Online Learning with Multiple Operator-valued Kernels

Julien Audiffren, Hachem Kadri

Online Learning with Multiple Operator-valued Kernels

Julien Audiffren   Hachem Kadri

Aix-Marseille Univ., LIF-QARMA, CNRS, UMR 7279, F-13013, Marseille, France

$\{firstname.name\}@lif.univ-mrs.fr$

November 5, 2013

Abstract

We consider the problem of learning a vector-valued function f in an online learning setting. The function f is assumed to lie in a reproducing Hilbert space of operator-valued kernels. We describe two online algorithms for learning f while taking into account the output structure. A first contribution is an algorithm, ONORMA, that extends the standard kernel-based online learning algorithm NORMA from scalar-valued to operator-valued setting. We report a cumulative error bound that holds both for classification and regression. We then define a second algorithm, MONORMA, which addresses the limitation of pre-defining the output structure in ONORMA by learning sequentially a linear combination of operator-valued kernels. Our experiments show that the proposed algorithms achieve good performance results with low computational cost.

1 Introduction

We consider the problem of learning a function $f : \mathcal{X} \to \mathcal{Y}$ in a reproducing kernel Hilbert space, where $\mathcal{Y}$ is a Hilbert space with dimension $d > 1$. This problem has received relatively little attention in the machine learning community compared to the analogous scalar-valued case where $\mathcal{Y} \subseteq R$. In the last decade, more attention has been paid on learning vector-valued functions [Micchelli and Pontil, 2005]. This attention is due largely to the developing of practical machine learning (ML) systems, causing the emergence of new ML paradigms, such as multi-task learning and structured output prediction, that can be suitably formulated as an optimization of vector-valued functions [Evgeniou et al., 2005, Kadri et al., 2013].

Motivated by the success of kernel methods in learning scalar-valued functions, in this paper we focus our attention to vector-valued function learning using reproducing kernels (see Álvarez et al., 2012, for more details). It is important to point out that in this context the kernel function outputs an operator rather than a scalar as usual. The operator allows to encode prior information about the outputs, and then take into account the output structure. Operator-valued kernels have been applied with success in the context of multi-task learning [Evgeniou et al., 2005], functional response regression [Kadri et al., 2010] and structured output prediction [Brouard et al., 2011, Kadri et al., 2013]. Despite these recent advances, one major limitation with using operator-valued kernels is the high computational expense. Indeed, in contrast to the scalar-valued case, the kernel matrix associated to a reproducing operator-valued kernel is a block matrix of dimension $td \times td$, where t is the number of examples and d the dimension of the output space. Manipulating and inverting matrices of this size become particularly problematic when dealing with large t and d. In this spirit we have asked whether, by learning the vector-valued function f in an online setting, one could develop efficient operator-valued kernel based algorithms with modest memory requirements and low computational cost.

Online learning has been extensively studied in the context of kernel machines [Kivinen et al., 2004, Crammer et al., 2006, Cheng et al., 2006, Ying and Zhou, 2006, Vovk, 2006, Orabona et al., 2009, Cesa-Bianchi et al., 2010, Zhang et al., 2013]. We refer the reader to Diethe and Girolami [2013] for a review of kernel-based online learning methods and a more complete list of relevant references. However, most of this work has focused on scalar-valued kernels and little attention has been paid to operator-valued kernels. The aim of this paper is to propose new algorithms for online learning with operator-valued kernels suitable for multi-output learning situations and able to make better use of complex (vector-valued) output data to improve non-scalar predictions. It is worth mentioning that recent studies have adapted existing batch multi-task methods to perform online multi-task learning [Cavallanti et al., 2010, Pillonetto et al., 2010, Saha et al., 2011]. Compared to this literature, our main contributions in this paper are as follows:

- we propose a first algorithm called ONORMA, which extends the widely known NORMA algorithm [Kivinen et al., 2004] to operator-valued kernel setting,

- we provide theoretical guarantees of our algorithm that hold for both classification and regression,

- we also introduce a second algorithm, MONORMA, which addresses the limitation of pre-defining the output structure in ONORMA by learning sequentially both a vector-valued function and a linear combination of operator-valued kernels,

- we provide an empirical evaluation of the performance of our algorithms which demonstrates their effectiveness on synthetic and real-world multi-task data sets.

2   Overview of the problem

This section presents the notation we will use throughout the paper and introduces the problem of online learning with operator-valued kernels.

Notation. Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space, $\mathcal{X}$ a Polish space, $\mathcal{Y}$ a (possibly infinite-dimensional) separable Hilbert space, $\mathcal{H}$ a separable Reproducing Kernel Hilbert Space (RKHS) $\subset \mathcal{Y}^{\mathcal{X}}$ with K its Hermitian positive-definite reproducing operator-valued kernel, and $L(\mathcal{Y})$ the space of continuous endomorphisms[1] of $\mathcal{Y}$ equipped with the operator norm. Let $t \in \mathbb{N}$ denotes the number of examples, $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$ the i-th example, $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$ a loss function, ∇ the gradient operator, and $R_{inst}(f, x, y) = \ell(f(x), y) + (\lambda/2)\|f\|_{\mathcal{H}}^2$ the instantaneous regularized error. Finally let $\lambda \in \mathbb{R}$ denotes the regularization parameter and η_t the learning rate at time t, with $\eta_t \lambda < 1$.

We now briefly recall the definition of the operator-valued kernel K associated to a RKHS $\mathcal{H}$ of functions that maps $\mathcal{X}$ to $\mathcal{Y}$. For more details see Micchelli and Pontil [2005].

Definition 2.1 *The application $K : \mathcal{X} \times \mathcal{X} \to L(\mathcal{Y})$ is called the Hermitian positive-definite reproducing operator-valued kernel of the RKHS $\mathcal{H}$ if and only if :*

1. $\forall x \in \mathcal{X}, \forall y \in \mathcal{Y}$, the application

$$K(., x)y : \mathcal{X} \to \mathcal{Y}$$
$$x' \mapsto K(x', x)y$$

belongs to $\mathcal{H}$.

2. $\forall f \in \mathcal{H}, \forall x \in \mathcal{X}, \forall y \in \mathcal{Y}$,

$$\langle f(x), y \rangle_{\mathcal{Y}} = \langle f, K(., x)y \rangle_{\mathcal{H}},$$

3. $\forall x_1, x_2 \in \mathcal{X}$,

$$K(x_1, x_2) = K(x_2, x_1)^* \in L(\mathcal{Y}) \text{ (* denotes the adjoint),}$$

4. $\forall n \geq 1, \forall (x_i, i \in \{1..n\}), (x_i', i \in \{1..n\}) \in \mathcal{X}^n, \forall (y_i, i \in \{1..n\}), (y_i', i \in \{1..n\}) \in \mathcal{Y}^n$,

$$\sum_{k,\ell=0}^{n} \langle K(., x_k)y_k, K(., x_\ell')y_\ell' \rangle_{\mathcal{Y}} \geq 0.$$

(i) and (ii) define a reproducing kernel, (iii) and (iv) corresponds to the Hermitian and positive-definiteness properties, respectively.

Online learning. In this paper we are interested in online learning in the framework of operator-valued kernels. On the contrary to batch learning, examples $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$ become available sequentially, one by one. The purpose is to construct a sequence $(f_i)_{i \in \mathbb{N}}$ of elements of $\mathcal{H}$ such that:

1. $\forall i \in \mathbb{N}$, f_i only depends on the previously observed examples $\{(x_j, y_j), 1 \leq j \leq i\}$,

2. The sequence $(f_i)_{i \in \mathbb{N}}$ try to minimize $\sum_i R_{inst}(f_i, x_{i+1}, y_{i+1})$.

---

[1] We denote by $My = M(y)$ the application of the operator $M \in L(\mathcal{Y})$ to $y \in \mathcal{Y}$.

In other words, the sequence of f_t, $t \in \{1, \ldots, m\}$, tries to predict the relation between x_t and y_t based on previously observed examples.

In section 3 we consider the case where the functions f_i are constructed in a RKHS $\mathcal{H} \subset \mathcal{Y}^{\mathcal{X}}$. We propose an algorithm, called ONORMA, which is an extension of NORMA [Kivinen et al., 2004] to operator-valued kernel setting, and we prove theoretical guarantees that holds for both classification and regression.

In section 4 we consider the case where the functions f_i are constructed as a linear combination of functions g_i^j, $1 \leq j \leq m$, of different RKHS $\mathcal{H}^j$. We propose an algorithm which construct both the j sequences $(g_i^j)_{i \in \mathbb{N}}$ and the coefficient of the linear combination. This algorithm, called MONORMA, is a variant of ONORMA that allows to learn sequentially the vector-valued function f and a linear combination of operator-valued kernels. MONORMA can be seen as an online version of the recent MovKL (multiple operator-valued kernel learning) algorithm [Kadri et al., 2012].

It is important to note that both algorithms, ONORMA and MONORMA, do not require to invert the block kernel matrix associated to the operator-valued kernel and have at most a linear complexity with the number of examples at each update. This is a really interesting property since in the operator valued kernel framework, the complexity of batch algorithms is extremely high and the inversion of kernel block matrices with elements in $L(\mathcal{Y})$ is an important limitation in practical deployment of such methods. For more details, see 'Complexity Analysis' in Sections 3 and 4.

3   ONORMA

In this section we consider online learning of a function f in the RKHS $\mathcal{H} \subset \mathcal{Y}^{\mathcal{X}}$ associated to the operator-valued kernel K. The key idea here is to perform a stochastic gradient descent with respect to R_{inst} as in Kivinen et al. [2004]. The update rule (i.e., the definition of f_{t+1} as a function of f_t and the input (x_t, y_t)) is

$$f_{t+1} = f_t - \eta_{t+1} \nabla_f R_{inst}(f, x_{t+1}, y_{t+1})|_{f=f_t}, \tag{1}$$

where η_{t+1} denotes the learning rate at time $t+1$. Since $\ell(f(x), y)$ can be seen as the composition of the following functions:

$$\mathcal{H} \to \mathcal{Y} \to \mathbb{R}$$
$$f \mapsto f(x) \mapsto \ell(f(x), y),$$

we have, $\forall h \in \mathcal{H}$,

$$\begin{aligned} D_f \ell(f(x), y)|_{f=f_t}(h) \\ &= D_z \ell(z, y)|_{z=f_t(x)} (D_f f(x)|_{f=f_t}(h)) \\ &= D_z \ell(z, y)|_{z=f_t(x)} (h(x)) \\ &= \left\langle \nabla_z \ell(z, y)|_{z=f_t(x)}, h(x) \right\rangle_{\mathcal{Y}} \\ &= \left\langle K(x, \cdot) \nabla_z \ell(z, y)|_{z=f_t(x)}, h \right\rangle_{\mathcal{H}}, \end{aligned}$$

where we used successively the linearity of the evaluation function, the Riesz representation theorem and the reproducing property. From this we deduce that

$$\nabla_f (\ell(f(x), y)|_{f=f_t} = K(x, \cdot) \nabla_z \ell(z, y)|_{z=f_t(x)}. \tag{2}$$

Combining (1) and (2), we obtain

$$\begin{aligned} f_{t+1} &= f_t - \eta_{t+1} \left(K(x_{t+1}, \cdot) \nabla_z \ell(z, y_{t+1})|_{z=f_t(x_{t+1})} + \lambda f \right) \\ &= (1 - \eta_{t+1} \lambda) f_t - \eta_t K(x_{t+1}, \cdot) \nabla_z \ell(z, y)|_{z=f_t(x_{t+1})}. \end{aligned} \tag{3}$$

From the equation above we deduce that if we choose $f_0 = 0$, then there exists $(\alpha_{i,j})_{i,j}$ a family of elements of $\mathcal{Y}$ such that $\forall t \geq 0$,

$$f_t = \sum_{i=1}^{t} K(x_i, \cdot) \alpha_{i,t}, \tag{4}$$

Moreover, using (3) we can easily compute the $(\alpha_{i,j})_{i,j}$ iteratively. This is Algorithm 1, called ONORMA, which is an extension of NORMA [Kivinen et al., 2004] to operator-valued setting.

Note that (4) is a similar expression to the one obtained in batch learning framework, where the representer theorem Micchelli and Pontil [2005] implies that the solution f of a regularized empirical risk

---

Algorithm 1   ONORMA

Input: parameter $\lambda, \eta_t \in \mathbb{R}_+^*$, $s_t \in \mathbb{N}$,
              loss function ℓ,

Initialization: $f_0 = 0$

At time t: Input : (x_t, y_t)

     1. **New coefficient:**
$$\alpha_{t,t} := -\eta_t \nabla_z \ell(z, y_t)|_{z=f_{t-1}(x_t)}$$

     2. **Update old coefficient:** $\forall 1 \leq i \leq t-1$,
$$\alpha_{i,t} := (1 - \eta_t \lambda)\alpha_{i,t-1}$$

     3. **(Optional) Truncation:** $\forall 1 \leq i \leq t - s_t$,
$$\alpha_{i,t} := 0$$

     4. **Obtain f_t:**
$$f_t = \sum_{i=1}^{i=t} K(x_i, \cdot)\alpha_{i,t}$$

---

minimization problem in the vector-valued function learning setting can be written as $f = \sum K(x_i, \cdot)\alpha_i$ with $\alpha_i \in \mathcal{Y}$.

Truncation. Similarly to the scalar-valued case, ONORMA can be modified by adding a truncation step. In its unmodified version, the algorithm needs to keep in memory all the previous input $\{x_i\}_{i=1}^t$ to compute the prediction value y_t, which can be costly. However, the influence of these inputs decreases geometrically at each iteration, since $0 < 1 - \lambda\eta_t < 1$. Hence the error induced by neglecting old terms can be controlled (see Theorem 2 for more details). The optional truncation step in Algorithm 1 reflects this possibility.

Cumulative Error Bound. The cumulative error of the sequence $(f_i)_{i \leq t}$, is defined by $\sum_{i=1}^t \ell(f_i(x_i), y_i)$, that is to say the sum of all the errors made by the online algorithm when trying to predict a new output. It is interesting to compare this quantity to the error made with the function $\mathbf{f}_t$ obtained from a regularized empirical risk minimization defined as follows

$$\mathbf{f}_t = \arg\min_{h \in \mathcal{H}} R_{reg}(h, t)$$
$$= \arg\min_{h \in \mathcal{H}} \frac{1}{t} \sum_{i=1}^t \ell(h(x_i), y_i) + \frac{\lambda}{2}\|h\|_{\mathcal{H}}^2.$$

To do so, we make the following assumptions:

Hypothesis 1 $\sup_{x \in \mathcal{X}} |k(x,x)|_{op} \leq \kappa^2$.

Hypothesis 2 ℓ is σ-admissible,i.e. ℓ is convex and σ-Lipschitz with regard to z.

Hypothesis 3

     i. $\ell(z, y) = \frac{1}{2}\|f(x) - z\|_{\mathcal{Y}}^2$,

    ii. $\exists C_y > 0$ such that $\forall t \geq 0$, $\|y_t\|_{\mathcal{Y}} \leq C_y$,

   iii. $\lambda > 2\kappa^2$.

The first hypothesis is a classical assumption on the boundedness of the kernel function. Hypothesis 2 requires the σ-admissibility of the loss function ℓ, which is also a common assumption. Hypothesis 3 provides a sufficient condition to recover the cumulative error bound in the case of least square loss function, which is not σ-admissible. It is important to note that while Hypothesis 2 is a usual assumption in the scalar case, bounding the least square cumulative error in regression using Hypothesis 3, as far as we know, is a new approach.

Theorem 1 *Let $\eta > 0$ such that $\eta\lambda < 1$. If Hypothesis 1 and either Hypothesis 2 or 3 holds, then there exists $U > 0$ such that, with $\eta_t = \eta t^{-1/2}$,*

$$\frac{1}{m}\sum_{i=1}^m R_{inst}(f_i, x_i, y_i) \leq R_{reg}(\mathbf{f}_m, m) + \frac{\alpha}{\sqrt{m}} + \frac{\beta}{m},$$

where $\alpha = 2\lambda U^2(2\eta\lambda + 1/(\eta\lambda))$, $\beta = U^2/(2\eta)$, and f_t denotes the function obtained at time t by Algorithm 1 without truncation.

A similar result can be obtained in the truncation case.

Theorem 2 *Let $\eta > 0$ such that $\eta\lambda < 1$ and $0 < \varepsilon < 1/2$. Then, $\exists t_0 > 0$ such that for*

$$s_t = \min(t, t_0) + 1_{t > t_0}\lfloor (t - t_0)^{1/2 + \varepsilon}\rfloor,$$

if Hypothesis 1 and either Hypothesis 2 or 3 holds, then there exists $U > 0$ such that, with $\eta_t = \eta t^{-1/2}$,

$$\frac{1}{m}\sum_{i=1}^m R_{inst}(f_i, x_i, y_i) \leq \inf_{g \in \mathcal{H}} R_{reg}(\mathbf{f}_m, m) + \frac{\alpha}{\sqrt{m}} + \frac{\beta}{m},$$

where $\alpha = 2\lambda U^2(10\eta\lambda + 1/(\eta\lambda))$, $\beta = U^2/(2\eta)$ and f_t denotes the function obtained at time t by Algorithm 1 with truncation.

The proof of Theorems 1 and 2 differs from the scalar case through several points which are grouped in Propositions 3.1 (for the σ-admissible case) and 3.2 (for the least square case). Due to the lack of space, we present here only the proof of Propositions 3.1 and 3.2, and refer the reader to Kivinen et al. [2004] for the proof of Theorems 1 and 2.

Proposition 3.1 *If Hypotheses 1 and 2 hold, we have*

    i. $\forall t \in \mathbb{N}^*$, $\|\alpha_{t,t}\|_{\mathcal{Y}} \leq \eta_t C$,

    ii. if $\|f_0\|_{\mathcal{H}} \leq U = C\kappa/\lambda$ then $\forall t \in \mathbb{N}^*$, $\|f_t\|_{\mathcal{H}} \leq U$,

    iii. $\forall t \in \mathbb{N}^*, \|\mathbf{f}_t\|_{\mathcal{H}} \leq U$.

PROOF : (i) is a consequence of the Lipschitz property and the definition of α,

$$\|\alpha_{t,t}\|_{\mathcal{Y}} = \|\eta_t\nabla_z\ell(z, y_t)|_{z = f_{t-1}(x_t)}\|_{\mathcal{Y}} \leq \eta_t C.$$

(ii) is proved using Eq. (3) and (i) by induction on t. (ii) is true for $t = 0$. If (ii) is true for $t = m$, then is is true for $m + 1$, since

$$\begin{aligned}
\|f_{m+1}\|_{\mathcal{H}} &= \|(1 - \eta_t\lambda)f_m + k(x, \cdot)\alpha_{m,m}\|_{\mathcal{H}} \\
&\leq (1 - \eta_t\lambda)\|f_m\|_{\mathcal{H}} + \sqrt{\|k(x,x)\|_{op}}\|\alpha_{m,m}\|_{\mathcal{H}} \\
&\leq \frac{C\kappa}{\lambda} - \kappa\eta_t C + \kappa\eta_t C = \frac{C\kappa}{\lambda}.
\end{aligned}$$

To prove (iii), one can remark that by definition of $\mathbf{f}_t$, we have $\forall \varepsilon > 0$,

$$\begin{aligned}
0 \leq &\frac{\lambda}{2}(\|(1 - \varepsilon)\mathbf{f}\|_{\mathcal{H}}^2 - \|\mathbf{f}\|_{\mathcal{H}}^2) \\
&+ \frac{1}{t}\sum_{i=1}^t \ell((1 - \varepsilon)\mathbf{f}(x_i), y_i) - \ell(\mathbf{f}(x_i), y_i) \\
\leq &\frac{\lambda}{2}(\varepsilon^2 - 2\varepsilon)\|\mathbf{f}\|_{\mathcal{H}}^2 + C\varepsilon\kappa\|\mathbf{f}\|_{\mathcal{H}}.
\end{aligned}$$

Since this quantity must be positive for any $\varepsilon > 0$, the dominant term in the limit when $\varepsilon \to 0$, i.e. the coefficient of ε, must be positive. Hence $\lambda\|\mathbf{f}\|_{\mathcal{H}} \leq C\kappa$. $\qquad\square$

Proposition 3.2 *If Hypotheses 1 and 3 hold, and $\|f_0\|_{\mathcal{H}} < C_y/\kappa \leq U = \max(C_y/\kappa, 2C_y/\lambda)$, then $\forall t \in \mathbb{N}^*$:*

   i. $\|f_t\|_{\mathcal{H}} < C_y/\kappa \leq U$,

   ii. $\exists V$ *in a 'neighbourhood' of f_t such that $\ell(\cdot, y_t)|_V$ is $2C_y$ Lipschitz,*

   iii. $\|\alpha_{t,t}\|_{\mathcal{Y}} \leq 2\eta_t C_y$,

   iv. $\|\mathbf{f}_t\|_{\mathcal{H}} \leq \frac{2C_y}{\lambda} \leq U$.

PROOF : We will first prove that $\forall t \in \mathbb{N}^*$,

$$(i) \implies (ii) \implies (iii).$$

$(i) \implies (ii)$**:** In the Least Square case, the application $z \mapsto \nabla_z \ell(z, y_t) = (z - y_t)$ is continuous. Hypothesis 3-(ii) and 1 combined with (i) imply that $\|\nabla_z \ell(z, y_t)|_{z=f_t(x_t)}\|_{\mathcal{Y}} < 2C_y$. Using the continuity property, we obtain (ii).

$(ii) \implies (iii)$**:** We use the same idea as in Proposition 3.1, and we obtain

$$\|\alpha_{t,t}\|_{\mathcal{Y}} = \|\eta_t \nabla_z \ell(z, y_t)|_{z=f_{t-1}(x_t)}\|_{\mathcal{Y}} \leq \eta_t 2 C_y.$$

Now we will prove (i) by induction:
Initialization (t=0). By hypothesis, $\|f_0\|_{\mathcal{H}} < C_y/\kappa$.
Propagation (t=m): If $\|f_m\|_{\mathcal{H}} < C_y/\kappa$, then using (ii) and (iii):

$$\begin{aligned}
\|f_{m+1}\|_{\mathcal{H}} &\leq (1 - \eta_t \lambda) \|f_m\|_{\mathcal{H}} + \sqrt{\|k(x,x)\|_{op}} \|\alpha_{m,m}\|_{\mathcal{H}} \\
&\leq (1 - \eta_t \lambda) C_y/\kappa + 2\kappa \eta_t C_y \\
&= C_y/\kappa + \eta_t C_y (2\kappa - \lambda/\kappa) \\
&< C_y/\kappa,
\end{aligned}$$

where the last transition is a consequence of Hypothesis 3-(iii).

Finally, note that by definition of $\mathbf{f}$, since $0 \in \mathcal{H}$,

$$\frac{\lambda}{2} \|\mathbf{f}\|_{\mathcal{H}}^2 + \frac{1}{t} \sum_{i=1}^{t} \ell(\mathbf{f}(x_i), y_i) \leq \frac{\lambda}{2} \|0\|_{\mathcal{H}}^2 + \frac{1}{t} \sum_{i=1}^{t} \ell(0, y_i)$$
$$\leq C_y^2.$$

Hence, $\|\mathbf{f}\|_{\mathcal{H}} \leq 2C_y/\lambda$. $\hfill\square$

Complexity analysis. Here we consider a naive implementation of ONORMA algorithm when $\dim \mathcal{Y} = d < \infty$. At iteration t, the calculation of the prediction has complexity $O(td^2)$ and the update of the old coefficient has complexity $O(td)$. In the truncation version, the calculation of the prediction has complexity $O(s_n d^2)$ and the update of the old coefficient has complexity $O(s_t d)$. The truncation step has complexity $O(s_t - s_{t-1})$ which is $O(1)$ since s_t is sublinear. Hence the complexity of the iteration t of ONORMA is $O(s_t d^2)$ with truncation (respectively $O(td^2)$ without truncation) and the complexity up to iteration t is respectively $O(ts_t d^2)$ and $O(t^2 d^2)$. Note that the complexity of a naive implementation of the batch algorithm for learning a vector-valued function with operator-valued kernels is $O(t^3 d^3)$. A major advantage of ONORMA is its lower computational complexity compared to classical batch operator-valued kernel-based algorithms.

4   MONORMA

In this section we describe a new algorithm called MONORMA, which extend ONORMA to the multiple operator-valued kernel (MovKL) framework [Kadri et al., 2012]. The idea here is to learn in a sequential manner both the vector-valued function f and a linear combination of operator-valued kernels. This allows MONORMA to learn the output structure rather than to pre-specify it using an operator-valued kernel which has to be chosen in advance as in ONORMA.

Algorithm 2   MONORMA

Input: parameter $\lambda, r > 0, \eta_t \in \mathbb{R}^*_+$, $s_t \in \mathbb{N}$, loss function ℓ, kernels $K^1,...,K^m$

Initialization: $f_0 = 0, g_0^j = 0$, $\delta_0^j = \frac{1}{n}, \gamma_0^j = 0$

At time $t \geq 1$: Input : (x_t, y_t)

    1. α **update phase:**

        1.1 **New coefficient:**
$$\alpha_{t,t} := -\eta_t \nabla_z \ell(z, y_t)|_{z=f_{t-1}(x_t)}$$

        1.2 **Update old coefficient:** $\forall 1 \leq i \leq t-1$,
$$\alpha_{i,t} := (1 - \eta_t \lambda)\alpha_{i,t-1}$$

    2. δ **update phase:**

        2.1. **For all** $1 \leq j \leq m$, **Calculate** $\gamma_t^j = \|g_t^j\|^2_{\mathcal{H}^j}$:
$$\gamma_t^j = (1 - \eta_t \lambda)^2 \gamma_{t-1}^j + \left\langle K^j(x_t, x_t)\alpha_{t,t}, \alpha_{t,t} \right\rangle_{\mathcal{Y}} + 2(1 - \eta_t \lambda) \left\langle g_{t-1}^j(x_t), \alpha_{t,t} \right\rangle_{\mathcal{Y}}$$

        2.2. **For all** $1 \leq j \leq m$, **Obtain** δ_t^j:
$$\delta_t^j = \frac{\left((\delta_{t-1}^j)^2 \gamma_t^j\right)^{1/r+1}}{\left(\sum_j \left((\delta_{t-1}^j)^2 \gamma_t^j\right)^{r/r+1}\right)^{1/r}}$$

    3. f_t **update phase:**

        3.1 **Obtain** g_t^i:
$$g_t^j = \sum_{i=1}^t K^j(x_i, \cdot)\alpha_{i,t}$$

        3.2 **Obtain** f_t:
$$f_t = \sum_{j=1}^n \delta_t^j g_t^j$$

Let $K^1,...,K^m$ be m different operator valued kernels associated to RKHS $\mathcal{H}^1,..,\mathcal{H}^m \subset \mathcal{Y}^{\mathcal{X}}$. In batch setting, MovKL consists in learning a function $\mathbf{f} = \sum_{i=1}^m \delta^i g^i \in \mathbf{E} = \mathcal{H}^1 + ... + \mathcal{H}^m$ with $g^i \in \mathcal{H}^i$ and $(\delta^i)_i \in D_r = \left\{(\omega^i)_{1 \leq i \leq m}, \text{ such that } \omega^i > 0 \text{ and } \sum_i (\omega^i)^r \leq 1\right\}$, which minimizes

$$\min_{(\delta_t^i) \in D} \min_{g_t^i \in \mathcal{H}^i} \frac{1}{t} \sum_{j=1}^t \ell(\sum_i \delta_t^i g_t^i, x_j, y_j) + \sum_i \|g_t^i\|_{\mathcal{H}^i}.$$

To solve this minimization problem, the batch algorithm proposed by Kadri et al. [2012] alternatively optimize until convergence: 1) the problem with respect to g_t^i when δ_t^i are fixed, and 2) the problem with respect to δ_t^i when g_t^i are fixed. Both steps are feasible. Step 1 is a classical learning algorithm with a fixed operator-valued kernel, and Step 2 admits the following closed form solution Micchelli and Pontil [2005]

$$\delta_t^i \leftarrow \frac{(\delta_t^i \|g_t^i\|_{\mathcal{H}^i})^{\frac{2}{r+1}}}{(\sum_i (\delta_t^i \|g_t^i\|_{\mathcal{H}^i})^{\frac{2r}{r+1}})^{\frac{1}{r}}}. \tag{5}$$

It is important to note that the function $\mathbf{f}$ resulting from the aforementioned algorithm can be written $\mathbf{f} = \sum_j K(x_j, \cdot)\alpha_j$ where $K(\cdot, \cdot) = \sum_i d_i K_i(\cdot, \cdot)$ is the resulting operator-valued kernel.

The idea of our second algorithm, MONORMA, is to combine the online operator-valued kernel learning algorithm ONORMA and the alternate optimization of MovKL algorithm. More precisely, we are looking for functions $f_t = \sum_i \delta_t^i \sum_{j=0}^t K_i(x_j, \cdot)\alpha_{j,t} \in \mathbf{E}$. Note that f_t can be rewritten as $f_t = \sum_i \delta_t^i g_t^i$ where $g_t^i = \sum_{j=0}^t K_i(x_j, \cdot)\alpha_{j,t} \in \mathcal{H}_i$. At each iteration of MONORMA, we update successively the $\alpha_{j,t}$ using the same rule as ONORMA, and the δ_t^i using (5).

Description of MONORMA. For each training example (x_t, y_t), MONORMA proceeds as follows: first, using the obtained value δ_{t-1}^i the algorithm determines α_t^i using a stochastic gradient descent step as in NORMA (see step 1.1 and 1.2 of MONORMA). Then, it computes $\gamma_t^i = \|g_t^i\|^2_{\mathcal{H}^i}$ using γ_{t-1}^i and α_t^i (step 2.1) and deduces δ_t^i (step 2.2). Step 3 is optional and allows to obtain the functions g_t^i and f^t if the user want to see the evolution of these functions after each update.

It is important to note that in MONORMA, $\|g_t^i\|$ is calculated without computing g_t^i itself. This is motivated by the observation that the computational complexity of computing $\|g_t^i\|$ directly from the

---

function g_t^i has complexity $O(t^2d^3)$ and thus is costly in an online setting (recall that t denotes the number of inputs and d denotes the dimension of $\mathcal{Y}$). To avoid the need to do the computation of g_t^i, one can use the following trick to save the time by calculating $\|g_t^i\|_{\mathcal{H}^i}^2$ from $\|g_{t-1}^i\|_{\mathcal{H}^i}^2$ and $\alpha_{t,t}$:

$$
\begin{aligned}
\|g_t^i\|_{\mathcal{H}^i}^2 &= \|(1-\eta_t\lambda)g_{t-1}^i + K^i(x_t,\cdot)\alpha_{t,t}\|_{\mathcal{H}^i}^2 \\
&= 2(1-\eta_t\lambda)\left\langle g_{t-1}^i, K^i(x_t,\cdot)\alpha_{t,t}\right\rangle_{\mathcal{H}^i} \\
&\quad + (1-\eta_t\lambda)^2\|g_{t-1}^i\|_{\mathcal{H}^i}^2 + \left\langle K^i(x_t,\cdot)\alpha_{t,t}, K^i(x_t,\cdot)\alpha_{t,t}\right\rangle_{\mathcal{H}^i} \\
&= 2(1-\eta_t\lambda)\left\langle g_{t-1}^i(x_t), \alpha_{t,t}\right\rangle_{\mathcal{Y}} + (1-\eta_t\lambda)^2\|g_{t-1}^i\|_{\mathcal{H}^i}^2 \\
&\quad + \left\langle K^i(x_t,x_t)\alpha_{t,t}, \alpha_{t,t}\right\rangle_{\mathcal{Y}},
\end{aligned}
\tag{6}
$$

where in the last transition we used the reproducing property. All of the terms that appear in (6) are easy to compute: $g_{t-1}^i(x_t)$ is already calculated during step 1.1 to obtain the gradient, and $\alpha_{t,t}$ is obtained from step 1. Hence, computing $\|g_t^i\|^2$ from (6) has complexity $O(d^2)$

The main advantages of ONORMA and MONORMA their ability to approach the performance of their batch counterpart with modest memory and low computational cost. Indeed, they do not require the inversion of the operator-valued kernel matrix which is extremely costly. Moreover, in contrast to previous operator-valued kernel learning algorithms [Minh and Sindhwani, 2011, Kadri et al., 2012, Sindhwani et al., 2013], ONORMA and MONORMA can also be used to non separable operator-valued kernels. Finally, a major advantage of MONORMA is that interdependencies between output variables are estimated instead of pre-specified as in ONORMA, which means that the important output structure does not have to be known in advance.

Complexity analysis. Here we consider a naive implementation of the MONORMA algorithm. At iteration t, the α update step has complexity $O(td)$. The δ update step has complexity $O(md)$. Evaluating g_j and f has complexity $O(td^2m)$. Hence the complexity of the iteration t of MONORMA is $O(td^2m)$ and the complexity up to iteration t is respectively $O(t^2d^2m)$.

5   Experiments

In this section, we conduct experiments on synthetic and real-world datasets to evaluate the efficiency of the proposed algorithms, ONORMA and MONORMA. In the following, we set $\eta = 1$, $\eta_t = 1/\sqrt{t}$, and $\lambda = 0,01$ for ONORMA and MONORMA. We compare these two algorithms with their batch counterpart algorithm[2], while the regularization parameter λ is chosen with a five-fold cross validation.

We use the following real-world dataset: Dermatology[3] (1030 instances, 9 attributes, $d = 6$), CPU performance[4] (8192 instances, 18 attributes, $d = 4$), School data[5] (15362 instances, 27 attributes, $d = 2$). Additionally, we also use a synthetic dataset (5000 instances, 20 attributes, $d = 10$) described in Evgeniou et al. [2005]. In this dataset, inputs $(x_1, ..., x_{20})$ are generated independently and uniformly over $[0,1]$ and the different output are computed as follows. Let $\phi(x) = (x_1^2, x_4^2, x_1x_2, x_3x_5, x_2, x_4, 1)$ and (w^i) denotes iid copies of a 7 dimensional Gaussian distribution with zero mean and covariance equal to $\text{Diag}(0.5; 0.25; 0.1; 0.05; 0.15; 0.1; 0.15)$. Then, the outputs of the different tasks are generated as $y^i = w^i\phi(x)$. We use this dataset with 500 instances and $d = 4$, and with 5000 instances and $d = 10$.

For our experiments, we used two different operator-valued kernels:
- a separable Gaussian kernel $K_\mu(x,x') = \exp(-\|x-x'\|_2^2/\mu)\mathbf{J}$ where μ is the parameter of the kernel varying from 10^{-3} to 10^2 and $\mathbf{J}$ denotes the $d \times d$ matrix with coefficient $\mathbf{J}_{i,j}$ equals to 1 if $i = j$ and $1/10$ otherwise.
- a non separable kernel $K_\mu(x,x') = \mu\langle x,x'\rangle\mathbf{1} + (1-\mu)\langle x,x'\rangle^2\mathbf{I}$ where μ is the parameter of the kernel varying from 0 to 1 and $\mathbf{1}$ denotes the matrix of size $d \times d$ with all its elements equal to 1 and $\mathbf{I}$ denotes the $d \times d$ identity matrix.

To measure the performance of the algorithms, we use the Mean Square Error (MSE) which refers to the mean cumulative error $\frac{1}{|Z|}\sum\|f_{i-1}(x_i) - y_i\|_{\mathcal{Y}}^2$ for ONORMA and MONORMA and to the mean empirical error $\frac{1}{|Z|}\sum\|f(x_i) - y_i\|_{\mathcal{Y}}^2$ for the batch algorithm.

---

[2]The batch algorithm refers to the usual regularized least squares regression with operator-valued kernels (see, e.g., Kadri et al. 2010).

[3]Available at UCI repository `http://archive.ics.uci.edu/ml/datasets`.

[4]Available at `http://www.dcc.fc.up.pt/~ltorgo/Regression`.
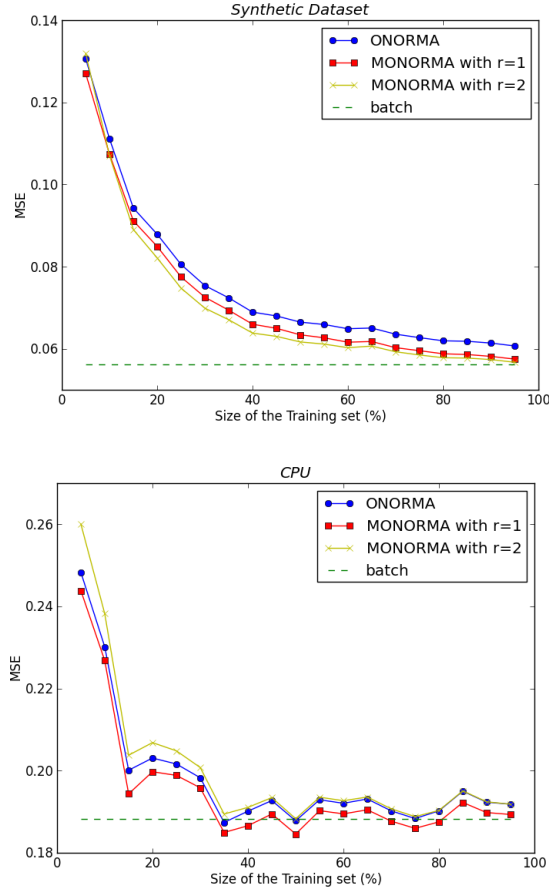
[5]see Argyriou et al. [2008].

Figure 1: Variation of MSE (Mean Square Error) of ONORMA (blue) and MONORMA (red and yellow) with the size of the dataset. The MSE of the batch algorithm (green) on the entire dataset is also reported. (left) Synthetic Dataset. (right) CPU Dataset.

Performance of ONORMA and MONORMA. In this experiment, the entire dataset Z is randomly split in two parts of equal size, Z_1 for training and Z_2 for testing. Figure 1 shows the MSE results obtained with ONORMA and MONORMA on the synthetic and the CPU performance datasets when the size of the dataset increases, as well as the results obtained with the batch algorithm on the entire dataset. For ONROMA, we used the non separable kernel with parameter $\mu = 0.2$ for the synthetic dataset and the Gaussian kernel with parameter $\mu = 1$ for the CPU performance dataset. For MONORMA, we used the kernels $\langle x, x' \rangle \mathbf{1}$ and $\langle x, x' \rangle^2 \mathbf{I}$ for the synthetic datasets, and three Gaussian kernels with parameter 0.1, 1 and 10 for the CPU performance dataset. In Figure 2, we report the MSE (or the misclassification rate for the Dermatology dataset) of the batch algorithm and ONORMA when varying the kernel parameter μ. The performance of MONORMA when learning a combination of operator-valued kernels with different parameters is also reported in Figure 2. Our results show that ONORMA and MONORMA achieve a level of accuracy close to the standard batch algorithm, but a significantly lower running time (see Table 1). Moreover, by learning a combination of operator-valued kernels and then learning the output structure, MONORMA performs better than ONORMA.

6 Conclusion

In this paper we presented two algorithms: ONORMA is an online learning algorithm that extends NORMA to operator-valued kernel framework and requires the knowledge of the output structure. MONORMA does not require this knowledge since it is capable to find the output structure by learning sequentially a linear combination of operators-valued kernels. We reported a cumulative error bound for ONORMA that holds both for classification and regression. We provided experiments on the performance of the proposed algorithms that demonstrate variable effectiveness. Possible future research directions include 1) deriving
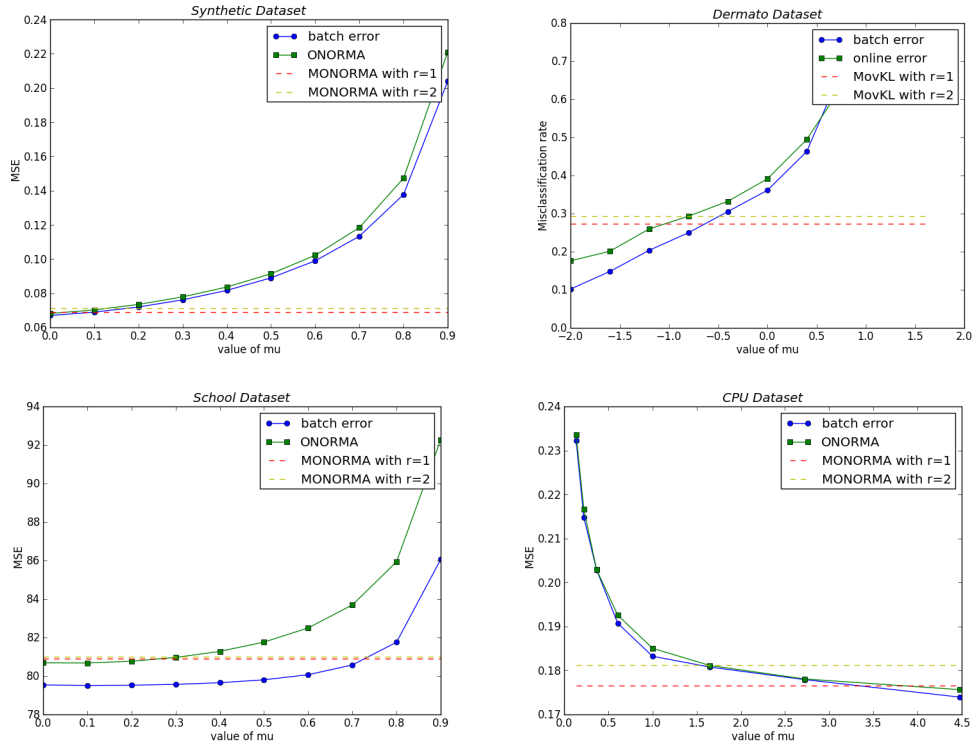
Figure 2: MSE of ONORMA (green) and the batch algorithm (blue) as a function of the parameter μ. The MSE of MONORMA is also reported. (top left) Synthetic Dataset. (top right) Dermatology Dataset. (bottom left) CPU Dataset. (bottom right) School Dataset.

Table 1: Running time in seconds of ONORMA, MONORMA and batch algorithms.

	Synthetic	Dermatology	CPU
Batch	162	11.89	2514
ONORMA	12	0.78	109
MONORMA	50	2.21	217

cumulative error bound for MONORMA, and 2) extending MONORMA to structured output learning.

References

M. A. Álvarez, L. Rosasco, and N. D. Lawrence. Kernels for vector-valued functions: a review. *Foundation and Trends in Machine Learning*, 4(3):195–266, 2012.

A. Argyriou, T. Evgeniou, and M. Pontil. Convex multi-task feature learning. *Machine Learning*, 73(3): 243–272, 2008.

C. Brouard, F. d'Alche Buc, and M. Szafranski. Semi-supervised penalized output kernel regression for link prediction. ICML, 2011.

G. Cavallanti, N.Cesa-bianchi, and C. Gentile. Linear algorithms for online multitask classification. *Journal of Machine Learning Research*, 11:2901–2934, 2010.

N. Cesa-Bianchi, S. Shalev-Shwartz, and O. Shamir. Online learning of noisy data with kernels. In *COLT*, 2010.

L. Cheng, S. V. N. Vishwanathan, D. Schuurmans, S. Wang, and T. Caelli. Implicit online learning with kernels. In *NIPS*, 2006.

K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online Passive-Aggressive Algorithms. *Journal of Machine Learning Research*, 7:551–585, 2006.

T. Diethe and M. Girolami. Online learning with (multiple) kernels: a review. *Neural computation*, 25 (3):567–625, 2013.

T. Evgeniou, C. A. Micchelli, and M. Pontil. Learning multiple tasks with kernel methods. *Journal of Machine Learning Research*, 6:615–637, 2005.

H. Kadri, E. Duflos, P. Preux, S. Canu, and M. Davy. Nonlinear functional regression: a functional RKHS approach. AISTATS, 2010.

H. Kadri, A. Rakotomamonjy, F. Bach, and P. Preux. Multiple operator-valued kernel learning. NIPS, 2012.

H. Kadri, M. Ghavamzadeh, and P. Preux. A generalized kernel approach to structured output learning. ICML, 2013.

J. Kivinen, A. J. Smola, and R. C. Williamson. Online learning with kernels. *Signal Processing, IEEE Transactions on*, 52(8):2165–2176, 2004.

C. A. Micchelli and M. Pontil. On learning vector-valued functions. *Neural Computation*, 17:177204, 2005.

H. Q. Minh and V. Sindhwani. Vector-valued manifold regularization. In *ICML*, 2011.

F. Orabona, J. Keshet, and B. Caputo. Bounded kernel-based online learning. *Journal of Machine Learning Research*, 10:2643–2666, 2009.

G. Pillonetto, F. Dinuzzo, and G. De Nicolao. Bayesian online multitask learning of gaussian processes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(2):193–205, 2010.

A. Saha, P. Rai, H. Daumé III, and S. Venkatasubramanian. Online learning of multiple tasks and their relationships. In *AISTATS*, 2011.

V. Sindhwani, H. Q. Minh, and A. C. Lozano. Scalable matrix-valued kernel learning for high-dimensional nonlinear multivariate regression and granger causality. In *UAI*, 2013.

V. Vovk. On-line regression competitive with reproducing kernel hilbert spaces. In *Theory and Applications of Models of Computation*, pages 452–463. Springer, 2006.

Y. Ying and D. Zhou. Online regularized classification algorithms. *Information Theory, IEEE Transactions on*, 52(11):4775–4788, 2006.

L. Zhang, J. Yi, R. Jin, M. Lin, and X. He. Online kernel learning with a near optimal sparsity bound. ICML, 2013.