Journal of Machine Learning Research 22 (2021) 1-40 Submitted 8/19; Revised 10/20; Published 1/21

Entangled Kernels - Beyond Separability

Riikka Huusari RIIKKA.HUUSARI@AALTO.FI
Helsinki Institute for Information Technology HIIT
Department of Computer Science
Aalto University
02150 Espoo, Finland

Hachem Kadri HACHEM.KADRI@LIS-LAB.FR
Department of Computer Science
Aix-Marseille University, CNRS, LIS
13013 Marseille, France

Editor: Arthur Gretton

Abstract

We consider the problem of operator-valued kernel learning and investigate the possibility of going beyond the well-known separable kernels. Borrowing tools and concepts from the field of quantum computing, such as partial trace and entanglement, we propose a new view on operator-valued kernels and define a general family of kernels that encompasses previously known operator-valued kernels, including separable and transformable kernels. Within this framework, we introduce another novel class of operator-valued kernels called *entangled kernels* that are not separable. We propose an efficient two-step algorithm for this framework, where the entangled kernel is learned based on a novel extension of kernel alignment to operator-valued kernels. We illustrate our algorithm with an application to supervised dimensionality reduction, and demonstrate its effectiveness with both artificial and real data for multi-output regression.

Keywords: Kernel Learning, Entangled Kernels, Operator-valued Kernels, Vector-valued RKHS, Multi-output Learning

1. Introduction

There is a growing body of learning problems for which each instance in the training set is naturally associated with a set of discrete and/or continuous labels (Izenman, 1975; Caruana, 1997; Micchelli and Pontil, 2005; Álvarez and Lawrence, 2011; Dembczyński et al., 2012; Baldassarre et al., 2012). Output kernel learning algorithms approach these problems by learning simultaneously a vector-valued function in a reproducing kernel Hilbert space (RKHS) and a positive semi-definite matrix that describes the relationships between the labels (Dinuzzo et al., 2011; Dinuzzo and Fukumizu, 2011; Ciliberto et al., 2015; Jawanpuria et al., 2015). The main idea of these methods is to learn a separable operator-valued kernel.

Operator-valued kernels appropriately generalize the well-known notion of reproducing kernels and provide a means for extending the theory of reproducing kernel Hilbert spaces from scalar- to vector-valued functions. They were introduced as a machine learning tool in Micchelli and Pontil (2005) and have since been investigated for use in various

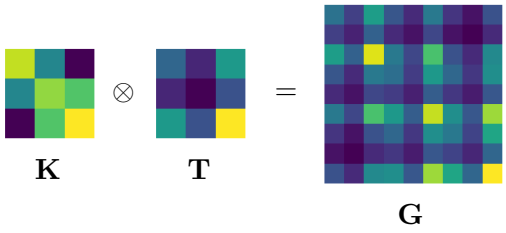

Figure 1: Illustration on the restrictiveness of a separable kernel. $\mathbf{K}$ is the scalar-valued kernel matrix, $\mathbf{T}$ is the output similarity matrix and $\mathbf{G}$ is the operator-valued kernel matrix. Every block of the big kernel matrix $\mathbf{G} = \mathbf{K} \otimes \mathbf{T}$ has the same structure, and thus models the output dependencies almost the same no matter the input interactions in $\mathbf{K}$.

machine learning tasks, including multi-task learning (Evgeniou et al., 2005), functional regression (Kadri et al., 2016), structured output prediction (Brouard et al., 2016), quantile learning (Sangnier et al., 2016), multi-view learning (Minh et al., 2016) and reinforcement learning (Lever et al., 2016). The kernel function evaluated on two data samples in this setting outputs a linear operator (a matrix in the case of finite-dimensional output spaces, $K(x, z) \in \mathbb{R}^{p \times p}$ with p the dimension of the output space) which encodes information about multiple output variables. A challenging question in vector-valued learning is what sort of interactions should the operator-valued kernel learn and quantify, and how should one build and design these kernels. This is the main question investigated in the paper in the context of non-separability between input and output variables.

Some classes of operator-valued kernels have been proposed in the literature (Caponnetto et al., 2008; Álvarez et al., 2012), with separable kernels being one of the most widely used for learning vector-valued functions due to their simplicity and computational efficiency. These kernels are formulated as a product between a kernel function for the input space alone, and a matrix that encodes the interactions among the outputs. Indeed, the name of the class refers to the fact that dependencies between input and output variables are considered separately. In order to overcome the need for choosing a kernel before the learning process, output kernel learning methods learn the output matrix from data (Dinuzzo et al., 2011; Ciliberto et al., 2015; Jawanpuria et al., 2015). However there are limitations in using separable kernels. These kernels use only one output matrix and one input kernel function, and then cannot capture different kinds of dependencies and correlations. Moreover the kernel matrix associated to separable operator-valued kernels is a rank-one kronecker product matrix (i.e, computed by only one kronecker product $\mathbf{K} \otimes \mathbf{T}$, where $\mathbf{K}$ is the scalar-valued kernel matrix and $\mathbf{T}$ is the output similarity matrix), which is restrictive as it assumes a strong repetitive structure in the operator-valued kernel matrix that models input and output interactions as illustrated in Figure 1.

To go beyond separable kernels, some attempts have been made to learn a weighted sum of them in the multiple kernel learning framework (Kadri et al., 2012; Sindhwani et al., 2013; Gregorová et al., 2017). Another approach, proposed by Lim et al. (2015), is to

learn a combination of a separable and a transformable kernel, the latter being a type of non-separable kernel based on representing the data via label-dependent transformations. In that work, the form of the transformable kernel is fixed in advance but allows to encode non-separable dependencies between inputs and outputs. Despite these previous investigations, the lack of knowledge about the full potential of operator-valued kernels and how to go beyond the restrictive separable kernel clearly hampers their widespread use in machine learning and other fields.

This paper deals with the problem of learning non-separable kernels. It is a significant extension of our previous conference paper (Huusari and Kadri, 2019), giving more thorough treatment of the background material, additional theoretical results, full proofs, and more insights to the developed framework. It also provides a theoretical analysis of the generalization error of the learning method, along with expanded experimental section. Our main contributions are:

- By leveraging tools from the field of quantum computing, we introduce a novel class of kernels based on the notion of partial trace which generalizes the trace operation to block matrices. This class of partial trace kernels we propose is very broad and encompasses previously known operator-valued kernels, including separable and transformable kernels, which we illustrate with examples.

- From the new class of partial-trace kernels we derive another new class of operator-valued kernels, called *entangled* kernels, that are not separable. As far as we are aware, this is the first time such an operator-valued kernel categorization has been performed.

- We further study this class of kernels and develop a new algorithm called EKL (Entangled Kernel Learning) that in two steps learns an entangled kernel and a vector-valued function. For the first step of kernel learning, we propose a novel definition of alignment between an operator-valued kernel and labels of a multi-output learning problem. To our knowledge, this is the first proposition on how to extend alignment to the context of operator-valued kernels. Our algorithm offers improvements to the high computational cost usually associated with learning with general operator-valued kernels.

- We prove a bound on the generalization error of our method using the notion of Rademacher complexity.

- We provide an empirical evaluation of EKL. First, we illustrate how EKL works by applying it to the task of supervised dimensionality reduction in the multi-task setting. We also thoroughly study its performance and demonstrate its effectiveness on artificial data as well as real benchmarks. Finally we compare the running times of learning with various classes of operator-valued kernels.

The remainder of this paper is organized as follows. We begin in Section 2 with a short background on quantum entanglement and learning with operator-valued kernels. In Section 3, we describe some known classes of operator-valued kernels and review previous work on learning separable operator-valued kernels. Section 4 then introduces the new

$\mathcal{X}$	input space	$\mathcal{Y}$	output space
$k(\cdot,\cdot)$	scalar-valued kernel	$K(\cdot,\cdot)$	operator-valued kernel
$\mathcal{K}$	reproducing kernel Hilbert space of k	$\mathcal{H}$	reproducing kernel Hilbert space of K
$\mathbf{K}$	the kernel matrix of k	$\mathbf{G}$	the (block) kernel matrix of K
ϕ	feature map of k (from $\mathcal{X}$ to $\mathcal{Y}$)	Γ	feature map of K (from $\mathcal{X}$ to $\mathcal{L}(\mathcal{Y},\mathcal{H})$)
$\mathcal{L}(\mathcal{A},\mathcal{B})$	the set of trace-class operators from $\mathcal{A}$ to $\mathcal{B}$	$\mathcal{L}(\mathcal{A})$	the set $\mathcal{L}(\mathcal{A},\mathcal{A})$
$\mathbf{A}^\top$, $\mathbf{u}^\top$	the transpose of a matrix $\mathbf{A}$ or a vector $\mathbf{u}$	$\mathbf{A}^*$, $\mathbf{u}^*$	the adjoint of an operator $\mathbf{A}$ or a vector $\mathbf{u}$
$\mathbf{A} \geq 0$	a positive semi-definite (psd) matrix	A($\mathbf{A}$,$\mathbf{B}$)	alignment between matrices $\mathbf{A}$ and $\mathbf{B}$
$\mathcal{A}_1 \otimes \mathcal{A}_2$	the tensor product of Hilbert spaces $\mathcal{A}_1$ and $\mathcal{A}_2$	$\mathbf{A}_1 \otimes \mathbf{A}_2$	the tensor product of operators $\mathbf{A}_1$ and $\mathbf{A}_2$
$\mathrm{tr}(\mathbf{A})$	the trace of a matrix or an operator $\mathbf{A} \in \mathcal{L}(\mathcal{A})$	$\mathrm{tr}_{\mathcal{A}_2}(\mathbf{A})$	the partial trace of a matrix or an operator $\mathbf{A} \in \mathcal{L}(\mathcal{A}_1 \otimes \mathcal{A}_2)$

Table 1: Notation summary.

classes of *partial-trace* and *entangled* kernels. Our new algorithm EKL for learning entangled kernels is given in Section 5, along with generalization analysis. In Section 6, we present our experimental results for both synthetic and real-life data. We conclude in Section 7 and present some technical details in the appendix.

1.1 Notation

We denote scalars, vectors and matrices as a, $\mathbf{a}$ and $\mathbf{A}$ respectively. The notation $\mathbf{A} \geq 0$ will be used to denote a positive semi-definite (psd) matrix. Throughout the paper we use n as the number of labeled data samples and p as the number of outputs corresponding to one data sample. We denote our set of data samples by $\{x_i, y_i\}_{i=1}^n$ on $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$ is a Polish space and $\mathcal{Y}$ is a separable Hilbert space. Usually, $\mathcal{X}$ and $\mathcal{Y}$ are respectively $\mathbb{R}^d$ and $\mathbb{R}^p$ equipped with the standard Euclidean metric. Without loss of generality, we can assume that $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \mathbb{R}^p$, and thus denote our data set as $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^n$. We use $k(\cdot,\cdot)$ as a scalar-valued, and $K(\cdot,\cdot)$ as an operator-valued kernel function; the corresponding kernel matrices are $\mathbf{K} \in \mathbb{R}^{n\times n}$ and $\mathbf{G} \in \mathbb{R}^{np\times np}$, the latter containing blocks of size $p \times p$. We denote by $\mathcal{K}$ and $\mathcal{H}$ the reproducing kernel Hilbert spaces (RKHS) associated to the kernels k and K, respectively. Table 1.1 summarizes the notation used in this paper.

2. Background

We now give some background about quantum entanglement, and review the basics of learning with operator-valued kernels.

2.1 Quantum Entanglement

The field of quantum computing is vast, and rapidly growing. This section is not intended to provide a broad overview or exhaustive survey of the literature on quantum etanglement, but gives some notions on entanglement as a quantum property of mixed composite quantum systems that inspired our entangled kernel design. We refer the reader to Horodecki et al. (2009), Bengtsson and Życzkowski (2017), and Rieffel and Polak (2011, chap. 10) for more background information. We will now start with very basics of quantum computation, for this we refer the reader to Rieffel and Polak (2011, chap. 2-3).

A major difference of quantum computing and quantum information theory to their classical counterparts is that instead of bits the "particles" carrying information are qubits. Unlike bits which have only two possible states, 0 and 1, qubits can exist in those and any combination of them. More formally, a qubit takes values $a\psi_0 + b\psi_1$ where ψ_0 and ψ_1 are orthonormal basis vectors and $a, b \in \mathbb{C}$ such that $|a|^2 + |b|^2 = 1$.[1] In quantum information theory the actual state $a\psi_0 + b\psi_1$ cannot be recovered by measurement; it is always measured as either ψ_0 or ψ_1 according to the probabilities proportional to multipliers a and b.

In the heart of quantum computing there is a notion of quantum systems, consisting of one or more qubits. For a system of one qubit, the system's basis consists of two-dimensional vectors. For a system of n qubits, the description requires a 2^n-dimensional Hilbert space to capture all possible combinations of the qubit values. This brings forward the notion of entanglement; any state that cannot be written as a tensor product of n single-qubit states is said to be entangled. Perhaps the simplest example of an entangled state is

$$\frac{1}{\sqrt{2}}\left(\psi_{00} + \psi_{11}\right) \tag{1}$$

as it cannot be written as

$$(a_1\psi_0 + b_1\psi_1) \otimes (a_2\psi_0 + b_2\psi_1) = a_1a_2\psi_{00} + a_1b_2\psi_{01} + b_1a_2\psi_{10} + b_1b_2\psi_{11}$$

with any multipliers a_1, a_2, b_1 and b_2, and where $\psi_{01} = \psi_0 \otimes \psi_1$, similarly for others.

A quantum system exists in a state, describing all the information that can be learned of the system. A quantum system can also be divided into parts or subsystems. We focus here only on *bipartite* quantum systems, i.e., systems composed of two distinct subsystems. The Hilbert space $\mathcal{F}$ associated with a bipartite quantum system is given by the tensor product $\mathcal{F}_1 \otimes \mathcal{F}_2$ of the spaces $\mathcal{F}_1$ anf $\mathcal{F}_2$ corresponding to each of the subsystems. A question to ask in this context is, what information of the system can be obtained by only considering a part of it, either $\mathcal{F}_1$ or $\mathcal{F}_2$? A quantum state can be either "pure" or "mixed". While states of pure systems can be represented by a state vector $\psi \in \mathcal{F}$, for mixed states the

1. In the field of quantum computing, it is more usual to use the Dirac's bra-ket notation for the basis vectors. In this notation "bra" $\langle x|$ denotes a row vector and "ket" $|x\rangle$ a column vector, and a qubit would take values $a|0\rangle + b|1\rangle$.

characterization is done with density operators (or matrices) ρ, positive Hermitian operators with trace equal to one. Pure states can also be modeled with a density operator allowing for uniform treatment, in this case $\rho = \psi\psi^\top$.

The entanglement present in a bipartite quantum system can be modeled through the *partial trace* of the system. Given the density operator ρ modeling the whole system, the state of, say, the first subsytem is described by a reduced density matrix, given by taking the partial trace of ρ over $\mathcal{F}_2$. If the system can be accurately represented with only the two subsystems, then it is not entangled. In the following we review the notions of partial trace, separability and entanglement of bipartite quantum systems in more detail.

We denote the set of bounded linear operators from a Hilbert space $\mathcal{B}$ to $\mathcal{B}$ with finite trace norm as $\mathcal{L}(\mathcal{B})$. Let $\mathcal{F}_1$ and $\mathcal{F}_2$ be separable Hilbert spaces.

Definition 1 *(partial trace)*
Let $\{e_i\}_i$ be an orthonormal basis for $\mathcal{F}_2$. For an operator $\mathbf{A}$ in $\mathcal{L}(\mathcal{F}_1 \otimes \mathcal{F}_2)$ its partial trace, $\mathrm{tr}_{\mathcal{F}_2}\mathbf{A}$, is an operator in $\mathcal{L}(\mathcal{F}_1)$ defined by the relation

$$\langle x, (\mathrm{tr}_{\mathcal{F}_2}\mathbf{A})y\rangle = \sum_i \langle x \otimes e_i, \mathbf{A}(y \otimes e_i)\rangle$$

for all $x, y \in \mathcal{F}_1$.

This definition follows the ones in Bhatia (2009, Equation 4.34) and Attal (2015b, Equation 2.10). In the finite-dimensional case where $\mathcal{F}_1 = \mathbb{R}^p$ and $\mathcal{F}_2 = \mathbb{R}^N$, the operator $\mathbf{A} \in \mathbb{R}^{pN \times pN}$ is a block matrix where each block is of size $N \times N$, and the partial trace is obtained by computing the trace of each block (see Figure 2). To see this, let us denote the orthonormal bases of $\mathcal{F}_1$ and $\mathcal{F}_2$ by $\{g_j\}_{j=1}^p$ and $\{e_i\}_{i=1}^N$, respectively. Any block matrix $\mathbf{A} \in \mathbb{R}^{pN \times pN}$ can be written as $\mathbf{A} = \sum_{s,t=1}^p g_s g_t^\top \otimes \mathbf{A}_{st}$, where $\mathbf{A}_{st} \in \mathbb{R}^{N \times N}$ is the (s,t)th block of $\mathbf{A}$. Now, when investigating one element of the operator $\mathrm{tr}_{\mathcal{F}_2}\mathbf{A}$ at position (l,m) we see that it exactly corresponds to the trace of block $\mathbf{A}_{lm}$:

$$\begin{aligned}
[\mathrm{tr}_{\mathcal{F}_2}\mathbf{A}]_{lm} &= \langle g_l, (\mathrm{tr}_{\mathcal{F}_2}\mathbf{A})g_m\rangle = \sum_{i=1}^N \langle g_l \otimes e_i, \mathbf{A}(g_m \otimes e_i)\rangle \\
&= \sum_{i=1}^N \sum_{s,t=1}^p \langle g_l \otimes e_i, (g_s g_t^\top \otimes \mathbf{A}_{st})(g_m \otimes e_i)\rangle = \sum_{i=1}^N \sum_{s,t=1}^p \langle g_l \otimes e_i, g_s g_t^\top g_m \otimes \mathbf{A}_{st} e_i\rangle \\
&= \sum_{i=1}^N \sum_{s=1}^p \langle g_l \otimes e_i, g_s \otimes \mathbf{A}_{sm} e_i\rangle = \sum_{i=1}^N \sum_{s=1}^p \langle g_l, g_s\rangle\langle e_i, \mathbf{A}_{sm} e_i\rangle = \sum_{i=1}^N \langle e_i, \mathbf{A}_{lm} e_i\rangle \\
&= tr(\mathbf{A}_{lm}).
\end{aligned}$$

The last equality is easy to see from the definition of trace. The following theorem shows how to compute partial trace for separable operators (Attal, 2015b, Theorem 2.29).

Theorem 2 *(partial trace of a tensor product of operators)*
Let $\mathbf{B}$ and $\mathbf{C}$ be operators in $\mathcal{L}(\mathcal{F}_1)$ and $\mathcal{L}(\mathcal{F}_2)$, respectively. If $\mathbf{A}$ is an operator in $\mathcal{L}(\mathcal{F}_1 \otimes \mathcal{F}_2)$ of the form $\mathbf{B} \otimes \mathbf{C}$, then

$$\mathrm{tr}_{\mathcal{F}_2}(\mathbf{A}) = \mathbf{B}\ \mathrm{tr}(\mathbf{C}).$$

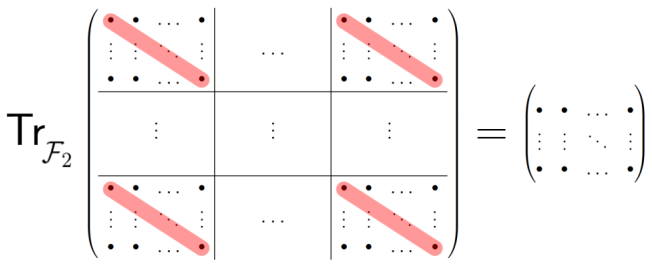

Figure 2: Illustration of partial trace operation. The partial trace operation applied to $N \times N$-blocks of a $pN \times pN$ matrix gives a $p \times p$ matrix as an output.

The notion of *partial trace* is a generalization of the trace operation to block structured matrices (Rieffel and Polak, 2011, chap. 10). Note that there are two ways of generalizing trace to block matrices. Another possibility would be the so-called block trace (Filipiak et al., 2018) which, informally, is defined as a sum of the diagonal blocks of a matrix; with $\mathbf{A} \in \mathbb{R}^{pN \times pN}$ it would be the sum $\sum_{t=1}^{p} \mathbf{A}_{tt}$ in which each $\mathbf{A}_{tt}$ is of size $N \times N$. However in this work we only consider the "blockwise trace" definition we discussed earlier.

In the case where the density matrix ρ of a mixed bipartite state can be written as $\rho = \rho_1 \otimes \rho_2$, where ρ_1 and ρ_2 are density matrices on $\mathcal{F}_1$ and $\mathcal{F}_2$ of the subsystems, the partial trace of ρ with respect to $\mathcal{F}_2$ is ρ_1. This form of mixed product states is restrictive and does not exhibit correlations between the two subsystems. A convex sum of different product states,

$$\rho = \sum_i p_i \, \rho_1^i \otimes \rho_2^i, \tag{2}$$

with $p_i \geq 0$ and $\sum_i p_i = 1$, however, will in general represent certain types of correlations between the subsystems of the composite quantum system. These correlations can be described in terms of the classical probabilities p_i, and are therefore considered classical. States of the form (2) thus are called *separable* mixed states. In contrast, a mixed state is *entangled* if it cannot be written as a convex combination of product states, i.e.,

$$\nexists \; \rho_1^i, \rho_2^i, p_i \geq 0 \quad \text{such that} \quad \rho = \sum_i p_i \, \rho_1^i \otimes \rho_2^i. \tag{3}$$

Entangled states are one of the most commonly encountered classes of bipartite states possessing quantum correlations (Mintert et al., 2009).

A challenging problem in quantum computing is to identify necessary and sufficient conditions for quantum separability. Given a density matrix ρ of a bipartite quantum state, the quantum separability problem asks whether ρ is entangled or separable. A useful and efficient necessary condition for checking if a given block density matrix is separable in some block size partition, is to use *positive partial transpose (PPT)* condition, sometimes also called Peres-Horodecki criterion (Peres, 1996; Horodecki, 1997). The partial transpose of a $pN \times pN$ block matrix $\mathbf{P}$ with blocks $(\mathbf{P}_{ij})_{i,j=1}^{p}$ is the block matrix of the same size containing the transposed blocks $(\mathbf{P}_{ij}^{\top})_{i,j=1}^{p}$. If a density matrix is separable, then it has positive partial transpose. It is necessary for any separable density matrix to have positive

partial transpose; yet in general this condition is not sufficient in guaranteeing separability, as there might be non-separable density matrices fulfilling the PPT condition. However this condition guarantees that if the partial transpose matrix has a negative eigenvalue, the state is entangled.

2.2 Learning with Operator-valued Kernels

We now review the basics of operator-valued kernels (OvKs) and their associated vector-valued reproducing kernel Hilbert spaces (RKHSs) in the setting of supervised learning. Vector-valued RKHSs were introduced to the machine learning community by Micchelli and Pontil (2005) as a way to extend kernel machines from scalar to vector outputs. Given a set of training samples $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^n$ on $\mathcal{X} \times \mathcal{Y}$, the optimization problem

$$\arg\min_{f \in \mathcal{H}} \sum_{i=1}^{n} V(\mathbf{y}_i, f(\mathbf{x}_i)) + \lambda \|f\|_{\mathcal{H}}^2, \tag{4}$$

where f is a vector-valued function and $V : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ is a convex loss function, can be solved in a vector-valued RKHS $\mathcal{H}$ by the means of a vector-valued extension of the representer theorem.

Definition 3 *(vector-valued RKHS)*
A Hilbert space $\mathcal{H}$ of functions from $\mathcal{X}$ to $\mathcal{Y}$ is called a reproducing kernel Hilbert space if there is a positive semi-definite $\mathcal{L}(\mathcal{Y})$-valued kernel K on $\mathcal{X} \times \mathcal{X}$ such that:

i. the function $\mathbf{z} \mapsto K(\mathbf{x}, \mathbf{z})\mathbf{y}$ belongs to $\mathcal{H}$, $\forall\, \mathbf{z}, \mathbf{x} \in \mathcal{X},\ \mathbf{y} \in \mathcal{Y}$,

ii. $\forall f \in \mathcal{H}, \mathbf{x} \in \mathcal{X},\ \mathbf{y} \in \mathcal{Y},\ \ \langle f, K(\mathbf{x}, \cdot)\mathbf{y}\rangle_{\mathcal{H}} = \langle f(\mathbf{x}), \mathbf{y}\rangle_{\mathcal{Y}}$ (reproducing property).

Definition 4 *(positive semi-definite operator-valued kernel)*
A $\mathcal{L}(\mathcal{Y})$-valued kernel K on $\mathcal{X} \times \mathcal{X}$ is a function $K(\cdot, \cdot) : \mathcal{X} \times \mathcal{X} \to \mathcal{L}(\mathcal{Y})$; it is positive semi-definite if:

i. $K(\mathbf{x}, \mathbf{z}) = K(\mathbf{z}, \mathbf{x})^*$, where superscript * denotes the adjoint operator,

ii. and, for every $n \in \mathbb{N}$ and all $\{(\mathbf{x}_i, \mathbf{y}_i)_{i=1,\dots,n}\} \in \mathcal{X} \times \mathcal{Y}$,

$$\sum_{i,j} \langle \mathbf{y}_i, K(\mathbf{x}_i, \mathbf{x}_j)\mathbf{y}_j\rangle_{\mathcal{Y}} \geq 0.$$

Theorem 5 *(bijection between vector-valued RKHS and positive semi-definite operator-valued kernel)*
An $\mathcal{L}(\mathcal{Y})$-valued kernel K on $\mathcal{X} \times \mathcal{X}$ is the reproducing kernel of some Hilbert space $\mathcal{H}$, if and only if it is positive semi-definite.

Theorem 6 *(representer theorem)*
Let K be a positive semi-definite operator-valued kernel and $\mathcal{H}$ its corresponding vector-valued RKHS. The solution $\hat{f} \in \mathcal{H}$ of the regularized optimization problem (4) has the following form

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{n} K(\mathbf{x}, \mathbf{x}_i)\mathbf{c}_i, \quad \textit{with} \quad \mathbf{c}_i \in \mathcal{Y}. \tag{5}$$

With regard to the classical representer theorem, here the kernel K outputs a matrix and the "weights" $\mathbf{c}_i$ are vectors. The proofs of Theorem 5 and 6 can be found in Micchelli and Pontil (2005) and Kadri et al. (2016). For further reading on operator-valued kernels and their associated RKHSs, see, e.g., Caponnetto et al. (2008); Carmeli et al. (2010); Álvarez et al. (2012).

3. Learning Operator-valued Kernels

In this section we first review some known classes of operator-valued kernels, before moving on to describing ways to learn them. Most of the works in this field consider separable kernels, but a few specialized methods exist also for non-separable kernels.

3.1 Known Classes of Operator-valued Kernels

Some well-known classes of operator-valued kernels include separable and transformable kernels. Note that here and throughout the rest of the manuscript we consider the case where the output space $\mathcal{Y}$ is of finite dimension p (i.e., $\mathcal{Y} = \mathbb{R}^p$ and $\mathcal{L}(\mathcal{Y}) = \mathbb{R}^{p\times p}$).

Definition 7 *(Separable operator-valued kernel)*
A separable operator-valued kernel is a function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^{p\times p}$, that can be written as

$$K(\mathbf{x}, \mathbf{z}) = k(\mathbf{x}, \mathbf{z})\mathbf{T}, \quad \forall\, \mathbf{x}, \mathbf{z} \in \mathcal{X}, \tag{6}$$

in which k is a scalar-valued kernel function, and $\mathbf{T} \in \mathbb{R}^{p\times p}$ is a positive semi-definite matrix.

This class of kernels is very attractive in terms of computational time, as it is easily decomposable. However the matrix $\mathbf{T}$ acts only on the outputs independently of the input data, which makes it difficult for these kernels to capture input-output relations. In the same spirit a more general class, sum of separable kernels, can be defined as follows.

Definition 8 *(Sum of separable operator-valued kernels)*
A sum of separable operator-valued kernels is a function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^{p\times p}$, that can be written as

$$K(\mathbf{x}, \mathbf{z}) = \sum_l k_l(\mathbf{x}, \mathbf{z})\mathbf{T}_l, \quad \forall\, \mathbf{x}, \mathbf{z} \in \mathcal{X}, \tag{7}$$

in which k_l are a scalar-valued kernels and $\mathbf{T}_l \in \mathbb{R}^{p\times p}$ are positive semi-definite.

This class of operator-valued kernels can capture more complex similarities, but still assumes that the unknown input-output dependencies can be decomposed into a product of two separate kernel functions that encode interactions among inputs and outputs independently.

Definition 9 *(Transformable operator-valued kernel)*
A transformable operator-valued kernel is a function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^{p\times p}$, that can be written as

$$K(\mathbf{x}, \mathbf{z}) = \left[\widetilde{k}(S_l\mathbf{x}, S_m\mathbf{z})\right]_{l,m=1}^{p}, \quad \forall\, \mathbf{x}, \mathbf{z} \in \mathcal{X}, \tag{8}$$

in which $\widetilde{k} : \widetilde{\mathcal{X}} \times \widetilde{\mathcal{X}} \to \mathbb{R}$ is a scalar-valued kernel function and $\{S_t\}_{t=1}^{p}$ are mappings from $\mathcal{X}$ to $\widetilde{\mathcal{X}}$.

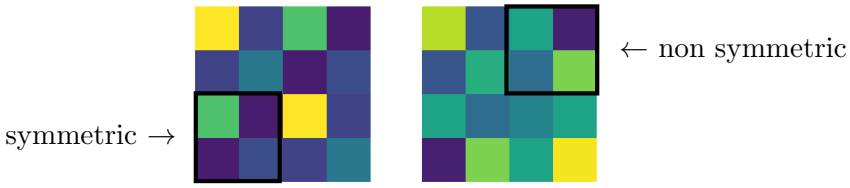

Figure 3: Illustration on differences of separable (left) and non-separable (right) operator-valued kernels. For separable kernels the $p \times p$ output matrix is always a psd symmetric matrix.

In transformable kernels the data is transformed with the mappings $\{S_t\}_{t=1}^p$ before feeding it to the scalar-valued kernel function; which transformations to use depends on which outputs the element in $K(\mathbf{x}, \mathbf{z})$ corresponds to. The mappings S_t operate on input data while depending on outputs; however they are not intuitive nor easy to interpret and determine. One example of such kernels, which was proposed in Caponnetto et al. (2008), is the kernel function $K(\mathbf{x}, \mathbf{z}) := (e^{\boldsymbol{\sigma}_{lm}\langle \mathbf{x}, \mathbf{z}\rangle} : l, m \in \{1, \ldots, p\})$ with $\boldsymbol{\sigma} = (\boldsymbol{\sigma}_{lm})$ a positive semi-definite matrix. In this transformable kernel the matrix entries outputted by the kernel are computed using linear transformations of the data. Indeed, it is easy to see that $K(\mathbf{x}, \mathbf{z}) = \left[\prod_{i=1}^p e^{\langle S_l^{(i)}\mathbf{x}, S_m^{(i)}\mathbf{z}\rangle}\right]_{l,m=1}^p$, where $\boldsymbol{\sigma} = \sum_{i=1}^p \lambda_i \mathbf{u}_i \mathbf{u}_i^\top$ is the eigenvalue decomposition of $\boldsymbol{\sigma}$ and $S_t^{(i)}\mathbf{x} := \sqrt{\lambda}\mathbf{u}_{it}\mathbf{x}$, for all $t = 1, \ldots, p$.

Separable kernels are the most common operator-valued kernels to be used and learned. As already mentioned, they are nevertheless a relatively restrictive class of kernels, as the relationships between inputs and outputs are modelled independently of each other as was already illustrated in Figure 1. Moreover, only certain types of interactions can be modelled, as $\mathbf{T}$ should be a psd matrix, and symmetric. Figure 3 illustrates this.

3.2 Learning Separable Operator-valued Kernels

Most works on learning operator-valued kernels consider the Output Kernel Learning (OKL) framework, where a separable operator-valued kernel is learned by fixing the scalar-valued kernel and learning the operator $\mathbf{T}$. The method is named for the observation that learning $\mathbf{T}$ does not depend on input data values, but only on the outputs.

All the output kernel learning algorithms are based on joint optimization, that is, the kernel is learned jointly with the learning problem, giving an optimization problem that generally can be written as

$$\min_{\mathbf{T},\mathbf{c}} \sum_{i=1}^n V(\mathbf{y}_i, f_{\mathbf{T},\mathbf{c}}(\mathbf{x}_i)) + \lambda\, \Omega(f_{\mathbf{T},\mathbf{c}}) + \gamma\, \Theta(\mathbf{T}).$$

Here V is the loss function for classification/regression and Ω is the accompanying regularization term, while Θ regularizes the output matrix. With separable operator-valued kernels, applying the representer theorem we get that $f_{T,\mathbf{c}}(\cdot) = \sum_{j=1}^n k(\mathbf{x}_j, \cdot)\mathbf{T}\mathbf{c}_j$.

Many algorithms solve the output kernel learning problem. The first output-kernel learning algorithm was introduced in Dinuzzo et al. (2011) with Frobenius norm regularizer on the output matrix $\mathbf{T}$. Dinuzzo and Fukumizu (2011) considers learning low-rank output kernels, that is, separable kernels where the rank of $\mathbf{T}$ is constrained to be less or equal to some r. The optimization is performed with having also a regularizer on $\mathrm{tr}(\mathbf{T})$ in addition to the rank constraint. More general or efficient formulations of output kernel learning have been proposed in Ciliberto et al. (2015) and Jawanpuria et al. (2015).

To go beyond the standard OKL, Kadri et al. (2012) extended the multiple kernel learning framework (Gönen and Alpaydın, 2011) that is popular in learning scalar-valued kernels into operator-valued kernel framework. The multiple kernel learning refers to paradigm where given multiple (scalar-valued) kernels k_i, a combination $k(\mathbf{x}, \mathbf{z}) = \sum_{i=1}^{l} \alpha_i k_i(\mathbf{x}, \mathbf{z})$ is learned and then used in the predictive learning problem at hand. Similarly, Kadri et al. (2012) focus on learning a finite linear combination of separable operator-valued kernels. They consider two formulations of the optimization problem. In the first one the separable operator-valued kernels all share the same output operator $\mathbf{T}$, meaning that the full kernel is

$$K(\mathbf{x}, \mathbf{z}) = \sum_{i=1}^{K} \alpha_i k_i(\mathbf{x}, \mathbf{z}) \mathbf{T}.$$

The second formulation considers the case where also the output operators differ across the operator-valued kernels in the sum, giving

$$K(\mathbf{x}, \mathbf{z}) = \sum_{i=1}^{K} \alpha_i k_i(\mathbf{x}, \mathbf{z}) \mathbf{T}_i.$$

In both of these versions only the multipliers α_i are learned, and the kernels are fixed, comparably to the case of multiple kernel learning (MKL) for scalar-valued kernels. Notably, the operators $\mathbf{T}$ and $\mathbf{T}_i$ are fixed in advance, which makes the use of this method difficult as it is not obvious how one should choose $\mathbf{T}$ without learning it.

Some works have continued this line of investigation. Sindhwani et al. (2013) considers learning a combination of separable kernels that share the operator $\mathbf{T}$, while optimizing both the combination of the basis scalar-valued kernels and the matrix $\mathbf{T}$. Another approach by Gregorová et al. (2017) considers combining a set of scalar-valued kernels with a set of output matrices $\mathbf{T}_i$. However they impose diagonal structure on the output matrices, restricting the types of relations they are able to model. In this setting the diagonal values can be interpreted as model weights of the kernel in standard MKL setting.

3.3 Learning Non-separable Operator-valued Kernels

There are very few works that consider learning non-separable kernels. Lim et al. (2015) considers an application to modelling time series data and goes further than separability by learning a combination of a separable and a transformable kernel. They consider a transformable kernel defined as

$$[K_{transf.}(\mathbf{x}, \mathbf{z})]_{st} = \exp(-\gamma(\mathbf{x}_s - \mathbf{z}_t)^2),$$

where $\mathbf{x}_s$ and $\mathbf{z}_t$ are the sth and tth elements of vectors $\mathbf{x}$ and $\mathbf{z}$ respectively. This transformable kernel applies a Gaussian kernel to pairs of elements of the data vectors, giving a $d \times d$-matrix as an output if the data dimension is d. The separable kernel in their work is

$$K_{sep.}(\mathbf{x}, \mathbf{z}) = \exp(-\gamma \|\mathbf{x} - \mathbf{z}\|^2)\, \mathbf{T},$$

and the full kernel matrix they consider in learning is $K = K_{transf.} \circ K_{sep.}$, a Hadamard or element-wise matrix product of the two operator-valued kernel matrices. When they learn this kernel, they consider learning the matrix $\mathbf{T}$ from the separable part of it. This class of kernels cannot generalize to the learning problems we consider. The greatest restriction is, that the kernel outputs a $d \times d$ matrix, d being the dimension of the input data. This is very rarely the same as the dimension of the outputs.

Another specialized operator-valued kernel is that of Huusari et al. (2018), where a non-separable kernel is learned in context of multi-view learning, by incorporating a learnable metric operating between the views into the kernel. The output of a kernel is a $v \times v$ matrix where v is the number of views in the data. Similarly to the previous work, this is not applicable for a general multi-output setting we consider. Having only few specialized works outside the separability framework motivates our more general entangled kernel learning paradigm.

4. Partial Trace and Entangled Kernels

This section first revisits the known classes of operator-valued kernels and discusses the inclusions between them. After that we introduce the two novel classes of operator-valued kernels, the partial trace kernels that encompass the known classes of operator-valued kernels, and the entangled kernels that are a class of kernels distinct from the separable.

While it is straightforward to see that separable kernels belong to the larger class of sum of separable, the picture is less clear for transformable kernels. The following examples clarify this situation.

Example 1 *(transformable but not separable kernel)*
On the space $\mathcal{X} = \mathbb{R}$, consider the kernel

$$K(x, z) = \begin{pmatrix} xz & xz^2 \\ x^2 z & x^2 z^2 \end{pmatrix}, \quad \forall\, x, z \in \mathcal{X}.$$

K is a transformable kernel, but not a (sum of) separable kernel. We obtain that K is transformable simply by choosing in Def. 9 the kernel $\widetilde{k}(x, z) = xz$, $S_1(x) = x$, and $S_2(x) = x^2$. From the property of positive semi-definiteness of the operator-valued kernel, it is easy to see that the matrix $\mathbf{T}$ of a separable kernel is symmetric (see Def. 7), and since the matrix $K(x, z)$ is not, K is not a separable kernel.

Example 2 *(transformable and separable kernel)*
Let K be the kernel function defined as

$$K(\mathbf{x}, \mathbf{z}) = \langle \mathbf{x}, \mathbf{z} \rangle \mathbf{T}, \quad \forall\, \mathbf{x}, \mathbf{z} \in \mathcal{X},$$

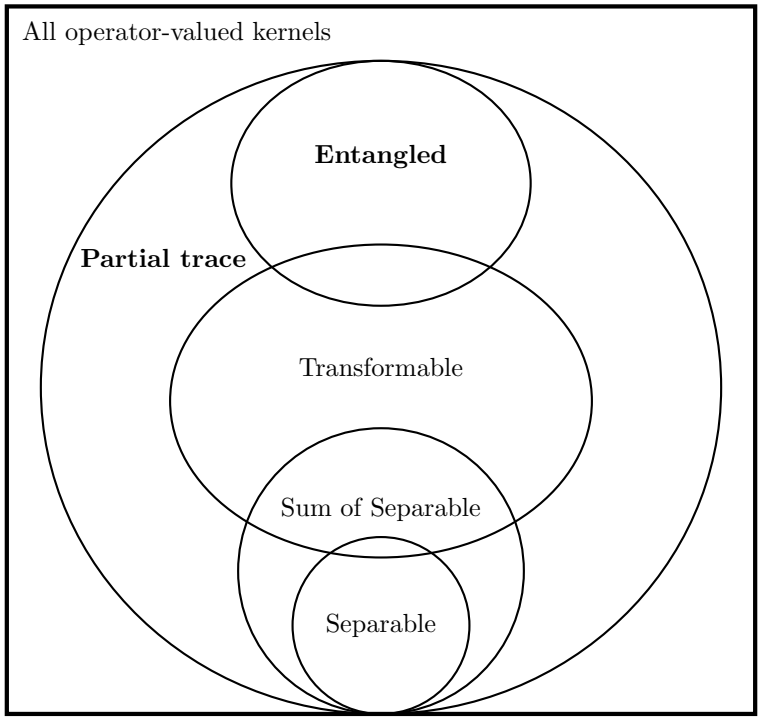

Figure 4: Illustration of inclusions among various operator-valued kernel classes.

where $\mathbf{T} \in \mathbb{R}^{p \times p}$ *is a rank one positive semi-definite matrix. K is both separable and transformable kernel. Since* $\mathbf{T}$ *is of rank one, it follows that* $\big(K(\mathbf{x}, \mathbf{z})\big)_{lm} = \mathbf{u}_l \mathbf{u}_m \langle \mathbf{x}, \mathbf{z} \rangle$*, with* $\mathbf{T} = \mathbf{u}\mathbf{u}^\top$*. We can see that K is transformable by replacing in Def. 9 the kernel* $\widetilde{k}(\mathbf{x}, \mathbf{z})$ *by* $\langle \mathbf{x}, \mathbf{z} \rangle$ *and* $S_t(\mathbf{x})$ *by* $\mathbf{u}_t \mathbf{x}$*,* $t = 1, \ldots, p$*. K is separable by construction.*

It is worth noting that separable kernels are not limited to finite-dimensional output spaces, while transformable kernels are. Figure 4 depicts inclusions among kernel classes discussed here and the two new families of operator-valued kernels we propose: *partial trace* kernels and *entangled* kernels.

We now define the two novel classes of operator-valued kernels. The first one, the class of partial trace kernels, encompasses both (sum of) separable and transformable kernels, while the second, entangled kernels, is a class of non-separable kernels. We start by introducing the more general class of partial trace kernels. The intuition behind this class of kernels is that in the scalar-valued case any kernel function k can be written as the trace of an operator in $\mathcal{L}(\mathcal{K})$, where $\mathcal{K}$ is the reproducing kernel Hilbert space associated to the scalar-valued kernel k. It is easy to see that $k(\mathbf{x}, \mathbf{z}) = \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle = \operatorname{tr}(\phi(\mathbf{x})\phi(\mathbf{z})^\top)$.[2] The following definition of partial trace kernels can be motivated as a generalization of the kernel trick by using the partial trace operator instead of trace.

2. There is some abuse of notation in using the transpose symbol $^\top$ for a feature map representation which can be infinite-dimensional. In this case, we can write $k(\mathbf{x}, \mathbf{z}) = \operatorname{tr}(\phi(\mathbf{x})\phi(\mathbf{z})^*)$, where $\phi(\mathbf{x})\phi(\mathbf{z})^*$ is the rank one operator defined for all $u \in \mathcal{K}$ by $(\phi(\mathbf{x})\phi(\mathbf{z})^*)u = \langle \phi(\mathbf{z}), u \rangle \phi(\mathbf{x})$.

Definition 10 *(Partial trace kernel)*
A partial trace kernel is an operator-valued kernel function K having the following form

$$K(\mathbf{x}, \mathbf{z}) = \mathrm{tr}_{\mathcal{K}}(\mathbf{P}_{\phi(\mathbf{x}),\phi(\mathbf{z})}), \tag{9}$$

where $\mathbf{P}_{\phi(\mathbf{x}),\phi(\mathbf{z})}$ is an operator on $\mathcal{L}(\mathcal{Y} \otimes \mathcal{K})$, and $\mathrm{tr}_{\mathcal{K}}$ is the partial trace on $\mathcal{K}$ (i.e., over the inputs).

Depending on the choice of the operator $\mathbf{P}_{\phi(\mathbf{x}),\phi(\mathbf{z})}$, partial trace kernels may not be positive semi-definite. Since the partial trace preserves positive semi-definiteness (Filipiak et al., 2018), it is clear that the partial trace kernel is positive semi-definite if for every $n \in \mathbb{N}$ and $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathcal{X}$ the matrix $[\mathbf{P}_{\phi(\mathbf{x}_i),\phi(\mathbf{x}_j)}]_{i,j=1}^{n}$ is positive. Even though all the kernel subclasses considered here and shown in Figure 4 are positive semi-definite, our definition of partial trace kernel is general and flexible enough to cover many kernels and allows the design of new ones that may or not be positive semi-definite. As for the scalar-valued case, operator-valued kernels which are not positive semi-definite may be useful for learning in reproducing kernel Kreĭn spaces (Ong et al., 2004; Saha and Palaniappan, 2020).

The class of partial trace kernels is very broad and encompasses the classes of separable and transformable kernels (see Figure 4). From the definition of the partial trace operation, we can see that if we choose $\mathbf{P}_{\phi(\mathbf{x}),\phi(\mathbf{z})} = \sum_l \mathbf{T}_l \otimes \big(\phi_l(\mathbf{x})\phi_l(\mathbf{z})^*\big)$, where $\phi(\mathbf{x})\phi(\mathbf{z})^*$ is the rank one operator defined in Footnote 2, we recover the case of sum of separable kernels. In the same way, if we fix $[\mathbf{P}_{\widetilde{\phi}(\mathbf{x}),\widetilde{\phi}(\mathbf{z})}]_{l,m=1}^{p} = \big(\widetilde{\phi} \circ S_l(\mathbf{x})\big)\big(\widetilde{\phi} \circ S_m(\mathbf{z})\big)^\top$ in Eq. 9, computing the trace of each block using the partial trace will give the transformable kernel. With this in mind, we can use the partial trace kernel formulation to induce a novel class for operator-valued kernels which are not separable, with the goal to characterize inseparable correlations between inputs and outputs.

Definition 11 *(Entangled kernel)*
An entangled operator-valued kernel K is defined as

$$K(\mathbf{x}, \mathbf{z}) = \mathrm{tr}_{\mathcal{K}}\left(\mathbf{U}\big(\mathbf{T} \otimes (\phi(\mathbf{x})\phi(\mathbf{z})^*)\big)\mathbf{U}^*\right), \tag{10}$$

in which $\mathbf{T} \in \mathcal{L}(\mathcal{Y})$ is a positive semi-definite operator, and $\mathbf{U} \in \mathcal{L}(\mathcal{Y} \otimes \mathcal{K})$ is not separable.

When $\mathcal{Y}$ and $\mathcal{K}$ have finite dimensions p and N, respectively, $\mathcal{L}(\mathcal{Y} \otimes \mathcal{K})$ is simply the set of matrices of dimensions $pN \times pN$ and $\phi(\mathbf{x})\phi(\mathbf{z})^*$ is the matrix $\phi(\mathbf{x})\phi(\mathbf{z})^\top$, where $\phi(\mathbf{z})^\top$ denotes the transpose of $\phi(\mathbf{z})$. In the following we abuse the notation and denote N as the dimensionality of feature representation $\phi(\mathbf{x})$. However we do not restrict ourselves to finite dimensions and N in this notation can also be infinite. In this definition, $\mathbf{U}$ not being separable means that it cannot be written as $\mathbf{U} = \mathbf{B} \otimes \mathbf{C}$, with $\mathbf{B} \in \mathbb{R}^{p \times p}$ and $\mathbf{C} \in \mathbb{R}^{N \times N}$. The term $\mathbf{T} \otimes (\phi(\mathbf{x})\phi(\mathbf{z})^\top)$ represents a separable kernel function over inputs and outputs, while $\mathbf{U}$ characterizes the entanglement shared between them. We note that $\mathbf{U}$ not being equal to $\mathbf{B} \otimes \mathbf{C}$ marks the crucial difference to separable kernels; if $\mathbf{U}$ were $\mathbf{B} \otimes \mathbf{C}$, then the class described above would be part of separable kernels (see Theorem 2).

Some intuition to $\mathbf{U}$ can be seen from its role of an "entangled" similarity in the joint feature space. It is entangled in the sense that it cannot be decomposed into two "sub"-matrices of similarity between inputs and between outputs independently. The partial

trace is the operation used to recover the sub-similarity matrix between the outputs from the entangled joint similarity matrix. In the particular case of separability, the partial trace will give the output metric.

Theorem 12 *Entangled kernels given by the Definition 11 are positive semi-definite kernels.*

Proof The proof is based on the observation that $\mathbf{T} \otimes (\phi(\mathbf{x})\phi(\mathbf{x})^\top) = \mathbf{O}_{\phi(\mathbf{x})}\mathbf{T}\mathbf{O}^*_{\phi(\mathbf{x})}$, where $\mathbf{O}_{\phi(\mathbf{x})}$ is the operator defined by

$$\begin{aligned}\mathbf{O}_{\phi(\mathbf{x})} : \mathcal{Y} &\longrightarrow \mathcal{Y} \otimes \mathcal{K} \\ \mathbf{y} &\longmapsto \mathbf{y} \otimes \phi(\mathbf{x}),\end{aligned}$$

and $\mathbf{O}^*_{\phi(\mathbf{x})}$ is its adjoint defined by:

$$\begin{aligned}\mathbf{O}^*_{\phi(\mathbf{x})} : \mathcal{Y} \otimes \mathcal{K} &\longrightarrow \mathcal{Y} \\ \mathbf{y} \otimes u &\longmapsto \langle \phi(\mathbf{x}), u\rangle \mathbf{y}.\end{aligned}$$

Indeed, $\forall\, \mathbf{y} \in \mathcal{Y}, u \in \mathcal{K}$, we have

$$\begin{aligned}\big(\mathbf{T} \otimes (\phi(\mathbf{x})\phi(\mathbf{z})^\top)\big)(\mathbf{y} \otimes u) &= \mathbf{T}\mathbf{y} \otimes \phi(\mathbf{x})\phi(\mathbf{z})^\top u \\ &= \langle \phi(\mathbf{z}), u\rangle \mathbf{T}\mathbf{y} \otimes \phi(\mathbf{x}) \\ &= \mathbf{O}_{\phi(\mathbf{x})}\mathbf{T}\langle \phi(\mathbf{z}), u\rangle \mathbf{y} \\ &= \mathbf{O}_{\phi(\mathbf{x})}\mathbf{T}\mathbf{O}^*_{\phi(\mathbf{z})}(\mathbf{y} \otimes u).\end{aligned}$$

Now, to show that the entangled kernel defined by Eq. 10 is positive semi-definite, let us compute

$$\begin{aligned}\sum_{i,j} \langle \mathbf{y}_i, K(\mathbf{x}_i, \mathbf{x}_j)\mathbf{y}_j\rangle_{\mathcal{Y}} &= \sum_{i,j} \left\langle \mathbf{y}_i, \mathrm{tr}_{\mathcal{K}} \left(\mathbf{U}\big(\mathbf{T} \otimes (\phi(\mathbf{x}_i)\phi(\mathbf{x}_j)^\top)\big)\mathbf{U}^\top\right) \mathbf{y}_j \right\rangle_{\mathcal{Y}} \\ &= \sum_{i,j} \mathrm{tr}\left(\big(\mathbf{y}_i^\top \otimes \mathbf{I}_N\big)\mathbf{U}\big(\mathbf{T} \otimes (\phi(\mathbf{x}_i)\phi(\mathbf{x}_j)^\top)\big)\mathbf{U}^\top \big(\mathbf{y}_j \otimes \mathbf{I}_N\big)\right) \quad {}^{3} \\ &= \sum_{i,j} \mathrm{tr}\left(\big(\mathbf{y}_i^\top \otimes \mathbf{I}_N\big)\mathbf{U}\big(O_{\phi(\mathbf{x}_i)}\mathbf{T}O^*_{\phi(\mathbf{x}_j)}\big)\mathbf{U}^\top \big(\mathbf{y}_j \otimes \mathbf{I}_N\big)\right) \\ &= \mathrm{tr}(S\mathbf{T}S^*) \geq 0,\end{aligned}$$

as clearly $\mathrm{tr}(S\mathbf{T}S^*) = \mathrm{tr}(S\mathbf{V}\mathbf{V}^\top S^*) = \mathrm{tr}((S\mathbf{V}(S\mathbf{V})^*)$ and trace preserves positivity. Here we have denoted $S = \sum_i \big(\mathbf{y}_i^\top \otimes \mathbf{I}_N\big)\mathbf{U}O_{\phi(\mathbf{x}_i)}$. Moreover, from the fact that $\mathrm{tr}_{\mathcal{K}}(\mathbf{A}^\top) = \mathrm{tr}_{\mathcal{K}}(\mathbf{A})^\top$, we immediately obtain that $K(\mathbf{x}, \mathbf{z}) = K(\mathbf{z}, \mathbf{x})^*$, which completes the proof. ■

While by definition entangled kernels cannot be separable, the following example shows that an entangled kernel can be transformable.

3. See Filipiak et al. (2018, Lemma 2.11).

Example 3 *(Entangled and transformable kernel)*
An entangled kernel given by Definition 11 with symmetric and supersymmetric $\mathbf{U}$ *(that is, all its* $N \times N$ *blocks are symmetric),* $\mathbf{T} = \mathbb{1}_p \mathbb{1}_p^T \in \mathbb{R}^{p \times p}$ *and with a scalar-valued kernel with finite-dimensional feature mapping,* $\phi(\mathbf{x}) \in \mathbb{R}^N$*, is a transformable kernel.*

This can be seen from the following calculation:

$$\begin{aligned}
\mathbf{P}_{\phi(\mathbf{x}),\phi(\mathbf{z})} &= \left[\sum_{i,j=1}^{p} \mathbf{U}_{lj} \phi(\mathbf{x}) \phi(\mathbf{z})^\top \mathbf{U}_{im}^\top \right]_{l,m=1}^{p} \\
&= \left[\left(\sum_{j=1}^{p} \mathbf{U}_{lj} \phi(\mathbf{x}) \right) \left(\sum_{i=1}^{p} \mathbf{U}_{im} \phi(\mathbf{z}) \right)^\top \right]_{l,m=1}^{p} \\
&= \left[\left(\sum_{j=1}^{p} \mathbf{U}_{lj} \phi(\mathbf{x}) \right) \left(\sum_{i=1}^{p} \mathbf{U}_{mi} \phi(\mathbf{z}) \right)^\top \right]_{l,m=1}^{p} .
\end{aligned}$$

Now $\mathbf{S}_l(\mathbf{x}) = \sum_{k=1}^{p} \mathbf{U}_{lk} \phi(\mathbf{x})$*; recall that transformable kernels can be obtained from partial trace kernels (9) with* $[\mathbf{P}_{\widetilde{\phi}(\mathbf{x}),\widetilde{\phi}(\mathbf{z})}]_{l,m=1}^{p} = \big(\widetilde{\phi} \circ S_l(\mathbf{x})\big)\big(\widetilde{\phi} \circ S_m(\mathbf{z})\big)^\top$.

Choice of the matrix $\mathbf{U}$ is crucial to the class of entangled kernels. In the next section we develop an algorithm that learns an entangled kernel from data.

5. Entangled Kernel Learning

In general, there is no knowing whether input and output data are or are not entangled. In this sense, learning the entangled K in Eq. 10 by imposing that $\mathbf{U}$ is inseparable can sometimes be restrictive. In our entangled kernel learning approach we do not impose any separability restriction, with the hope that our learning algorithm can automatically detect the lack or presence of entanglement. Key to our method is a reformulation of the entangled kernel K (Eq. 10) via Choi-Kraus representation.

Theorem 13 *(Choi-Kraus representation Choi, 1975; Kraus, 1983; Rieffel and Polak, 2011) The map* $K(\mathbf{x}, \mathbf{z}) = \mathrm{tr}_{\mathcal{K}} \left(\mathbf{U} \big(\mathbf{T} \otimes (\phi(\mathbf{x}) \phi(\mathbf{z})^\top) \big) \mathbf{U}^\top \right)$ *can be generated by an operator sum representation*

$$K(\mathbf{x}, \mathbf{z}) = \sum_{i=1}^{r} \mathbf{M}_i \phi(\mathbf{x}) \phi(\mathbf{z})^\top \mathbf{M}_i^\top, \tag{11}$$

where r *is called the* Kraus rank *and* $\mathbf{M}_i \in \mathbb{R}^{p \times N}$ *are the* Kraus operators*.*

Proof See Appendix A ■

Using this formulation, entangled kernel learning consists of finding a (possibly low-rank when r is small) decomposition of the kernel by learning the matrices $\mathbf{M}_i$, $i = 1, \ldots, r$, where these matrices "merge" the matrices $\mathbf{T}$ and $\mathbf{U}$.

Looking at the simpler class of separable kernels, they model the relations between tasks (i.e., ouputs) using only the output matrix $\mathbf{T}$ of size $p \times p$. The matrix $\mathbf{T}$ acts as a

covariance matrix on the labels independently of the inputs and if connected to the existing deep neural network approaches for multi-task learning (i.e. the task-decoder methods which learn a shared representation at a high-level layer followed by task-specific decoders, see, e.g., Meyerson and Miikkulainen, 2018), can then play the role of a decoder that predicts labels using the outputs of related tasks. Entangled kernels, however, capture task relatedness through the matrices $\mathbf{M}_i$, of size $p \times N$ modeling the relationships of tasks and features. A small value of r is a (Kraus) low-rank assumption which reduces the number of parameters and promotes sharing information and knowledge between tasks. This would be more similar to the column-based approaches in deep multi-task learning, which consider multiple deep neural networks in parallel and share the layer parameters between these networks (Meyerson and Miikkulainen, 2018).

It is not easy to see from the theorem how exactly $\mathbf{T}$, $\mathbf{U}$ and $\mathbf{M}_i$ interact. While the proof of the theorem (see Appendix A) gives the explicit relation between them, in order to make this more clear let us consider only one element of the output of the kernel. From the Choi-Kraus representation (11) we have

$$[K(\mathbf{x}, \mathbf{z})]_{s,t} = \sum_{i=1}^{r} \mathbf{M}_i[s, :]\phi(\mathbf{x})\phi(\mathbf{z})^\top (\mathbf{M}_i[t, :])^\top,$$

and on the other hand from the definition of the entangled kernels (Definition 11) we get that

$$[K(\mathbf{x}, \mathbf{z})]_{s,t} = tr\left(\sum_{l=1}^{p}\sum_{k=1}^{p} \mathbf{U}_{sk}\mathbf{T}_{kl}\phi(\mathbf{x})\phi(\mathbf{z})^\top \mathbf{U}_{lt}^\top\right) = \sum_{l,k=1}^{p} \mathbf{T}_{kl}\, \phi(\mathbf{z})^\top \mathbf{U}_{lt}^\top \mathbf{U}_{sk}\phi(\mathbf{x}).$$

Here we have used notation $\mathbf{M}_i[s, :]$ to refer to the row s of matrix $\mathbf{M}_i$, $\mathbf{T}_{ij}$ to refer to the element in position (i, j) in $\mathbf{T}$, and $\mathbf{U}_{ij}$ similarly ordered to the block of size $N \times N$ in $\mathbf{U}$. From this we see that both the rows of $\mathbf{M}_i$ and rows of (blocks of) $\mathbf{U}$ act on transforming the features $\phi(\mathbf{x})$. If we restrict the "rank" of the entangled kernel by restricting the r in (11), we are in essence restricting the row space of (blocks of) $\mathbf{U}$.

It is important to note, that while every entangled kernel can be represented like this, the representation is not restricted only to entangled kernels. Thus by learning the $\mathbf{M}_i$ we expect to learn the meaningful relationships in the data, be they entangled or not. Yet, when learning even a low-rank entangled kernel, we are bound to learn more parameters in the $\mathbf{M}_i$ than we could learn from just (full-rank) $\mathbf{T}$, thus making the class of kernels we consider much more expressive.

To make this explicit, let us consider the separable kernels in the Choi-Kraus represenation framework. For the class of separable kernels $\mathbf{U}$ equals identity, and the $\mathbf{T}$ in (10) is the same as the $\mathbf{T}$ in Definition 7. We can describe the operator $\mathbf{T}$ with a set of vectors $\mathbf{t}_j \in \mathbb{R}^p$ for which $\mathbf{T} = \sum_j \mathbf{t}_j \mathbf{t}_j^\top$.

Remark 14 *(Choi-Kraus representation of seprabale kernels)*
For separable kernels, each $\mathbf{M}_i$ in the Choi-Kraus represenation (11) can be described as $\mathbf{M}_i = \mathbf{M}_{jk} = \mathbf{t}_j \mathbf{e}_k^\top$, with $j = 1, ..., p$, $k = 1, ..., N$ and $\{\mathbf{e}_k\}_k$ is an orthonormal basis of $\mathbb{R}^N$.

This can be confirmed with the following calculation:

$$\begin{aligned} K(\mathbf{x},\mathbf{z}) &= \sum_{i=1}^{r} \mathbf{M}_i \phi(\mathbf{x})\phi(\mathbf{z})^\top \mathbf{M}_i^\top = \sum_{j=1}^{p}\sum_{k=1}^{N} \mathbf{t}_j \mathbf{e}_k^\top \phi(\mathbf{x})\phi(\mathbf{z})^\top \mathbf{e}_k \mathbf{t}_j^\top \\ &= \sum_{k=1}^{N} \mathbf{e}_k^\top \phi(\mathbf{x})\phi(\mathbf{z})^\top \mathbf{e}_k \sum_{j=1}^{p} \mathbf{t}_j \mathbf{t}_j^\top = \sum_{k=1}^{N} \langle \phi(\mathbf{x})\phi(\mathbf{z})^\top \mathbf{e}_k, \mathbf{e}_k \rangle \mathbf{T} \\ &= \operatorname{tr}\left(\phi(\mathbf{x})\phi(\mathbf{z})^\top\right) \mathbf{T} = \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle \mathbf{T} = k(\mathbf{x},\mathbf{z})\,\mathbf{T}. \end{aligned}$$

It is clear that the class of entangled kernels is much more expressive than the class of separable kernels. With this in mind, we now turn our attention to describing the framework in which we can learn the $\mathbf{M}_i$ in entangled kernels.

Because the feature space of the scalar-valued kernel can be of very large dimensionality (or infinite-dimensional), we consider an approximation to speed up the computation. For example random Fourier features or Nyström approximation (Rahimi and Recht, 2008; Williams and Seeger, 2001) give us $\hat{\phi}$ such that

$$k(\mathbf{x},\mathbf{z}) = \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle \approx \langle \hat{\phi}(\mathbf{x}), \hat{\phi}(\mathbf{z}) \rangle.$$

We note that our approximation is on scalar-valued kernels, not operator-valued, although there are methods for approximating them, too, directly (Brault et al., 2016; Minh, 2016). Our approximated kernel is thus

$$\hat{K}(\mathbf{x},\mathbf{z}) = \sum_{i=1}^{r} \hat{\mathbf{M}}_i \hat{\phi}(\mathbf{x})\hat{\phi}(\mathbf{z})^\top \hat{\mathbf{M}}_i^\top, \tag{12}$$

where $\hat{\phi}(\mathbf{x}) \in \mathbb{R}^m$ and $\hat{\mathbf{M}}_i \in \mathbb{R}^{p\times m}$, from where our goal is to learn the $\hat{\mathbf{M}}_i$. We can write our $np \times np$ kernel matrix as

$$\begin{aligned} \hat{\mathbf{G}} &= \sum_{i=1}^{r} \operatorname{vec}(\hat{\mathbf{M}}_i \hat{\mathbf{\Phi}}) \operatorname{vec}(\hat{\mathbf{M}}_i \hat{\mathbf{\Phi}})^\top \qquad (13) \\ &= \sum_{i=1}^{r} (\hat{\mathbf{\Phi}}^\top \otimes \mathbf{I}_p) \operatorname{vec}(\hat{\mathbf{M}}_i) \operatorname{vec}(\hat{\mathbf{M}}_i)^\top (\hat{\mathbf{\Phi}} \otimes \mathbf{I}_p) \quad [4] \end{aligned}$$

where $\hat{\mathbf{\Phi}} = [\hat{\phi}(\mathbf{x}_1), \cdots, \hat{\phi}(\mathbf{x}_n)]$ is of size $m \times n$. Further, if we denote

$$\mathbf{D} = \sum_{i=1}^{r} \operatorname{vec}(\hat{\mathbf{M}}_i) \operatorname{vec}(\hat{\mathbf{M}}_i)^\top, \tag{14}$$

we can simply write

$$\hat{\mathbf{G}} = (\hat{\mathbf{\Phi}}^\top \otimes \mathbf{I}_p)\mathbf{D}(\hat{\mathbf{\Phi}} \otimes \mathbf{I}_p). \tag{15}$$

To learn an entangled kernel, we need to learn the psd matrix $\mathbf{D}$.

4. Matrix cookbook Eq. (520): `http://www.math.uwaterloo.ca/~hwolkowi/matrixcookbook.pdf`.

It is good to note that also from this formulation of the learnable entangled kernel, we can easily recover the class of separable kernels. Indeed, as we already mentioned, the representation (11) is not restricted to only entangled kernels, and thus it is possible to recover other frameworks also from (15). In this case, if we choose $\mathbf{D} = (\mathbf{I}_N \otimes \mathbf{T})$ with $\mathbf{T} \in \mathbb{R}^{p\times p}$ (note that requirement for $\mathbf{D}$ to be psd and symmetric also restricts $\mathbf{T}$ to be such), we can write

$$\hat{\mathbf{G}} = (\hat{\mathbf{\Phi}}^\top \otimes \mathbf{I}_p)(\mathbf{I}_N \otimes \mathbf{T})(\hat{\mathbf{\Phi}} \otimes \mathbf{I}_p) = (\hat{\mathbf{\Phi}}^\top \mathbf{I}_N \hat{\mathbf{\Phi}} \otimes \mathbf{I}_p \mathbf{T} \mathbf{I}_p) = \mathbf{K} \otimes \mathbf{T},$$

and see that we have recovered a separable kernel. Notably, we can see that our proposed class of learnable kernels are much more expressive than that of separable kernels, as they can be recovered as a special case.

5.1 The Learning Algorithm

We will now describe how to learn the psd matrix $\mathbf{D}$ from the entangled kernel (15), and how to efficiently formulate the vector-valued learning problem.

Kernel alignment (Cristianini et al., 2002; Cortes et al., 2012) is a measure of similarity between two (scalar-valued) kernels. Alignment between two matrices $\mathbf{M}$ and $\mathbf{N}$ is defined as

$$A(\mathbf{M}, \mathbf{N}) = \frac{\langle \mathbf{M}_c, \mathbf{N}_c \rangle_F}{\|\mathbf{M}_c\|_F \|\mathbf{N}_c\|_F}, \tag{16}$$

where subscript c refers to centered matrices, that is, $\mathbf{M}_c = \mathbf{HMH}$ where $\mathbf{H} = \mathbf{I}_n - \frac{1}{n}\mathbf{1}\mathbf{1}^\top$, if $\mathbf{M}$ is a $n \times n$ matrix. Here $0 \leq A(\mathbf{M}, \mathbf{N}) \leq 1$, with higher value showing better alignment and higher similarity. Kernel alignment has been used in learning a kernel by considering the so-called ideal kernel as the target of the alignment. In the context of binary classification, the ideal kernel is defined as $\mathbf{y}\mathbf{y}^\top$ where $\mathbf{y} = [y_1, ..., y_n]^\top$ is the n-vector of class labels (Cristianini et al., 2002). The work of Kandola et al. (2002) extends the ideal kernel to regression and to unbalanced classification where there are more samples from one class than from the other.

We extend the concept of alignment into the case of multiple outputs. As already mentioned, in the previous works the ideal kernel has been the linear kernel defined on output values, $\mathbf{y}\mathbf{y}^\top$. Extended to multi-output setting, it is natural to consider the linear kernel $\mathbf{K}_Y = \mathbf{Y}^\top \mathbf{Y}$, where $\mathbf{Y}$ is of size $p \times n$, containing the labels associated to data sample i on its ith column. Yet, if we wish to use this ideal kernel in learning our entangled kernel we face problems as we would be trying to align a $np \times np$ matrix with a $n \times n$ one. We thus propose as the first part of our optimization problem to align the partial trace of our entangled kernel matrix, $\mathrm{tr}_p(\hat{\mathbf{G}})$, to the output kernel $\mathbf{K}_Y$. As this term does not consider the full operator-valued kernel matrix, for the second term we consider $\mathbf{y} = \mathrm{vec}(\mathbf{Y})$, a vectorization of the matrix containing the labels, and take matrix $\mathbf{K}_y = \mathbf{y}\mathbf{y}^\top$ as the second ideal kernel in our setting. In this kernel each $p \times p$ block is an outer product between the labels associated with the samples, or $[\mathbf{K}_y]_{ij} = \mathbf{y}_i \mathbf{y}_j^\top$, and we can directly align the $np \times np$ matrix $\mathbf{K}_y$ to the full entangled kernel $\hat{\mathbf{G}}$. Our optimization problem is thus a convex combination

$$\max_{\mathbf{D}} \quad (1-\gamma)\, A\left(\mathrm{tr}_p(\hat{\mathbf{G}}), \mathbf{Y}^\top \mathbf{Y}\right) + \gamma\, A\left(\hat{\mathbf{G}}, \mathbf{y}\mathbf{y}^\top\right), \tag{17}$$

Algorithm 1 Entangled Kernel Learning (EKL)

Input: matrix of features $\mathbf{\Phi}$, labels $\mathbf{Y}$
// 1) Kernel learning:
Solve for $\mathbf{Q}$ in eq.17 ($\mathbf{D} = \mathbf{Q}\mathbf{Q}^\top$) within a sphere manifold
// 2) Learning the predictive function:
if Predict with scalar-valued kernel **then**
  $\mathbf{c}_K = (\mathrm{tr}_p(\hat{\mathbf{G}}) + \lambda\mathbf{I})^{-1}\mathbf{Y}^\top$ $\quad \mathcal{O}(m^3 + mnp)$
else
  $\mathbf{c}_G = (\hat{\mathbf{G}} + \lambda\mathbf{I})^{-1}\,\mathrm{vec}(\mathbf{Y})$ $\quad \mathcal{O}(r^3 + mnp^2)$
Return $\mathbf{D} = \mathbf{Q}\mathbf{Q}^\top$, $\mathbf{c}$

where $\gamma \in [0, 1]$. We note that by applying Lemma 2.11 from Filipiak et al. (2018), we can write $\mathrm{tr}_p(\hat{\mathbf{G}}) = \hat{\mathbf{\Phi}}^\top \mathrm{tr}_p(\mathbf{D})\hat{\mathbf{\Phi}}$.

Intuitively the first alignment learns a scalar-valued kernel matrix that can be obtained via partial trace applied to the more complex operator-valued kernel, while the second term focuses on the possibly entangled relationships in the data. One possibility for using the entangled kernel framework is to learn a scalar-valued kernel for multi-output problem using the partial-trace formulation and to use it in a kernel machine.

The optimization problem is solved with a gradient-based approach. To make sure that the resulting kernel is valid (psd), we write $\mathbf{D} = \mathbf{Q}\mathbf{Q}^\top$ with $\mathbf{Q}$ of size $mp \times r$ with r at most mp, and perform the optimization over $\mathbf{Q}$. The gradients for alignment terms are straightforward to calculate. The optimization is performed on sphere manifold as a way to normalize $\mathbf{D}$.[5]

After we have learned the entangled kernel, we solve the learning problem by choosing the squared loss function

$$\min_{\mathbf{c}} \|\,\mathrm{vec}(\mathbf{Y}) - \hat{\mathbf{G}}\mathbf{c}\|^2 + \lambda\langle\hat{\mathbf{G}}\mathbf{c}, \mathbf{c}\rangle. \tag{18}$$

For this $\mathbf{c}$ update we can find the classical closed-form solution, $\mathbf{c} = (\hat{\mathbf{G}}+\lambda\mathbf{I})^{-1}\,\mathrm{vec}(\mathbf{Y})$. Note that this computation is, by considering the entangled structure of $\hat{\mathbf{G}}$, more computationally efficient than a general (say, some transformable) operator-valued kernel. Generally we can say that the complexity of calculating the predictive function with nonseparable operator-valued kernels is $\mathcal{O}(n^3p^3)$. In our proposed network, however, we can apply the Woodbury formula for the matrix inversion and only invert a $r \times r$ matrix. We can assume that $m \ll n$ and often in multi-output data sets $p \ll n$. Furthermore as $r \leq mp$, we can see that n dominates the complexity calculations. Thus we write the complexity of the $\mathbf{c}$-update as $\mathcal{O}(r^3 + mnp^2)$, where we have also considered the most costly steps of the matrix multiplications involved.

Moreover it is possible, with our kernel learning framework, to learn a scalar-valued kernel by first learning the entangled kernel as proposed and then extracting the scalar-valued one by using partial trace operator. The scalar-valued kernel proposed to be used in predicting is now $\mathbf{K} = \mathrm{tr}_p(\hat{\mathbf{G}})$, and it can be used in regression setting as usual; calculating $\mathbf{c}_K = (\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{Y}^\top$ and using that to obtain predictions. We note that even here our

5. We used the toolbox from `pymanopt.github.io` for the implementation (Townsend et al., 2016).

	learning $\mathbf{c}$	predicting $\mathbf{Y}$
No structure	$\mathcal{O}(n^3p^3)$	$\mathcal{O}(tnp^2)$
Separable	$\mathcal{O}(n^3+p^3)$	$\mathcal{O}(tnp+np^2)$
Low-rank separable	$\mathcal{O}(m^3+m^2n+p^3)$	$\mathcal{O}(tp^2+(n+t)pm)$
Entangled	$\mathcal{O}(r^3+mnp^2)$	$\mathcal{O}((t+n+r)pm)$
Entangled, ptr	$\mathcal{O}(m^3+mnp)$	$\mathcal{O}(tm^2+(n+t)pm)$

Table 2: Complexities of calculating parameters of predictive function (vector $\mathbf{c}$), and predicting labels (calculating $\mathbf{G}_t\mathbf{c}$ with $\mathbf{G}_t$ of size $tp \times np$) with various operator-valued kernels. The "low-rank separable" refers to the case when the scalar-valued kernel matrix of the separable kernel has a low-rank approximation, i.e. $\mathbf{G} = \mathbf{K} \otimes \mathbf{T} = \mathbf{U}\mathbf{U}^\top \otimes \mathbf{T}$ with some $\mathbf{U}$ of size $n \times m$ with $m < n$.

framework brings forward advancements in computational complexity; after having an entangled kernel the cost of calculating $\mathbf{c}_K$ using again Woodbury formula is centered around inverting a $m \times m$ matrix. Again when we assume $m \ll n$, and consider the most dominant term arising from the matrix multiplications, the complexity of the $\mathbf{c}_K$-step is $\mathcal{O}(m^3+mnp)$.

There are also gains in predictive complexity. Predicting with a general operator-valued kernel has the complexity $\mathcal{O}(tnp^2)$. With an entangled kernel of our formulation, this reduces to $\mathcal{O}((n+t+r)mp)$. For the ptrEKL kernel, assuming partial trace of $\mathbf{D}$ is already calculated, the predictive complexity is $\mathcal{O}(tm^2+(n+t)mp)$ instead of $\mathcal{O}(tnp)$, beneficial with small m. The complexities of calculating the parameters ($\mathbf{c}$) of the predictive function and predicting with various operator-valued kernels are summarized in Table 2. It is good to note that the complexities for separable kernels could be reduced by approximating the scalar-valued kernel matrix, as well as that it would be also possible to approximate a general operator-valued kernel matrix. While we show this approximation for separable kernels in Table 2 for completeness, it is good to remember that these kind of approximations are not intrinsic to the approaches, unlike with our entangled kernels. Additionally we have assumed in the complexities for general and separable kernels that the big operator-valued kernel matrix $\mathbf{G}$ and scalar-valued kernel matrix $\mathbf{K}$ as well as the output matrix $\mathbf{T}$ are already known at the time of the computations. Similarly, we have assumed that for entangled kernels the Φ and $\mathbf{Q}$ are already available at the time of these calculations.

5.2 Rademacher Generalization Bound

We now provide a generalization analysis of our EKL algorithm using Rademacher complexities (Bartlett and Mendelson, 2002). The notion of Rademacher complexity has been generalized to vector-valued hypothesis spaces (Maurer, 2006; Sindhwani et al., 2013; Sangnier et al., 2016). Previous work has analyzed the case where the matrix-valued kernel is fixed prior to learning, while our analysis considers the kernel learning problem, similarly to the bound in Huusari et al. (2018). It provides a Rademacher bound for our algorithm when both the vector-valued function f and the kernel via $\mathbf{Q}$, are learnt. We start by recalling that the feature map associated to the matrix-valued kernel K is the mapping

$\Gamma : \mathcal{X} \to \mathcal{L}(\mathcal{Y}, \mathcal{H}_K)$, where $\mathcal{X}$ is the input space, $\mathcal{Y} = \mathbb{R}^p$, and $\mathcal{L}(\mathcal{Y}, \mathcal{H}_K)$ is the set of bounded linear operators from $\mathcal{Y}$ to $\mathcal{H}_K$ (see, e.g., Micchelli and Pontil, 2005; Carmeli et al., 2010 for more details). It is known that $K(\mathbf{x}, \mathbf{z}) = \Gamma(\mathbf{x})^* \Gamma(\mathbf{z})$. We denote by $\Gamma_{\mathbf{Q}}$ the feature map associated to our entangled kernel (Equation 15).

The hypothesis class of EKL is

$$\mathcal{H}_\beta = \{x \mapsto f_{u,\mathbf{Q}}(\mathbf{x}) = \Gamma_{\mathbf{Q}}(\mathbf{x})^* u : \mathbf{Q} \in \Delta, \|u\|_{\mathcal{H}} \leq \beta\},$$

with $\Delta = \{\mathbf{Q} : \|\mathbf{Q}\|_F = 1\}$ and β is a regularization parameter. Let $\boldsymbol{\sigma}_1, \ldots, \boldsymbol{\sigma}_v$ be an iid family of vectors of independent Rademacher variables where $\boldsymbol{\sigma}_i \in \mathbb{R}^p$, $\forall\, i = 1, \ldots, n$. The empirical Rademacher complexity of the vector-valued class $\mathcal{H}_\beta$ is the function $\hat{\mathcal{R}}_n(\mathcal{H}_\beta)$ defined as

$$\hat{\mathcal{R}}_n(\mathcal{H}_\beta) = \frac{1}{n} \mathbb{E}\left[\sup_{f \in \mathcal{H}} \sup_{\mathbf{Q} \in \Delta} \sum_{i=1}^{n} \boldsymbol{\sigma}_i^\top f_{u,\mathbf{Q}}(\mathbf{x}_i) \right].$$

Theorem 15 *(Rademacher complexity bound for EKL)*
The empirical Rademacher complexity of $\mathcal{H}_\beta$ can be upper bounded as

$$\hat{\mathcal{R}}_n(\mathcal{H}_\beta) \leq \beta \sqrt{\frac{\kappa p}{n}}$$

for kernels k that satisfy $k(\mathbf{x}, \mathbf{x}) \leq \kappa, \forall \mathbf{x} \in \mathcal{X}$.

The proof for the theorem can be found in Appendix B. We now make use of this result to bound the generalization error of EKL for the case where the second stage of the algorithm is kernel ridge regression (Equation 18). Similar results can be given using other algorithms such as SVMs in the second stage.

Corollary 16 *(Generalization bound for EKL)*
Let $M > 0$. Assume that $\|f(\mathbf{x}) - \mathbf{y}\|_{\mathcal{Y}} \leq M$ for all $(\mathbf{x}, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$ and $f \in \mathcal{H}_\beta$. Then, the following holds with probability larger than $1 - \delta$ over samples of size n for all $f \in \mathcal{H}_\beta$:

$$R(f) \leq \hat{R}(f) + 4\sqrt{2} M \sqrt{\frac{\beta^2 \kappa p}{n}} + 3M \sqrt{\frac{\log \frac{2}{\delta}}{2n}}, \tag{19}$$

where $R(f) := \mathbb{E}_{(\mathbf{x},\mathbf{y})}\left[\|f(\mathbf{x}) - \mathbf{y}\|_{\mathcal{Y}}^2\right]$ is the expected risk w.r.t. the square loss and $\hat{R}(f) := \frac{1}{n} \sum_{i=1}^{n} \|f(\mathbf{x}_i) - \mathbf{y}_i\|_{\mathcal{Y}}^2$ is the empirical risk of f.

The proof of Corollary 16, which is deferred to Appendix C, is based on a general Rademacher complexity learning bound provided in Bartlett and Mendelson (2002) (see also Mohri et al. 2018, Theorem 3.3) and a vector-contraction inequality for the Rademacher complexity of classes of vector-valued functions proved in Maurer (2016).

6. Experiments

In this section we investigate the performance of our algorithm with real and artificial data sets. We start with an illustration of its behaviour, and move on to experiments on the predictive performance. In this latter setting, we compare our proposed Entangled Kernel Learning (EKL)[6] algorithm to OKL (Dinuzzo et al., 2011); a kernel learning method for

6. Code is available at RH's personal website, `riikkahuusari.com`.

separable kernels (we use the code provided by the authors [7]), and KRR; kernel ridge regression. Furhtermore, we investigate performance of predicting with scalar-valued kernel extracted from the operator-valued kernel matrix EKL learns, and call this ptrEKL. In all the experiments we cross-validate over various regularization parameters λ, and for EKL also the γs controlling the combination of alignments. In the experiments we consider (normalized) mean squared error (nMSE) and normalized improvement to KRR (nI) (Ciliberto et al., 2015) as error measures. Finally we conclude the experimental section by comparing the performance of learning and predicting with various operator-valued kernels, showing the advantage of entangled kernels.

6.1 Illustration

We consider the digits data set available at scikit-learn [8] (Pedregosa et al., 2011) and make it into multi-task data set by encoding class-memberships into vectors consisting of values $-1, +1$. We consider only the first four classes (digits 0-3) for clarity of illustration; in this case for example label vector $\begin{bmatrix}-1 & -1 & +1 & -1\end{bmatrix}$ stands for digit 2. We took 25 data samples from each class as training samples, and further 25 from each for test data samples.

We trained EKL by restricting the number of columns in matrix $\mathbf{Q}$, r, to two. This drastic reduction of learnable parameters is not expected to produce the best accuracy (as shown in the later experiments), but instead to allow us to visualize and illustrate a part on how EKL works.

In this setting the learned EKL matrix is not separable, according to the PPT condition (end of Section 2.1). Figure 5 shows parts of the entangled kernel matrix, so that in each part the samples making it up belong to one class of digits. We can see that for each class the task relationships are modelled differently. This is vastly different from the OKL approach, where the $p \times p$ output matrix has the same structure for all the data samples, as we illustrated in Figure 1. In our case the output matrix structure is extremely dependent on the inputs.

Another way to see this dependency is via an application to supervised multi-task dimensionality reduction. We note that $\mathbf{Z} = (\hat{\mathbf{\Phi}}^\top \otimes \mathbf{I}_p)\mathbf{Q}$, is a matrix of size $np \times r$ and acts as an approximated feature map of the operator-valued kernel matrix, in the sense that the full kernel is $\mathbf{Z}\mathbf{Z}^\top$. We can interpret the $\mathbf{Z}$ to give dimensionality reduction of the data with respect to all the outputs individually, as in we have p dimensionality reductions of size $n \times r$. As a way to project data to lower dimensions, our illustrative approach closely resembles supervised dimensionality reduction (Fukumizu et al., 2004; Sugiyama, 2006). We note that the focus of our work is not in this specific task, nor is our framework directly applicable to those works. We consider multi-output learning, making the application of EKL to this problem the first in supervised multi-task dimensionality reduction, as far as we know.

Figure 6 shows the results of the low-dimensional projection for each of the tasks. The supervised dimensionality reduction is successful for all the tasks: we can see that each task is modelled individually, as the samples corresponding to task under question are projected to a separate cluster from the other samples. An advantage of our approach is that our

7. http://people.tuebingen.mpg.de/fdinuzzo/okl.html
8. https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_digits.html

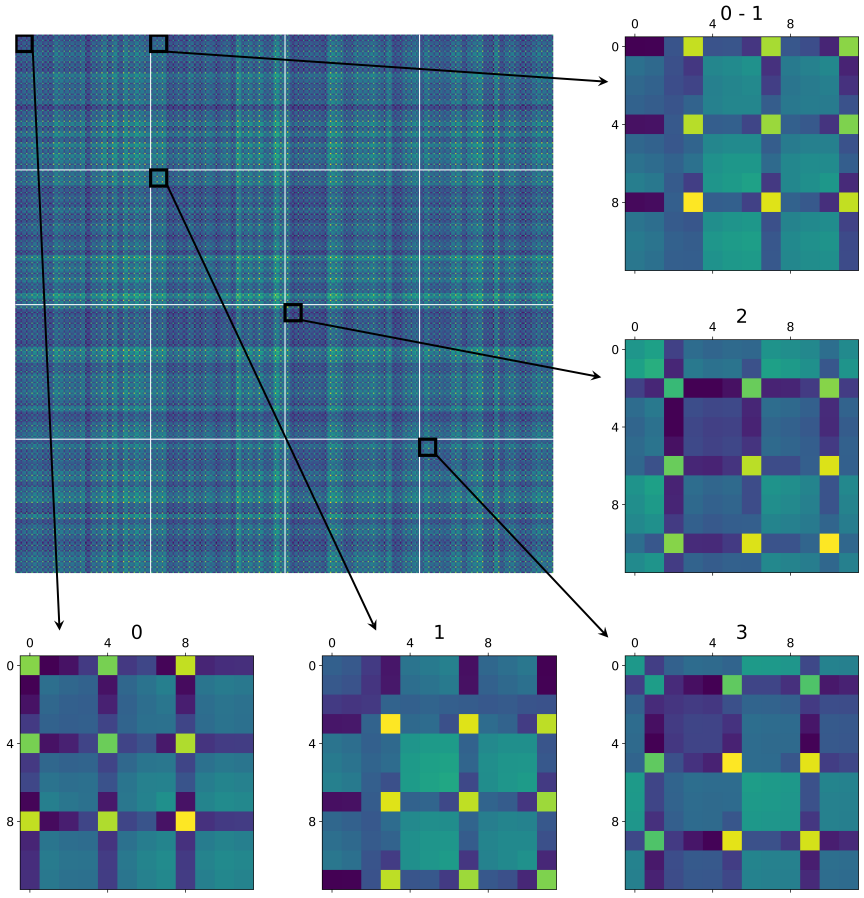

Figure 5: Small selections (bottom and right images) of the $np \times np$ entangled kernel matrix $\mathbf{G}$ (top left) trained on digits data set. The data in big matrix is ordered so, that all samples of class 0 come first, then 1, 2 and finally of 3, and for clarity of illustration white lines separate the classes. Each small selection contains 3 samples of the class(es) indicated in the title of the subfigure. For this data the output of the kernel function is a 4×4 matrix. The structure of the output matrix is extremely dependent on the inputs. The off-diagonal $p \times p$ blocks of the kernel matrix are not symmetrical unlike with separable kernels.

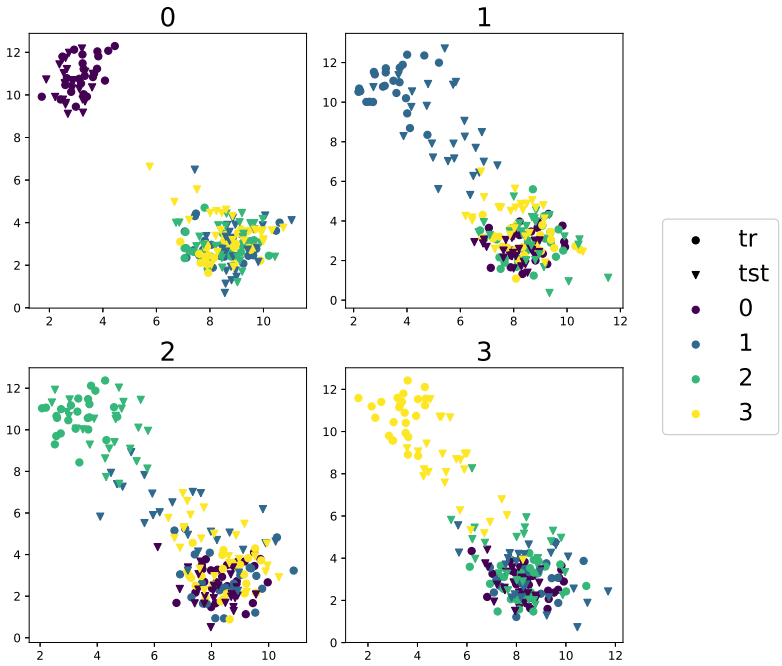

Figure 6: Supervised dimensionality reduction results on two dimensions for each of the four tasks in digits data sets, task in question indicated by the subfigure title. The classes of the projected data samples are indicated with colours, and the shape of the blob indicates whether the sample was used in training stage (round) or projected as test sample (triangle).

model allows calculating new reduced features for new data samples; given $\hat{\mathbf{\Phi}}_t$ a matrix of features of new samples, the "predicted reduced features" are now simply $\mathbf{Z}_t = (\hat{\mathbf{\Phi}}_t^\top \otimes \mathbf{I}_p)\mathbf{Q}$. These projections are for the most part very accurate as shown in Figure 6, respecting the task clusters especially for the easiest classes.

6.2 Performance of EKL

We now turn to consider the predictive performance of EKL. We first show experiments on artificial data, before turning to real-world data sets.

6.2.1 Artificial data

EKL is expected to learn complex relationships within the data. To illustrate this, we created data with bi-linear model $\mathbf{TCA} + \mathbf{ICK} = \mathbf{Y}$, where $\mathbf{T}$, $\mathbf{C}$ and $\mathbf{A}$ are randomly created $p \times p$, $p \times n$, and $n \times n$ matrices respectively. $\mathbf{K}$ is linear kernel calculated from randomly generated data $\mathbf{X} \in \mathbb{R}^{n \times d}$; this scalar-valued kernel is given to all the learning algorithms along with noisy labels $\mathbf{Y}$.

The results are shown in Figure 7. We can see that when p is larger than n (or comparable) the predictive capabilities of EKL are much better than for other methods. Here

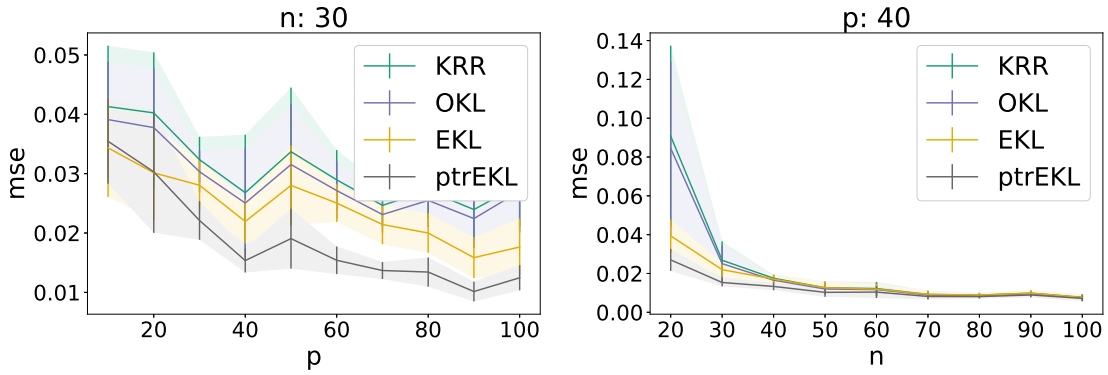

Figure 7: Results (mean squared error) of the simulated experiments with fixed amount of inputs and varying number of outputs (top), and fixed amount of outputs and varying inputs (bottom).The advantage of learning complex relationships is the biggest with small n.

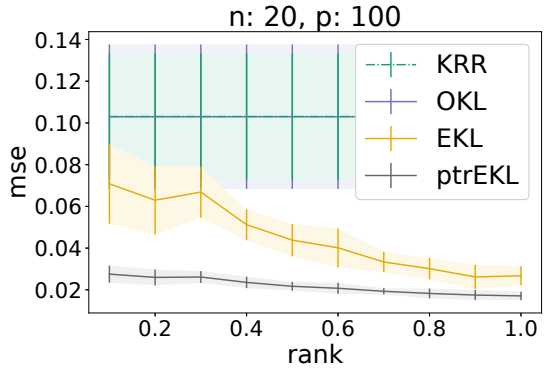

Figure 8: Results (mean squared error) on the simulated data set with varying number of columns used in learning $\mathbf{Q}$ with $n = 20$ and $p = 100$. The number of columns, or rank level, is shown as percentage of the full possible rank. OKL And KRR are plotted as a reference, naturally the rank constraint of EKL does not affect these methods.

predicting with the scalar-valued kernel extracted form learned entangled kernel gives the best results.

We also investigated the effect the choice of rank r (number of columns in matrix $\mathbf{Q}$) has for EKL performance (Figure 8). As the rank increases, the performance of EKL gets better. This is true to an extent also for ptrEKL, however there seems not to be as strong effect as with EKL. This is not so surprising; ptrEKL has fewer parameters affecting predictive performance, so decreasing the amount of them shouldn't change the results as much as for full EKL.

	$n = 50$		$n = 100$		$n = 200$		$n = 1000$	
method	nMSE	nI	nMSE	nI	nMSE	nI	nMSE	nI
KRR	0.2418 ± 0.0281	0.0000	0.1668 ± 0.0097	0.0000	0.1441 ± 0.0037	0.0000	0.1273 ± 0.0006	0.0000
OKL	0.2445 ± 0.0296	-0.0109	0.1672 ± 0.0099	-0.0026	0.1442 ± 0.0037	-0.0009	0.1273 ± 0.0006	-0.0000
EKL/ptrEKL	0.2381 ± 0.0250	0.0139	0.1661 ± 0.0097	0.0040	0.1440 ± 0.0037	0.0003	0.1273 ± 0.0006	0.0001

Table 3: Results on Sarcos data set with various number of training samples used, averaged over 10 data partitions. The advantage of learning complex relationships decreases with amount of data samples increasing.

	$n = 5$		$n = 10$		$n = 15$	
method	nMSE	nI	nMSE	nI	nMSE	nI
KRR	0.951 ± 0.101	0.000	0.813 ± 0.141	0.000	0.761 ± 0.037	0.000
OKL	1.062 ± 0.250	-0.092	0.900 ± 0.196	-0.094	0.788 ± 0.058	-0.034
EKL/ptrEKL	0.840 ± 0.084	0.124	0.722 ± 0.036	0.107	0.728 ± 0.033	0.044

Table 4: Results on Weather data set averaged over 5 data partitions.

6.2.2 Real data

We consider the following regression data sets: Concrete slump test[9] with 103 data samples and three output variables; Sarcos[10] is a data set characterizing robot arm movements with 7 tasks; Weather[11] has daily weather data ($p = 365$) from 35 stations. With these data sets, we considered linear kernels and used the original features in EKL, and full rank in learning $\mathbf{Q}$. Furthermore, we consider the uWaveGesture data set[12], a multi-view data set for classification, which we use in a setting of predicting a view from another, giving us a regression problem with 314 output variables. We again consider linear kernels with the data set, and investigate also the effect of chosen rank of a matrix $\mathbf{Q}$.

The main advantage of learning complex dependencies in the data lies in the setting where number of samples is relatively low; a phenomenon observed already in output kernel learning setting (Ciliberto et al., 2015; Jawanpuria et al., 2015). With small amounts of data learning the complex relationships in EKL is even more beneficial than learning the output dependencies of OKL. Figure 9 shows this advantage on Concrete data set when number of instances used in training is small. Here, in contrast to our simulated data, EKL performs better than ptrEKL. For Sarcos data set we consider the setting in Ciliberto et al. (2015) and show the results in Table 3 (predicting is done to all 5000 test samples). As can be expected, the results with the Sarcos data set which has very few outputs ($p = 7$) do not show improvement over the compared methods when the number of samples is large. However we can ascertain that our EKL finds the same solutions than the other methods with the large sample sizes - indeed the methods perform identically when n increases. We expect that the main improvement of the EKL lies in the cases when n and p are comparable,

9. UCI data set repository
10. `www.gaussianprocess.org/gpml/data/`
11. `https://www.psych.mcgill.ca/misc/fda/`
12. `http://www.cs.ucr.edu/∼eamonn/time_series_data/`

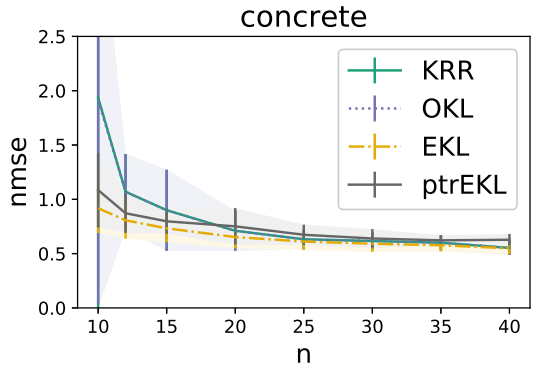

Figure 9: Results (normalized mean squared error) on the concrete data set with varying amount of training data (n) used. The advantage of learning complex relationships is biggest on small n.

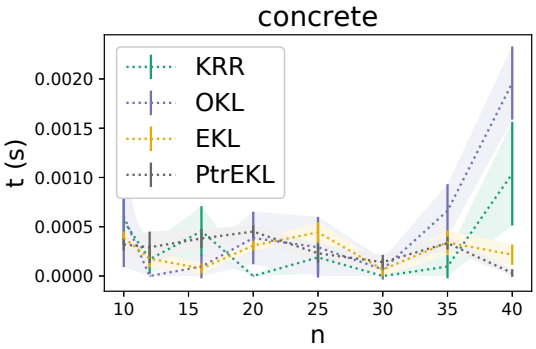

Figure 10: Running times in seconds (including calculating the predictive function given a kernel and predicting) on the concrete data set with varying amount of training data (n) used.

	$n = 12$		$n = 20$		$n = 40$	
method	nMSE	nI	nMSE	nI	nMSE	nI
KRR	1.070 ± 0.347	0.000	0.710 ± 0.183	0.000	0.552 ± 0.065	0.000
OKL	1.069 ± 0.347	0.001	0.710 ± 0.183	0.001	0.552 ± 0.064	0.000
EKL	0.796 ± 0.164	0.266	0.634 ± 0.103	0.097	0.547 ± 0.046	0.007
ptrEKL	0.843 ± 0.186	0.212	0.726 ± 0.170	-0.023	0.627 ± 0.058	-0.130

Table 5: Results on Concrete data set, averaged over 10 data partitions.

		$n = 20$	$n = 30$	$n = 40$
Class 1	KRR	3.012 ± 0.2337	2.853 ± 0.0780	2.262 ± 0.3157
	OKL	3.403 ± 0.0899	3.232 ± 0.1877	2.440 ± 0.3451
	EKL	1.136 ± 0.0937	1.123 ± 0.0852	1.107 ± 0.1019
	ptrEKL	1.323 ± 0.1807	1.208 ± 0.0932	1.132 ± 0.1221
Class 2	KRR	1.773 ± 0.0655	2.008 ± 0.3391	1.937 ± 0.3170
	OKL	1.802 ± 0.0603	2.096 ± 0.3673	2.074 ± 0.3483
	EKL	0.991 ± 0.0517	0.943 ± 0.0237	0.908 ± 0.0130
	ptrEKL	1.026 ± 0.0509	0.940 ± 0.0143	0.914 ± 0.0138
Class 3	KRR	1.081 ± 0.1437	1.028 ± 0.1235	0.902 ± 0.1020
	OKL	1.173 ± 0.1720	1.146 ± 0.1753	0.993 ± 0.1486
	EKL	0.671 ± 0.0202	0.638 ± 0.0214	0.632 ± 0.0325
	ptrEKL	0.681 ± 0.0226	0.651 ± 0.0345	0.632 ± 0.0307

Table 6: Normalized mean squared errors over three runs on the uWaveGesture data set, classes 1-3, with varying amount of training data and full rank of the EKL.

		rank 100%	rank 60%	rank 20%
Class 1	EKL	1.123 ± 0.0852	1.134 ± 0.0762	1.171 ± 0.1199
	ptrEKL	1.208 ± 0.0932	1.228 ± 0.0970	1.325 ± 0.1630
Class 2	EKL	0.943 ± 0.0237	0.939 ± 0.0091	0.941 ± 0.0074
	ptrEKL	0.940 ± 0.0143	0.958 ± 0.0053	0.987 ± 0.0137
Class 3	EKL	0.638 ± 0.0214	0.644 ± 0.0222	0.654 ± 0.0271
	ptrEKL	0.651 ± 0.0345	0.647 ± 0.0344	0.647 ± 0.0394

Table 7: Normalized mean squared errors over three runs with $n = 30$ on the uWaveGesture data set, classes 1-3, with varying rank of the EKL.

or $p > n$. This is clearly seen with the Weather data set, where number of outputs is much larger than the number of data samples (Table 4).

While we investigate the running times of the compared algorithms more in depth in next subsection, we here in Figure 10 display them for the Concrete data set. The Figure shows the combined time for both calculating the predictive function given the kernel, and predicting.

To show further the advantage of our method in the case where $n < p$, we turn our attention to uWaveGesture data set with three views. Each of the views is of dimensionality 314. The training set partition contains in total 896 data samples from the three views and eight classes. However as we want to investigate the case with large distinction between n and p, we only consider the smaller sets of data belonging to each class, with around 100 samples each which we divide into training and testing sets for the compared algorithms. Tables 6 and 7 show the results. In Table 6 we present the errors for the first three classes

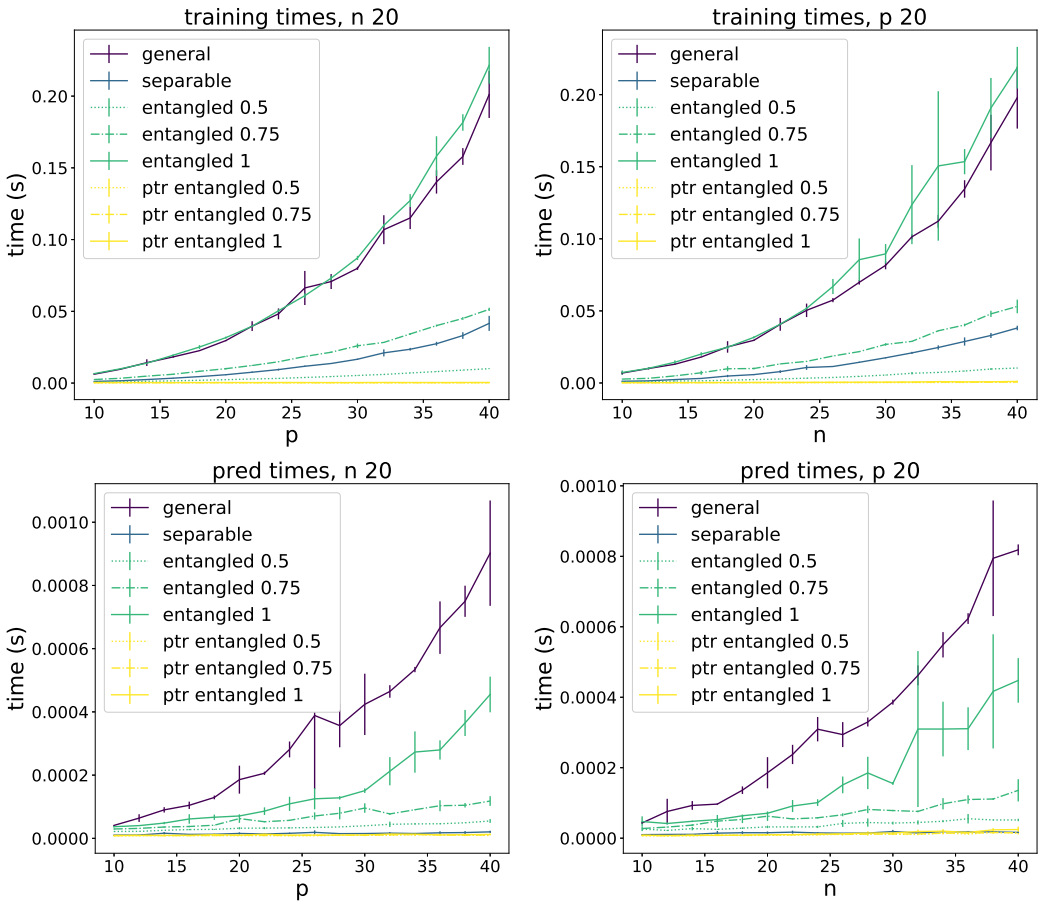

Figure 11: Times for training (top row) or predicting (bottom row) with the various operator-valued kernels with fixed n/p indicated in the subtitle of the figures. For entangled and partial trace entangled we show the results with various choices of m and r, indicated with the number after method name in the legend. For example "entangled 0.75" means that $m = 0.75 \cdot n$ and $r = 0.75 \cdot mp$, meaning both are 75% of the largest possible value. $t = n$ for all the experiments.

with full rank parameter r in the EKL algorithms. This setting is very extreme with the tiny sample size and large p, and it is clear that our method obtains the best result. We also detail in Table 7 the EKL results with respect to the rank parameter r on 100%, 60% and 20% of the full rank. Even though the error rises a little in most cases, it is not significant especially compared to the results with KRR and OKL. This further justifies the use of our algorithm in the more efficient, low-rank setting.

6.3 Running Times of Learning with OvKs

As we show in Table 2, there is a big difference in computational complexities between various operator-valued kernels. Namely, the complexities under question are those of cal-

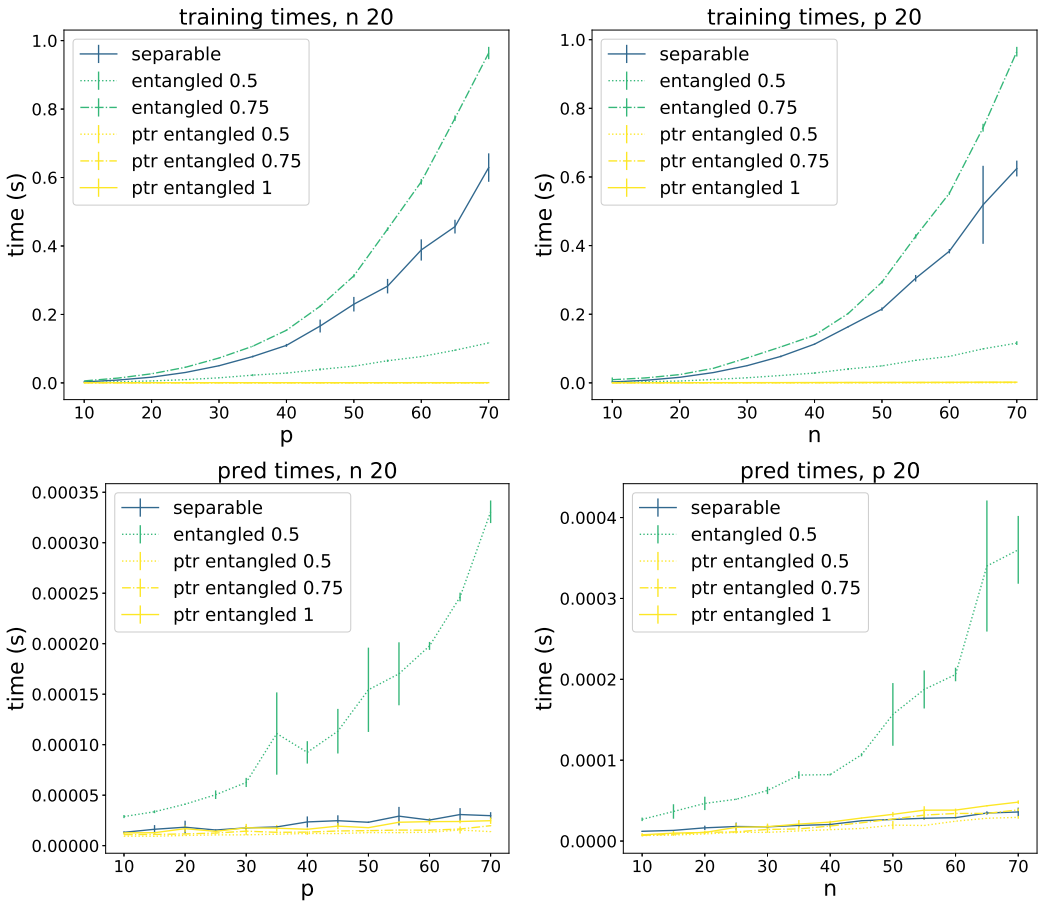

Figure 12: A closer look at the times for training (top row) or predicting (bottom row) with the various operator-valued kernels with fixed n/p indicated in the subtitle of the figures. For entangled and partial trace entangled we show the results with various choices of m and r, indicated with the number after method name in the legend. For example "entangled 0.75" means that $m = 0.75 \cdot n$ and $r = 0.75 \cdot mp$, meaning both are 75% of the largest possible value. $t = n$ for all the experiments.

culating the parameters $\mathbf{c}$ of the predictive function, and of calculating the label predictions. In this section we perform experiments to highlight these differences. We note that here we do not learn the kernels, but only compare in the experiments the differences on pre-defined entangled/separable or general operator-valued kernels.

In our experiments we created random data (random kernel matrices, random $\mathbf{c}$ from which labels were calculated) with which we performed our calculations. We repeated our experiments five times with different random data, and each time timed the execution of learning and predicting five times. In our results (Figures 11 and 12) we present averages of these runs.

Figure 11 shows the times of the runs for all the methods listed in Table 2. As expected, the operator-valued kernels with no structure are among the worst in every run, with scalar-valued kernels much better than them. The performance of entangled kernels naturally rests on the crucial parameters m and r. In the experiments we modify both at the same time, from full values to 75% and 50% of the full value. With maximum m and r entangled kernels are as slow as the general operator-valued kernels, but outperform them in predictive step. With 75% they train comparably to separable kernels, while outperforming them with 50%. Partial trace version of entangled kernels naturally has the lowest computational cost in training step. In predicting, entangled kernels are always better than general operator-valued kernels. Separable and partial trace kernels perform similarly, which can be seen better in Figure 12 showing a closer look to separable and entangled kernels.

7. Conclusion

In this work we shed new light on meaning of inseparable kernels by defining a general framework for constructing operator-valued kernels based on the notion of partial trace and using ideas borrowed from the field of quantum computing. Instances of our framework include entangled kernels, a new conceptually interesting class of kernels that is designed to capture more complex dependencies between input and output variables as the more restricted class of separable kernels. We have proposed a new algorithm, entangled kernel learning (EKL), that learns this entangled kernel in kernel alignment framework. The first step uses a definition of kernel alignment, extended here for use with operator-valued kernels with help of partial trace operator. In contrast to output kernel learning (OKL), EKL is able to learn inseparable kernels and can model a larger variety of interactions between input and output data. Moreover, the structure of the entangled kernels enables more efficient computation than that with general operator-valued kernels. Our illustration on artificial data and experiments on real data give validation to our approach.

Like with previous work with separable kernels, the main advantage of learning complex relations in the data lies in setting where number of data samples is relatively low. In the experimental section, we observed that difference in performance between EKL and OKL decreases as sample size increases. We think that one possible reason for that is that number of outputs relative to samples is not as large in this case, and it will be interesting to thoroughly investigate EKL in the setting where the number of outputs is even larger. Moreover, the effect of choosing the number of columns in matrix Q (or choosing the rank of the entangled kernel) would warrant further study, especially with small values (low rank kernel setting). It is also well-known that feature representation of a kernel is not unique, and we leveraged this in our work. Studying the effect on this could be worthwhile, as well as any theoretical consequences it has.

Using the Kraus representation, entangled kernels can be viewed as a mapping from a covariance matrix on the input features to a covariance matrix on the output labels. Recently, Riemannian networks for symmetric positive definite (SPD) learning have been introduced in Huang and Gool (2017). These networks receive SPD matrices, such as covariance descriptors, as inputs, and preserve their SPD structure across the layers. The SPD structure is preserved via the bilinear mapping (BiMap) layer, which can be seen as a particular case of a Choi-Kraus representation with Kraus rank equal to 1. Generalizing

SPD networks to higher Kraus ranks could provide a way to efficiently learn entangled kernels with deep learning machinery, and would be an interesting direction to explore in future work.

Overall, we hope that our work featuring the first comprehensive description on how to learn non-separable operator-valued kernels will give a boost to the field of learning inseparable kernels.

Acknowledgments

We thank the anonymous reviewers for their helpful comments and suggestions, which improved the quality of the paper. We also thank F. Denis, P. Arrighi and G. Di Molfetta for fruitful discussions. This work has been funded by the French National Research Agency (ANR) project QuantML (grant number ANR-19-CE23-0011). A large part of this research was done while R.H. was at Aix-Marseille University; the part in Aalto University in part been funded by Academy of Finland grants 334790 (MAGITICS) and 310107 (MACOME).

Appendix A. Proof for Choi-Kraus representation theorem (Theorem 13)

The following result is useful to prove Theorem 13.

Theorem 17 *(Watrous, 2018, Theorem 2.22)*
Let Φ be a linear map from $\mathcal{L}(\mathcal{K})$ to $\mathcal{L}(\mathcal{Y})$, where $\mathcal{K}$ and $\mathcal{Y}$ are Euclidean spaces. The following statements are equivalent:

1. *There exists an operator $A \in \mathcal{L}(\mathcal{K}, \mathcal{Y} \otimes \mathcal{Z})$ for a some choice of an Euclidean space $\mathcal{Z}$, such that*
$$\Phi(\mathbf{x}) = \mathrm{tr}_{\mathcal{Z}}(AXA^*)$$
for all $X \in \mathcal{X}$.

2. *There existes a collection $\{A_a : a \in \Sigma\}$, for some choice of an alphabet Σ, for which*
$$\Phi(\mathbf{x}) = \sum_a A_a X A_a^*$$
for all $X \in \mathcal{X}$.

As mentioned in Watrous (2018), this theorem is an amalgamation of results that are generally attributed to Stinespring (1955); Kraus (1971, 1983); Choi (1975), and presents only the finite-dimensional analogues of the results they proved which hold for infinite-dimensional spaces.

Proof of Theorem 13 The proof follows the same arguments as the proof of Theorem 6.5 from Attal (2015a), which holds for infinite dimensional spaces. It is based on the observation that for $\mathbf{v} \in \mathcal{Y}$ we have $\mathbf{v}\mathbf{v}^\top \otimes (\phi(\mathbf{x})\phi(\mathbf{x})^\top) = \tilde{O}_{\mathbf{v}}(\phi(\mathbf{x})\phi(\mathbf{x})^\top)\tilde{O}_{\mathbf{v}}^*$, in which $\tilde{O}_{\mathbf{v}}$ is the

operator defined by

$$\begin{aligned}\tilde{O}_{\mathbf{v}} : \mathcal{K} &\longrightarrow \mathcal{Y} \otimes \mathcal{K} \\ u &\longmapsto \mathbf{v} \otimes u,\end{aligned}$$

and $\tilde{O}_{\mathbf{v}}^*$ is its adjoint defined by

$$\begin{aligned}\tilde{O}_{\mathbf{v}}^* : \mathcal{Y} \otimes \mathcal{K} &\longrightarrow \mathcal{K} \\ \mathbf{y} \otimes u &\longmapsto \langle \mathbf{v}, \mathbf{y} \rangle u.\end{aligned}$$

The entangled kernel K can now be written as follows

$$\begin{aligned}K(\mathbf{x}, \mathbf{z}) &= \mathrm{tr}_{\mathcal{K}} \left(\mathbf{U} \big(\mathbf{T} \otimes (\phi(\mathbf{x})\phi(\mathbf{z})^\top) \big) \mathbf{U}^\top \right) \\ &= \sum_t \mathrm{tr}_{\mathcal{K}} \left(\mathbf{U} \big(\mathbf{v}_t \mathbf{v}_t^\top \otimes (\phi(\mathbf{x})\phi(\mathbf{z})^\top) \big) \mathbf{U}^\top \right) \\ &= \sum_t \mathrm{tr}_{\mathcal{K}} \left(\mathbf{U} \big(\tilde{O}_{\mathbf{v}_t} (\phi(\mathbf{x})\phi(\mathbf{z})^\top) \tilde{O}_{\mathbf{v}_t}^* \big) \mathbf{U}^\top \right) \\ &= \sum_t \mathrm{tr}_{\mathcal{K}} \left(S_t (\phi(\mathbf{x})\phi(\mathbf{z})^\top) S_t^* \right),\end{aligned}$$

where $S_t = \mathbf{U}\tilde{O}_{\mathbf{v}_t}$. Using Theorem 17, which is also valid for infinite-dimensional spaces (Watrous, 2018), we obtain that there exists a set of matrices $\{\mathbf{M}_i, 1 \leq i \leq r\}$ for which $K(\mathbf{x}, \mathbf{z}) = \sum_{i=1}^r \mathbf{M}_i \phi(\mathbf{x})\phi(\mathbf{z})^\top \mathbf{M}_i^\top$. This completes the proof.

■

Appendix B. Proof of Theorem 15 (Rademacher bound for EKL)

We provide here the proof for our Rademacher complexity bound.

Proof of Theorem 15 We start by recalling that the feature map associated to the operator-valued kernel K is the mapping $\Gamma : \mathcal{X} \to \mathcal{L}(\mathcal{Y}, \mathcal{H})$, where $\mathcal{X}$ is the input space, $\mathcal{Y} = \mathbb{R}^p$, and $\mathcal{L}(\mathcal{Y}, \mathcal{H})$ is the set of bounded linear operators from $\mathcal{Y}$ to $\mathcal{H}$ (see, e.g., Micchelli and Pontil (2005); Carmeli et al. (2010) for more details). It is known that $K(\mathbf{x}, \mathbf{z}) = \Gamma(\mathbf{x})^* \Gamma(\mathbf{z})$. We denote by $\Gamma_{\mathbf{Q}}$ the feature map associated to our entangled kernel (Equation 15). We also define the matrix $\mathbf{\Sigma} = (\boldsymbol{\sigma})_{i=1}^n \in \mathbb{R}^{np}$

$$\begin{aligned}\hat{\mathcal{R}}_n(\mathcal{H}_\beta) &= \frac{1}{n} \mathbb{E} \left[\sup_{f \in \mathcal{H}} \sup_{\mathbf{Q} \in \Delta} \sum_{i=1}^n \boldsymbol{\sigma}_i^\top f_{u,\mathbf{Q}}(\mathbf{x}_i) \right] \\ &= \frac{1}{n} \mathbb{E} \left[\sup_u \sup_{\mathbf{Q}} \sum_{i=1}^n \langle \boldsymbol{\sigma}_i, \Gamma_{\mathbf{Q}}(\mathbf{x}_i)^* \, u \rangle_{\mathbb{R}^p} \right] \\ &= \frac{1}{n} \mathbb{E} \left[\sup_u \sup_{\mathbf{Q}} \sum_{i=1}^n \langle \Gamma_{\mathbf{Q}}(\mathbf{x}_i) \boldsymbol{\sigma}_i, u \rangle_{\mathcal{H}} \right] \quad (1)\end{aligned}$$

$$
\begin{aligned}
&\leq \frac{\beta}{n}\,\mathbb{E}\left[\sup_{\mathbf{Q}} \|\sum_{i=1}^{n} \Gamma_{\mathbf{Q}}(\mathbf{x}_i)\boldsymbol{\sigma}_i\|_{\mathcal{H}}\right] \quad (2)\\
&= \frac{\beta}{n}\,\mathbb{E}\left[\sup_{\mathbf{Q}} \left(\sum_{i,j=1}^{n} \langle \boldsymbol{\sigma}_i, K_{\mathbf{Q}}(\mathbf{x}_i,\mathbf{x}_j)\boldsymbol{\sigma}_j\rangle_{\mathbb{R}^p}\right)^{\frac{1}{2}}\right] \quad (3)\\
&= \frac{\beta}{n}\,\mathbb{E}\left[\sup_{\mathbf{Q}} \left(\langle \boldsymbol{\Sigma}, \mathbf{G}_{\mathbf{Q}}\boldsymbol{\Sigma}\rangle_{\mathbb{R}^{np}}\right)^{1/2}\right]\\
&= \frac{\beta}{n}\,\mathbb{E}\left[\sup_{\mathbf{Q}} \langle \boldsymbol{\Sigma}, [\boldsymbol{\Phi}^\top \otimes \mathbf{I}_p]\mathbf{Q}\mathbf{Q}^\top[\boldsymbol{\Phi} \otimes \mathbf{I}_p]\boldsymbol{\Sigma}\rangle^{1/2}\right]\\
&= \frac{\beta}{n}\,\mathbb{E}\left[\sup_{\mathbf{Q}} tr([\boldsymbol{\Phi} \otimes \mathbf{I}_p]\boldsymbol{\Sigma}\boldsymbol{\Sigma}^\top[\boldsymbol{\Phi}^\top \otimes \mathbf{I}_p]\mathbf{Q}\mathbf{Q}^\top)^{1/2}\right]\\
&\leq \frac{\beta}{n}\,\mathbb{E}\left[\sup_{\mathbf{Q}} tr([[\boldsymbol{\Phi} \otimes \mathbf{I}_p]\boldsymbol{\Sigma}\boldsymbol{\Sigma}^\top[\boldsymbol{\Phi}^\top \otimes \mathbf{I}_p]]^2)^{1/4} tr(\mathbf{Q}\mathbf{Q}^\top\mathbf{Q}\mathbf{Q}^\top)^{1/4}\right] \quad (4)\\
&\leq \frac{\beta}{n}\,\mathbb{E}\left[\sup_{\mathbf{Q}} tr([\boldsymbol{\Phi} \otimes \mathbf{I}_p][\boldsymbol{\Phi}^\top \otimes \mathbf{I}_p]\boldsymbol{\Sigma}\boldsymbol{\Sigma}^\top)^{1/2} tr(\mathbf{Q}\mathbf{Q}^\top\mathbf{Q}\mathbf{Q}^\top)^{1/4}\right]\\
&\leq \frac{\beta}{n}\,\mathbb{E}\left[\sup_{\mathbf{Q}} tr([\boldsymbol{\Phi}\boldsymbol{\Phi}^\top \otimes \mathbf{I}_p]\boldsymbol{\Sigma}\boldsymbol{\Sigma}^\top)^{1/2}\right]\\
&= \frac{\beta}{n}\,\mathbb{E}\left[tr([\mathbf{K} \otimes \mathbf{I}_p]\boldsymbol{\Sigma}\boldsymbol{\Sigma}^\top)^{1/2}\right]\\
&\leq \frac{\beta}{n}\left(\mathbb{E}\left[tr([\mathbf{K} \otimes \mathbf{I}_p]\boldsymbol{\Sigma}\boldsymbol{\Sigma}^\top)\right]\right)^{1/2} \quad (5)\\
&= \frac{\beta}{n}\left(tr\left[[\mathbf{K} \otimes \mathbf{I}_p]\,\mathbb{E}(\boldsymbol{\Sigma}\boldsymbol{\Sigma}^\top)\right]\right)^{1/2}\\
&= \frac{\beta}{n}\sqrt{tr(\mathbf{K})p}.
\end{aligned}
$$

Here (1) and (3) are obtained with reproducing property, (2) and (4) with Cauchy-Schwarz inequality, and (5) with Jensen's inequality.

For kernels k that satisfy $tr(\mathbf{K}) \leq \kappa n$, we obtain that

$$\hat{\mathcal{R}}_n(\mathcal{H}_\beta) \leq \frac{\beta\sqrt{\kappa p}}{\sqrt{n}}.$$

■

Appendix C. Proof of Corollary 16 (Generalization bound for EKL)

The following two results are useful to prove Corollary 16. First, Bartlett and Mendelson (2002) provides the following generalization bound based on Rademacher complexity (see also Mohri et al. 2018, Theorem 3.3).

Theorem 18 *(Mohri et al. 2018, Theorem 3.3)*
Let $\mathcal{G}$ be a family of functions mapping from an arbitrary input space $\mathcal{Z}$ to [0,M]. Then, for any $\delta > 0$, with probability at least $1-\delta$ over the draw of an i.i.d. sample S of size n, the following holds for all $g \in \mathcal{G}$:

$$\mathbb{E}[g(\mathbf{z})] \leq \frac{1}{n}\sum_{i=1}^{n} g(\mathbf{z}_i) + 2\hat{\mathcal{R}}_n(\mathcal{G}) + 3M\sqrt{\frac{\log\frac{2}{\delta}}{2n}}, \tag{20}$$

where $\hat{\mathcal{R}}_n(\mathcal{G})$ is the empirical Rademacher complexity of $\mathcal{G}$.

The second result, from Maurer (2016), provides a contraction inequality for the Rademacher complexity of classes of vector-valued functions.

Corollary 19 *(Maurer 2016, Corollary 1)*
Let $\mathcal{X}$ be any set, $(\mathbf{x}_1, \ldots, \mathbf{x}_n) \in \mathcal{X}^n$, let $\mathcal{F}$ be a class of functions $f : \mathcal{X} \to \ell_2$ and let $h_i : \ell_2 \to \mathbb{R}$ have Lipschitz norm L. Then

$$\mathbb{E}\sup_{f\in\mathcal{F}} \sum_i \sigma_i h_i(f(\mathbf{x}_i)) \leq \sqrt{2}L\mathbb{E}\sup_{f\in\mathcal{F}} \sum_{i,j=1} \sigma_{ij} f_j(\mathbf{x}_i), \tag{21}$$

where σ_{ij} is an independent doubly indexed Rademacher sequence and $f_j(\mathbf{x}_i)$ is the j-th component of $f(\mathbf{x}_i)$.

We now make use of the above results to prove the generalization error bound of EKL.

Proof of Corollary 16

Since $\|f(\mathbf{x}) - \mathbf{y}\|_{\mathcal{Y}} \leq M$ for all $(\mathbf{x}, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$ and $f \in \mathcal{H}_\beta$, for any $\mathbf{y}'$ the function $y \mapsto \|\mathbf{y} - \mathbf{y}'\|_{\mathcal{Y}}^2$ is $2M$-Lipschitz. Then by Corollary 19, for any sample $S = ((\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n))$, the Rademacher complexity of the family $\mathcal{G} = \{(\mathbf{x}, \mathbf{y}) \mapsto \|f(\mathbf{x}) - \mathbf{y}\|_{\mathcal{Y}}^2 : f \in \mathcal{H}_\beta\}$ is upper bounded as follows:

$$\hat{\mathcal{R}}_n(\mathcal{G}) \leq 2\sqrt{2}M\hat{\mathcal{R}}_n(\mathcal{H}_\beta). \tag{22}$$

Combining this inequality with the general Rademacher complexity learning bound of Theorem 18 and the Rademacher complexity bound of $\mathcal{H}_\beta$ given in Theorem 15 completes the proof.

■

References

Mauricio A Álvarez and Neil D Lawrence. Computationally efficient convolved multiple output gaussian processes. *Journal of Machine Learning Research*, 12:1459–1500, 2011.

Mauricio A Álvarez, Lorenzo Rosasco, Neil D. Lawrence, et al. Kernels for vector-valued functions: A review. *Foundations and Trends® in Machine Learning*, 4(3):195–266, 2012.

Stephane Attal. Lectures in quantum noise theory. Lecture 6: Quantum channels, 2015a.

Stephane Attal. Lectures in quantum noise theory. Lecture 2: Tensor Products and Partial Traces, 2015b.

Luca Baldassarre, Lorenzo Rosasco, Annalisa Barla, and Alessandro Verri. Multi-output learning via spectral filtering. *Machine learning*, 87(3):259–301, 2012.

Peter L Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3:463–482, 2002.

Ingemar Bengtsson and Karol Życzkowski. *Geometry of quantum states: an introduction to quantum entanglement*. Cambridge university press, 2017.

Rajendra Bhatia. *Positive definite matrices*, volume 24. Princeton university press, 2009.

Romain Brault, Markus Heinonen, and Florence d'Alché Buc. Random fourier features for operator-valued kernels. In *Asian Conference on Machine Learning (ACML)*, pages 110–125, 2016.

Céline Brouard, Marie Szafranski, and Florence d'Alché Buc. Input output kernel regression: Supervised and semi-supervised structured output prediction with operator-valued kernels. *Journal of Machine Learning Research*, 17(176):1–48, 2016.

Andrea Caponnetto, Charles A Micchelli, Massimiliano Pontil, and Yiming Ying. Universal multi-task kernels. *Journal of Machine Learning Research*, 9(Jul):1615–1646, 2008.

Claudio Carmeli, Ernesto De Vito, Alessandro Toigo, and Veronica Umanita. Vector valued reproducing kernel hilbert spaces and universality. *Analysis and Applications*, 08(01):19–61, 2010.

Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

Man-Duen Choi. Completely positive linear maps on complex matrices. *Linear algebra and its applications*, 10(3):285–290, 1975.

Carlo Ciliberto, Youssef Mroueh, Tomaso Poggio, and Lorenzo Rosasco. Convex learning of multiple tasks and their structure. In *International Conference on Machine Learning (ICML)*, 2015.

Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. Algorithms for learning kernels based on centered alignment. *The Journal of Machine Learning Research*, 13(1):795–828, 2012.

Nello Cristianini, John Shawe-Taylor, Andre Elisseeff, and Jaz S Kandola. On kernel-target alignment. In *Advances in Neural Information Processing Systems*, pages 367–373, 2002.

Krzysztof Dembczyński, Willem Waegeman, Weiwei Cheng, and Eyke Hüllermeier. On label dependence and loss minimization in multi-label classification. *Machine Learning*, 88(1-2):5–45, 2012.

Francesco Dinuzzo and Kenji Fukumizu. Learning low-rank output kernels. In *Asian Conference on Machine Learning (ACML)*, 2011.

Francesco Dinuzzo, Cheng S. Ong, Gianluigi Pillonetto, and Peter V Gehler. Learning output kernels with block coordinate descent. In *International Conference on Machine Learning (ICML)*, pages 49–56, 2011.

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

Katarzyna Filipiak, Daniel Klein, and Erika Vojtková. The properties of partial trace and block trace operators of partitioned matrices. *Electronic Journal of Linear Algebra*, 33, 2018.

Kenji Fukumizu, Francis R Bach, and Michael I Jordan. Dimensionality reduction for supervised learning with reproducing kernel hilbert spaces. *Journal of Machine Learning Research*, 5(Jan):73–99, 2004.

Mehmet Gönen and Ethem Alpaydın. Multiple kernel learning algorithms. *Journal of Machine Learning Research*, 12(Jul):2211–2268, 2011.

Magda Gregorová, Alexandros Kalousis, and Stéphane Marchand-Maillet. Forecasting and granger modelling with non-linear dynamical dependencies. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD)*, 2017.

Paweł Horodecki. Separability criterion and inseparable mixed states with positive partial transposition. *Physical Review Letters*, 232:333, 1997.

Ryszard Horodecki, Paweł Horodecki, Michał Horodecki, and Karol Horodecki. Quantum entanglement. *Reviews of modern physics*, 81(2):865, 2009.

Zhiwu Huang and Luc Van Gool. A riemannian network for spd matrix learning. In *AAAI Conference on Artificial Intelligence*, pages 2036–2042, 2017.

Riikka Huusari and Hachem Kadri. Entangled kernels. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2019.

Riikka Huusari, Hachem Kadri, and Cécile Capponi. Multi-view metric learning in vector-valued kernel spaces. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2018.

Alan J Izenman. Reduced-rank regression for the multivariate linear model. *Journal of multivariate analysis*, 5(2):248–264, 1975.

Pratik Jawanpuria, Maksim Lapin, Matthias Hein, and Bernt Schiele. Efficient output kernel learning for multiple tasks. In *Advances in Neural Information Processing Systems*, 2015.

Hachem Kadri, Alain Rakotomamonjy, Philippe Preux, and Francis R Bach. Multiple operator-valued kernel learning. In *Advances in Neural Information Processing Systems*, 2012.

Hachem Kadri, Emmanuel Duflos, Philippe Preux, Stéphane Canu, Alain Rakotomamonjy, and Julien Audiffren. Operator-valued kernels for learning from functional response data. *Journal of Machine Learning Research*, 16:1–54, 2016.

Jaz Kandola, John Shawe-Taylor, and Nello Cristianini. On the extensions of kernel alignment. 2002.

Karl Kraus. General state changes in quantum theory. *Annals of Physics*, 64(2):311–335, 1971.

Karl Kraus. *States, effects and operations: fundamental notions of quantum theory*. Springer–Verlag, 1983.

Guy Lever, John Shawe-Taylor, Ronnie Stafford, and Csaba Szepesvári. Compressed conditional mean embeddings for model-based reinforcement learning. In *AAAI Conference on Artificial Intelligence*, 2016.

Néhémy Lim, Florence d'Alché Buc, Cédric Auliac, and George Michailidis. Operator-valued kernel-based vector autoregressive models for network inference. *Machine learning*, 99(3): 489–513, 2015.

Andreas Maurer. The Rademacher complexity of linear transformation classes. In *International Conference on Computational Learning Theory (COLT)*, pages 65–78, 2006.

Andreas Maurer. A vector-contraction inequality for rademacher complexities. In *International Conference on Algorithmic Learning Theory (ALT)*, pages 3–17, 2016.

Elliot Meyerson and Risto Miikkulainen. Beyond shared hierarchies: Deep multitask learning through soft layer ordering. In *International Conference on Learning Representations (ICLR)*, 2018.

Charles A Micchelli and Massimiliano Pontil. On learning vector-valued functions. *Neural Computation*, 17:177–204, 2005.

Ha Quang Minh. Operator-valued bochner theorem, fourier feature maps for operator-valued kernels, and vector-valued learning. *arXiv preprint arXiv:1608.05639*, 2016.

Ha Quang Minh, Loris Bazzani, and Vittorio Murino. A unifying framework in vector-valued reproducing kernel hilbert spaces for manifold regularization and co-regularized multi-view learning. *Journal of Machine Learning Research*, 17(25):1–72, 2016.

Florian Mintert, Carlos Viviescas, and Andreas Buchleitner. Basic concepts of entangled states. In *Entanglement and Decoherence*, pages 61–86. Springer, 2009.

Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2018.

Cheng Soon Ong, Xavier Mary, Stéphane Canu, and Alexander J Smola. Learning with non-positive kernels. In *International Conference on Machine Learning (ICML)*, page 81, 2004.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Asher Peres. Separability criterion for density matrices. *Physical Review Letters*, 77(8): 1413, 1996.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems*, pages 1177–1184, 2008.

Eleanor G Rieffel and Wolfgang H Polak. *Quantum computing: A gentle introduction*. MIT Press, 2011.

Akash Saha and Balamurugan Palaniappan. Learning with operator-valued kernels in reproducing kernel kreĭn spaces. In *Advances in Neural Information Processing Systems*, 2020.

Maxime Sangnier, Olivier Fercoq, and Florence d'Alché Buc. Joint quantile regression in vector-valued rkhss. In *Advances in Neural Information Processing Systems*, pages 3693–3701, 2016.

Vikas Sindhwani, Minh Ha Quang, and Aurélie C. Lozano. Scalable matrix-valued kernel learning for high-dimensional nonlinear multivariate regression and granger causality. In *Uncertainty in Artificial Intelligence (UAI)*, 2013.

Forrest W Stinespring. Positive functions on C*-algebras. *Proceedings of the American Mathematical Society*, 6(2):211–216, 1955.

Masashi Sugiyama. Local fisher discriminant analysis for supervised dimensionality reduction. In *International Conference on Machine Learning (ICML)*, pages 905–912, 2006.

James Townsend, Niklas Koep, and Sebastian Weichwald. Pymanopt: A Python toolbox for optimization on manifolds using automatic differentiation. *Journal of Machine Learning Research*, 17(137):1–5, 2016.

John Watrous. *The theory of quantum information*. Cambridge University Press, 2018.

Christopher KI Williams and Matthias Seeger. Using the nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems*, pages 682–688, 2001.