Max–Planck–Institut für biologische Kybernetik
Max Planck Institute for Biological Cybernetics

Technical Report No. 128

Multivariate Regression with Stiefel Constraints

Gökhan H. Bakır,[1] Arthur L. Gretton,[1] Matthias O. Franz,[1] Bernhard Schölkopf[1]

July 2004

This Technical Report has been approved by:

Director at MPIK

Postdoc at MPIK

Max–Planck–Institut für biologische Kybernetik
Max Planck Institute for Biological Cybernetics

Technical Report No. 128

Multivariate Regression with Stiefel Constraints

Gökhan H. Bakır,[1] Arthur L. Gretton,[1] Matthias O. Franz,[1] Bernhard Schölkopf[1]

July 2004

[1] Department Schölkopf, email: gb;arthur;mof;bs@tuebingen.mpg.de

Multivariate Regression with Stiefel Constraints

Gökhan H. Bakır, Arthur L. Gretton, Matthias O. Franz, and Bernhard Schölkopf

Abstract. We introduce a new framework for regression between multi-dimensional spaces. Standard methods for solving this problem typically reduce the problem to one-dimensional regression by choosing features in the input and/or output spaces. These methods, which include PLS (partial least squares), KDE (kernel dependency estimation), and PCR (principal component regression), select features based on different a-priori judgments as to their relevance. Moreover, loss function and constraints are chosen not primarily on statistical grounds, but to simplify the resulting optimisation. By contrast, in our approach the feature construction and the regression estimation are performed jointly, directly minimizing a loss function that we specify, subject to a rank constraint. A major advantage of this approach is that the loss is no longer chosen according to the algorithmic requirements, but can be tailored to the characteristics of the task at hand; the features will then be optimal with respect to this objective. Our approach also allows for the possibility of using a regularizer in the optimization. Finally, by processing the observations sequentially, our algorithm is able to work on large scale problems.

---

1 Introduction

The problem of regressing between a high dimensional input space and a *continuous, univariate* output has been studied in considerable detail: classical methods are described in [1], and methods applicable when the input is in a reproducing kernel Hilbert space are discussed in [2]. When the output dimension is high (or even infinite), however, it becomes inefficient or impractical to apply univariate methods separately to each of the outputs, and specialized multivariate techniques must be used. Examples include

- regression to subsets of $\mathbb{R}^d$ for large d, where for instance we might wish to predict properties of a manufactured product on the basis of the machinery settings [3];
- regression to a reproducing kernel Hilbert space, which can be used to recover images from incomplete or corrupted data; or as a means of classification, through mapping to a suitable output space [4];
- regression between discrete spaces, such as finite state automata [5] and graphs [6], on which similarity measures may be defined via kernels.

We propose a novel method for regression between two spaces $\mathcal{F}_x$ and $\mathcal{F}_y$, where both spaces can have arbitrarily large dimension. Our algorithm works by choosing low dimensional subspaces in both $\mathcal{F}_x$ and $\mathcal{F}_y$,and finding the mapping between these subspaces for which a particular loss is small.[1] There are several reasons for learning a mapping between low dimensional subspaces, rather than between $\mathcal{F}_x$ and $\mathcal{F}_y$ in their entirety. First, $\mathcal{F}_x$ and $\mathcal{F}_y$ may have high dimension, yet our data are generally confined to smaller subspaces. Second, the outputs may be statistically dependent, and learning all of them at once allows us to exploit this dependence. Third, it is common practice (for instance in principal component regression (PCR)) to ignore certain directions in the input and/or output spaces, which decreases the variance in the regression coefficients (at the expense of additional bias): this is a form of regularization.

Given a particular subspace dimension, classical multivariate regression methods use a variety of heuristics for subspace choice.[2] The mapping between subspaces is then achieved as a second, independent step. By contrast, our method, Multivariate Regression via Stiefel Constraints (MRS), jointly optimises over the subspaces *and* the mapping; its goal is to find the subspace/mapping *combination* with the smallest possible loss. Drawing on results from differential geometry [7], we represent each subspace projection operator as an element on a Stiefel manifold. Our method then conducts gradient descent over these projections.

On small-scale problems, the subspace/mapping estimation problem may be solved using a batch method, in which optimisation is carried out over the entire data set. When the data set is large, however, this can have prohibitive

[1] The loss is specified by the user.

[2] For instance, PCR generally retains the input directions with highest *variance*, whereas partial least squares (PLS) approximates the input directions along which *covariance* with the outputs is high.

computational and memory requirements. For such cases, we propose an online variant of our algorithm, called Sequential Multivariate Regression via Stiefel Constraints (S-MRS), which processes a series of small subsets of the data, rather than requiring all of it at once. The sequential approach has two advantages: it allows us to easily update our prediction in the light of new observations, and to learn from large data sets by breaking the problem down into smaller learning tasks. The choice of subspaces and mapping must then be constrained, in that both are not permitted to change too much from those previously learned. This ensures we do not disregard past observations when given new data. The similarity of the new solution to the prior solutions is enforced using a *regularizing* term,[3] which is added to the original batch-based loss.

We begin our discussion in Section 2 with some basic definitions, and give a formal description of the multivariate regression setting (both batch and sequential). We then summarise various classical methods for multivariate regression (PLS, PCR, and KDE) in Section 3. In each case, the relevant subspaces in $\mathcal{F}_x$ and $\mathcal{F}_y$ are described, and the heuristics used to determine these subspaces are elucidated. Next, in Section 4, we introduce the MRS procedure for both the L_1 and L_2 losses, and the S-MRS procedure for the L_2 loss. The sets $\mathcal{X}$ and $\mathcal{Y}$ in which our inputs and outputs respectively exist may not correspond to $\mathcal{F}_x$ and $\mathcal{F}_y$; thus, in Section 5, we describe mappings from $\mathcal{X}$ to $\mathcal{F}_x$, and from $\mathcal{F}_y$ to $\mathcal{Y}$. Finally, in Section 6, we apply our method in three settings: an inverse kinematics problem for a robot arm, a (relatively) low dimensional image denoising application (restoring partly corrupted hand-written digits), and a high dimensional image denoising transformation (restoring corrputed images of faces). The first two experiments are a test of the batch algorithm, while the final one demonstrates the sequential version.

2 Problem setting and motivation

2.1 General description, batch setting

We first describe our regression framework in more detail, and introduce the variables we will use. We begin in the present section with the batch method, and then describe the online setting in Section 2.2. We are given m pairs of input and output variables, $\boldsymbol{z} := ((\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_m, \mathbf{y}_m))$, where $\mathbf{x}_i \in \mathcal{F}_x$, $\mathbf{y}_i \in \mathcal{F}_y$, and $\mathcal{F}_x$ and $\mathcal{F}_y$ are reproducing kernel Hilbert spaces with respective dimension l_x and l_y (we assume for the moment that these are finite; the implications of these being infinite, as in the RKHS associated with the Gaussian kernel, will be addressed later in Section 5). We write the matrices of *centered* observations as

$$\mathbf{X} := \left[\begin{array}{ccc} \mathbf{x}_1 & \ldots & \mathbf{x}_m \end{array} \right] \mathbf{H}, \quad \mathbf{Y} := \left[\begin{array}{ccc} \mathbf{y}_1 & \ldots & \mathbf{y}_m \end{array} \right] \mathbf{H},$$

[3] This regularization should not be confused with the regularizer used in ridge regression, which can *also* be used to constrain the solution obtained for each data *subset*; or indeed for the entire data set in the batch case.

where $\mathbf{H} := \mathbf{I} - \frac{1}{m}\mathbf{1}\mathbf{1}^\top$, and $\mathbf{1}$ is the $m \times 1$ matrix of ones.

We now specify our batch learning problem: given observations $\mathbf{X}$ and $\mathbf{Y}$, a loss function $\mathsf{L}(\mathbf{Y}, \mathbf{X}, \mathbf{F}_{(r)})$, and a regularizer Ω_d (this can for instance be a 2-norm as in ridge regression: see Section 4.4), we want to find the best predictor $\mathbf{F}_{(r)}$, defined as

$$\mathbf{F}_{(r)} \quad = \quad \min_{\mathbf{G} \in \mathcal{H}_{(r)}} \mathsf{L}(\mathbf{Y}, \mathbf{X}, \mathbf{G}) + \Omega_d(\mathbf{G}), \tag{2.1}$$

where

$$\mathcal{H}_{(r)} := \left\{ \mathbf{F} \in {\mathcal{F}_y}^{\mathcal{F}_x} \mid \text{rank } \mathbf{F} = r \right\} \tag{2.2}$$

and ${\mathcal{F}_y}^{\mathcal{F}_x}$ denotes the set of mappings from $\mathcal{F}_x$ to $\mathcal{F}_y$. This rank constraint is crucial to our approach: it allows us to restrict ourselves to subspaces *smaller* than those spanned by the input and/or output observations, which can reduce the variance in our estimate of the mapping $\mathbf{F}_{(r)}$ while increasing the bias. We select the rank that optimises over this bias/variance tradeoff using cross validation.

As we shall see in Section 4, our approach is not confined to any particular loss function. That said, in this study we address only the *least squares* loss function,

$$\mathsf{L}_2(\mathbf{Y}, \mathbf{X}, \mathbf{F}_{(r)}) \quad = \quad \|\mathbf{Y} - \mathbf{F}_{(r)}\mathbf{X}\|_F^2, \tag{2.3}$$

and the L_1 loss,

$$\mathsf{L}_1(\mathbf{Y}, \mathbf{X}, \mathbf{F}_{(r)}) \quad = \quad \sum_{i=1}^{m} \|\mathbf{y}_i - \mathbf{F}_{(r)}\mathbf{x}_i\|_1. \tag{2.4}$$

We now transform the rank constraint in (2.1) and (2.2) into a form more amenable to optimisation. By diagonalizing the predictor $\mathbf{F}_{(r)}$ via its singular basis, we obtain

$$\mathbf{F}_{(r)} \quad = \quad \mathbf{V}_{(r)} S_{(r)} {\mathbf{W}_{(r)}}^\top, \tag{2.5}$$

where

$$\begin{aligned} {\mathbf{V}_{(r)}}^\top \mathbf{V}_{(r)} &= \mathbf{I}_{r,r}, & (2.6)\\ {\mathbf{W}_{(r)}}^\top \mathbf{W}_{(r)} &= \mathbf{I}_{r,r}, & (2.7)\\ S &\in \text{diagonal } \mathbb{R}^{r \times r}. & (2.8) \end{aligned}$$

In other words, $\mathbf{W}_{(r)} \in \mathbb{S}(l_y, r)$ and $\mathbf{V}_{(r)} \in \mathbb{S}(l_x, r)$, where $\mathbb{S}(n, r)$ is called the *Stiefel* manifold, and comprises the matrices with n rows and r orthonormal columns. In the L_2 case, finding a rank constrained predictor (2.5) is thus equivalent to finding the triplet $\theta = (\mathbf{V}_{(r)}, S_{(r)}, \mathbf{W}_{(r)})$ for which

$$\theta \quad = \quad \underset{\mathbf{V}_{(r)}, S_{(r)}, \mathbf{W}_{(r)}}{\arg\min} \|\mathbf{Y} - \mathbf{V}_{(r)} S_{(r)} {\mathbf{W}_{(r)}}^\top \mathbf{X}\|_F^2, \tag{2.9}$$

subject to constraints (2.6)-(2.8)[4]. We will refer to $\mathbf{W}_{(r)}$ and $\mathbf{V}_{(r)}$ as *feature matrices.*

It is clear from (2.9) that $\mathbf{W}_{(r)}$ and $\mathbf{V}_{(r)}$ determine particular subspaces in $\mathcal{F}_x$ and $\mathcal{F}_y$ respectively, and that the regression procedure is a mapping between these subspaces. A number of classical multivariate regression methods also have this property, although the associated subspaces are not determined according to the criteria we propose. In Section 3, we will review the subspace selection methods used in three established regression algorithms, before returning in Section 4 to the solution of (2.9). First, however, we describe the online framework used in S-MRS.

2.2 Online setting

In the sequential case, we do not observe the examples all at once: rather, at every time instance k, we obtain only m observations. These are written as

$$\mathbf{X_k} := \left[\begin{array}{ccc} \mathbf{x}_{k_1} & \ldots & \mathbf{x}_{k_m} \end{array} \right], \quad \mathbf{Y_k} := \left[\begin{array}{ccc} \mathbf{y}_{k_1} & \ldots & \mathbf{y}_{k_m} \end{array} \right], \quad \text{and } \mathbf{Z}_k = [\mathbf{X}_k, \mathbf{Y}_k].$$

The online setting requires (2.1) and (2.2) to be slightly modified: thus, for a particular $\mathbf{G} \in \mathcal{H}_{(r)}$, we can define a loss $\mathsf{L}(\mathbf{Z}_k, \mathbf{G})$ measuring the prediction error of $\mathbf{G}$ on $\mathbf{Z}_k$, and a regularization functional $\Omega_o(\mathbf{G}, \mathbf{F}_{(r)}^{(k-1)})$ that penalises large differences between $\mathbf{G}$ and a reference mapping $\mathbf{F}_{(r)}^{(k-1)}$. We then want to find the best predictor $\mathbf{F}_{(r)}^{(k)}$, defined as

$$\mathbf{F}_{(r)}^{(k)} = \underset{\mathbf{G} \in \mathcal{H}_{(r)}}{\arg\min}\ \mathcal{L}(\mathbf{Z_k}, \mathbf{G}, \mathbf{F}_{(r)}^{(k-1)}) \tag{2.10}$$

$$= \underset{\mathbf{G} \in \mathcal{H}_{(r)}}{\arg\min}\ \mathsf{L}(\mathbf{Z}_k, \mathbf{G}) + \Omega_o(\mathbf{G}, \mathbf{F}_{(\mathbf{r})}^{(\mathbf{k-1})}). \tag{2.11}$$

We may understand the purpose of Ω_o by reference to the (non-sequential) ridge regression method, for which $\mathbf{F}_{(r)}^{(k-1)} = \mathbf{0}$ and

$$\Omega_o(\mathbf{G}, \mathbf{0}) = \|\mathbf{G}\|_F^2 = \Omega_d(\mathbf{G}) :$$

in this case the regularization prevents the norm $\|\mathbf{G}\|_F^2$ from growing too large (see (2.1), in which $\Omega_d(\mathbf{G})$ is used in this manner). In the online case, the reference mapping is the solution $\mathbf{F}_{(r)}^{(k-1)}$ obtained previously, in keeping with our goal of choosing a new mapping close to the former solution. By way of comparison, the Kalman-Bucy filter can also be written in the form (2.10) with an analogous reference mapping [9]. The specific form taken by Ω_o in our multivariate regression setting will be described in Section 4.5.

We again transform the rank constraint in (2.10) using the decomposition in (2.5) and (2.6)-(2.8). Thus, for the kth set of observations, finding a rank constrained predictor (2.5) is equivalent to finding the triplet $\theta = (\mathbf{V}_{(r)}, S_{(r)}, \mathbf{W}_{(r)})$

[4]This is a more general form of the *Procrustes* problem [8], for which $\mathbf{F}_{(r)}$ is orthogonal rather than being rank constrained.

for which

$$\theta \quad = \quad \underset{\mathbf{V}_{(r)}, S_{(r)}, \mathbf{W}_{(r)}}{\arg\min} \; \mathcal{L}(\mathbf{V}_{(r)} S_{(r)} \mathbf{W}_{(r)}{}^{\top}, \mathbf{F}_{(r)}^{(k-1)}) \tag{2.12}$$

subject to constraints (2.6)-(2.8).

3 Existing Multivariate Techniques

In this section, we describe PCR, PLS, and KDE, paying particular attemtion to the way in which a small number r of features is chosen, and how these are used to construct the mapping $\mathbf{F}_{(r)}$ (the features may be chosen in the input *or* output spaces, depending on the algorithm; the rank of the mapping is unaffected). We consider only the batch case for each algorithm. Since each of these methods relies on the L_2 loss in (2.3), we begin with a description of the *least squares solution* associated with this loss, and demonstrate how this solution changes when the *inputs* are projected onto a small number of features. This solution is then used as a basis for principal component regression (PCR) and partial least squares (PLS), although the choice of features differs between these algorithms. Finally, we describe kernel dependency estimation (KDE), in which features are chosen in the *output* space.

3.1 Least squares regression in a restricted basis

The multivariate least squares solution is equivalent to l_y separate univariate problems; thus, we first describe the univariate case. We wish to solve

$$\mathbf{b}^*_{(r)} \quad := \quad \underset{\mathbf{b}}{\arg\min} \left\| \mathbf{y} - \mathbf{X}^{\top} \mathbf{W}_{(r)} \mathbf{b} \right\|^2, \tag{3.1}$$

where $\mathbf{y}^{\top} = \left[\begin{array}{ccc} y_1 & \ldots & y_m \end{array} \right] \mathbf{H}$, and the solution $\mathbf{f}_{(r)}{}^* := \mathbf{W}_{(r)} \mathbf{b}^*_{(r)}$ is expressed in terms of a linear combination of the columns of $\mathbf{W}_{(r)}$ (*i.e.*, the features) with coefficients $\mathbf{b}$. Taking the derivative with respect to $\mathbf{b}$ and setting this to zero yields

$$\mathbf{b}^*_{(r)} = \left(\mathbf{W}_{(r)}{}^{\top} \mathbf{X} \mathbf{X}^{\top} \mathbf{W}_{(r)} \right)^{-1} \mathbf{W}_{(r)}{}^{\top} \mathbf{X} \mathbf{y},$$

and thus

$$\mathbf{f}_{(r)}{}^* \quad = \quad \mathbf{W}_{(r)} \left(\mathbf{W}_{(r)}{}^{\top} \mathbf{X} \mathbf{X}^{\top} \mathbf{W}_{(r)} \right)^{-1} \mathbf{W}_{(r)}{}^{\top} \mathbf{X} \mathbf{y}. \tag{3.2}$$

It helps in interpreting the features if the columns of $\mathbf{W}_{(r)}$ are orthogonal or conjugate, but this is certainly not required. In particular, all that matters is the *subspace* spanned by the columns of $\mathbf{W}_{(r)}$, in that we can replace $\mathbf{W}_{(r)}$ by $\mathbf{U}_{(r)} := \mathbf{W}_{(r)} \mathbf{C}$ for any invertible matrix $\mathbf{C}$ and still get the same $\mathbf{f}_{(r)}{}^*$.

We now describe the multivariate case. In the absence of any restriction of the input to a subspace, we would solve

$$\mathbf{F}^* := \arg\min_{\mathbf{F}} \|\mathbf{Y} - \mathbf{F}\mathbf{X}\|_F^2 \tag{3.3}$$

$$= \sum_{k=1}^{l_y} \left\|\mathbf{y}_k^\top - \mathbf{f}_k^\top \mathbf{X}\right\|^2 \tag{3.4}$$

where $\mathbf{y}_k^\top$ and $\mathbf{f}_k^\top$ denote the kth *row* of $\mathbf{Y}$ and $\mathbf{F}$ respectively, and $\mathbf{F}$ is $l_y \times l_x$. An important point to note in (3.4) is that each coordinate in the output $\mathbf{y}$ is predicted entirely using a particular row in $\mathbf{F}$. A least squares solution is thus

$$(\mathbf{F}^*)^\top := \left(\mathbf{X}\mathbf{X}^\top\right)^\dagger \mathbf{X}\mathbf{Y}^\top,$$

which is the concatenation of the separate least squares solutions for the l_y individual rows of $\mathbf{Y}$. We can again project the input $\mathbf{X}$ onto $\mathbf{W}_{(r)}$; the least squares solution is then

$$\left(\mathbf{F}_{(r)}{}^*\right)^\top := \mathbf{W}_{(r)} \left(\mathbf{W}_{(r)}{}^\top \mathbf{X}\mathbf{X}^\top \mathbf{W}_{(r)}\right)^{-1} \mathbf{W}_{(r)}{}^\top \mathbf{X}\mathbf{Y}^\top, \tag{3.5}$$

which has rank at most r.

3.2 Principal component regression

The simplest method of selecting features in the input space is principal component regression. We write the singular value decomposition of $\mathbf{X}$ as

$$\mathbf{X} = \begin{bmatrix} \mathbf{W}_{(r)} & \mathbf{W}_{(r)\perp} \end{bmatrix} \begin{bmatrix} \mathbf{S}_{(r)} & \mathbf{0} \\ \mathbf{0} & \mathbf{S}_{(r)\perp} \end{bmatrix} \begin{bmatrix} \mathbf{U}_{(r)}{}^\top \\ \mathbf{U}_{(r)\perp}{}^\top \end{bmatrix},$$

where the r largest singular values are in $\mathbf{S}_{(r)}$, and the $l_x - r$ smallest in $\mathbf{S}_{(r)\perp}$. We can approximate $\mathbf{X}$ using only

$$\widehat{\mathbf{X}} := \mathbf{W}_{(r)} \mathbf{S}_{(r)} \mathbf{U}_{(r)}{}^\top,$$

where we retain the directions of $\mathbf{X}$ with highest variance (which are assumed to have greatest significance in predicting $\mathbf{y}$). The features $\mathbf{W}_{(r)}$ are then used in (3.5).

3.3 PLS in one and more dimensions

The method of partial least squares [10] is widely applied in chemometrics, where it is known to perform particularly well in cases where the input data are highly colinear. Our discussion in this section largely follows [11, 3]. A modification in which PLS is performed in the dual was proposed in [12], and the algorithm was kernelised in [13, 14] for use when $\mathcal{F}_x$ is a reproducing kernel Hilbert space. To

our knowledge the present paper is the first that deals with kernelised multiple outputs.

We again start with the one-dimensional case, since it forms the basis of the multidimensional algorithm. We give two descriptions of the feature matrix $\mathbf{W}_{(i)} := \left[\begin{array}{ccc} \mathbf{w}_1 & \ldots & \mathbf{w}_i \end{array} \right]$ obtained: the first, which is by far the best known, uses orthogonal features, whereas in the second the features are conjugate with respect to $\mathbf{X}\mathbf{X}^\top$. The subspaces spanned by the features are in both cases identical, however. We initialise with $\mathbf{X}_0 := \mathbf{X}$, and begin at $i = 1$. We then iterate the following steps over i.

$$\begin{aligned} \mathbf{w}_i &= \mathbf{X}_{i-1}\mathbf{y} && (3.6) \\ \mathbf{t}_i &= \mathbf{X}_{i-1}^\top \mathbf{w}_i && (3.7) \\ \mathbf{X}_i &= \mathbf{X}_{i-1}\left(\mathbf{I} - \frac{\mathbf{t}_i\mathbf{t}_i^\top}{\mathbf{t}_i^\top\mathbf{t}_i}\right) && (3.8) \end{aligned}$$

A useful result is that the $\mathbf{t}_i$ are mutually orthogonal.[5] In addition (3.8) can be rewritten more compactly as

$$\mathbf{X}_i = \left[\mathbf{I} - \mathbf{T}_{(i)}\left(\mathbf{T}_{(i)}^\top\mathbf{T}_{(i)}\right)\mathbf{T}_{(i)}^\top\right]\mathbf{X}. \qquad (3.9)$$

It is proved in [11] that following r iterations, PLS predicts the output at new observations using (3.2).

A motivating heuristic often employed to justify PLS[6] is that the *covariance* between input and output is used to weight each component of $\mathbf{x}$ when predicting y. This is certainly true of the choice of the first feature $\mathbf{w}_1 = \mathbf{X}^\top y$, which is equal to this covariance. This explanation is less useful in subsequent iterations, however, since the features $\mathbf{w}_i$ are *not* chosen by projecting out $\mathbf{W}_{i-1}$ from $\mathbf{X}$, and computing the covariance[7] using this deflated $\mathbf{X}$ (that said, it is easy to prove[8] $\mathbf{X}_i$ is orthogonal to $\mathbf{w}_j$ for all $j \leq i$, and that the $\mathbf{w}_i$ are mutually orthogonal).

---

[5] *Proof:* After $j \geq 1$ steps, $\mathbf{X}_{(i+j-1)} = \mathbf{X}_{(i-1)}\prod_{l=i}^{l=i+j-1}\left(\mathbf{I} - \frac{\mathbf{t}_{(l)}\mathbf{t}_{(l)}^\top}{\mathbf{t}_{(l)}^\top\mathbf{t}_{(l)}}\right)$. Thus the rows of $\mathbf{X}_{(i+j-1)}$ are orthogonal to each of $\{\mathbf{t}_{(i)}, \ldots, \mathbf{t}_{(i+j-1)}\}$ (a special case being $j = 1$, and $\mathbf{X}_{(i)}$ having rows orthogonal to $\mathbf{t}_{(i)}$). Then since $\mathbf{t}_{(i+j)}$ is a linear combination of the rows of $\mathbf{X}_{(i+j-1)}$ (from (3.7)), it follows that $\mathbf{t}_{(i)}$ is orthogonal to $\mathbf{t}_{(i+j)}$ for all $j \geq 1$.

[6] Note that PLS was originally derived in terms of a factor model, in which an underlying variable $\mathbf{t}$ is assumed to generate $\mathbf{x}$ and y. This motivation justifies the deflation procedure, but gives an incomplete view of the predictive performance.

[7] If we did project out the previous $\mathbf{w}$, our update (3.8) would become $\mathbf{X}_{(i)} = \left(\mathbf{I} - \frac{\mathbf{w}_{(i)}\mathbf{w}_{(i)}^\top}{\mathbf{w}_{(i)}^\top\mathbf{w}_{(i)}}\right)\mathbf{X}_{(i-1)} = \mathbf{X}_{(i-1)}\left(\mathbf{I} - \frac{\mathbf{y}_{(i)}\mathbf{t}_{(i)}^\top}{\mathbf{y}_{(i)}^\top\mathbf{t}_{(i)}}\right)$; thus the intuition that the features maximise the covariance is only true to the extent that $\mathbf{t}$ approximates $\mathbf{y}$. The SIMPLS algorithm [15] in fact updates $\mathbf{X}$ by projecting out $\mathbf{w}$, although this algorithm also uses a deflation on $\mathbf{y}$ which causes it to return the same features $\mathbf{w}$ as NIPALS.

[8] *Proof:* After one step, $\mathbf{X}_{(i)}^\top\mathbf{w}_{(i)} = \left(\mathbf{I} - \frac{\mathbf{t}_{(i)}\mathbf{t}_{(i)}^\top}{\mathbf{t}_{(i)}^\top\mathbf{t}_{(i)}}\right)\mathbf{X}_{(i-1)}^\top\mathbf{w}_{(i)} = \left(\mathbf{I} - \frac{\mathbf{t}_{(i)}\mathbf{t}_{(i)}^\top}{\mathbf{t}_{(i)}^\top\mathbf{t}_{(i)}}\right)\mathbf{t}_{(i)} = 0$. It follows from (3.6) that $\mathbf{w}_{(i+j)}$ is orthogonal to $\mathbf{w}_{(i)}$.

In the univariate case, there is a more satisfying explanation for PLS peformance than the above heuristic, which is that PLS yields an indentical solution to conjugate gradient (CG) descent [16] on $\left\|\mathbf{y}-\mathbf{X}^\top\mathbf{f}\right\|^2$, with respect to the mapping $\mathbf{f}$. The CG method builds a feature matrix $\mathbf{D}_{(i)}$ by setting each $\mathbf{d}_{i+1}$ as close as possible to the direction of steepest descent at the present solution $\mathbf{f}_{(i)} = \mathbf{D}_{(i)}\left(\mathbf{D}_{(i)}^\top\mathbf{X}\mathbf{X}^\top\mathbf{D}_{(i)}\right)^{-1}\mathbf{D}_{(i)}^\top\mathbf{X}\mathbf{y}$, subject to being conjugate to all the columns in $\mathbf{D}_{(i)}$ with respect to $\mathbf{X}\mathbf{X}^\top$. The solution returned after r steps is then obtained by setting $\mathbf{W}_{(r)} = \mathbf{D}_{(r)}$ in (3.2). The choice of conjugate features $\mathbf{D}_{(r)}$ helps to explain the good performance of PLS when the inputs are highly colinear,[9] and these features serve a clear purpose in minimising the loss (being approximately aligned with the directions of steepest descent). A property of the CG method is that the columns of $\mathbf{W}_{(r)}$ are equal to the terms of the rth order Krylov sequence $\mathcal{K}_{(r)}(\mathbf{X}^\top\mathbf{X}, \mathbf{X}^\top\mathbf{y})$, meaning

$$\mathbf{W}_{(r)} := \left[\begin{array}{cccc} \left(\mathbf{X}^\top\mathbf{y}\right) & \left(\mathbf{X}^\top\mathbf{X}\right)\left(\mathbf{X}^\top\mathbf{y}\right) & \ldots & \left(\mathbf{X}^\top\mathbf{X}\right)^{r-1}\left(\mathbf{X}^\top\mathbf{y}\right) \end{array}\right].$$

The simplest method for generalising to the multivariate case is to perform univariate regression on each output variable: indeed, this was found to outperform the multivariate NIPALS algorithm in [3]. This approach is obviously not feasible when l_y is very large; thus regression to a lower dimensional subspace in $\mathcal{F}_\mathcal{Y}$ is required. There are several multivariate PLS methods that accomplish this task. We describe NIPALS, which is the most widely used. Features are generated as follows:

$$\mathbf{w}_i = \arg\max_{\|\mathbf{w}\|\leq 1} \mathbf{w}^\top\mathbf{X}_{i-1}\mathbf{Y}^\top\mathbf{Y}\mathbf{X}_{i-1}^\top\mathbf{w} \tag{3.10}$$

$$\mathbf{t}_i = \mathbf{X}_{i-1}^\top\mathbf{w}_i \tag{3.11}$$

$$\mathbf{X}_i = \mathbf{X}_{i-1}\left(\mathbf{I}-\frac{\mathbf{t}_i\mathbf{t}_i^\top}{\mathbf{t}_i^\top\mathbf{t}_i}\right) \tag{3.12}$$

We note that $\mathbf{w}_i$ is equal to the left singular vector of the empirical covariance $\mathbf{X}_{i-1}\mathbf{Y}^\top$, rather than using (3.6). Thus, writing as $\mathbf{l}_i$ the corresponding right singular vector, the projections of $\mathbf{X}_i$ and $\mathbf{Y}$ onto $\mathbf{w}_i$ and $\mathbf{l}_i$ respectively have the largest covariance of any possible such projections. This conforms to a common intuition behind PLS in a single dimension, namely that features are chosen so as to maximise covariance between input and output. The deflation procedure again belies this interpretation, however, in that we do not project the features $\mathbf{w}$ out of $\mathbf{X}$. We again set $\mathbf{W}_{(r)} = \mathbf{W}_r$ after r iterations, and use (3.5) in predicting $\mathbf{y}^*$ for a test point $\mathbf{x}^*$ [3, Theorem 3.2]. The $\mathbf{w}_i$ remain mutually orthogonal (as do the $\mathbf{t}_i$), which is shown using the same method as in the univariate proofs. To our knowledge, however, there is no established link between multivariate PLS and conjugate gradient descent.

[9] A conceptually similar method to improve performance on colinear data is to use orthogonal features following a pre-whitening step. This is not always numerically stable, however.

Finally, we remark that the maximum singular value of the covariance between input and output has a useful interpretation when $\mathcal{F}_x$ and $\mathcal{F}_y$ are RKHSs with universal kernels (that is, the RKHSs must be dense in the space of continuous functions on X and Y respectively, which is true for instance of those induced by the Gaussian kernel; see [17]). This property is described in the following theorem.

Theorem 1 (The maximum singular value is zero at independence). *Defining the unit balls* $\widetilde{\mathcal{F}} : \{f \in \mathcal{F} : \|f\| \leq 1\}$ *and* $\widetilde{\mathcal{G}} : \{g \in \mathcal{G} : \|g\| \leq 1\}$, *where the reproducing kernel Hilbert spaces* $\mathcal{F}$ *and* $\mathcal{G}$ *have universal kernels, then*

$$\sup_{f \in \widetilde{\mathcal{F}}, g \in \widetilde{\mathcal{G}}} \operatorname{cov}(f(\mathsf{x})g(\mathsf{y})) = 0 \tag{3.13}$$

if and only if the random variables x, y *are independent.*

This theorem is proved for the case of the Gaussian kernel in [18], and for the general case in [19]. This result implies that PLS chooses its features so as to maximise a general measure of statistical dependence, when mapping between universal RKHSs.

3.4 KDE

The kernel dependency estimation method [4] differs from the previous algorithms in that the features are defined in the *output* space. These features, which we denote as the r columns of a matrix $\mathbf{V}_{(r)}$, correspond to the left singular vectors of $\mathbf{Y}$ with r largest singular values; it is thus assumed that the directions of $\mathbf{Y}$ with largest variance will most easily be predicted from $\mathbf{X}$. This projection reduces our problem to an r dimensional multivariate setting, where we wish to predict $\mathbf{C} := {\mathbf{V}_{(r)}}^\top \mathbf{Y}$; the output $\mathbf{y}^*$ at some new $\mathbf{x}^*$ is found by first computing $\mathbf{c}^*$, and then setting $\hat{\mathbf{y}}^* = \mathbf{V}_{(r)}\mathbf{c}^*$. Since $r \ll l_y$, the prediction of $\mathbf{c}$ may accomplished via univariate ridge regression on each of its r dimensions. The ith output projection c_i^* is thus found using $c_i^* = \mathbf{x}^\top \mathbf{b}_i^*$, where $\mathbf{b}_i^* := \arg\min_{\mathbf{b}} \left\|\mathbf{c}_i - \mathbf{X}^\top \mathbf{b}_i\right\|^2 + \Omega_d(\mathbf{b}_i)$, and Ω_d is a regularizer. The mapping from $\mathcal{F}_x$ to $\mathcal{F}_y$ is then ${\mathbf{F}_{(r)}}^* := \mathbf{V}_{(r)} \left(\mathbf{B}_{(r)}^*\right)^\top$, where $\mathbf{B}_{(r)}^* := \left[\begin{array}{ccc} \mathbf{b}_1^* & \ldots & \mathbf{b}_r^* \end{array}\right]$.

4 Multivariate Regression via Stiefel Constraints

We return now to the original multivariate regression setting in Section 2, and present a direct solution of the optimisation problem defined in (2.1) and (2.2).

We begin by noting that the alternative statement of the rank constraint (2.2), which consists in writing the mapping $\mathbf{F}_{(r)}$ in the form (2.5), still leaves us with a non-trivial optimisation problem (2.9). To see this, let us consider an iterative approach to obtain an approximate solution to (2.9), by constructing a sequence of predictors $\mathbf{F}_{(r)_1}, \ldots, \mathbf{F}_{(r)_i}$ such that

$$\mathsf{L}(\mathbf{X}, \mathbf{Y}, \mathbf{F}_{(r)_i}) \quad \geq \quad \mathsf{L}(\mathbf{X}, \mathbf{Y}, \mathbf{F}_{(r)_{i+1}}). \tag{4.1}$$

We might think to obtain this sequence by updating $\mathbf{V}_{i+1}, S_{i+1}$ and $\mathbf{W}_{i+1}$ according to their *free* matrix gradients $\frac{\partial \mathsf{L}}{\partial \mathbf{V}}|_{\theta_i}$, $\frac{\partial \mathsf{L}}{\partial \mathsf{S}}|_{\theta_i}$, and $\frac{\partial \mathsf{L}}{\partial \mathbf{W}}|_{\theta_i}$ respectively, where θ_i denotes the solution $(\mathbf{V}_i, \mathbf{W}_i, S_i)$ at the ith iteration (*i.e.*, the point at which the gradients are evaluated). This is unsatisfactory, however, in that updating $\mathbf{V}$ and $\mathbf{W}$ linearly along their free gradients does *not* result in matrices with orthogonal columns.

Thus, to define a sequence along the lines of (4.1), we must first show how to optimise over $\mathbf{V}$ and $\mathbf{W}$ in such a way as to retain orthogonal columns. As we saw in Section 2, the feature matrices are elements on the Stiefel manifold; thus any optimisation procedure must take into account the geometrical structure of this manifold. The resulting optimisation problem is *non-convex*, since the Stiefel manifold $\mathbb{S}(n, r)$ is not a convex set.

In the next section, we describe how to update $\mathbf{V}$ and $\mathbf{W}$ as we move along geodesics on the Stiefel manifold $\mathbb{S}(n, r)$; in the two sections that follow, we use these updates to conduct the minimisation of the L_2 and L_1 losses respectively in the absence of regularization. We introduce regularisation for the batch case in Section 4.4, and describe the online algorithm in Section 4.5.

4.1 Dynamics on stiefel manifolds.

We begin with a description of the geodesics for the simpler case of $\mathbb{S}(n, n)$, followed by a generalisation to $\mathbb{S}(n, r)$ when $n > r$. Let $O(n)$ denote the group of orthogonal matrices. Suppose we are given a matrix $\mathbf{V}(t) \in O(n)$ that depends on a parameter t, where $\mathbf{V}(t)$ describes a geodesic on the manifold $O(n)$. Our goal in this subsection is to describe how $\mathbf{V}(t)$ changes as we move along the geodesic. Since $O(n)$ is not only a manifold but also a *Lie group* (a special manifold whose elements form a group), there is an elegant way of moving along geodesics which involves an exponential map. We will give an informal but intuitive derivation of this map; for a formal treatment, see [7, 20]. We begin by describing a useful property of the derivative of $\mathbf{V}(t)$;

$$\begin{aligned}
\mathbf{I} &= \mathbf{V}(t)^\top \mathbf{V}(t), \\
\mathbf{0} &= \frac{d}{dt}(\mathbf{V}(t)^\top \mathbf{V}(t)), \\
\mathbf{0} &= (\frac{d}{dt}\mathbf{V}(t))^\top \mathbf{V}(t) + \mathbf{V}(t)^\top (\frac{d}{dt}\mathbf{V}(t)), \\
\mathbf{0} &= \mathbf{Z}(t)^\top + \mathbf{Z}(t),
\end{aligned}$$

with

$$\mathbf{Z}(t) := \mathbf{V}(t)^\top (\frac{d}{dt}\mathbf{V}(t)). \tag{4.2}$$

The matrix $\mathbf{Z}(t)$ is skew symmetric, which we write as $\mathbf{Z}(t) \in \mathfrak{s}(n, n)$, where $\mathfrak{s}$ consists of the set of all skew symmetric matrices of size $n \times n$.

We next consider curves corresponding to 1-parameter subgroups of $O(n)$; in other words, curves satisfying $\mathbf{V}(0) = \mathbf{I}$ and $\mathbf{V}(t+s) = \mathbf{V}(t)\mathbf{V}(s)$ (in particular,

$\mathbf{V}(t)^{-1} = \mathbf{V}(-t)$) for all s, t. For our group $O(n)$, we can obtain such a subgroup by fixing an n-dimensional axis and considering all matrices describing rotations around that axis. In this case, the parameters t, s can be thought of as rotation angles. Returning to (4.2) with $\mathbf{V}(t)$ in this 1-parameter subgroup, we have

$$\begin{aligned}
\mathbf{Z}(t) &= \mathbf{V}(t)^\top \left(\frac{\mathbf{V}(t+dt) - \mathbf{V}(t)}{dt} \right) \\
&= \frac{\mathbf{V}(t)^\top \mathbf{V}(t+dt) - \mathbf{I}}{dt} = \frac{\mathbf{V}(-t)\mathbf{V}(t+dt) - \mathbf{I}}{dt} \\
&= \frac{\mathbf{V}(dt) - \mathbf{V}(0)}{dt} = \frac{d}{dt}\Big|_0 \mathbf{V}(t) = \mathbf{Z}(0),
\end{aligned}$$

which means $\mathbf{Z}(t)$ is constant. Multiplying (4.2) with $\mathbf{V}(t)$ from the left yields an ordinary differential equation of the form

$$\begin{aligned}
\frac{d}{dt}\mathbf{V}(t) &= \mathbf{V}(t)\mathbf{Z} \qquad (4.3) \\
\text{with } \mathbf{V}(0) &= \mathbf{V},
\end{aligned}$$

which has solution

$$\mathbf{V}(t) = \mathbf{V}(0)e^{t\mathbf{Z}}, \qquad (4.4)$$

where $e^{\mathbf{Z}}$ denotes the matrix exponential [21].[10] We can see from (4.3) that the skew-symmetric matrix $\mathbf{Z}$ specifies a tangent at the point $\mathbf{V}(0)$ on the Stiefel manifold $\mathbb{S}(n, n)$.

We now generalize to the case where $\mathbf{V}$ does not have full rank, *i.e.* $\mathbf{V} \in \mathbb{S}(n, r)$ and $r < n$. We can embed $\mathbb{S}(n, r)$ into $O(n)$ by extending any $\mathbf{V} \in \mathbb{S}(n, r)$ with an $n - r$ matrix $\mathbf{V}_\perp \in \mathbb{S}(n, n-r)$ such that $\mathbb{R}^n = \mathbf{V}_\perp \bigoplus \mathbf{V}$. Therefore $\mathbf{V}_\perp$ spans the orthogonal complement to the space spanned by the columns of $\mathbf{V}$. Two orthogonal matrices $\mathbf{A}, \mathbf{B}$ in $O(n)$ are considered to be the same from the viewpoint of $\mathbb{S}(n, r)$ if they relate as

$$\mathbf{B} = [\mathbf{I}_{n,r}, \mathbf{P}]\,\mathbf{A} \qquad (4.5)$$

for any matrix $\mathbf{P} \in \mathbb{S}(n, n-r)$. Therefore (4.3) can be written as

$$\frac{d}{dt}\left[\mathbf{V}(t), \mathbf{V}_\perp(t)\right]\Big|_{t=0} = \left[\mathbf{V}(0), \mathbf{V}_\perp(0)\right]\hat{\mathbf{Z}}, \qquad (4.6)$$

and

$$\mathbf{V}(t) = \left[\mathbf{V}(0), \mathbf{V}_\perp(0)\right] e^{t\hat{\mathbf{Z}}} \left[\mathbf{I}_{n,r}, \mathbf{0}\right]. \qquad (4.7)$$

To conduct gradient descent, we need to find the tangent direction $\hat{\mathbf{G}}$ (for use in (4.4)) which is as close as possible to the free gradient $\mathbf{G}$, since $\mathbf{G}$ does

[10] When verifying that (4.4) is indeed a solution for (4.3) note that the skew-symmetric matrix $\mathbf{Z}$ is *normal*, *i.e.* $\mathbf{Z}\mathbf{Z}^\top = \mathbf{Z}^\top\mathbf{Z}$.

not in general have the factorization (4.3). The constrained gradient $\hat{\mathbf{G}}$ can be calculated directly by projecting the free gradient onto the tangent space of $\mathbf{V}$; see [7]. Intuitively speaking, we do this by removing the symmetric part of $\mathbf{G}^\top\mathbf{V}$, leaving the skew symmetric remainder.[11] The constrained gradient is thus

$$\hat{\mathbf{G}} \quad = \quad \mathbf{G} - \mathbf{V}\mathbf{G}^\top\mathbf{V}. \tag{4.8}$$

Finally, the skew symmetric matrix $\hat{\mathbf{Z}} \in \mathbb{R}^{n\times n}$ is given by (4.2) as

$$\hat{\mathbf{Z}} \quad = \quad \begin{pmatrix} \hat{\mathbf{G}}^\top\mathbf{V} & -(\hat{\mathbf{G}}^\top\mathbf{V}_\perp)^\top \\ \hat{\mathbf{G}}^\top\mathbf{V}_\perp & \mathbf{0} \end{pmatrix}. \tag{4.9}$$

We can now describe the nonlinear update operator π_{Stiefel}.

Algorithm 1 ($\pi_{\mathbf{Stiefel}}(\mathbf{V}, \mathbf{G}, t)$). *Given a free gradient $\mathbf{G} \in \mathbb{R}^{n,r}$, an orthogonal matrix $\mathbf{V} \in \mathbb{S}(n,r)$ and a scalar step parameter γ, the update of $\mathbf{V}$ specified by $\mathbf{G}$ and γ can be calculated as follows:*

1) Calculate constrained gradient in (4.8).

2) Calculate orthogonal basis $\mathbf{V}_\perp$ for the orthogonal complement of $\mathbf{V}$.

3) Calculate the tangent coordinates $\mathbf{Z}$ in (4.9),

4) $\mathbf{V}(\gamma) = [\mathbf{V}, \mathbf{V}_\perp]e^{\gamma\mathbf{Z}}[\mathbf{I}_{n,r}, \mathbf{0}]$.

4.2 Multivariate regression with L_2 loss

Now that we have defined π_{Stiefel}, we can apply a gradient descent approach to (2.9). We first calculate the free gradients,

$$\frac{\partial \mathsf{L}_2}{\partial \mathbf{V}}|_{\theta_i} \quad = \quad -\mathbf{Y}\mathbf{X}^\top\mathbf{W}_i S_i, \tag{4.10}$$

$$\frac{\partial \mathsf{L}_2}{\partial \mathbf{W}}|_{\theta_i} \quad = \quad -\mathbf{X}\mathbf{Y}^\top\mathbf{V}_i S_i + \mathbf{X}\mathbf{X}^\top\mathbf{W}_i S_i^2, \tag{4.11}$$

$$\frac{\partial \mathsf{L}_2}{\partial S}|_{\theta_i} \quad = \quad -\mathbf{I}_{r,r} \odot \mathbf{W}_i^\top\mathbf{X}\mathbf{Y}^\top\mathbf{V}_i + \mathbf{I}_{r,r} \odot \mathbf{W}_i^\top\mathbf{X}\mathbf{X}^\top\mathbf{W}_i S_i, \tag{4.12}$$

where $\odot$ denotes the Hadamard (element-wise) product. The multivariate regression algorithm for the L_2 loss is then:

Algorithm 2. *MRS for L_2 loss function.*

Initialization

$\mathbf{V}_0 = \mathbf{I}_{\dim \mathcal{F}_x, r} \quad S_0 = \mathbf{I}_{r,r} \quad \mathbf{W}_0 = \mathbf{I}_{\dim \mathcal{F}_y, r}$

$\theta_0 = (\mathbf{V}_0, S_0, \mathbf{W}_0) \quad \mathbf{F}_{(r)_0} = \mathbf{W}_0 S_0 \mathbf{V}_0 \quad i = 0$

Repeat until convergence:

[11] Any square matrix can be expressed as a unique sum of a symmetric and a skew-symmetric matrix.

1) Calculate free gradients, equations (4.10)-(4.12)

2) $t^*_{\mathbf{V}}, t^*_S, t^*_{\mathbf{W}} = \underset{t_{\mathbf{V}}, t_S, t_W}{\arg\min} \mathsf{L}_2(\mathbf{V}(t_{\mathbf{V}}), S(t), \mathbf{W}(t_{\mathbf{W}}))$

with $\mathbf{W}(t_{\mathbf{W}}) = \pi_{stiefel}(\mathbf{W}_i, \frac{\partial \mathcal{L}}{\partial \mathbf{W}}|_{\theta_i}, t_{\mathbf{W}})$,

$\mathbf{V}(t_{\mathbf{V}}) = \pi_{stiefel}(\mathbf{V}_i, \frac{\partial \mathcal{L}}{\partial \mathbf{V}}|_{\theta_i}, t_{\mathbf{V}})$, *and*

$S(t) = S_i + t_S \frac{\partial \mathcal{L}}{\partial S}|_{\theta_i}$

3) $\mathbf{V}_{i+1} = \mathbf{V}(t^*_{\mathbf{V}}), \quad \mathbf{W}_{i+1} = \mathbf{W}(t^*_{\mathbf{W}}), \quad S_{i+1} = S(t^*_S)$

4) $\mathbf{F}_{(r)_{i+1}} = \mathbf{V}_{i+1} S_{i+1} \mathbf{W}_{i+1}$

5) $\theta_{i+1} = (\mathbf{V}_{i+1}, S_{i+1}, \mathbf{W}_{i+1})$

6) $i = i + 1$

After convergence : $\mathbf{F}_{(r)} = \mathbf{F}_{(r)_i}$

4.3 Multivariate regression with L_1 loss

The L_1 loss,[12] given in (2.4), has the advantage that it does not weight outliers as strongly as the L_2 loss. Therefore it is significantly more robust in applications requiring resistance to outliers, compared with L_2 based methods. This loss can be written

$$\mathsf{L}_1(\mathbf{Y}, \mathbf{X}, \mathbf{F}_{(r)}) = \mathbf{1}_m^\top (\mathcal{E} \odot \sigma(\mathcal{E})) \mathbf{1}_m \tag{4.13}$$

with

$$\mathcal{E} := \mathbf{Y} - \mathbf{F}_{(r)}\mathbf{X}, \text{ and } \sigma(x) := \begin{cases} 1 & x > 0 \\ 0 & x = 0 \\ -1 & x < 0 \end{cases} \tag{4.14}$$

where the signum function σ is applied to each matrix entry. Note that the L_1 loss is not differentiable. To calculate the gradient, an approach taken in the neural network literature is to replace the σ function by a differentiable approximation, which we denote as $\hat{\sigma}$. A possible $\hat{\sigma}$ function and its derivative are

$$\hat{\sigma}(x) := \tanh(\alpha x) \quad \text{and} \quad \hat{\sigma}'(x) := \alpha \operatorname{sech}^2(\alpha x) \tag{4.15}$$

respectively, where $\alpha > 0$ is a scaling parameter which controls the slope of the $\hat{\sigma}$ function. We can thus approximate the $\mathsf{L}_1(\mathbf{Y}, \mathbf{X}, \mathbf{F}_{(r)})$ function as

$$\mathsf{L}_1(\mathbf{Y}, \mathbf{X}, \mathbf{F}_{(r)}) \approx \hat{\mathsf{L}}_1(\mathbf{Y}, \mathbf{X}, \mathbf{F}_{(r)}) = \mathbf{1}_m^\top (\mathcal{E} \odot \hat{\sigma}(\mathcal{E})) \mathbf{1}_m. \tag{4.16}$$

The free gradients can be identified as

[12] This loss is also known as least absolute deviation (LAD) loss.

$$\frac{\partial \hat{\mathsf{L}}_1}{\partial \mathbf{V}}|_\theta = \mathbf{Y}\tanh(\alpha\mathcal{E})^\top + \alpha\mathbf{Y}\left(\mathcal{E}\odot\operatorname{sech}^2(\alpha\mathcal{E})\right)^\top, \tag{4.17}$$

$$\frac{\partial \hat{\mathsf{L}}_1}{\partial S}|_\theta = -\mathbf{I}_{r,r}\odot\left(\tanh(\alpha\mathcal{E})\mathbf{X}^\top\mathbf{W}_0 + \alpha\mathcal{E}\odot\operatorname{sech}^2(\alpha\mathcal{E})\mathbf{X}^\top\mathbf{W}_0\right), \tag{4.18}$$

$$\frac{\partial \hat{\mathsf{L}}_1}{\partial \mathbf{W}}|_\theta = -\mathbf{X}\tanh(\alpha\mathcal{E})^\top S_0 - \alpha\mathbf{X}(\mathcal{E}\odot\operatorname{sech}^2(\alpha\mathcal{E}))^\top S_0. \tag{4.19}$$

We can use a monotone increasing sequence $(\alpha_k)_{k=1\ldots z}$ for α to make $\hat{\sigma}$ approach σ. The multivariate regression algorithm for the $\hat{\mathsf{L}}_1$ loss is then:

Algorithm 3. *Stiefel regression for* $\hat{\mathsf{L}}_1$

Initialization

$\mathbf{V}_0 = \mathbf{I}_{\dim\mathcal{F}_x,r}\quad S_0 = \mathbf{I}_{r,r}\quad \mathbf{W}_0 = \mathbf{I}_{\dim\mathcal{F}_y,r}$

$\theta_0 = (\mathbf{V}_0, S_0, \mathbf{W}_0)\quad \mathbf{F}_{(r)_0} = \mathbf{W}_0 S_0 \mathbf{V}_0\quad i = 0$

For $\alpha \in (\alpha_1, \ldots, \alpha_z)$ *repeat until convergence:*

1) Calculate free gradients, equations (4.17)-(4.19)

2) $t^*_{\mathbf{V}}, t^*_S, t^*_{\mathbf{W}} = \underset{t_{\mathbf{V}}, t_S, t_W}{\arg\min}\,\hat{\mathsf{L}}_1(\mathbf{V}(t_{\mathbf{V}}), S(t), \mathbf{W}(t_{\mathbf{W}}))$

with $\mathbf{W}(t_{\mathbf{W}}) = \pi_{stiefel}(\mathbf{W}_i, \frac{\partial\mathcal{L}}{\partial\mathbf{W}}|_{\theta_i}, t_{\mathbf{W}})$,

$\mathbf{V}(t_{\mathbf{V}}) = \pi_{stiefel}(\mathbf{V}_i, \frac{\partial\mathcal{L}}{\partial\mathbf{V}}|_{\theta_i}, t_{\mathbf{V}})$, *and*

$S(t) = S_i + t_S\frac{\partial\mathcal{L}}{\partial S}|_{\theta_i}$

3) $\mathbf{V}_{i+1} = \mathbf{V}(t^*_{\mathbf{V}}),\quad \mathbf{W}_{i+1} = \mathbf{W}(t^*_{\mathbf{W}}),\quad S_{i+1} = S(t^*_S)$

4) $\mathbf{F}_{(r)_{i+1}} = \mathbf{V}_{i+1}S_{i+1}\mathbf{W}_{i+1}$

5) $\theta_{i+1} = (\mathbf{V}_{i+1}, S_{i+1}, \mathbf{W}_{i+1})$

6) $i = i + 1$

After convergence : $\mathbf{F}_{(r)} = \mathbf{F}_{(r)_i}$

4.4 Regularization

It is easy to incorporate a regularizer Ω_d into the batch algorithm which penalizes the squared norm $||\mathbf{F}_{(r)}||^2$ (in a manner analogous to ridge regression), since $||\mathbf{F}_{(r)}||^2_F = ||S||^2_F$. In the L_2 case, (2.9) becomes

$$\theta = \underset{\mathbf{F}_{(r)}=\mathbf{V}S\mathbf{W}^\top}{\arg\min}\,\|\mathbf{Y} - \mathbf{V}S\mathbf{W}^\top\mathbf{X}\|^2_F + \lambda\|S\|^2_F, \tag{4.20}$$

subject to constraints (2.6-2.8). The only change in Algorithm 2 is to replace the gradient $\frac{\partial\mathsf{L}_2}{\partial S}|_\theta$ by

$$\frac{\partial \hat{\mathsf{L}}_2}{\partial S}|_\theta = \frac{\partial \mathsf{L}_2}{\partial S}_\theta + \lambda S. \tag{4.21}$$

In the same way, it is possible to add a regularizer in the L_1 case.

4.5 Online variant: S-MRS

In this section, we describe the sequential implementation of MRS, known as S-MRS. The main difference, compared with the batch method, is the addition to the loss of a regularizing functional[13] Ω_o, which controls the tradeoff between fit to the newly observed data block $\mathbf{Z}_k = [\mathbf{X}_k, \mathbf{Y}_k]$ (presently measured with the L_2 loss), and distance to the predictor $\mathbf{F}_{(r)}^{(k-1)}$ obtained from previous data $\mathbf{Z}_j$ with $j < k$ (see (2.11) in Section 2.2 for more detail). Thus the regularization functional takes the form

$$\Omega_o\left(\mathbf{V}_{(r)}S_{(r)}\mathbf{W}_{(r)}{}^\top, \mathbf{V}_{(r)}^{(k-1)}S_{(r)}^{(k-1)}\mathbf{W}_{(r)}^{(k-1)\top}\right) \quad = \quad \gamma\left\|\mathbf{V}_{(r)}S_{(r)}\mathbf{W}_{(r)}{}^\top - \mathbf{V}_{(r)}^{(k-1)}S_{(r)}^{(k-1)}\mathbf{W}_{(r)}^{(k-1)\top}\right\|^2.$$

Since Ω_o (and thus $\mathcal{L}$) is differentiable, the free gradients $\frac{\partial\mathcal{L}}{\partial\mathbf{V}}|_{\theta_i}$, $\frac{\partial\mathcal{L}}{\partial\mathbf{W}}|_{\theta_i}$ and $\frac{\partial\mathcal{L}}{\partial S}|_{\theta_i}$ can be calculated analytically.

The regularization parameter γ controls convergence behaviour. For example, we can increase γ for every new block k to progressively decrease the influence of new observations. On the other hand, a fixed γ can be used to implement adaptive behaviour if the distribution generating the observations changes over time.

Given this choice of Ω_o, and starting with $\mathbf{F}_{(r)} = \mathbf{I}$ at $k = 0$, we can adapt Algorithm 2 to solve (2.12) for each $k > 0$ and associated observation sequence $\mathbf{X}_k, \mathbf{Y}_k$:

Algorithm 4. *Sequential Multivariate Regression via Stiefel Constraints (S-MRS)*

Given data $\mathbf{X}_k, \mathbf{Y}_k$, *a learning rate* γ *and a reference mapping* $\mathbf{F}_{(r)}^{(k-1)} = \mathbf{V}_{(r)}^{(k-1)}S_{(r)}^{(k-1)}\mathbf{W}_{(r)}^{(k-1)\top}$

Initialization

$\mathbf{V}_0 = \mathbf{V}_{(r)}^{(k-1)} \quad S_0 = S_{(r)}^{(k-1)} \quad \mathbf{W}_0 = \mathbf{W}_{(r)}^{(k-1)}$

$\theta_0 = (\mathbf{V}_0, S_0, \mathbf{W}_0) \quad \mathbf{F}_0 = \mathbf{W}_0 S_0 \mathbf{V}_0 \quad i = 0$

Repeat until convergence:

1) Calculate free gradients $\frac{\partial\mathcal{L}}{\partial\mathbf{V}}|_{\theta_i}$, $\frac{\partial\mathcal{L}}{\partial\mathbf{W}}|_{\theta_i}$ *and* $\frac{\partial\mathcal{L}}{\partial S}|_{\theta_i}$.

2) $t_\mathbf{V}^*, t_S^*, t_\mathbf{W}^* = \underset{t_\mathbf{V}, t_S, t_W}{\arg\min}\mathcal{L}(\mathbf{Z}_k, \mathbf{V}(t_\mathbf{V})\,S(t_S)\,\mathbf{W}(t_\mathbf{W}), \mathbf{F}_{(r)}^{(k-1)})$

with $\mathbf{W}(t_\mathbf{W}) = \pi_{stiefel}(\mathbf{W}_i, \frac{\partial\mathcal{L}}{\partial\mathbf{W}}|_{\theta_i}, t_\mathbf{W})$,

$\mathbf{V}(t_\mathbf{V}) = \pi_{stiefel}(\mathbf{V}_i, \frac{\partial\mathcal{L}}{\partial\mathbf{V}}|_{\theta_i}, t_\mathbf{V})$, *and*

$S(t) = S_i + t_S\frac{\partial\mathcal{L}}{\partial S}|_{\theta_i}$

3) $\mathbf{V}_{i+1} = \mathbf{V}(t_\mathbf{V}^*), \quad \mathbf{W}_{i+1} = \mathbf{W}(t_\mathbf{W}^*), \quad S_{i+1} = S(t_S^*)$

4) $\mathbf{F}_{i+1} = \mathbf{V}_{i+1}S_{i+1}\mathbf{W}_{i+1}$

5) $\theta_{i+1} = (\mathbf{V}_{i+1}, S_{i+1}, \mathbf{W}_{i+1})$

6) $i = i + 1$

After convergence : $\mathbf{F}_{(r)} = \mathbf{F}_i$, $k = k + 1$

[13] This should not be confused with the regularizer Ω_d introduced in the previous section, which controls $||\mathbf{F}_{(r)}||^2$ for a solution obtained with a particular batch of data.

Algorithm 4 describes a gradient descent procedure on the quantities $\mathbf{W}, \mathbf{V}$ and S. The updates of the orthogonal matrices $\mathbf{W}, \mathbf{V}$ are performed using $\pi_{stiefel}$ defined in Algorithm 1, where $\pi_{stiefel}$ ensures that $\mathbf{W}$ and $\mathbf{V}$ always remain in the Stiefel manifold. Note that it is possibly to similarly modify Algorithm 3, which would yield a sequential algorithm for the L_1 case.

5 Nonlinear multivariate regression

So far, we have only described linear implementations of the MRS algorithm. We now address the *kernelization* of this algorithm, for use when the input and output spaces are *both* reproducing kernel Hilbert spaces of possibly infinite dimension, with respective kernels k and l.

5.1 Nonlinear input features

One convenient way to deal with high dimensional RKHSs is to apply the kernel PCA map [22]. That is, we calculate a *new* data representation $\mathbf{x}_i \in \mathbb{R}^m$ for all $i = 1 \ldots m$, given as

$$\mathbf{x}_i \quad = \quad K_X{}^{\frac{1}{2}} \mathbf{k}_x(x_i), \tag{5.1}$$

where $\mathbf{k}_x(x) \in \mathbb{R}^m$ is a vector containing $k(x, x_i)$ for all training inputs x_i. The linear dot product $\mathbf{x}_i^\top \mathbf{x}_j$ equals the nonlinear dot product $k(x_i, x_j)$ by construction, *i.e.* the Gram matrix $K_{\mathbf{X}} = \mathbf{X}^\top \mathbf{X}$ is equal to the kernel matrix K_X obtained with the kernel k on the training data.

5.2 Nonlinear output features

Nonlinear output features corresponding to the output kernel l can be generated in the same way as the input features; this was proposed in [4]. As in the input case, we use the kernel PCA map to construct explicit representations of the outputs in $\mathbb{R}^N$,

$$\mathbf{y}_i \quad = \quad L_Y{}^{\frac{1}{2}} \mathbf{l}_y(y_i), \tag{5.2}$$

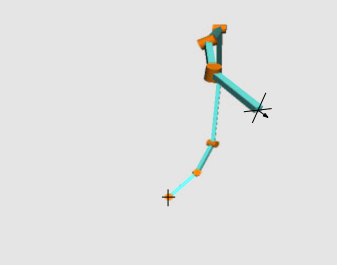
where $\mathbf{l}_y(y) \in \mathbb{R}^m$ is a vector containing $l(y, y_i)$ for each training label y_i. Since we are interested in points in $\mathcal{Y}$ rather than in $\mathcal{F}_y$, we have to reconstruct the pattern $\hat{y}$ from its *estimated* feature representation $\hat{\mathbf{y}} \in \mathcal{F}_y$. The problem of learning a mapping $\Gamma : \mathcal{F}_y \to \mathcal{Y}$ is known as the *pre-image problem*, and possible solutions are described in [23, 24]. An overview over the final regression scheme is given in Figure 5.1.

6 Applications

In this section, we give three examples demonstrating multivariate regression with Stiefel constraints. The first two experiments are on relatively small-scale

$$\begin{array}{ccc} \phi(\mathcal{X}) \subset \mathcal{F}_x & \xrightarrow{\mathbf{F}_{(r)}} & \phi(\mathcal{Y}) \subset \mathcal{F}_y \\ {\scriptstyle\phi}\uparrow & & {\scriptstyle\phi}\uparrow \;\; \downarrow{\scriptstyle\Gamma} \\ \mathcal{X} & & \mathcal{Y} \end{array} \tag{5.3}$$

Figure 5.1: Mappings between original sets $\mathcal{X}, \mathcal{Y}$ and corresponding feature spaces $\mathcal{F}_x$, $\mathcal{F}_y$.



a	α	d	$\underline{\theta}$
100	0	0	0
100	1	0	0
300	$\frac{1}{2}$	0	0
200	0	0	0
200	$\frac{1}{2}$	0	0
200	1	0	0
100	0	0	0

Table 1: Simulated Manipulator and its Denavit-Hartenberg specification.

problems (respectively an inverse kinematics problem for a robot arm and a hand-written digit reconstruction task), so we use MRS initialised by PLS. The final problem is concerned with the reconstruction of face images, and involves substantially larger data sets than the previous tasks: consequently, we apply S-MRS.

6.1 Inverse Kinematics

We begin by demonstrating the usage of MRS on a robot control task, where we want to predict the joint angles given a desired manipulator posture. The kinematic chain of the robot manipulator is given by its Denavit-Hartenberg parameters [25]; see Table 1.

We randomly generated 5000 robot postures, of which we reserevd 500 for training. We defined our input as the complete posture information given by the elbow position $e_i \in \mathbb{R}^3$ and manipulator tip $p_i \in \mathbb{R}^3$, so $\mathbf{x}_i = [e_i, p_i] \in \mathbb{R}^6$. As output, we used the 7 joint values $q_i \in \mathbb{R}^7$. The input space $\mathcal{F}_x$ was an RKHS with an RBF kernel, $k(x_i, x_j) = \exp^{\frac{1}{\gamma}\|x_i - x_j\|^2}$, while no kernel was used on the outputs. To determine the kernel width γ and the number r of features, we used threefold cross validation over the training set. This led to $\gamma = 42$ and $r = 4$ PLS iterations.

We initialised our algorithm using the NIPALS training scheme for PLS (see 3.10), which returned a predictor $\mathbf{F}_{(r)PLS}$. To do this, we decomposed $\mathbf{F}_{(r)PLS}$ into $\mathbf{U}S\mathbf{V}^\top$ via the singular value decomposition, and set $\mathbf{V}_0$, S_0, and $\mathbf{W}_0$ to $\mathbf{U}$, $\mathbf{I}_{4,4}$, and $\mathbf{V}$ respectively. We see in Table 2 that MRS improves on the PLS performance, both in terms of training error and generalization. We also see the

Algorithm	MSE (Training)	MSE (Testing)
PLS(4)	13.57 ± 0.01	58.0 ± 0.1
MRS(4)	8.96 ± 0.01	$\mathbf{53.5 \pm 0.1}$
RR	$\mathbf{8.60 \pm 0.01}$	54.0 ± 0.1

Table 2: Comparison of PLS, MRS, and ridge regression (RR) on the inverse kinematics task. The numbers in parentheses in the first column constitute the rank r of the mapping.

Algorithm	MSE (Training)	MSE (Testing)
$PLS(4)$	6.71 ± 0.01	1200 ± 4
$MRS(4)$ (L_1)	$\mathbf{1.41 \pm 0.01}$	$\mathbf{151 \pm 4}$

Table 3: Robust regression behavior due to L_1 minimisation. Note that the training error is evaluated on the uncorrupted training points.

training error obtained using ridge regression (RR) is slightly better than that of MRS, although the test error of MRS is slightly lower than RR.

As a second example we illustrate the power of using the L_1 loss when outliers are present. We perturb joint value q_1 of a single data entry by setting it to a very large number (1000). As α sequence in Algorithm 3, we chose $[0.25, 0.5, 1, 5, 10, 50, 100, 500, 1000]$. Results are shown in Table 3.

6.2 Image Restoration

We next demonstrate the application of MRS to an artificial image restoration task. The goal is to restore the corrupted part of an image, given examples of corrupted images and the corresponding clean images. The images are taken from the USPS postal database, which consists of 16×16 grayscale patches representing handwritten digits. We independently perturbed the gray values of each pixel in the lower half of each image with Gaussian noise having standard deviation 0.1. Our data consisted of 2000 digits chosen at random, with 1000 reserved for training.

To perform restoration, we first applied kernel PCA to extract 500 nonlinear features from the noisy digits,[14] using a Gaussian kernel of width 10. Thus the restoration task is a regression problem with a 500 dimensional input space $\mathcal{F}_x$, where we predict the *entire* clean digits in a 256 dimensional output space $\mathcal{F}_y$.

In our experiments, we compared ridge regression (RR), PLS, and MRS. We used the ridge parameter $1e-6$, which we optimised using 5-fold cross validation. For our PLS solution, we used a rank 123 mapping, again finding this optimal rank with 5-fold cross validation. We initialised MRS using a low rank approximation to the predictor $\mathbf{F}_{(r)_{RR}}$ found by ridge regression. To do

[14] For more detail on this feature extraction method, see [22].

	RR	RR(110)	PLS(123)	MRS(110)
RMSE	552.5 ± 0.1	554.9 ± 0.1	648.5 ± 0.1	$\mathbf{550.53 \pm 0.1}$

Table 4: Test error (using squared loss) of MRS, PLS, and RR for the digit restoration problem, with results averaged over 1000 digits. The first column gives the performance of RR alone. The second column uses a low rank (*i.e.* rank 110) approximation to the RR solution. The third and fourth columns respectively show the PLS and MRS results with the rank in parentheses, where MRS was initialised using the low rank RR solution.

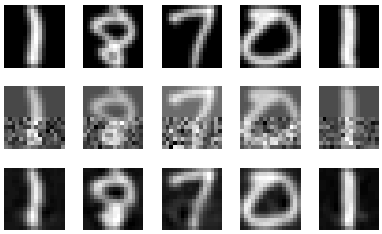


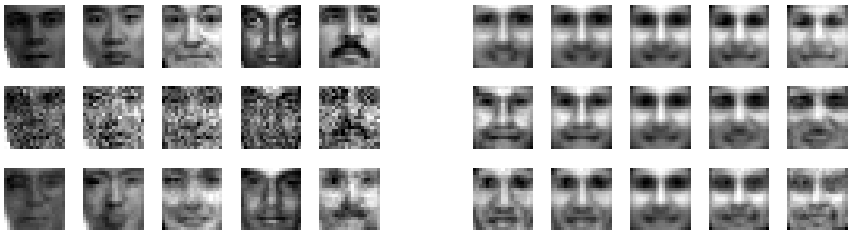
Figure 6.1: Example of image denoising using MRS. Each column contains a hand-written digit chosen at random from the 1000 images in our test set. The first row displays the original images, the second row contains the noisy images, and the third row shows the images as reconstructed using MRS.

this, we decomposed $\mathbf{F}_{(r)_{RR}}$ as $\mathbf{U}S\mathbf{V}^\top$ via the singular value decomposition, and set $\mathbf{U}_0$ and $\mathbf{V}_0$ to be the first 110 components (determined by cross validation on the MRS solution) of $\mathbf{U}$ and $\mathbf{V}$ respectively, while initialising $S_0 = \mathbf{I}$. We give sample results in Figure 6.1.

Table 4 shows that MRS improves on the RR generalization performance. We also give the result obtained by simply using the first 110 components of the SVD of $\mathbf{F}_{(r)_{RR}}$: the performance is worse than both ridge regression and MRS. In this image restoration task, it appears that MRS has a small advantage in eliminating irrelevant features from the subspaces used in prediction, as opposed to ridge regression (which shrinks the weights assigned to *all* the features).

6.3 Face denoising: a large-scale problem

In our final experiment we apply *sequential* MRS (S-MRS) to a larger image denoising task than in the previous section, where we draw our data from the face database in [26]. This database contains a training set of 2429 faces and 4548 non-faces, where every face is a 19 x 19 grayscale image. We perturbed the gray values of each pixel with univariate random noise of magnitude 0.01. The noisy images served as input, and our goal was to reconstruct the original unperturbed image.



a) b)

Figure 6.2: a) Denoising performance on 5 face images of the testing set. The original images are on the first row, the noisy images on the second row, and the predicted noise-free images on the third row. b) Illustration of three output feature vectors $\mathbf{v}_i$ in $\mathbf{V}$, one in each row (in a nutshell, the feature vectors span the subspace of the output relevant to the regression process). The significance of these features is shown by perturbing a particular image in the direction parameterised by each of the vectors. The middle column corresponds to the mean image over all faces in the training set, while to the left and right we add $\alpha\,\mathbf{v}_i$ for a small α. The columns correspond respectively to $\alpha \in [-0.2, -0.1, 0, 0.1, 0.2]$.

To perform denoising, we first applied kernel PCA to extract 500 nonlinear features from 2400 randomly sampled images, using a Gaussian kernel of width 5. Each input to S-MRS was then given by projecting an image onto these features, yielding a 500 dimensional input space $\mathcal{F}_x$. We set the rank r of the mapping to 50, and trained the algorithm for 50 epochs. In each epoch we trained S-MRS with $m = 50$ randomly chosen images (projected onto the 500 dimensional kernel PCA basis). The training outputs were simply the corresponding unperturbed face images of size 19×19.

Denoising performance is shown in Figure 6.3-a. Since our output space is linear and corresponds to 19×19 images, it is possible to visualise the output basis $\mathbf{V}$ by perturbing the mean image in the direction parameterised by each of the column vectors in $\mathbf{V}$ (see Figure 6.3-b).

7 Conclusion

We have introduced a new framework, MRS, and a sequential variant, S-MRS, for regression between multi-dimensional spaces. The complexity of MRS is $O(m^3)$, which is of the same order as multivariate PLS (with worse constants, though — in the experiments, our method was slower than PLS). Future work will explore relaxation methods to find a convex approximation to our optimization criteria, and regression between discrete spaces, such as graphs [6, 27], on which similarity measures may be defined via kernels. Finally, different regularization and loss functions should be investigated, so as to obtain sparse input and sparse output multivariate regression methods.

References

[1] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, New York, 2001.

[2] B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

[3] D. di Ruscio. A weighted view on the partial least squares algorithm. *Automatica*, 36:831–850, 2000.

[4] J. Weston, O. Chapelle, A. Elisseeff, B. Schölkopf, and V. Vapnik. Kernel dependency estimation. In *Advances in Neural Information Processing Systems*, volume 15, Cambridge, MA, USA, 2003. MIT Press.

[5] C. Cortes, P. Haffner, and M. Mohri. Rational kernels. In *Advances in Neural Information Processing Systems (NIPS 2002)*, volume 15, Cambridge, MA, 2003. MIT Press.

[6] A.J. Smola and R. Kondor. Kernels and regularization on graphs. In *Proceedings of the Annual Conference on Computational Learning Theory and Kernel Workshop*. Springer, 2003.

[7] A. Edelman, T. A. Arias, and S. T. Smith. The geometry of algorithms with orthogonality constraints. *SIAM Journal on Matrix Analysis and Applications*, 20(2):303–353, 1999.

[8] A. W. Bojanczyk and A. Lutoborski. The Procrustes problem for orthogonal Stiefel matrices. *SIAM Journal on Scientific Computing*, 21(4):1291–1304, 2000.

[9] Ali H. Sayed. *Fundamentals of Adaptive Filtering*. Wiley, 2003.

[10] S. Wold, H. Ruhe, H. Wold, and W. J. Dunne III. The collinarity problem in linear regression. the partial least squares (pls) approach to the generalized inverse. *SIAM Journal of Scientific and Statistical Computations*, 5:735–743, 1984.

[11] I. Helland. On the structure of partial least squares regression. *Commun. Statist. Simula.*, 17(2):581–607, 1988.

[12] F. Lindgren, P. Geladi, and S. Wold. The kernel algorithm for pls. *Journal of Chemometrics*, 7:45–60, 1993.

[13] R. Rosipal and L. Trejo. Kernel partial least squares regression in reproducing kernel hilbert spaces. *Journal of Machine Learning Research*, 1(2):97–123, 2001.

[14] K. P. Bennett and M. J. Embrechts. An optimization perspective on partial least squares regression. In J.A.K. Suykens, G. Horvath, S. Basu, C. Micchelli, and J. Vandewalle, editors, *Advances in Learning Theory: Methods, Models and Applications*, pages 227–250. IOS Press, 2003.

[15] S. de Jong. Simpls: An alternative approach to partial least squares regression. *Chemometrics and Intelligent Laboratory Systems*, 18:251–263, 1993.

[16] J. R. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. Technical report, Carnegie Mellon University, 1994.

[17] I. Steinwart. On the influence of the kernel on the consistency of support vector machines. *Journal of Machine Learning Research*, 2:67–93, 2001.

[18] F. Bach and M. Jordan. Kernel independent component analysis. *Journal of Machine Learning Research*, 3:1–48, 2002.

[19] A. Gretton, R. Herbrich, and A. Smola. The kernel mutual information. Technical report, Cambridge University Engineering Department and Max Planck Institute for Biological Cybernetics, 2003.

[20] M. Crampin and F. A. E. Pirani. *Applicable Differential Geometry*. Cambridge University Press, Cambridge, U. K., 1986.

[21] G. H. Golub and C. F. Van Loan. *Matrix Computations*. John Hopkins University Press, Baltimore, MD, 3rd edition, 1996.

[22] B. Schölkopf and A. Smola. *Learning with Kernels*. MIT press, Cambridge, MA, 2002.

[23] J.T. Kwok and I.W. Tsang. Finding the pre images in kernel principal component analysis. In *NIPS'2002 Workshop on Kernel Machines*, 2002.

[24] G. H. Bakır, J. Weston, and B. Schölkopf. Learning to find pre-images. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems (NIPS 2003)*, Cambridge, 2004. MIT Press.

[25] M.W. Spong and M. Vidyasagar. *Robot Dynamics and Control*. John Wiley, New-York, 1991.

[26] H. A. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):23–38, 1998.

[27] H. Kashima, K. Tsuda, and A. Inokuchi. Marginalized kernels between labeled graphs. In *ICML 2003*, pages 321–328, 2003.