Partial Trace Regression and Low-Rank Kraus Decomposition

Hachem Kadri 1 Stéphane Ayache 1 Riikka Huusari 2 Alain Rakotomamonjy 3,4 Liva Ralaivola 4

Abstract

The trace regression model, a direct extension of the well-studied linear regression model, allows one to map matrices to real-valued outputs. We here introduce an even more general model, namely the partial-trace regression model, a family of linear mappings from matrix-valued inputs to matrix-valued outputs; this model subsumes the trace regression model and thus the linear regression model. Borrowing tools from quantum information theory, where partial trace operators have been extensively studied, we propose a framework for learning partial trace regression models from data by taking advantage of the so-called low-rank Kraus representation of completely positive maps. We show the relevance of our framework with synthetic and real-world experiments conducted for both i) matrix-to-matrix regression and ii) positive semidefinite matrix completion, two tasks which can be formulated as partial trace regression problems.

1. Introduction

Trace regression model. The trace regression model or, in short, trace regression, is a generalization of the well-known linear regression model to the case where input data are matrices instead of vectors (Rohde & Tsybakov, 2011; Koltchinskii et al., 2011; Slawski et al., 2015), with the output still being real-valued. This model assumes, for the pair of covariates \((X,y)\), the following relation between the matrix-valued random variable \(X\) and the real-valued random variable \(y\):

\[
y = \text{tr} (B^\top X) + \epsilon,
\]

where \(\text{tr}(\cdot)\) denotes the trace, \(B^*\) is some unknown matrix of regression coefficients, and \(\epsilon\) is random noise. This model has been used beyond mere matrix regression for problems such as phase retrieval (Candes et al., 2013), quantum state tomography (Gross et al., 2010), and matrix completion (Klopp, 2011).

Given a sample \(S = \{(X_i,y_i)\}_{i=1}^\ell\), where \(X_i\) is a \(p_1 \times p_2\) matrix and \(y_i \in \mathbb{R}\), and each \((X_i,y_i)\) is assumed to be distributed as \((X,y)\), the training task associated with statistical model (1) is to find a matrix \(\hat{B}\) that is a proxy to \(B^*\). To this end, Koltchinskii et al. (2011); Fan et al. (2019) proposed to compute an estimation \(\hat{B}\) of \(B^*\) as the solution of the regularized least squares problem

\[
\hat{B} = \arg \min_{B} \sum_{i=1}^\ell (y_i - \text{tr}(B^\top X_i))^2 + \lambda \|B\|_1,
\]

where \(\|\cdot\|_1\) is the trace norm (or nuclear norm), which promotes a low-rank \(\hat{B}\), a key feature for the authors to establish bounds on the deviation of \(\hat{B}\) from \(B^*\). Slawski et al. (2015); Koltchinskii & Xia (2015) have considered the particular case where \(p = p_1 = p_2\) and \(B^*\) is assumed to be from \(S^+_p\), the cone of positive semidefinite matrices of order \(p\), and they showed that guarantees on the deviation of \(\hat{B}\) from \(B^*\) continue to hold when \(\hat{B}\) is computed as

\[
\hat{B} = \arg \min_{B \in S^+_p} \sum_{i=1}^\ell (y_i - \text{tr}(BX_i))^2.
\]

Here, the norm regularization of (2) is no longer present and it is replaced by an explicit restriction for \(\hat{B}\) to be in \(S^+_p\) (as \(B^*\)). This setting is tied to the learning of completely positive maps developed hereafter.

1Aix-Marseille University, CNRS, LIS, Marseille, France. 2Helsinki Institute for Information Technology HIIT, Department of Computer Science, Aalto University, Espoo, Finland. 3Université Rouen Normandie, LITIS, Rouen, France. 4Criteo AI Lab, Paris, France. Correspondence to: Hachem Kadri <hachem.kadri@univ-amu.fr>.
Partial trace regression model. Here, we propose the partial trace regression model, that generalizes the trace regression model to the case when both inputs and outputs are matrices, and we go a step farther from works that are assuming either matrix-variate inputs (Zhou & Li, 2014; Ding & Cook, 2014; Slawski et al., 2015; Luo et al., 2015) or matrix/tensor-variate outputs (Kong et al., 2019; Li & Zhang, 2017; Rabusseau & Kadri, 2016). Key to our work is the so-called partial trace, explained in the following section and depicted in Figure 1.

This novel regression model that maps matrix-to-matrix is of interest for several application tasks. For instance, in Brain-Computer Interfaces, covariance matrices are frequently used as a feature for representing mental state of a subject (Barachant et al., 2011; Congedo et al., 2013) and those covariance matrices need to be transformed (Zanini et al., 2018) into other covariance matrices to be discriminative for some BCI tasks. Similarly in Computer Vision and especially in the subfield of 3D Shape retrieval, covariance matrices are of interest as descriptors (Guo et al., 2018; Hariri et al., 2017), while there is a surging interest in deep learning methods for defining trainable layers with positive maps, and show the optimization problem to which code linear mappings from matrix-valued spaces to matrix-valued outputs; this model encompasses previously known results show that partial trace regression model yields good effectiveness.

Contributions. We make the following contributions. i) We introduce the partial trace regression model, a family of linear predictors from matrix-valued inputs to matrix-valued outputs; this model encompasses previously known regression models, including the trace regression model and thus the linear regression model; ii) borrowing concepts from quantum information theory, where partial trace operators have been extensively studied, we propose a framework for learning a partial trace regression model from data; we take advantage of the low-rank Kraus representation of completely positive maps to cast learning as an optimization problem which we are able to handle efficiently; iii) we provide statistical guarantees for the model learned under our framework, thanks to a provable estimation of pseudodimension of the class of functions that we can envision; iv) finally, we show the relevance of the proposed framework for the tasks of matrix-to-matrix regression and positive semidefinite matrix completion, both of them are amenable to a partial trace regression formulation; our empirical results show that partial trace regression model yields good performance, demonstrating wide applicability and effectiveness.

2. Partial Trace Regression

Here, we introduce the partial trace regression model to encode linear mappings from matrix-valued spaces to matrix-valued spaces. We specialize this setting to completely positive maps, and show the optimization problem to which learning with the partial trace regression model translates, together with a generalization error bound for the associated learned model. In addition, we present how the problem of (block) positive semidefinite matrix completion can be cast as a partial trace regression problem.

2.1. Notations, Block Matrices and Partial Trace

Notations. Our notational conventions are summarized in Table 1. For $n,m \in \mathbb{N}$, $\mathbb{M}_{n \times m} = \mathbb{M}_{n \times m}(\mathbb{R})$ denotes the space of all $n \times m$ real matrices. If $n = m$, then we write $\mathbb{M}_n$ instead of $\mathbb{M}_{n \times n}$. If $M$ is a matrix, $M_{ij}$ denotes its $(i,j)$-th entry. For $M \in \mathbb{M}_n$, $M \succeq 0$ will be used to indicate that $M$ is positive semidefinite (PSD); we may equivalently write $M \in \mathbb{S}^+$. Throughout, $\{(X_i, Y_i)\}_{i=1}^l$ denotes a training sample of $l$ examples, with each $(X_i, Y_i)$ assumed to be drawn IID from a fixed but unknown distribution on $\mathcal{X} \times \mathcal{Y}$ where, from now on, $\mathcal{X} \doteq \mathbb{M}_p$ and $\mathcal{Y} \doteq \mathbb{M}_q$.

Block matrices. We will make extensive use of the notion of block matrices, i.e., matrices that can be partitioned into submatrices of the same dimension. If $M \in \mathbb{M}_{nm}$, the number of block partitions of $M$ directly depends on the number of factors of $n$ and $m$; to uniquely identify the partition we are working with, we will always consider a $n \times n$ block partition, where $n$ will be clear from context —the number of rows and columns of the matrix at hand will thus be multiples of $n$. The set $\mathbb{M}_n(\mathbb{M}_m)$ will then denote the space of $n \times n$ block matrices $M = [[M_{ij}]]$ whose $i,j$ entry is an element of $\mathbb{M}_m$.

Partial trace operators. Partial trace, extensively studied and used in quantum computing (see, e.g., Rieffel & Polak 2011, Chapter 10), generalizes the trace operation to block matrices. The definition we work with is the following.

Definition 1 (Partial trace, see, e.g., Bhatia, 2003.) The partial trace operator, denoted by $\text{tr}_m(\cdot)$, is the linear map

$$
\text{tr}_m(\cdot) : \mathbb{M}_m(\mathbb{M}_m) \rightarrow \mathbb{M}_m
$$

1We also use the standard notations such as $\mathbb{R}^n$ and $\mathbb{M}_n$.

2The space $\mathbb{M}_n(\mathbb{M}_m)$ is isomorphic to $\mathbb{M}_n \otimes \mathbb{M}_m$. 

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$i, j, m, n, p, q$</td>
<td>integers</td>
</tr>
<tr>
<td>$\alpha, \beta, \gamma, \ldots$</td>
<td>real numbers</td>
</tr>
<tr>
<td>$\mathcal{X}, \mathcal{Y}, \mathcal{H}, \ldots$</td>
<td>vector spaces¹</td>
</tr>
<tr>
<td>$x, y, k, \ldots$</td>
<td>vectors (or functions)</td>
</tr>
<tr>
<td>$\mathcal{X}, \mathcal{Y}, K, \ldots$</td>
<td>matrices (or operators)</td>
</tr>
<tr>
<td>$\mathcal{X}, \mathcal{Y}$</td>
<td>block matrices</td>
</tr>
<tr>
<td>$\Phi, \Lambda, \Gamma, \ldots$</td>
<td>linear maps on matrices</td>
</tr>
<tr>
<td>$\top$</td>
<td>transpose</td>
</tr>
</tbody>
</table>

Table 1. Notational conventions used in the paper.
We now turn our attention to the question of how to estimate $q$. When

We now are set to define the partial trace regression model.

\[ \hat{\Phi} = \begin{bmatrix} \hat{\Phi}_{11} & \hat{\Phi}_{12} \\ \hat{\Phi}_{21} & \hat{\Phi}_{22} \end{bmatrix} \]

where $\hat{\Phi}$ is the partial trace of $\Phi$.

Remark 1 (Alternative partial trace) The other way of generalizing the trace operator to block matrices is the so-called block trace (Filipiak et al., 2018), which sums the diagonal blocks of block matrices. We do not use it here.

2.2. Model

We now are set to define the partial trace regression model.

Definition 2 (Partial trace regression model) The partial trace regression model assumes for a matrix-valued covariate pair $(X, Y)$, with $X$ taking value in $\mathbb{M}_p$ and $Y$ taking value in $\mathbb{M}_q$.

\[ Y = \text{tr}_m(A_\ast X B_\ast^\top) + \epsilon, \]

where $A_\ast, B_\ast \in \mathbb{M}_{qm \times p}$ are the unknown parameters of the model and $\epsilon$ is some matrix-valued random noise.

This assumes a stochastic linear relation between the input $X$ and the corresponding output $Y$ that, given an IID training sample $\{(X_i, Y_i)\}_{i=1}^l$ drawn according to (4), points to the learning of a linear mapping $\hat{\Phi} : \mathbb{M}_p \rightarrow \mathbb{M}_q$ of the form

\[ \hat{\Phi}(X) = \text{tr}_m(A \hat{X} B^\top), \]

where $\hat{\Phi} \in \mathbb{M}_{M,q}$ are the parameters to be estimated.

When $q = 1$, we observe that $\text{tr}_m(A_\ast X B_\ast^\top) = \text{tr}(A_\ast X B_\ast^\top)$, which is exactly the trace regression model (1), with a parametrization of the regression matrix as $B_\ast^\top A_\ast$.

We now turn our attention to the question of how to estimate the matrix parameters of the partial trace regression model while, as was done in (2) and (3), imposing some structure on the estimated parameters, so for the learning to come with statistical guarantees. As we will see, our solution to this problem takes inspiration from the fields of quantum information and quantum computing, and amounts to the use of the Kraus representation of completely positive maps.

2.3. Completely Positive Maps, Kraus Decomposition

The space $\mathcal{L}(M_p, M_q)$ of linear maps from $M_p$ to $M_q$ is a real vector space that has been thoroughly studied in the fields of mathematics, physics, and more specifically quantum computation and information (Bhatia, 2009; Nielsen & Chuang, 2000; Størmer, 2012; Watrous, 2018).

Operators from $\mathcal{L}(M_p, M_q)$ that have special properties are the so-called completely positive maps, a family that builds upon the notion of positive operators.

Definition 3 (Positive maps, Bhatia, 2009) A linear map $\Phi \in \mathcal{L}(M_p, M_q)$ is positive if for all $M \in \mathbb{S}_m$, $\Phi(M) \in \mathbb{S}_q$.

To define completely positive maps, we are going to deviate a bit from the block matrix structure advocated before and consider block matrices from $\mathbb{M}_m(M_p)$ and $\mathbb{M}_m(M_q)$.

Definition 4 (Completely positive maps, Bhatia, 2009) $\Phi \in \mathcal{L}(M_p, M_q)$ is $m$-positive if the associated map $\Phi_m : \mathcal{L}(\mathbb{M}_m(M_p), \mathbb{M}_m(M_q))$ which computes the $(i, j)$-th block of $\Phi_m(M)$ as $\Phi(M_{ij})$ is positive.

$\Phi$ is completely positive if it is $m$-positive for any $m \geq 1$.

The following theorem lays out the connection between partial trace regression models and positive maps.

Theorem 1 (Stinespring representation, Watrous, 2018) Let $\Phi \in \mathcal{L}(M_p, M_q)$. $\Phi$ writes as $\Phi(X) = \text{tr}_m(A X A^\top)$ for some $m \in \mathbb{N}$ and $A \in \mathbb{M}_{qm \times p}$ if and only if $\Phi$ is completely positive.

This invites us to solve the partial trace learning problem by looking for a map $\hat{\Phi} \in \mathcal{L}(M_p, M_q)$ of the form

\[ \hat{\Phi}(X) = \text{tr}_m(A \hat{X} A^\top), \]

where now, in comparison to the more general model of (5), the operator $\hat{\Phi}$ to be estimated is a completely positive map that depends on a sole matrix parameter $\hat{A}$. Restricting ourselves to such maps might seem restrictive but i) this is nothing but the partial trace version of the PSD contrained trace regression model of (3), which allows us to establish statistical guarantees, ii) the entailed optimization problem can take advantage of the Kraus decomposition of completely positive maps (see below) and iii) empirical performance is not impaired by this modelling choice.

Now that we have decided to focus on learning completely positive maps, we may introduce the last ingredient of our model, the Kraus representation.

Theorem 2 (Kraus representation, Bhatia, 2009) Let $\Phi \in \mathcal{L}(M_p, M_q)$ be a completely positive linear map. Then there exist $A_j \in \mathbb{M}_{q \times r}$, $1 \leq j \leq r$, with $r \leq pq$ such that

\[ \forall X \in \mathbb{M}_p, \quad \Phi(X) = \sum_{j=1}^r A_j X A_j^\top. \]
The matrices $A_j$ are called Kraus operators and $r$ the Kraus rank of $\Phi$.

With such a possible decomposition, learning a completely positive map $\Phi$ can reduce to finding Kraus operators $A_j$, for some fixed hyperparameter $r$, where small values of $r$ correspond to low-rank Kraus decomposition (Aubrun, 2009; Lancien & Winter, 2017), and favor computational efficiency and statistical guarantees. Given a sample $\{(X_i, Y_i)\}_{i=1}$, the training problem can now be written as:

$$\arg\min_{A_j \in M_{q \times p}} \sum_{j=1}^{r} \ell(Y_i, \sum_{j=1}^{r} A_j X_i A_j^\top),$$

where $\ell$ is a loss function. The loss function we use in our experiments is the square loss $\ell(Y, Y') = \|Y - Y'\|_F^2$, where $\|\cdot\|_F$ is the Frobenius norm. When $\ell$ is the square loss and $q = 1$, problem (8) reduces to the PSD-constrained trace regression problem (3), with a parametrization of the regression matrix as $\sum_{j=1}^{r} A_j^\top A_j$.

**Remark 2** Kraus and Stinespring representations can fully characterize completely positive maps. It has been shown that for a Kraus representation of rank $r$, there exists a Stinespring representation with dimension $m$ equal to $r$ (Watrous, 2018, Theorem 2.22). Note that the Kraus representation is rather computationally friendly compared to Stinespring representation. It has a simpler form and is easier to store and manipulate, as no need to create the matrix $A$ of size $qm \times p$. It also allows us to derive a generalization bound for partial trace regression, as we will see later.

**Optimization.** Assuming that the loss function $\ell(\cdot, \cdot)$ is convex in its second argument, the resulting objective function is non-convex with respect to $A_j$. If we further assume that the loss is coercive and differentiable, then the learning problem admits a solution that is potentially a local minimizer. In practice, several classical approaches can be applied to solve this problem. We have for instance tested a block-coordinate descent algorithm (Luo & Tseng, 1992) that optimizes one $A_j$ at a time. However, given current algorithmic tools, we have opted to solve the learning problem (8) using autodifferentiation (Baydin et al., 2017) and stochastic gradient descent, since the model can be easily implemented as a sum of product of matrices. This has the advantage of being efficient and allows one to leverage on efficient computational hardware, like GPUs.

Note that at this point, although not backed by theory, we can consider multiple layers of mappings by composing several mappings $\{\Phi_k\}$. This way of composing would extend the BiMap layer introduced by Huang & Van Gool (2017) which limits their models to rank 1 Kraus decomposition. Interestingly, they also proposed a ReLU-like layer for PSD matrices that can be applicable to our work as well. For two layers, this would boil down to $\Phi_2 \circ \Phi_1(X) = \sum_{j=1}^{r_2} A_j^{(2)} \Gamma \sum_{j_1=1}^{r_1} A_j^{(1)} X A_j^{(1)\top} A_j^{(2)\top}$, where $\Gamma$ is a nonlinear activation that preserves positive semidefiniteness. We investigate also this direction in our experiments.

**Generalization.** We now examine the generalization properties of partial trace regression via low-rank Kraus decomposition. Specifically, using the notion of pseudo-dimension, we provide an upper bound on the excess-risk for the function class $\mathcal{F}$ of completely positive maps with low Kraus rank, i.e.,

$$\mathcal{F} = \{ \Phi : M_p \to M_q : \Phi \text{ is completely positive and its Kraus rank is equal to } r \}.$$

Recall that the expected loss of any hypothesis $h \in \mathcal{F}$ is defined by $R(h) = \mathbb{E}_{(X,Y)} [\ell(Y, h(X))]$ and its empirical loss by $\hat{R}(h) = \frac{1}{l} \sum_{i=1}^{l} \ell(Y_i, h(X_i))$.

The analysis presented here follows the lines of Srebro (2004) in which generalization bounds were derived for low-rank matrix factorization (see also Rabusseau & Kadri (2016) where similar results were obtained for low-rank tensor regression). In order to apply known results on pseudo-dimension, we consider the class of real-valued functions $\mathcal{F}$ with domain $M_p \times [q] \times [q]$, where $[q] \equiv \{1, \ldots, q\}$, defined by

$$\tilde{\mathcal{F}} = \{(X, s, t) \mapsto (\Phi(X))_{st} : \Phi(X) = \sum_{j=1}^{r} A_j X A_j^\top \}.$$

**Lemma 3** The pseudo-dimension of the real-valued function class $\tilde{\mathcal{F}}$ is upper bounded by $pq r \log(\frac{3 epq}{r})$.

We can now invoke standard generalization error bounds in terms of the pseudodimension (Mohri et al., 2018, Theorem 10.6) to obtain:

**Theorem 4** Let $\ell : M_q \to \mathbb{R}$ be a loss function satisfying

$$\ell(Y, Y') = \frac{1}{q^2} \sum_{s,t} \ell'(Y_{st}, Y'_{st})$$

for some loss function $\ell' : \mathbb{R} \to \mathbb{R}^+$ bounded by $\gamma$. Then for any $\delta > 0$, with probability at least $1 - \delta$ over the choice of a sample of size $l$, the following inequality holds for all $h \in \mathcal{F}$:

$$R(h) \leq \hat{R}(h) + \gamma \sqrt{\frac{pq r \log(\frac{3 epq}{r}) \log(\frac{1}{\delta})}{l}} + \gamma \sqrt{\frac{\log(\frac{1}{\delta})}{2l}}.$$

The proofs of Lemma 3 and Theorem 4 are provided in the Supplementary Material. Theorem 4 shows that the Kraus
rank $r$ plays the role of a regularization parameter. Our generalization bound suggests a trade-off between reducing the empirical error which may require a more complex hypothesis set (large $r$), and controlling the complexity of the hypothesis set which may increase the empirical error (small $r$).

### 2.4. Application to PSD Matrix Completion

Our partial trace model is designed to address the problem of matrix-to-matrix regression. We now show how it can also be applied to the problem of matrix completion. We start by recalling how the matrix completion problem fits into standard trace regression model. Let $B^* \in \mathbb{M}_m$ be a matrix whose entries $B_{ij}^*$ are given only for some $(i, j) \in \Omega$. Low-rank matrix completion can be addressed by solving the following optimization problem:

$$
\min_B \|P_{\Omega}(B) - P_{\Omega}(B^*)\|^2, \text{ s.t. } \text{rank}(B) = r, \tag{9}
$$

where $P_{\Omega}(B)_{ij} = B_{ij}$ if $(i, j) \in \Omega$, 0 otherwise. This problem can be cast as a trace regression problem by considering $y_{ij} = P_{\Omega}(B^*)_{ij}$ and $X_{ij} = E_{ij}$, where $E_{ij}$, $1 \leq i, j \leq m$, are the matrix units of $\mathbb{M}_m$. Indeed, it is easy to see that in this case problem (9) is equivalent to

$$
\min_B \sum_{(i,j)\in \Omega} \left( y_{ij} - \text{tr}(BX_{ij}) \right)^2 \text{ s.t. } \text{rank}(B) = r. \tag{10}
$$

Since the partial trace regression is a generalization of the trace regression model, one can ask what type of matrix completion problems can be viewed as partial trace regression models. The answer to this question is given by the following theorem.

**Theorem 5** (Hiai & Petz, 2014, Theorem 2.49)

Let $\Phi : \mathbb{M}_p \rightarrow \mathbb{M}_q$ be a linear mapping. Then the following conditions are equivalent:

1. $\Phi$ is completely positive.

2. The block matrix $M \in \mathbb{M}_p(\mathbb{M}_q)$ defined by

$$
M_{ij} = \Phi(E_{ij}), \ 1 \leq i, j \leq p, \tag{11}
$$

is positive, where $E_{ij}$ are the matrix units of $\mathbb{M}_p$.

Theorem 5 makes the connection between PSD block decomposable matrices and completely positive maps. Our partial trace regression formulation is based on learning completely positive maps via low-rank Kraus decomposition, and thus can be applied to the problem of PSD matrix completion. The most straightforward application of Theorem 5 to PSD matrix completion is to consider the case where the matrix is block-structured with missing blocks. This boils down to solving the following optimization problem

$$
\min_{A_k \in \mathbb{M}_{q \times p}} \sum_{i,j=1}^p \|Y_{ij} - \Phi(X_{ij})\|_F^2 \text{ s.t. } \Phi(\cdot) = \sum_{k=1}^r A_k A_k^\top,
$$

where $\| \cdot \|_F$ is the Frobenius norm, $Y_{ij}$ are the observed blocks of the matrix and $X_{ij}$ are the corresponding matrix units. So, a completely positive map $\Phi$ can be learned by our approach from the available blocks and then can be used to predict the missing blocks. This would be a natural approach to take into account local structures in the matrix and then improve the completion performance. This approach can be also applicable in situations where no information about the block-decomposability of the data matrix to be completed is available. In this case, the size of the blocks $q$ and the number of the blocks $p$ can be viewed as hyperparameters of the model, and can be tuned to fit the data. Note that when $q = 1$, our method reduces to the standard trace regression-based matrix completion.

### 3. Experiments

In this section we turn our attention to evaluating our proposed partial trace regression (PTR) method. We conduct experiments on two tasks where partial trace regression can help, both in a simulated setting and exploiting real data. In all the experiments, the PTR model is implemented in a keras/Tensorflow framework and learned with Adam with default learning rate (0.001) and for 100 epochs. Batch size is typically 16. Our code is available at https://github.com/Stef-hub/partial_trace_kraus.

#### 3.1. PSD to PSD Matrix Regression

We will now describe experiments where the learning problem can be easily described as learning a mapping between two PSD matrices; first with simulated data and then applied to learning covariance matrices for Brain-Computer Interfaces.

##### 3.1.1. Experiments on Simulated Data

Our first goal is to show the ability of our model to accurately recover mappings conforming to its assumptions. We randomly draw a set of matrices $X_i$ and $A_r$, and build the matrices $Y_i$ using Equation 7, for various Kraus ranks $r$, and $p$, $q$ the size of input and output spaces respectively. We train the model with 100, 1000 and 10000 samples on two simulated datasets with Kraus ranks 5 and 20, and show the results for maps $20 \times 20 \rightarrow 10 \times 10$ and $100 \times 100 \rightarrow 40 \times 40$ in Figure 2. While 100 samples is not enough to get a good estimation, with more data the PTR is able to accurately represent the model with correct rank.
Figure 2. PSD to PSD predictive performance (mean squared error) of PTR as a function of its Kraus rank, on simulated data from map $20 \times 20 \to 10 \times 10$ (top) and map $100 \times 100 \to 40 \times 40$ (bottom).

We also aim to show, in this setting, a comparison of our All the models are trained using 10000 simulated examples with a low-rank constraint on $B$ (Novikov et al., 2015; Cao & Rabusseau, 2017).

Note that, in contrast to PTR, tensor train NN does not preserve the PSD property.

We ran these experiments also with fewer examples (100 instead of 10000). In this case PTR performs better than tensor train NN (0.007 ± 0.013 for PTR and 0.662 ± 0.364 for tensor train NN) and again better than the other baselines.

3.1.2. Mapping Covariance Matrices for Domain Adaptation

In some applications like 3D shape recognition or Brain-Computer Interfaces (Barachant et al., 2011; Tabia et al., 2014), features take the form of covariance matrices and algorithms taking into account the specific geometry of these objects are needed. For instance, in BCI, "minimum distance to the mean (in the Riemannian sense)" classifier has been shown to be a highly efficient classifier for categorizing motor imagery tasks.

For these tasks, distribution shifts usually occur in-between sessions of the same subject using the BCI. In such situations, one solution is to consider an optimal transport mapping of the covariance matrices from the source to the target domain (the different sessions) (Yair et al., 2019; Courty et al., 2016). Here, our goal is to learn such a covariance matrix mapping and to perform classification using covariance matrix from one session (the source domain) as training data and those of the other session (the target domain) as test data. For learning the mapping, we will consider only matrices from the training session.

We adopt the experimental setting described by Yair et al. (2019) for generating the covariance matrix. From the optimal transport-based mapping obtained in an unsupervised way from the training session, we have couples of input-output matrices of size $22 \times 22$ from which we want to learn our partial trace regressor. While introducing noise into the classification process, the benefit of such regression function is to allow out-of-sample domain adaptation as in

\begin{table}[h]
\centering
\begin{tabular}{|c|c|}
\hline
Model & MSE \\
\hline
Partial Trace Regression & 0.001 ± 0.0008 \\
Trace Regression & 0.028 ± 0.0144 \\
Multivariate Regression & 0.058 ± 0.0134 \\
Reduced-rank Multivariate Regression & 0.245 ± 0.1023 \\
HOPLS & 1.602 ± 0.0011 \\
Tensor Train NN & 0.001 ± 0.0009 \\
Tucker NN & 0.595 ± 0.0252 \\
\hline
\end{tabular}
\caption{Comparison of various regression models on PSD to PSD task on simulated data with map $20 \times 20 \to 10 \times 10$ and true Kraus rank of the model 5. We report the best performance among various tested model ranks from 1 to 100.}
\end{table}
Table 3. Accuracy of a “minimum distance to the mean” classifier on domain adaptation BCI task. We report the results for the same subjects as in (Yair et al., 2019).

<table>
<thead>
<tr>
<th>Subject</th>
<th>NoAdapt</th>
<th>FullAdapt</th>
<th>OoSAdapt</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>73.66</td>
<td>73.66</td>
<td>72.24</td>
</tr>
<tr>
<td>3</td>
<td>65.93</td>
<td>72.89</td>
<td>68.86</td>
</tr>
<tr>
<td>7</td>
<td>53.42</td>
<td>64.62</td>
<td>59.20</td>
</tr>
<tr>
<td>8</td>
<td>73.80</td>
<td>75.27</td>
<td>73.06</td>
</tr>
<tr>
<td>9</td>
<td>73.10</td>
<td>75.00</td>
<td>76.89</td>
</tr>
</tbody>
</table>

Perrot et al. (2016). This means that we do not need to solve an optimal transport problem every time a new sample (or a batch of new samples) is available. In practice, we separate all the matrices (about 270) from the training session in a training/test group, and use the training examples (230 samples) for learning the covariance mapping. For evaluating the quality of the learned mapping, we compare the classification performance of a minimum distance to the mean classifier in three situations:

- No adaptation between domains is performed. The training session data is used as is. The method is denoted as NoAdapt.
- All the matrices from the training session are mapped using the OT mapping. This is a full adaptation approach, denotes as FullAdapt
- Our approach denoted as OoSAdapt (from Out of Sample Adaptation) uses the 230 covariance matrices mapped using OT and the other matrices mapped using our partial trace regression model.

For our approach, we report classification accuracy for a model of rank 20 and depth 1 trained using an Adam optimize of learning rate 0.001 during 500 iterations. We have also tested several other hypeparameters (rank and depth) without much variations in the average performance.

Classification accuracy over the test set (the matrice from the second session) is reported in Table 3. We first note that for all subjects, domain adaptation helps in improving performance. Using the estimated mapping instead of the true one for projecting source covariance matrix into the target domain, we expect a loss of performance with respect to the FullAdapt method. Indeed, we observe that for Subject 1 and 8, we occur small losses. For the other subjects, our method allows to improve performance compared to no domain adaptation while allowing for out-of-sample classification. Interestingly, for Subject 9, using estimated covariance matrices performs slightly better than using the exact ones.

Figure 3. Completion performance on simulated $28 \times 28$ matrix, with $p = 7$ and $q = 4$. Top left, middle and right: original matrix, original matrix with missing values, our result with rank 1 Kraus decomposition. Bottom left, middle and right: Our results with Kraus rank 5, 30 and 100.

Table 4. Comparison of partial trace regression and trace regression and tensor train neural networks on (block) PSD matrix completion on simulated $28 \times 28$ matrix with missing blocks.

<table>
<thead>
<tr>
<th>Model</th>
<th>MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partial Trace Regression</td>
<td>0.572 ± 0.019</td>
</tr>
<tr>
<td>Trace Regression</td>
<td>2.996 ± 1.078</td>
</tr>
<tr>
<td>Tensor Train NN</td>
<td>3.942 ± 1.463</td>
</tr>
</tbody>
</table>

3.2. PSD Matrix Completion

We now describe our experiments in the matrix completion setting, first by illustrative examples with simulated data, then more comprehensively in the setting of multi-view kernel matrix completion.

3.2.1. Experiments on Simulated Data

We consider the problem of matrix completion in two settings: filling in fully missing blocks, and filling in individually missing values in a matrix. We perform our experiments on full rank PSD matrices of size $28 \times 28$.

We show the results on block completion in Figure 3, where we have trained our partial trace regression model (without stacking) with Kraus ranks 1, 5, 30 and 100. While using Kraus rank 1 is not enough to retrieve missing blocs, rank 5 gives already reasonable results, and ranks 30 and 100 are able to infer diagonal values that were totally missing from the training blocks. Completion performance in terms of mean squared error are reported in Table 4, showing that our PTR method performs favorably against trace regression and tensor train neural networks.

Figures 4 and 5 illustrate the more traditional completion...
Partial Trace Regression and Low-Rank Kraus Decomposition

Figure 4. Completion performance on simulated $28 \times 28$ matrix, with $p = 7$ and $q = 4$. Top row: target matrix and target matrix with missing data. Bottom row: Our completion results with depth=1 (left) and depth=2 (right).

Figure 5. Completion performance on simulated $28 \times 28$ matrix, with $p = 7$ and $q = 4$. Top row: target matrix and target matrix with missing data. Bottom row: Our completion results with depth=1 (left) and depth=2 (right).

Figure 6. Sum of the matrix completion errors for trace regression (TR) and our partial trace regression (PTR), over the views of the multiple features digits dataset as a function of missing samples with model depths 1 and 2.

We fix $p$ and $q$ to 7 and 4, respectively, and investigate the effect that stacking the models has on the completion performance. Note that $p$ and $q$ may be chosen via cross-validation. With only a little missing data (Figure 4) there is very little difference between the results obtained with one-layer and two-layer models. However we observe that for the more difficult case (Figure 5), stacking partial trace regression models significantly improves the reconstruction performance.

3.2.2. Similarity Matrix Completion

For our last set of experiments, we evaluate our partial trace regression model in the task of matrix completion on a multi-view dataset, where each view contains missing samples. This scenario occurs in many fields where the data comes from various sensors of different types producing heterogeneous view data (Rivero et al., 2017). Following Huusari et al. (2019), we consider the multiple features digits dataset, available online.\(^3\) The task is to classify digits (0 to 9) from six descriptions extracted from the images, such as Fourier coefficients or Karhunen-Loève coefficients. We selected 20 samples from all the 10 classes, and computed RBF kernels for views with data samples in $\mathbb{R}^d$, and $\text{Chi}^2$ kernels for views with data samples in $\mathbb{Z}^d$, resulting in six $200 \times 200$ kernel matrices. We then randomly introduced missing samples within views, leading to missing values (rows/columns) into kernel matrices. We vary the level of total missing samples in the whole dataset from 10% to 90%, by taking care that all the samples are observed at least in one view, and that all views have observed samples.

We first measure the matrix completion performance on reconstruction quality by computing $\sum_v \|K_v - \hat{K}_v\|_F$ with $K_v$ the original kernel matrix for view $v$ and $\hat{K}_v$ the completed one. We then analyse the success of our method in the classification task by learning an SVM classifier from a simple average combination of input kernels. Here we compare our partial trace regression method to two very simple baselines of matrix completion, namely zero and mean input, as well as the more relevant CVKT method presented in Huusari et al. (2019). We perform the matrix completion with our algorithm with three block-structure configurations; $p, q = 20, 10$, $p, q = 10, 20$, and finally with $p, q = 200, 1$, which corresponds to trace regression. We consider both depths 1 and 2 for our partial trace regression model, as well as Kraus ranks 50, 100 and 200.

Figure 6 shows the sum of matrix completion errors $\sum_v \|K_v - \hat{K}_v\|_F$ for our method and the trace regression method in various configurations. For depth 2, the partial trace regression clearly outperforms the more simple trace regression ($p, q = 200, 1$). With model depth 1 all the methods perform similarly. It might be that the real data

\(^3\)https://archive.ics.uci.edu/ml/datasets/Multiple+Features
considered in this experiment is more complex and non-linear than our model assumes, thus stacking our models is useful for the performance gain. However the traditional trace regression does not seem to be able to capture the important aspects of this data even in the stacked setting.

Figure 7 shows in the left panel the SVM classification accuracies obtained by using the kernel matrices from various completion methods, and detailed results focusing on our method in the middle and right side panels. For these experiments, we selected to use in SVMs the kernel matrices giving the lowest completion errors. We observe that our model provides excellent kernel completion since the classification accuracy is close to the performance of the setting with no missing data. The stacked model seems to be able to accurately capture the data distribution, giving rise to very good classification performance. The middle panel confirms the observations from Figure 6: the depth of our model plays a crucial part on its performance, with depth 2 outperforming the depth 1 in almost every case, except the choice of \( p, q = 200, 1 \) corresponding to trace regression. The Kraus rank does not have a strong effect on classification performance (right panel). Indeed, this justifies the usage of our method in a low-rank setting.

### 4. Conclusion

In this paper, we introduced partial trace regression model, a family of linear predictors from matrix-valued inputs to matrix-valued outputs that generalizes previously proposed models such as trace regression and linear regression. We proposed a novel framework for estimating the partial trace regression model from data by learning low-rank Kraus decompositions of completely positive maps, and derived an upper bound on the generalization error. Our empirical study with synthetic and real-world datasets shows the promising performance of our proposed approach on the tasks of matrix-to-matrix regression and positive semidefinite matrix completion.

### Acknowledgements

This work has been funded by the French National Research Agency (ANR) project QuantML (grant number ANR-19-CE23-0011). Part of this work was performed using computing resources of CRIANN (Normandy, France). The work by RH has in part been funded by Academy of Finland grants 334790 (MAGITICS) and 310107 (MACOME).

### References


Partial Trace Regression and Low-Rank Kraus Decomposition


Supplementary Material

In this supplementary material, we prove Lemma 3 and Theorem 4 in Section 2.3 of the main paper. Let us first recall the definition of pseudo-dimension.

Definition 5 (Shattering Mohri et al., 2018, Def. 10.1)
Let $G$ be a family of functions from $X$ to $\mathbb{R}$. A set $\{x_1, ..., x_m\} \subset X$ is said to be shattered by $G$ if there exist $t_1, ..., t_m \in \mathbb{R}$ such that,

$$f(x) = \left\{ \left[ \begin{array}{c} \text{sign}(g(x_1) - t_1) \\ \vdots \\ \text{sign}(g(x_m) - t_m) \end{array} \right] : g \in G \right\} = 2^m.$$

Definition 6 (pseudo-dimension Mohri et al., 2018, Def. 10.2)
Let $G$ be a family of functions from $X$ to $\mathbb{R}$. Then, the pseudo-dimension of $G$, denoted by $\text{Pdim}(G)$, is the size of the largest set shattered by $G$.

In the following we consider that the expected loss of any hypothesis $h \in \mathcal{F}$ is defined by $R(h) = E_{X,Y}[\ell(Y, h(X))]$ and its empirical loss by $\hat{R}(h) = \frac{1}{m} \sum_{i=1}^{m} \ell(Y, h(X))$. To prove Lemma 3 and Theorem 4, we need the following two results.

Theorem 6 (Srebro, 2004, Theorem 35)
The number of sign configurations of $m$ polynomials, each of degree at most $d$, over $n$ variables is at most $\left( \frac{4 \text{Pdim}}{n} \right)^n$ for all $m > n > 2$.

Theorem 7 (Mohri et al., 2018, Theorem 10.6)
Let $H$ be a family of real-valued functions and let $G = \{x \mapsto L(h(x), f(x)) : h \in H\}$ be the family of loss functions associated to $H$. Assume that the pseudo-dimension of $G$ is bounded by $d$ and that the loss function $L$ is bounded by $M$. Then, for any $\delta > 0$, with probability at least $\delta$ over the choice of a sample of size $m$, the following inequality holds for all $h \in H$:

$$R(h) \leq \hat{R}(h) + M \sqrt{\frac{2d \log \left( \frac{4 \text{Pdim}}{n} \right)}{m}} + M \sqrt{\frac{\log \left( \frac{1}{\delta} \right)}{2m}}.$$

5. Proof of Lemma 3

We now prove Lemma 3 in Section 2.3 of the main paper.

Lemma 3 The pseudo-dimension of the real-valued function class $\mathcal{F}$ with domain $\mathbb{M}_p \times [q] \times [q]$ defined by

$$\mathcal{F} = \{(X, s, t) \mapsto (\Phi(X))_{st} : \Phi(X) = \sum_{j=1}^{r} A_j X A_j^T \}$$
we obtain that the number of sign configurations, which is equal to its dimension (Mohri et al., 2018, Theorem 10.5). Since $\mathcal{F}$ is a subspace of the $p^2q^2$-dimensional vector space
\[
\{(X, s, t) \mapsto (\Phi(X))_{st} : \Phi \in \mathcal{L}(M_p, M_q)\}
\]
of real-valued functions with domain $M_p \times [q] \times [q]$ the pseudo-dimension of $\mathcal{F}$ is bounded by $p^2q^2$.

Now, let $m \leq p^2q^2$ and let $\{(X_k, s_k, t_k)\}_{k=1}^m$ be a set of points that are pseudo-shattered by $\mathcal{F}$ with thresholds $t_1, \ldots, t_m \in \mathbb{R}$. Then for each binary labeling $(u_1, \ldots, u_m) \in \{-, +\}^m$, there exists $\tilde{\Phi} \in \mathcal{F}$ such that $\text{sign}(\tilde{\Phi}(X_k, s_k, t_k) - v_k) = u_k$. Any function $\tilde{\Phi} \in \mathcal{F}$ can be written as
\[
\tilde{\Phi}(X, s, t) = \left( \sum_{j=1}^r A_j X A_j^\top \right)_{st},
\]
where $A_j \in M_{q \times p}, \forall j \in [r]$. If we consider the $pqr$ entries of $A_j$, $j = 1, \ldots, r$, as variables, the set $\{(\tilde{\Phi}(X_k, s_k, t_k) - v_k)\}_{k=1}^m$ can be seen (using Eq. 12) as a set of $m$ polynomials of degree 2 over these variables. Applying Theorem 6 above, we obtain that the number of sign configurations, which is equal to $2^m$, is bounded by $\left( \binom{8m}{pqr} \right)^{pqr}$. The result follows since $m \leq p^2q^2$.

6. Proof of Theorem 4

In this section, we prove Theorem 4 in Section 2.3 of the main paper.

**Theorem 4** Let $\ell : M_q \rightarrow \mathbb{R}$ be a loss function satisfying
\[
\ell(Y, Y') = \frac{1}{q^2} \sum_{s,t} \ell'(Y_{st}, Y'_{st})
\]
for some loss function $\ell' : \mathbb{R} \rightarrow \mathbb{R}^+$ bounded by $\gamma$. Then for any $\delta > 0$, with probability at least $1 - \delta$ over the choice of a sample of size $l$, the following inequality holds for all $h \in \mathcal{F}$:
\[
R(h) \leq \hat{R}(h) + \gamma \sqrt{\frac{pqr \log \left( \frac{8pq}{r} \right) \log \left( \frac{1}{\delta} \right)}{l}} + \gamma \sqrt{\frac{\log \left( \frac{1}{\delta} \right)}{2l}}.
\]

**Proof:** For any $h : M_p \rightarrow M_q$ we define $\tilde{h} : M_p \times [q] \times [q] \rightarrow \mathbb{R}$ by $\tilde{h}(X, s, t) = (h(X))_{st}$. Let $\mathcal{D}$ denote the distribution of the input-output data. We have
\[
R(h) = \mathbb{E}_{(X,Y) \sim \mathcal{D}}[\ell(Y, h(X))]
\]
\[
= \frac{1}{q^2} \sum_{s,t} \mathbb{E}_{(X,Y) \sim \mathcal{D}}[\ell'(Y_{st}, h(X)_{st})]
\]
\[
= \mathbb{E}_{(X,Y) \sim \mathcal{D}}[\ell'(Y_{st}, \tilde{h}(X, s, t))],
\]
where $\mathcal{U}(q)$ denotes the discrete uniform distribution on $[q]$. It follows that $R(h) = R(\tilde{h})$. By the same way, we can show that $\hat{R}(h) = R(\tilde{h})$. The generalization bound is then obtained using Theorem 7 above. 

\[\blacksquare\]