano

Abstract — **The performance of reproducing kernel Hilbert space-based methods is known to be sensitive to the choice of the reproducing kernel. Choosing an adequate reproducing kernel can be challenging and computationally demanding, especially in data-rich tasks without prior information about the solution domain. In this paper, we propose a learning scheme that scalably combines several single kernel-based online methods to reduce the kernel-selection bias. The proposed learning scheme applies to any task formulated as a regularized empirical risk minimization convex problem. More specifically, our learning scheme is based on a multi-kernel learning formulation that can be applied to widen any single-kernel solution space, thus increasing the possibility of finding higher-performance solutions. In addition, it is parallelizable, allowing for the distribution of the computational load across different computing units. We show experimentally that the proposed learning scheme outperforms the combined single-kernel online methods separately in terms of the cumulative regularized least squares cost metric.**

B.1 Introduction

Reproducing kernel Hilbert space (RKHS)-based methods allow modeling highly non-linear relationships at a moderate computational cost [60]. Thanks to their simplicity and generality, they have been successfully adopted in a wide range of signal-processing applications [90, 91].

The performance of any RKHS-based method strongly relies on a preselected reproducing kernel (RK). The efficient selection of an adequate RK presumes some task-specific prior information, such as knowledge about the data domain, invariant data transformations, geometrical data structures, or some properties of the underlying data generating process [92]. For example, spline interpolation RKs are best suited for smooth data [24, 78]. Similarly, radial basis RKs can perform poorly if their associated hyperparameters are not properly tuned to the task. The kernel-selection issue cannot be easily mitigated via cross-validation [93] because the associated computational load grows prohibitively with the number of RKs. On the other hand, efficiently computable approximations of the leave-one-out error [61] or hyperparameter optimization techniques [94] usually involve non-convexity and may lead to undesirable local minima.

Multi-kernel methods compensate for the lack of task-specific prior information using a predefined set of RKs known as *dictionary*. The dictionary can be formed by integrating different types of RKs, the same RK with different hyperparameter values, or a mix of both. Typically, the preselected RK is constructed as a combination of several RKs from the dictionary [62]. Therefore, how the dictionary is formed and how the preselected RK is constructed have a pivotal impact on the resulting accuracy and complexity of the method. For instance, the larger the dictionary

is, the more likely it is to reduce the kernel-selection bias compared to a particular RK or hyperparameter choice. In addition, larger dictionaries allow for greater adaptability when learning from data samples since, in practice, these samples may come from a combination of different sources. On the other hand, increasing the dictionary size becomes computationally demanding or even prohibitive. For this reason, a commonly sought goal for multi-kernel methods is to find a compromise between performance and a computationally light and compact representation of the proposed solution in terms of the dictionary elements [95].

Another related scaling issue that also applies to single-kernel methods is the curse of kernelization, i.e., potentially unbounded linear growth in model size with the amount of data [82]. This drawback is generally addressed through online approaches, which may rely on sparsification procedures [96, 97, 98, 99, 100] or dimensionality reduction approximations [101, 102], among others [103].

In this context, some works [104, 105, 106] have explored the use of online methods for determining the optimal solution associated with each single RK within the dictionary, as well as the best combination of these single-kernel solutions under a given task. Following a conceptually similar approach, this paper proposes a novel multi-kernel learning scheme that can be parallelized across RKs by efficiently combining the solution of several single RK-based online methods running concurrently. This provides scalability with respect to the number of data samples and adaptability across different data patterns. Moreover, it allows to distribute the computational load across different computing units as the dictionary size increases. Our proposed scheme applies to any task that can be formulated as a regularized empirical risk minimization (RERM) convex problem [39, 107]. Finally, the performance of the proposed learning scheme is experimentally validated in terms of the cumulative regularized least squares cost metric.

B.2 Problem formulation

Supervised learning is arguably one of the core topics in machine learning [108]. Many supervised learning tasks can be formulated as RERM convex problems whose solution admits a kernel representation. That is, given a set of N data samples $\mathcal{S} = \{(x^{(n)}, y^{(n)})\}_{n=1}^{N} \subseteq \mathcal{X} \times \mathcal{Y}$, and an RKHS[1] $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$, the goal is to find a function estimate $f \in \mathcal{H}$ minimizing the following regularized functional cost

$$\mathcal{C}_\eta\left(f; \mathcal{S}\right) = \sum_{n=1}^{N} \ell\left(f(x^{(n)}), y^{(n)}\right) + \frac{\eta}{2}\|f\|_{\mathcal{H}}^2, \tag{B.1}$$

where the loss $\ell : \mathcal{Y}^2 \to \mathbb{R} \cup \{\infty\}$ is a proper convex function used as a goodness-of-fit metric, the regularizer $\|\cdot\|_{\mathcal{H}}^2 : \mathcal{H} \to \mathbb{R}$ is the squared RKHS $\mathcal{H}$ induced norm, and the hyperparameter $\eta \in \mathbb{R}_+$ controls the model complexity of the solution.

Under a multi-kernel learning framework, one typically constructs a valid RK by adequately combining the RKs within a preselected dictionary [62]. Particularly, finding the RK within a convex hull of P positive definite RKs that yields the function estimate incurring the lowest functional cost (B.1) is equivalent to obtaining a solution from $\mathcal{H}$ built as the RKHS direct sum $\mathcal{H}_1 \oplus \cdots \oplus \mathcal{H}_P$, where each pth

[1]The notation $\mathcal{Y}^{\mathcal{X}}$ refers to the set of functions from $\mathcal{X}$ to $\mathcal{Y}$.

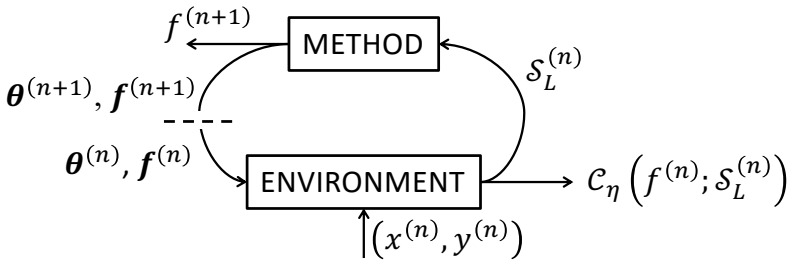

Figure B.1: Visual description of the online setting considered.

RKHS $\mathcal{H}_p = \overline{\text{span}}\{k_p(x, \cdot) : x \in \mathcal{X}\}$, being $k_p : \mathcal{X}^2 \to \mathbb{R}$ its associated RK [109]. The solution $f \in \mathcal{H}$ that minimizes (B.1), can be expressed without loss of generality as $f = \boldsymbol{\theta}^\top \boldsymbol{f}$, where $\boldsymbol{f} = [f_1, \ldots, f_P]^\top \in \mathcal{H}_{1:P}$, with $\mathcal{H}_{1:P} \triangleq \mathcal{H}_1 \times \cdots \times \mathcal{H}_P$ and $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_P]^\top \in \Delta^{P-1}$, with $\Delta^{P-1} \triangleq \{\boldsymbol{\beta} \in \mathbb{R}^P : \boldsymbol{\beta} \succeq \mathbf{0}$ and $\mathbf{1}^\top \boldsymbol{\beta} = 1\}$ denoting a simplex [110]. Thus, the RERM problem posed before becomes

$$\min_{\boldsymbol{\theta} \in \Delta^{P-1}, \boldsymbol{f} \in \mathcal{H}_{1:P}} \mathcal{C}_\eta\left(f; \mathcal{S}\right) \tag{B.2a}$$

$$\text{subject to: } f = \boldsymbol{\theta}^\top \boldsymbol{f}. \tag{B.2b}$$

Optimization problem (B.2) is bi-convex, meaning that it is convex in $\boldsymbol{\theta}$ for a fixed $\boldsymbol{f}$ and vice-versa. Still, it is not jointly convex in both optimization variables. It can be tackled via specialized methods that primarily exploit the convex substructures of the problem [111]. However, these methods do not scale well with the number of RKs and data samples, denoted as P and N, respectively.

This paper presents a method to solve efficiently (B.2) for large P and N.

B.3 Proposed solution

This section describes how to synergize an online formulation and an upper bound on the objective (B.2a) to solve (B.2) scalably with respect to N and P.

B.3.1 Online setting

Online settings [63, 64] can be adopted to solve (B.2) achieving low run-time complexity with respect to N while incurring a certain tolerable (cumulative) cost. They usually trade-off solution accuracy for speed, e.g., by processing only a few samples every iteration, for low memory complexity, e.g., by discarding samples after a few processing steps, or for model complexity control, i.e., bounded model size regardless of whether N increases.

The online setting considered in this work can be cast as a method-environment iterative game [112]. The data samples in $\mathcal{S}$ are assumed to be available sequentially. Then, at each iteration step n, the method chooses a function estimate $f^{(n)} \in \mathcal{H}_1 \oplus \cdots \oplus \mathcal{H}_P$ expressible as $f^{(n)} = {\boldsymbol{\theta}^{(n)}}^\top \boldsymbol{f}^{(n)}$ with $\boldsymbol{\theta}^{(n)} \in \Delta^{P-1}$ and $\boldsymbol{f}^{(n)} \in \mathcal{H}_{1:P}$. In response, the environment penalizes the proposed function estimate $f^{(n)}$ with the incurred cost $\mathcal{C}_\eta(f^{(n)}; \mathcal{S}_L^{(n)})$, where $\mathcal{S}_L^{(n)} = \{(x^{(i)}, y^{(i)})\}_{i=n_L}^{n} \subseteq \mathcal{S}$ is a sliding window of L data samples and $n_L \triangleq \max\{n - L + 1, 1\}$. Finally, once the nth function

Algorithm 3 (Quadratic program) Projection onto simplex.

Input: The function estimate components $\{f_p^{(n)}\}_{p=1}^P$, the data window $\mathcal{S}_L^{(n-1)}$ and the loss ℓ.

1: Set the components of $\boldsymbol{a}$ and $\boldsymbol{b}$ as in (B.5).

2: Sort $\boldsymbol{b}$ in ascending order, denoted as $\boldsymbol{u}$: $u_1 \leq u_2 \leq \cdots \leq u_P$ and rearrange $\boldsymbol{a}$ accordingly into $\boldsymbol{v}$.

3: Find $\rho = \max\left\{1 \leq j \leq P : u_j - \frac{\sum_{i=1}^{j} \frac{u_i}{v_i} + 2}{\sum_{i=1}^{j} \frac{1}{v_i}} < 0\right\}$.

4: Define $\mu = -\frac{2 + \sum_{i=1}^{\rho} \frac{u_i}{v_i}}{\sum_{i=1}^{\rho} \frac{1}{v_i}}$.

5: Compute $\theta_p = \max\left\{-\frac{1}{2a_p}(b_p + \mu), 0\right\}$ for $p = 1, \ldots, P$.

Output: $\boldsymbol{\theta}$.

estimate $f^{(n)}$ is chosen, the method receives the nth data window $\mathcal{S}_L^{(n)}$, which can be used at the next iteration[2] step $n+1$. Fig. B.1 visually describes the procedure.

B.3.2 Upper bound on the functional cost

As we show next, we can improve scalability with respect to the number of RKs, by making use of the following upper bound:

$$\mathcal{C}_\eta(f; \mathcal{S}) = \sum_{n=1}^{N} \ell\left(\boldsymbol{\theta}^\top \boldsymbol{f}(x^{(n)}), y^{(n)}\right) + \frac{\eta}{2}\|\boldsymbol{\theta}^\top \boldsymbol{f}\|_{\mathcal{H}}^2 \tag{B.3a}$$

$$\leq \sum_{n=1}^{N} \sum_{p=1}^{P} \theta_p \ell\left(f_p(x^{(n)}), y^{(n)}\right) + \theta_p^2 \frac{\eta}{2}\|f_p\|_{\mathcal{H}_p}^2 \tag{B.3b}$$

$$\triangleq \breve{\mathcal{C}}_\eta(\boldsymbol{\theta}, \boldsymbol{f}; \mathcal{S}). \tag{B.3c}$$

The first upper-bounded term in (B.3b) follows directly from Jensen's inequality [113], whereas the second term is obtained by invoking the definition of the RKHS direct sum norm [114]. Even though the second term in (B.3b) could have also been upper bounded through Jensen's inequality, as in [110], exploiting the definition of the RKHS direct sum norm, leads to a tighter bound because $\theta_p^2 \leq \theta_p$ for $\theta_p \in [0, 1]$.

The key advantage of the upper bound cost (B.3c) is that it is separable across the P RKs within the dictionary, hence allowing for parallelization at the expense of some loss in optimality, albeit with still satisfactory performance.

B.3.3 Parallelizable learning scheme

Our proposed learning scheme consists of executing at each iteration step n the following consecutive operations:

1) Every pth function estimate component $f_p^{(n)}$ is chosen through a single-kernel online method operating over the pth RK within the dictionary. The methods are selected by the user, and they can be different for each RK as long as all of

[2] At the first iteration step $n = 1$, the method has not received any data sample, thus $f^{(1)}$ is set as some arbitrary initial function estimate.

them adopt the online setting described in Sec. B.3.1. For example, stochastic gradient descent methods for function estimation [12], and associated variants [81, 1], can be readily used. Since the function estimate components of $\boldsymbol{f}^{(n)}$ can be computed in parallel across P different computing units, the computational cost can be distributed.

2) Next, the convex weights in $\boldsymbol{\theta}^{(n)}$ are chosen as the ones that minimize the partially evaluated upper bound cost (B.3c) at $\boldsymbol{f}^{(n)}$ and $\mathcal{S}_L^{(n-1)}$, referred from now on as learning cost. Mathematically,

$$\boldsymbol{\theta}^{(n)} = \arg\min_{\boldsymbol{\theta}\in\Delta^{P-1}} \breve{\mathcal{C}}_\eta\left(\boldsymbol{\theta}, \boldsymbol{f}^{(n)}; \mathcal{S}_L^{(n-1)}\right) \tag{B.4a}$$

$$= \arg\min_{\boldsymbol{\theta}\in\Delta^{P-1}} \boldsymbol{\theta}^\top \boldsymbol{A}^{(n)}\boldsymbol{\theta} + \boldsymbol{b}^{(n)\top}\boldsymbol{\theta}, \tag{B.4b}$$

where $\boldsymbol{A}^{(n)} \triangleq \operatorname{diag}(\boldsymbol{a}^{(n)}) \in \boldsymbol{S}_{++}^P$, with $\boldsymbol{a}^{(n)} \in \mathbb{R}_+^P$, and $\boldsymbol{b}^{(n)} \in \mathbb{R}^P$ whose components are computed[3] as

$$a_p^{(n)} = \frac{\eta}{2}\left\|f_p^{(n)}\right\|_{\mathcal{H}_p}^2, \tag{B.5a}$$

$$b_p^{(n)} = \sum_{i\in\mathcal{I}_L^{(n-1)}} \ell\left(f_p^{(n)}(x^{(i)}), y^{(i)}\right), \tag{B.5b}$$

for all $p \in \mathbb{N}^{[1,P]}$, where $\mathcal{I}_L^{(n-1)}$ corresponds to the index set associated with the data samples in $\mathcal{S}_L^{(n-1)}$. We adapt the projection onto the simplex algorithm discussed in [115, 116, 117] by extending its applicability to any quadratic problem described by a diagonal positive definite matrix with simplex constraints. As a result, the proposed **Algorithm 3** can solve (B.4) exactly. Its computational complexity is bottlenecked by a sorting step; that is, an asymptotic average complexity $\mathcal{O}(P\log P)$ [118]. It should be mentioned that this complexity can be further reduced to $\mathcal{O}(P)$ on average by using a randomized pivot algorithm variation that identifies the parameter ρ (**Algorithm 3**, line 3) using a divide and conquer procedure instead of sorting [119], but this is out of the scope of the present paper.

3) Finally, the function estimate $f^{(n)} = \boldsymbol{\theta}^{(n)\top}\boldsymbol{f}^{(n)}$ is proposed.

In summary, our scheme can be seen as a higher-level learner that iteratively chooses the lowest incurring learning cost combination of function estimates provided by lower-level learners, namely, single RK methods.

B.4 Performance analysis

Under an online setting, as the one described in Sec. B.3.1, the incurred cost accumulated over time receives the name of *cumulative* cost (CC). In our case, the CC up to the nth time step is given by $\sum_{i=1}^n \mathcal{C}_\eta(f^{(i)}; \mathcal{S}_L^{(i)})$. From here, recall that every ith function estimate $f^{(i)}$ is proposed via Sec. B.3.3, before the ith data window $\mathcal{S}_L^{(i)}$ becomes available; thus, the CC is a measure of performance protecting against overfitting. Intuitively, the lower the growth of the incurred CC with respect to n,

[3]Notice that nothing prevents $f_p^{(n)}$ to be zero-valued and thus $\boldsymbol{A}^{(n)}$ from being singular and positive semi-definite. However, we can always set $\boldsymbol{A}^{(n)} = \operatorname{diag}(\boldsymbol{a}^{(n)}) + \delta\boldsymbol{I}_P$ where δ is an arbitrarily small positive value.

the better the expected performance over unseen data. In fact, popular measures of performance, such as the dynamic regret, are constructed as the difference between the CC incurred by the sequence of function estimates proposed by a method and a sequence of comparators [66].

In order to validate our scheme experimentally, we pose a signal reconstruction online problem from synthetically generated streaming data. Specifically, we use the squared loss, i.e., $\ell(f(x), y) = (f(x) - y)^2$, and a dictionary of $P = 20$ Gaussian kernels, i.e., $k_p(x, t) = \exp\left(-\frac{1}{2}(x-t)^2/\sigma_p^2\right)$, with different widths σ_p linearly spaced between 0.1 and 10. The method associated with each RK is an augmented naive online R_{reg} minimization algorithm (NORMA) [12] with a window length of $L = 10$ data samples, a budget of $\tau = 100$ kernel expansion terms (beyond the allowed budget we truncate the oldest terms of the kernel expansion), and a fixed learning rate $\lambda_{\text{NORMA}} = 0.05$. That is, before any possible truncation, each nth function estimate associated with the pth RK is constructed as $f_p^{(n)} = \sum_{i=1}^{n-1} \alpha_{p,i}^{(n)} \, k_p(x^{(i)}, \cdot)$, where each $\alpha_{p,i}^{(n)} \in \mathbb{R}$ denotes a kernel-expansion coefficient obtained from the following NORMA update:

$$f_p^{(n)} = f_p^{(n-1)} - \lambda_{\text{NORMA}} \, \partial_f \, \mathcal{C}_\eta \left(f; \mathcal{S}_L^{(n-1)} \right) \Big|_{f = f_p^{(n-1)}}, \tag{B.6}$$

which, after some algebraic steps, leads to the next closed-form update rule:

$$\alpha_i^{(n)} = \begin{cases} -\lambda_{\text{NORMA}} \, \ell_{p,i}'^{(n-1)} & \text{if } i = n-1, \\ \gamma \alpha_i^{(n-1)} - \lambda_{\text{NORMA}} \, \ell_{p,i}'^{(n-1)} & \text{if } i \in \mathcal{I}_L^{(n-1)} \backslash \{n-1\}, \\ \gamma \alpha_i^{(n-1)} & \text{otherwise}, \end{cases} \tag{B.7}$$

where $\ell_{p,i}'^{(n)} \triangleq \ell'\left(f_p^{(n)}(x^{(i)}), y^{(i)} \right) = 2\left(f^{(n)}(x^{(i)}) - y^{(i)} \right)$ and $\gamma \triangleq (1 - \lambda_{\text{NORMA}} \eta) \in \mathbb{R}_+^{(0,1)}$. The regularization parameter is chosen as $\eta = 0.01$. Lastly, the data samples have been generated via a stable AR(1) process $y^{(n)} = \varphi y^{(n-1)} + u^{(n)}$, with $\varphi = 0.5488135$, $u^{(n)} \overset{\text{iid}}{\sim} \mathcal{N}(0, 0.71519837)$, $y^{(0)} = 0$ and unit time stamps uniformly arranged in time, i.e., $x^{(n)} = n$.

Additionally, we compare our scheme with the online multiple kernel regression (OMKR) algorithm [104], arguably the closest approach conceptually. More specifically, we compare against the budget OMKR gradient-based variant method over the same experimental setting described above. Briefly, the considered OMKR method can be described, at each iteration step n, by the following three-stage scheme:

1) The set of function estimates, in this case, regressors proposed by each one of the P NORMAs, is updated as (B.7) and collected in $\boldsymbol{f}^{(n)} \in \mathcal{H}_{1:P}$.

2) Then, the P weights for combining the multiple regressors are updated as

$$\boldsymbol{w}^{(n)} = \boldsymbol{w}^{(n-1)} - \lambda_{\text{OMKR}}^{(n)} \, \nabla_{\boldsymbol{w}} \, \mathcal{C}_\eta \left(\boldsymbol{w}^\top \boldsymbol{f}^{(n)}; \mathcal{S}_L^{(n-1)} \right) \Big|_{\boldsymbol{w} = \boldsymbol{w}^{(n-1)}}. \tag{B.8}$$

In this case, we use an initial learning rate $\lambda_{\text{OMKR}}^{(1)} = 8 \cdot 10^{-4}$ that is halved every 50 steps until a minimum value of 10^{-5}. After some algebraic manipulations and making use of the definition of the RKHS direct sum norm [114], the evaluated gradient in (B.8) equals to

$$\sum_{i \in \mathcal{I}_L^{(n-1)}} \boldsymbol{f}^{(n)}(x^{(i)}) \ell'\left(\boldsymbol{w}^{(n-1)^\top} \boldsymbol{f}^{(n)}(x^{(i)}), y^{(i)} \right) + \eta \boldsymbol{A}^{(n)} \boldsymbol{w}^{(n-1)}. \tag{B.9}$$

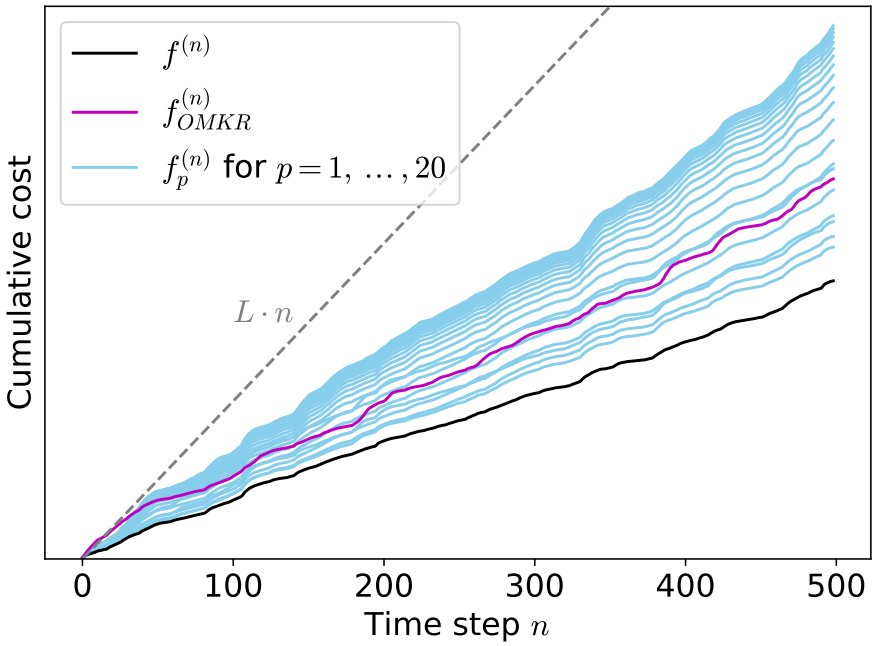

Figure B.2: Cumulative cost up to time step n incurred by our learning scheme, the OMKR algorithm, and the combined single RK NORMA regressors individually. The dashed-dotted line $L \cdot n$ is shown as a reference.

The matrix $\boldsymbol{A}^{(n)}$ corresponds to the one introduced in (B.5a), and the initial combination weights are set as $\boldsymbol{w}^{(1)} = \boldsymbol{0}_P$.

3) Finally, the function estimate $f^{(n)} = \boldsymbol{w}^{(n)\top} \boldsymbol{f}^{(n)}$ is proposed.

Unlike our scheme, the OMKR algorithm can eventually learn any linear combination of single-kernel function estimates. However, due to the additive nature of the update step in (B.8), the OMKR algorithm usually suffers from slow convergence rates. Moreover, it requires a sequence of learning rates $\lambda^{(n)}_{\text{OMKR}}$ whose tuning involves optimization techniques or task-specific knowledge, hence adversely affecting performance, e.g., poor learning or instabilities, if not carried out adequately.

Our experimental results in Fig. B.2 show that the CC incurred by our proposed learning scheme outperforms the lowest CC incurred by any of the combined single RK NORMA regressors separately. In the same figure, it can also be observed that our scheme incurs a CC that increases at a lower rate than the one incurred by the OMKR algorithm, thus allowing our scheme to outperform the best NORMA regressor much sooner. The reason behind this observation is arguably the additive nature of the OMKR algorithm, which requires numerous updates to completely remove the residual contributions of irrelevant regressors. For example, see in Fig. B.3 the "spiky" shape of the OMKR signal estimate due to some regressors constructed with a narrow-valued σ_p RK.

Regarding computational resources, both the OMKR algorithm and our scheme can be parallelized across the RKs within the dictionary, which in our experiments means a constant time complexity $\mathcal{O}(\tau)$, and a combination step of complexity $\mathcal{O}(P)$ and $\mathcal{O}(P \log P)$, respectively. However, as mentioned in Sec. B.3.3, the complexity of our scheme can be further reduced to $\mathcal{O}(P)$ on average, making both of the compared approaches computationally equivalent.

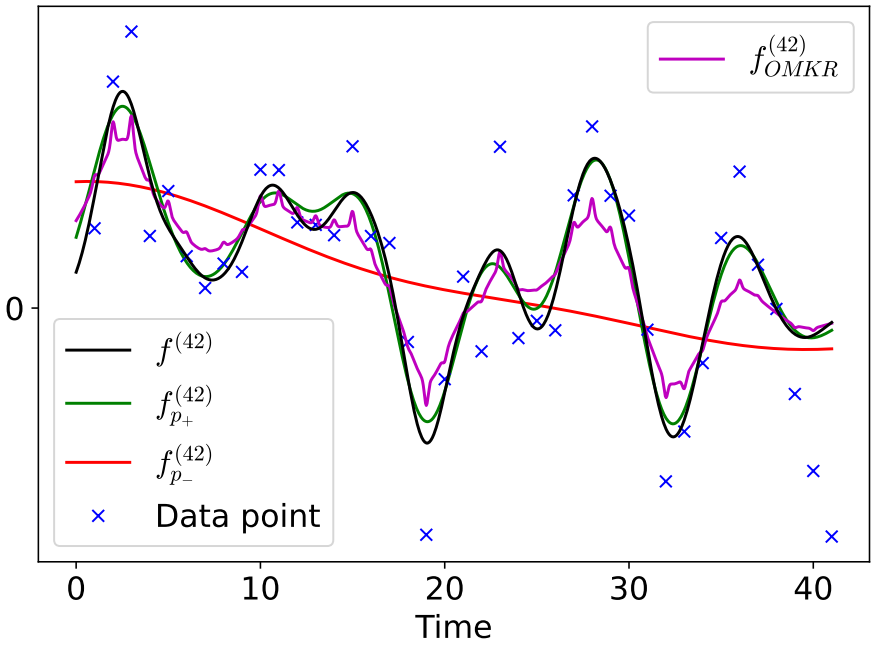

Figure B.3: Snapshot of the 42nd signal estimate obtained by the OMKR algorithm, our learning scheme, and the best and the worst of the combined single RK NORMA regressors (denoted by the indices p_+ and p_-, respectively) in terms of the so-far ($n = 42$) incurred cumulative cost.

B.5 Conclusion

We present a multi-kernel learning scheme that experimentally outperforms the best of the combined single RK methods, in terms of the cumulative regularized least squares cost metric, with a comparable computational load per computing unit. This corroborates the ability of the proposed scheme to effectively accommodate a larger function space (from which to draw function estimates) of multi-kernel methods while keeping the lower computational complexity of online single RK methods. Furthermore, although **Algorithm 3** has been expressly designed for the task discussed in this paper, it can be used to solve any other problem that accepts a formulation as in (B.4b).

Correctness of Algorithm 1

A Lagrangian of problem (B.4) is

$$\mathrm{L}(\boldsymbol{\theta}, \boldsymbol{\lambda}, \mu) = \boldsymbol{\theta}^\top \mathrm{diag}(\boldsymbol{a}^{(n)})\boldsymbol{\theta} + \boldsymbol{b}^{(n)\top}\boldsymbol{\theta} - \boldsymbol{\lambda}^\top\boldsymbol{\theta} + \mu(\mathbf{1}^\top\boldsymbol{\theta} - 1), \tag{B.10}$$

being $\mu \in \mathbb{R}$ and $\boldsymbol{\lambda} \in \mathbb{R}^P$ the Lagrange multipliers associated with the equality and inequality constraints, respectively. At the optimal solution $\boldsymbol{\theta}^{(n)}$, the following KKT conditions [120] hold:

$$2\theta_p^{(n)} a_p^{(n)} + b_p^{(n)} - \lambda_p + \mu = 0, \quad p = 1, \ldots, P \tag{B.11a}$$

$$\theta_p^{(n)} \geq 0, \quad p = 1, \ldots, P \tag{B.11b}$$

$$\lambda_p \geq 0, \quad p = 1, \ldots, P \tag{B.11c}$$

$$\lambda_p \theta_p^{(n)} = 0, \quad p = 1, \ldots, P \tag{B.11d}$$

$$\sum_{p=1}^{P} \theta_p^{(n)} = 1. \tag{B.11e}$$

From the complementary slackness, stated in (B.11d), we can deduce that if the primal inequality constraint in (B.11b) is slacked, i.e., greater than zero, then $\lambda_p = 0$ and from the stationarity condition (B.11a), the solution fulfils

$$\theta_p^{(n)} = -\frac{1}{2a_p^{(n)}}(b_p^{(n)} + \mu) > 0. \tag{B.12}$$

On the other hand, if the primal inequality constraint is tight, i.e., $\theta_p^{(n)} = 0$, then the dual constraint (B.11c) is not binding. Again, from the stationarity condition in (B.11a), we can identify those non-binding constraints as those that satisfy the following expression:

$$b_p^{(n)} + \mu = \lambda_p \geq 0. \tag{B.13}$$

In this way, it is clear from (B.13) that the components of the optimal solution that are zero, if any, correspond to the larger components of $\boldsymbol{b}^{(n)}$. Without loss of generality, we can assume that the components of $\boldsymbol{b}^{(n)}$ are sorted in ascending order as long as the components of $\boldsymbol{a}^{(n)}$ are rearranged accordingly. Thus, by comparing $\boldsymbol{b}^{(n)}$ with the solution as follows:

$$\begin{aligned} b_1^{(n)} \leq \cdots \leq b_\rho^{(n)} \leq & b_{\rho+1}^{(n)} \leq \cdots \leq b_P^{(n)}, \\ & \theta_{\rho+1}^{(n)} = \cdots = \theta_P^{(n)} = 0, \end{aligned} \tag{B.14}$$

it can be concluded that the index $\rho \in \mathbb{N}^{[1,P]}$ determines the number of components in the solution that are nonzero. From here, and rewriting the equality primal constraint (B.11e) as

$$\sum_{p=1}^{P} \theta_p^{(n)} = \sum_{p=1}^{\rho} \theta_p^{(n)} = -\frac{1}{2} \sum_{p=1}^{\rho} \frac{1}{a_p^{(n)}}(b_p^{(n)} + \mu) = 1, \tag{B.15}$$

the Lagrangian multiplier associated with the equality constraint can be isolated and computed as

$$\mu = -\frac{2 + \sum_{p=1}^{\rho} \frac{b_p^{(n)}}{a_p^{(n)}}}{\sum_{p=1}^{\rho} \frac{1}{a_p^{(n)}}}, \tag{B.16}$$

as long as the index ρ is known.

Theorem 1: Let ρ be the number of positive components in the solution of optimization problem (4), then

$$\rho = \max \left\{ 1 \leq j \leq P : b_j - \frac{2 + \sum_{i=1}^{j} \frac{b_i}{a_i}}{\sum_{i=1}^{j} \frac{1}{a_i}} < 0 \right\}, \tag{B.17}$$

where $\boldsymbol{b}$ is obtained by sorting $\boldsymbol{b}^{(n)}$ components in ascending order and $\boldsymbol{a}$ corresponds to $\boldsymbol{a}^{(n)}$ rearranged accordingly.

Proof. Let us first define the quantities $\varphi_j \triangleq b_j - (2 + \sum_{i=1}^{j} \frac{b_i}{a_i}) / \sum_{i=1}^{j} \frac{1}{a_i}$ and $s_{j:k} \triangleq \sum_{i=j}^{k} \frac{1}{a_i}$. Then, the goal is to show that $j = \rho$ is the largest index in $\{1, \ldots, P\}$ for which φ_j remains negative.

For $j < \rho$, we have that

$$\varphi_j = \frac{1}{s_{1:j}} \left(s_{1:j} b_j - \left(2 + \sum_{i=1}^{j} \frac{b_i}{a_i} \right) \right) \tag{B.18a}$$

$$= \frac{1}{s_{1:j}} \left(s_{1:j} b_j - 2 - \sum_{i=1}^{\rho} \frac{b_i}{a_i} + \sum_{i=j+1}^{\rho} \frac{b_i}{a_i} \right) \tag{B.18b}$$

$$= \frac{1}{s_{1:j}} \left(s_{1:j} b_j + s_{1:\rho} \mu + \sum_{i=j+1}^{\rho} \frac{b_i}{a_i} \right) \tag{B.18c}$$

$$= b_j + \mu + \frac{1}{s_{1:j}} \sum_{i=j+1}^{\rho} \frac{1}{a_i} (\mu + b_i) < 0, \tag{B.18d}$$

where in step (B.18b) we use the equivalence $\sum_{i=1}^{j} \frac{b_i}{a_i} = \sum_{i=1}^{\rho} \frac{b_i}{a_i} - \sum_{i=j+1}^{\rho} \frac{b_i}{a_i}$. Next, in step (B.18c), we make use of the relation in (B.16). Finally, the step (B.18d) holds thanks to the relation in (B.12) and because $s_{1:j}, a_i \geq 0 \ \forall i, j$.

For $j = \rho$, and thanks to (B.12), we have $\varphi_\rho = b_\rho + \mu < 0$. Then, using the relation in (B.16), we can verify that (B.17) holds.

For $j > \rho$, we can follow similar algebraic steps as in (B.18) to obtain

$$\varphi_j = \frac{1}{s_{1:j}} \left(s_{1:\rho} (b_j + \mu) + \sum_{i=\rho+1}^{j} \frac{1}{a_i} (b_j - b_i) \right) \geq 0. \tag{B.19}$$

The inequality in (B.19) holds thanks to the relation in (B.13) and the fact that $b_j \geq b_i \ \forall i$.