MDPI

Article

Aiding Dictionary Learning through Multi-Parametric Sparse Representation

Florin Stoican [1], Paul Irofti [2]

[1] Department of Automatic Control and Computers University Politehnica of Bucharest 313 Spl. Independenței, 060042 Bucharest, Romania; florin.stoican@acse.pub.ro
[2] The Research Institute of the University of Bucharest (ICUB) and Department of Computer Science University of Bucharest, Romania; paul@irofti.net
* Correspondence: florin.stoican@acse.pub.ro; Tel.: +40-760-148-950

Version May 22, 2019 submitted to Algorithms

Abstract: The ℓ_1 relaxations of the sparse and cosparse representation problems which appear in the dictionary learning procedure are usually solved repeatedly (varying only the parameter vector), thus making them well-suited to a multi-parametric interpretation. The associated constrained optimization problems differ only through an affine term from one iteration to the next (i.e., the problem's structure remains the same while only the current vector, which is to be (co)sparsely represented, changes). We exploit this fact by providing an explicit, piecewise affine with a polyhedral support, representation of the solution. Consequently, at runtime, the optimal solution (the (co)sparse representation) is obtained through a simple enumeration throughout the non-overlapping regions of the polyhedral partition and the application of an affine law. We show that, for a suitably large number of parameter instances, the explicit approach outperforms the classical implementation.

Keywords: dictionary learning; multi-parametric problem; sparse representation; cosparse representation; cross-polytopic constraint; rank-deficient quadratic cost.

1. Introduction

We commonly represent m-dimensional data, or signals, through a collection of m linearly independent vectors that together form a base. For certain m-dimensional signals, or classes of signals, we have prior information or insight pointing to the fact that they are in fact perturbed versions of a much simpler, original, true signal that lies in an s-dimensional sub-space. In order to be able to retrieve the original signal we need to have access to a proper base, containing the right s vectors that were used when the true signal was created. With such limited information it is hard to pick such a base, especially for a large class of signals, which is why it is common to extend it with more vectors, making it a redundant base, also called a frame or, as will be used through out this paper, a dictionary. The dictionary has $n > m$ columns, that are also called atoms. The idea is to have more options from which to choose s vectors.

Assume we are given a good dictionary $\boldsymbol{D} \in \mathbb{R}^{m \times n}$ that we want to use to represent a signal $\boldsymbol{y} \in \mathbb{R}^m$ with just s atoms. This leads to $\binom{s}{n}$ possible choices. As the number of atoms increases, so does the complexity of finding the best choice. Even when we decide on a set of atoms, there is still the question of their associated coefficients in the direction they are pointing towards.

The task of finding the atoms and their coefficients is called the sparse representation problem [1]. Existing algorithms pursuit this task for each signal $\boldsymbol{y}$ that they receive, by solving a particular optimization problem based on $\boldsymbol{y}$ that provides the s-sparse representation $\boldsymbol{x}$. In this paper we propose a new scheme where we solve a more complex optimization problem in the beginning, providing

us with an affine model that permits to represent any signal $\boldsymbol{y}$ with just a couple of matrix-vector multiplications.

The dictionary representation step (in either its sparse or cosparse form) consists from the repeated resolution of a constrained optimization problem which differs only through the m-dimensional vector $\boldsymbol{y}$ which is to be represented. This has led us to consider a multi-parametric interpretation where the vector $\boldsymbol{y}$ becomes a parameter which governs the optimization's problem solution and its domain of validity.

While the multi-parametric approach is well-known in the optimization field [2], to the best of our knowledge, it appears not to have been employed by the DL community.

Conceptually, the idea is simple: instead of solving repeatedly very similar optimization problems (with the same structure and differing only in the current value of a parameter) we instead break the problem into two distinct stages:

i) the offline stage: find the way in which the optimal solution depends on parameter $\boldsymbol{y}$ and store this information for later use in the online stage;
ii) the online stage: for the current value of $\boldsymbol{y}$, retrieve and apply the a priori computed solution.

Further assuming that the cost is quadratic and the constraints linear leads, in the offline stage, to a piecewise formulation with a polyhedral support [3,4]. That is, the optimal solution is an affine law in parameter $\boldsymbol{y}$ which is active on a so-called polyhedral *critical region*. Enumerating [5] all these disjoint critical regions covers completely the the parameter space (or the restriction over which $\boldsymbol{y}$ is defined). Thus, at the online stage it remains only to find the region in which the current $\boldsymbol{y}$ lies [6–8], retrieve the associated affine law and apply it. For further details with a practical bent, see the survey [9] which deals with the subtleties of multi-parametric problem in the context of *explicit MPC*.

A couple of issues differentiate the DL problem from other formulations encountered in the literature (e.g., [8,10] which solve a generic complementarity problem):

i) the quadratic cost is unavoidably rank-deficient (the dictionary is over-determined which means that the quadratic cost is characterized by a semi-definite matrix);
ii) the ℓ_1 norm used here (as a relaxation from the sparse restriction induced by the ℓ_0 norm) leads to a very-particular set of linear constraints (they define a cross-polytope [11]).

These issues lead to a particular multi-parametric formulation: the first requires a careful decomposition of the matrices (in order to avoid degeneracy in the formulations [12]); the second requires to consider a vertex-representation of the cross-polytope (as this is a more compact[1] representation than its equivalent half-space representation [13]).

Not in the least we mention some of the difficulties remaining in the multi-parametric representation: the over-approximation of quadratic constraints (as appear whenever the ℓ_2 norm is used as a constraint; the enumeration problem when dealing with large-size parameters and some ideas for alleaviating the issue [14].

The rest of the paper is organized as follows. Section 2 presents the sparse and cosparse representations. Section 3 gives a multi-parametric solution for a generic optimization problem which is then particularized for the DL representation case in Section 4. The multi-parametric approach is tested over a standard example and comparison with the standard case are shown in Section 5. The conclusions are drawn in Section 6.

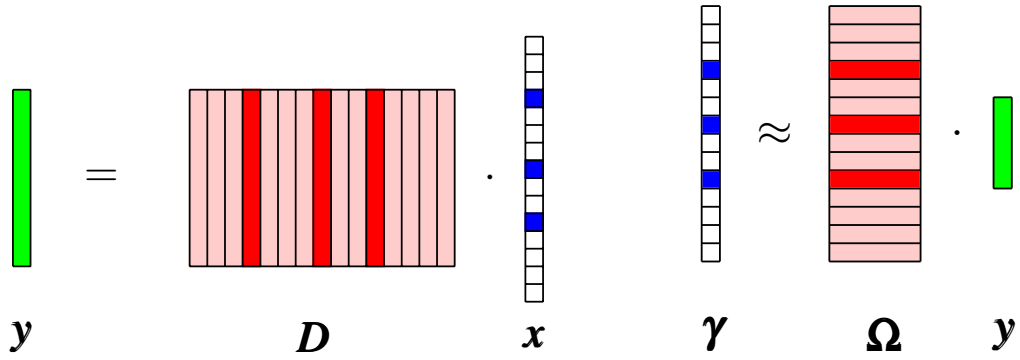
[1] In general, the opposite is true: to a reasonable number of linear inequalities corresponds a significantly larger number of vertices. Thus, most if not all of space partitioning induced by the multi-parametric representation exploit the "half-space" description of the feasible domain [9].

2. The dictionary learning problem

Given the signal $\boldsymbol{y} \in \mathbb{R}^m$ and the fixed dictionary $\boldsymbol{D} \in \mathbb{R}^{m \times n}$, the sparse representation problem imposes an error bound ε and seeks the sparsest representation

$$\min_{\boldsymbol{x}} \quad \|\boldsymbol{x}\|_0 \quad \text{s.t.} \quad \|\boldsymbol{y} - \boldsymbol{D}\boldsymbol{x}\| \leq \varepsilon \tag{1}$$

where $\|.\|_0$ is the ℓ_0 pseudo-norm counting the non-zero entries of $\boldsymbol{x}$. The process is depicted in Figure 1a where the full signal $\boldsymbol{y}$ uses $s = 3$ dictionary atoms for its sparse representation $\boldsymbol{x}$.



(a) Sparse representation. **(b)** Cosparse representation.

Figure 1. Sparse/cosparse representation of a signal [15, Chapter 1 and 10]. The used atoms are red and the nonzero coefficients are blue. The unused atoms are pink.

Due to the non-convex nature of the ℓ_0-norm, we prefer its convex relaxation

$$\min_{\boldsymbol{x}} \quad \|\boldsymbol{y} - \boldsymbol{D}\boldsymbol{x}\|^2 \quad \text{s.t.} \quad \|\boldsymbol{x}\|_1 \leq \theta \tag{2}$$

which is known as the *lasso* problem [16], . Here, θ is a sparsity promoting bound for the 1-norm of the solution. The focus in (2) is on the approximation, such that the objective is minimized up to a point where $\boldsymbol{x}$ has too many large non-zero elements. If instead we desire the sparsest ℓ_1 solution satisfying a given approximation quality ε, the optimization problem becomes

$$\min_{\boldsymbol{x}} \quad \|\boldsymbol{x}\|_1 \quad \text{s.t.} \quad \|\boldsymbol{y} - \boldsymbol{D}\boldsymbol{x}\|^2 \leq \varepsilon \tag{3}$$

Given a set of $\boldsymbol{Y} \in \mathbb{R}^{m \times N}$ training signals, the dictionary learning (DL) problem [15] is

$$\min_{\boldsymbol{D},\boldsymbol{X}} \quad \|\boldsymbol{Y} - \boldsymbol{D}\boldsymbol{X}\|_F^2 \quad \text{s.t.} \quad \|\boldsymbol{x}_\ell\|_1 \leq \theta,\ \ell = 1:N, \quad \left\|\boldsymbol{d}_j\right\| = 1,\ j = 1:n \tag{4}$$

where $\boldsymbol{D} \in \mathbb{R}^{m \times n}$ is the dictionary, whose columns are also called atoms, and $\boldsymbol{X} \in \mathbb{R}^{n \times N}$ are the resulting sparse representations. Note that (4), in contrast to (1), does not only find the sparse representation $\boldsymbol{X}$ but also learns the dictionary $\boldsymbol{D}$.

As depicted in Algorithm 1, problem (4) is usually solved by alternative optimization: fixing $\boldsymbol{D}$ we enter the sparse representation stage (step 2) where we compute the representations $\boldsymbol{X}$ by solving either of (1)–(3) for each signal $\boldsymbol{y} \in \boldsymbol{Y}$, and then we enter the dictionary update stage (step 3) where we fix $\boldsymbol{X}$ and update $\boldsymbol{D}$ by taking each atom $\boldsymbol{d} \in \boldsymbol{D}$ and refining it based on the signals that use it in their sparse representation. The process is repeated K times (step 1) until the objective in (4) converges or the error becomes satisfactory.

Algorithm 1: Alternate optimization dictionary learning

Data: signals set $\boldsymbol{Y} \in \mathbb{R}^{m \times N}$
initial dictionary $\boldsymbol{D} \in \mathbb{R}^{m \times n}$
number of iterations K
representation criteria s, ε, or θ
Result: trained dictionary $\boldsymbol{D}$

for $k = 1 : K$ **do**
  Sparse representation: keeping $\boldsymbol{D}$ fixed, obtain $\boldsymbol{x_i}$ by solving (1), (2), or (3), $i = 1 : N$
  Dictionary update: keeping the resulting representations $\boldsymbol{X}$ fixed, solve (4) to obtain dictionary $\boldsymbol{D}$;

2.1. The Cosparse Model

While sparse representation is interested in finding the non-zero coefficients corresponding to a low-dimensional subspace where signal $\boldsymbol{x}$ lies, cosparse representation is interested in finding its complementary subspace, the subspace orthogonal to $\boldsymbol{x}$. The sparse model can be seen as a synthesis process, focused on constructing $\boldsymbol{x}$ based on the dictionary, and the cosparse model as its analysis counter-part, where the focus is on the original signal $\boldsymbol{y}$ and its null subspace.

As depicted in Figure 1b, in the cosparse model [17], we apply the analysis dictionary $\boldsymbol{\Omega} \in \mathbb{R}^{n \times m}$ to the signal $\boldsymbol{y}$ and obtain the sparse vector

$$\gamma = \boldsymbol{\Omega} \boldsymbol{y}. \tag{5}$$

The atoms are now the rows of dictionary $\boldsymbol{\Omega}$, of which only a few are used in the multiplication with $\boldsymbol{y}$. Unlike the sparse model, here we are interested in the atoms that are not used by $\boldsymbol{y}$, the atoms that are orthogonal to it and create the zero entries in γ.

Assume that signal $\boldsymbol{y}$ is the noisy version of the original, unknown, signal $\boldsymbol{z} \in \mathbb{R}^m$ that lies in a lower-dimensional subspace

$$\boldsymbol{y} = \boldsymbol{z} + \boldsymbol{v}, \tag{6}$$

where $\boldsymbol{v}$ is white additive noise. Let $\boldsymbol{\Omega} \in \mathbb{R}^{n \times m}$ be the given cosparse dictionary. then recovering $\boldsymbol{z}$ means finding the low-dimensional subspace to which it belongs and implies finding the set of atoms Λ from $\boldsymbol{\Omega}$ orthogonal to $\boldsymbol{z}$. In the literature, Λ is sometimes called the cosupport of $\boldsymbol{z}$.

The ℓ_0 optimization problem becomes

$$\min_{\boldsymbol{z},\Lambda} \quad \|\boldsymbol{y} - \boldsymbol{z}\|^2 \quad \text{s.t.} \quad \boldsymbol{\Omega}_\Lambda \boldsymbol{z} = 0, \quad \text{rank}(\boldsymbol{\Omega}_\Lambda) = m - s \tag{7}$$

where we are interested in minimizing the distance between $\boldsymbol{y}$ and $\boldsymbol{z}$ such that the cosupport defines an orthogonal $m - s$ subspace in which $\boldsymbol{z}$ lies. Thus, applying the cosparse dictionary leads to an s-sparse signal as in Figure 1b.

For our purposes, we relax the equality constraints of (7) and move to its convex form

$$\min_{\boldsymbol{z},\Lambda} \quad \|\boldsymbol{y} - \boldsymbol{z}\|^2 \quad \text{s.t.} \quad \|\boldsymbol{\Omega}_\Lambda \boldsymbol{z}\|_1 \leq \theta, \quad \text{rank}(\boldsymbol{\Omega}_\Lambda) \leq m - s \tag{8}$$

where we induce sparsity using the ℓ_1-norm just like we did in (2). Focusing on sparsity instead of approximation, we can rewrite (8) as

$$\min_{\boldsymbol{z},\Lambda} \quad \|\boldsymbol{\Omega}_\Lambda \boldsymbol{z}\|_1 \quad \text{s.t.} \quad \|\boldsymbol{y} - \boldsymbol{z}\|^2 \leq \varepsilon, \quad \text{rank}(\boldsymbol{\Omega}_\Lambda) \leq m - s \tag{9}$$

3. Analysis of the multi-parametric formulation induced by the DL problem

Let us consider the constrained optimization problem

$$\begin{aligned} \min_{\boldsymbol{\xi}} \quad & \|\boldsymbol{T}\boldsymbol{\xi} - \boldsymbol{y}\|_2^2 \\ \text{s.t.} \quad & \|\Delta\boldsymbol{\xi}\|_1 \leq \delta, \end{aligned} \tag{10}$$

with $\boldsymbol{\xi} \in \mathbb{R}^n$, $\boldsymbol{y} \in \mathbb{R}^m$, $\boldsymbol{T} \in \mathbb{R}^{m\times n}$, $\boldsymbol{\Delta} \in \mathbb{R}^{p\times n}$. $\delta \in \mathbb{R}^+$ is a positive scalar.

Note that we make no assumption on the relation between space dimensions m, n, p. In fact, we will be mostly interested in the degenerate case $n > m$ which implies that mapping $\boldsymbol{T}\boldsymbol{\xi}$ is a projection from a higher dimensional space to a lower one (from $\mathbb{R}^n$ to $\mathbb{R}^m$).

The goal of (10) is to provide the constrained optimal solution $\boldsymbol{\xi}^\star$ minimizing the cost while simultaneously respecting the constraints. While multiple [2] tools and theoretical algorithms can be employed to solve (10), in here we propose to exploit two of its particularities:

i) the ℓ_1 norm constraint characterizes a scaled cross-polytope;
ii) the cost is affinely parametrized after parameter $\boldsymbol{y}$.

3.1. Geometrical interpretation of the ℓ_1 norm

The first item stems from a well-known result from polyhedral sets state of the art [18]: the norm inequality $\|\boldsymbol{\xi}\|_1 \leq 1$ describes the n-th order cross-polytope (the dual of a hypercube):

$$C_n = \text{conv}\{\pm\boldsymbol{e}_i\} = \{\boldsymbol{\xi} \in \mathbb{R}^n : \|\boldsymbol{\xi}\|_1 \leq 1\} \tag{11}$$

where e_i are the column vectors which have '1' on the i-th position and '0' anywhere else. Consequently, it follows that $\|\Delta\boldsymbol{\xi}\|_1 \leq \delta$ is equivalent with

$$\{\boldsymbol{\xi} \in \mathbb{R}^n : \|\boldsymbol{\Delta}\boldsymbol{\xi}\|_1 \leq \delta\} = \Delta C_n \cdot \delta = \text{conv}\{\pm\Delta_i\} \cdot \delta, \tag{12}$$

with $\Delta_i \in \mathbb{R}^n$, the column vectors of matrix Δ. Writing explicitly the convex sum from (12) we arrive at

$$\{\xi \in \mathbb{R}^n : \|\Delta\xi\|_1 \leq \delta\} = \{\xi : \xi = \boldsymbol{F}\boldsymbol{\alpha},\ \boldsymbol{\alpha} \geq 0,\ \mathbf{1}^\top\boldsymbol{\alpha} = \delta\}. \tag{13}$$

Note that we made the notation[2] (for compactness reasons) $\boldsymbol{F} = \begin{bmatrix}\Delta & -\Delta\end{bmatrix}$ and that $\alpha \in \mathbb{R}^{2n}$.

With these notations we rewrite (10) into the form:

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \|\boldsymbol{T}\boldsymbol{F}\boldsymbol{\alpha} - \boldsymbol{y}\|_2^2 \\ \text{s.t.} \quad & \boldsymbol{\alpha} \geq 0,\ \mathbf{1}^\top\boldsymbol{\alpha} = \delta. \end{aligned} \tag{14}$$

Note that the optimal solution of (14) is $\alpha^\star$ and, thus, $\xi^\star = F\alpha^\star$.

Remark 1. *Any polyhedral set has a dual representation: half-space (as a finite intersection of linear inequalities) and convex (as a convex sum of extreme points and linear combination of rays – in the case in which is not bounded). Most often the half-space representation is preferred since it is more compact (there are usually many fewer constraints than extreme points). On the other hand, the cross-polytope is the go-to counter-example: there are $2n$ extreme points in (11) but to these correspond 2^n inequalities in the half-space representation. It is thus significantly more efficient to exploit the convex representation in this particular case.*

[2] Any permutation of column vectors $\pm\Delta_i$ in $\boldsymbol{F}$ is equaly valid. We have simply chosen this form as it permits to compactly write $\boldsymbol{F}$ with respect to $\boldsymbol{\Delta}$.

3.2. Karush-Kuhn-Tucker form

The second particularity of (10) (or, equivalently, of (14)) is that the optimization depends on parameter $\boldsymbol{y}$. Inspired by the approach followed in explicit MPC constructions [9] (and which, to the best of our knowledge, has no equivalence in the dictionary classification state of the art) we propose to provide an explicit description of the optimal solution in terms of parameter $\boldsymbol{y}$.

The first step is to rewrite (14) in its dual form (i.e., the Karash-Kuhn-Tucker representation, see [2] or any standard optimization book for a more in-depth description):

$$\begin{aligned}
\text{stationarity:} \quad & \nabla\left(\|\boldsymbol{y}-\boldsymbol{TF}\alpha\|_2^2\right)+\nabla\left(-\alpha\right)\cdot\lambda+\nabla\left(\mathbf{1}^\top\alpha\right)\cdot\mu=0, && \text{(15a)}\\
\text{primal feasibility:} \quad & -\alpha\leq 0,\ \mathbf{1}^\top\alpha=\delta && \text{(15b)}\\
\text{dual feasibility:} \quad & \lambda\geq 0, && \text{(15c)}\\
\text{complementarity:} \quad & \lambda\times(-\alpha)=0. && \text{(15d)}
\end{aligned}$$

The ∇ operator denotes the gradient operation applied, in order, to: the cost, the inequalities and the equality appearing in the primal problem (14). $\lambda\in\mathbb{R}^{2n}$ and $\mu\in\mathbb{R}$ are the Lagrangian multipliers (also called dual variables) associated with the inequalities, respectively the equality constraint of (14). '$\times$' denotes the complementarity condition which requires that either the i-th Lagrangian multiplier λ_i is zero or the i-th inequality is active ($-\alpha_i\leq 0\mapsto -\alpha_i=0$).

Applying the gradient, the KKT form (15) becomes

$$\begin{aligned}
2\left(\boldsymbol{TF}\right)^\top\cdot\left(\boldsymbol{TF}\alpha-\boldsymbol{y}\right)-\lambda+\mathbf{1}\mu=0, && \text{(16a)}\\
\lambda\geq 0,\ -\alpha\leq 0,\ \mathbf{1}^\top\alpha=\delta && \text{(16b)}\\
\lambda\times(-\alpha)=0. && \text{(16c)}
\end{aligned}$$

Under Slater's conditions of optimality [2], the primal variable α and dual variables λ,μ verifying (16) describe the optimum (primal and dual).

Using the well-known result that $\text{rank}(AB)\leq\min(\text{rank}(A)\,\text{rank}(B))$ and that for a matrix $A\in\mathbb{R}^{p\times q}$, $\text{rank}(A)\leq\min(p,q)$ we have that $\text{rank}(\boldsymbol{TF})\leq\min\left(\min(m,n),\min(n,2n)\right)=\min(m,n)$. Thus, the semidefinite matrix $\left(\boldsymbol{TF}\right)^\top\boldsymbol{TF}$ appearing in (16a) is rank-deficient (is a square matrix of dimension $2n$ but its rank is at most $\min(m,n)$). Hereinafter we consider[3] that $m\leq n$ and proceed accordingly (i.e., $\min(m,n)=m$).

As next step, we consider the SVD decomposition $\boldsymbol{TF}=U\Sigma V^\top$ with $U\in\mathbb{R}^{m\times m}$ and $V\in\mathbb{R}^{2n\times 2n}$ orthogonal matrices and $\Sigma\in\mathbb{R}^{m\times 2n}$ the matrix containing the singular values of $\boldsymbol{TF}$. For further use we consider the decompositions $\Sigma=\begin{bmatrix}\Sigma_1 & 0\end{bmatrix}$ and $V^\top=\begin{bmatrix}V_1^\top\\ V_2^\top\end{bmatrix}$ where $\Sigma_1\in\mathbb{R}^{m\times m}$ and $V_1^\top\in\mathbb{R}^{m\times 2n}$, $V_2^\top\in\mathbb{R}^{(2n-m)\times 2n}$. Introducing these notations in (16) allows to rewrite (16a)–(16b) as follows:

$$\begin{aligned}
V_1^\top\alpha-\left(\Sigma_1^{-1}\right)^\top U^\top\boldsymbol{y}-\frac{1}{2}\left(\Sigma_1^{-2}\right)^\top V_1^\top\left(\lambda-\mathbf{1}\mu\right)=0, && \text{(17a)}\\
V_2^\top\left(\lambda-\mathbf{1}\mu\right)=0. && \text{(17b)}
\end{aligned}$$

Remark 2. *Note that we assume that Σ_1 is full-rank which, coupled with its square form, implies that it is non-singular (and consequently admits an inverse). Particular cases where Σ_1 is singular can be treated similarly, by extracting the non-singular part and resizing accordingly $V_1^\top, V_2^\top$.*

[3] The opposite case would unfold in the same manner and is not followed here.

3.3. Multi-parametric interpretation

The difficulty of solving (16), even when making use of (17), lies in the complementarity condition (16c). There are multiple formulations in the literature [4,8,10], each proposing a particular 'flavor'. In what follows we make use of one of the most common approaches: the *active set* method [6]. This approach enumerates[4] combinations of active (those for which $\lambda_i > 0$) and inactive constraints (those for which $\lambda_i = 0$). Subsequently, problem (16) is solved for each of the particular combinations of active constraints

In what follows we form submatrices associated to the active/inactive set of constraints ($\widetilde{(\cdot)}$ for these corresponding to active constraints and $\widehat{(\cdot)}$ for these corresponding to inactive constraints[5]) and reformulate (16) correspondingly:

$$\begin{bmatrix} \hat{V}_1^\top \\ \mathbf{1}^\top \end{bmatrix} \hat{\alpha} - \begin{bmatrix} \left(\Sigma_1^{-1}\right)^\top U^\top \boldsymbol{y} + \frac{1}{2}\left(\Sigma_1^{-2}\right)^\top \left(\tilde{V}_1^\top \tilde{\lambda} - V_1^\top \mathbf{1}\mu\right) \\ \theta \end{bmatrix} = 0, \tag{18a}$$

$$\tilde{V}_2^\top \tilde{\lambda} - V_2^\top \mathbf{1}\mu = 0, \tag{18b}$$

$$\tilde{\lambda} > 0,\ \tilde{\alpha} = 0, \tag{18c}$$

$$\hat{\lambda} = 0,\ \hat{\alpha} > 0. \tag{18d}$$

We note with $\tilde{m}, \hat{m}$ the number of active, respectively inactive, constraints (hence $\tilde{m} + \hat{m} = 2n$).

At this stage, most multi-parametric formulations employ various heuristics for the selection of the set of active constraints [5,8] and solve (18). We proceed similarly with the observation that the cardinality of the set of active constraints has to lead to a well-defined problem (18). In other words, we expect that $\hat{m} \leq m + 1$ (i.e., the number of rows in the matrix left-multiplying α in (18a) is at least equal with the number of columns, which means that the matrix admits a full-rank pseudo-inverse).

Having to accommodate the case $\hat{m} < m + 1$ requires a further partitioning of the matrices appearing in (18), this time in row-blocks: we add the subscript '1' for the sub-matrices formed from the first $\hat{m} - 1$ rows and subscript '2' for the sub-matrices formed from the remaining $m - \hat{m} + 1$ rows[6]. For example, $\hat{V}_1^\top \in \mathbb{R}^{m \times \hat{m}}$ is partitioned into $\hat{V}_1^\top = \begin{bmatrix} V_{11} & V_{12} \end{bmatrix}^\top$ where $V_{11}^\top \in \mathbb{R}^{(\hat{m}-1)\times \hat{m}}$ and $V_{12}^\top \in \mathbb{R}^{(m-\hat{m}+1)\times \hat{m}}$. With these notations, (18a)–(18b) become:

$$\begin{bmatrix} \hat{V}_{11}^\top \\ \mathbf{1}^\top \end{bmatrix} \hat{\alpha} - \begin{bmatrix} \left(\Sigma_{11}^{-1}\right)^\top U^\top \boldsymbol{y} + \frac{1}{2}\left(\Sigma_{11}^{-2}\right)^\top \left(\tilde{V}_1^\top \tilde{\lambda} - V_1^\top \mathbf{1}\mu\right) \\ \theta \end{bmatrix} = 0, \tag{19a}$$

$$\hat{V}_{12}^\top \hat{\alpha} - \left[\left(\Sigma_{12}^{-1}\right)^\top U^\top \boldsymbol{y} + \frac{1}{2}\left(\Sigma_{12}^{-2}\right)^\top \left(\tilde{V}_1^\top \tilde{\lambda} - V_1^\top \mathbf{1}\mu\right)\right] = 0, \tag{19b}$$

$$\tilde{V}_2^\top \tilde{\lambda} - V_2^\top \mathbf{1}\mu = 0. \tag{19c}$$

---

[4] The selection of sets of active constraints and the stop criteria for the search of feasible combinations are subject to significant research and still an open topic [14].

[5] For example, $\tilde{V}_2^\top$ is the matrix constructed from the columns of $V_2^\top$ which correspond to active constraints – those for which $\tilde{\alpha}_i = 0$.

[6] Note that, as before, the partitioning is not unique and any permutation which does not change the number of rows is admissible.

A few matrix manipulations lead to the following compact form:

$$\begin{bmatrix} \hat{V}_{11}^\top \\ \mathbf{1}^\top \end{bmatrix} \hat{\alpha} = \begin{bmatrix} \left(\Sigma_{11}^{-1}\right)^\top U^\top \\ 0 \end{bmatrix} \boldsymbol{y} + \begin{bmatrix} 0 \\ \theta \end{bmatrix} + \begin{bmatrix} \left(\frac{1}{2}\Sigma_{11}^{-2}\right)^\top \\ 0 \end{bmatrix} \begin{bmatrix} \tilde{V}_1^\top & -V_1^\top \mathbf{1} \end{bmatrix} \begin{bmatrix} \tilde{\lambda} \\ \mu \end{bmatrix}, \tag{20a}$$

$$\begin{bmatrix} \frac{1}{2}\left(\Sigma_{12}^{-2}\right)^\top \tilde{V}_1^\top & -\frac{1}{2}\left(\Sigma_{12}^{-2}\right)^\top V_1^\top \mathbf{1} \\ \tilde{V}_2^\top & -V_2^\top \mathbf{1} \end{bmatrix} \begin{bmatrix} \tilde{\lambda} \\ \mu \end{bmatrix} = \begin{bmatrix} \hat{V}_{12}^\top \\ 0 \end{bmatrix} \hat{\alpha} - \begin{bmatrix} \left(\Sigma_{12}^{-1}\right)^\top U^\top \\ 0 \end{bmatrix} \boldsymbol{y}. \tag{20b}$$

Note that the matrices appearing to the left of the equal sign in both (20a) and (20b) are square: the first is in $\mathbb{R}^{\hat{m}\times\hat{m}}$ and the second is in $\mathbb{R}^{(2n-\hat{m}+1)\times(2n-\hat{m}+1)}$. Thus, we can proceed as follows:

i) we obtain $\begin{bmatrix} \tilde{\lambda}^\top & \mu \end{bmatrix}^\top$ from (20b) as a function of $\hat{\alpha}$ and $\boldsymbol{y}$;
ii) we introduce it in (20a) and obtain $\hat{\alpha}$ as an affine form of $\boldsymbol{y}$;
iii) we go back in (20b), replace the now-known $\hat{\alpha}$ and obtain $\begin{bmatrix} \tilde{\lambda}^\top & \mu \end{bmatrix}^\top$ as an affine form of $\boldsymbol{y}$.

For ease of notation we assume that the steps mentioned above lead to

$$\hat{\alpha}(\boldsymbol{y}) = \hat{M}\boldsymbol{y} + \hat{N}, \quad \tilde{\lambda}(\boldsymbol{y}) = \tilde{M}_\lambda \boldsymbol{y} + \tilde{N}_\lambda, \quad \tilde{\mu}(\boldsymbol{y}) = \tilde{M}_\mu \boldsymbol{y} + \tilde{N}_\mu. \tag{21}$$

Combining with the fact that $\tilde{\alpha} = 0, \hat{\lambda} = 0$, as required in (18c)–(18d), allows to write (by interlacing rows of zeroes in matrices $\hat{M}$, $\hat{N}$, $\tilde{M}_\lambda$, $\tilde{N}_\lambda$) the explicit solutions for α, λ:

$$\alpha(\boldsymbol{y}) = M_\alpha \boldsymbol{y} + N_\alpha, \quad \lambda(\boldsymbol{y}) = M_\lambda \boldsymbol{y} + N_\lambda, \quad \tilde{\mu}(\boldsymbol{y}) = \tilde{M}_\mu \boldsymbol{y} + \tilde{N}_\mu. \tag{22}$$

Recalling that $\tilde{\lambda} > 0$ and $\hat{\alpha} > 0$, as required in (18c)–(18d), and using the notation from (21) gives the feasible domain over which the affine laws (21) are valid:

$$\mathcal{R} = \{\boldsymbol{y} : \hat{M}\boldsymbol{y} + \hat{N} > 0,\ M_\lambda \boldsymbol{y} + \tilde{N}_\lambda\} = \left\{\boldsymbol{y} : \begin{bmatrix} \hat{M} \\ M_\lambda \end{bmatrix} \boldsymbol{y} + \begin{bmatrix} \hat{N} \\ \tilde{N}_\lambda \end{bmatrix} > 0\right\}. \tag{23}$$

To recapitulate: starting from an initially chosen collection of active constraints, we have arrived at explicit formulations, parametrized affinely in $\boldsymbol{y}$, for the primal solution α, the dual solutions λ, μ as in (22) and the region (23) for which these solutions are optimal. Note that (23) may turn to be infeasible (the empty set). In this case it simply means that the associated collection of active constraints is not a feasible one and it should be discarded from the enumeration.

Enumerating all feasible combinations of active constraints will [9], thus, partition the parameter domain (the space $\mathbb{R}^p$ or some restriction of it in which $\boldsymbol{y}$ lies) into a non-overlapping collection of regions (23) which is the support of a piecewise law affine in $\boldsymbol{y}$, given by (22):

$$\alpha(\boldsymbol{y}) = M_\alpha^i \boldsymbol{y} + N_\alpha^i, \quad \forall y \in \mathcal{R}^i = \{\boldsymbol{y} : \hat{M}^i \boldsymbol{y} + \hat{N}^i > 0,\ M_\lambda^i \boldsymbol{y} + \tilde{N}_\lambda^i > 0\}. \tag{24}$$

Index i appearing in (24) augments the shorthand 'hat' and 'tilde' notation used earlier for compactness (that is, each time we had, e.g., $\tilde{V}_1$ we should had have written $\tilde{V}_1^i$ to highlight that we are considering the i-th combination of active constraints).

Remark 3. *With (24) computed offline, at runtime we only need to identify the index i for which $\boldsymbol{y} \in \mathcal{R}^i$ holds and apply the corresponding law $\alpha(y)$. For large dimensions this may create storing and retrieval problems (as the number of regions $\mathcal{R}^i$ increases exponentially). Various solutions, with limited succes, are proposed in the literature [4,7,14].*

Remark 4. *Due to the continuity of the piecewise law $\boldsymbol{\alpha}(y)$ and the fact that partition $\cup_i \mathcal{R}^i$ covers the entire domain, we do not require the analysis of cases where more than $m+1$ constraints are inactive [12].*

Illustrative example

For illustration purposes we depict a proof-of-concept example. We take $T = \begin{bmatrix} 1 & 0 & \sqrt{2} \\ 0 & 1 & \sqrt{2} \end{bmatrix}$, $\Delta = \boldsymbol{I}_3$, $\delta = 1$, thus the the space dimensions are $m = 2$, $n = 3$, $p = 3$. Having taken $(\Delta = \boldsymbol{I}_3, \delta = 1)$ means that $\|\Delta x\|_1 \leq \delta$ describes the 3-dimensional cross-polytope $C_3 = \text{conv}\,(\pm e_1, \pm e_2, \pm e_3)$, as shown (semi-transparent red object) in Figure 2. The geometrical interpretation of solving (10) is that

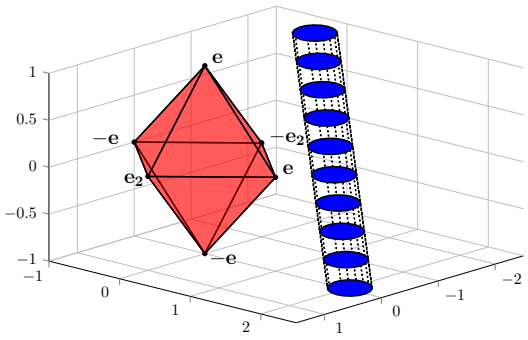
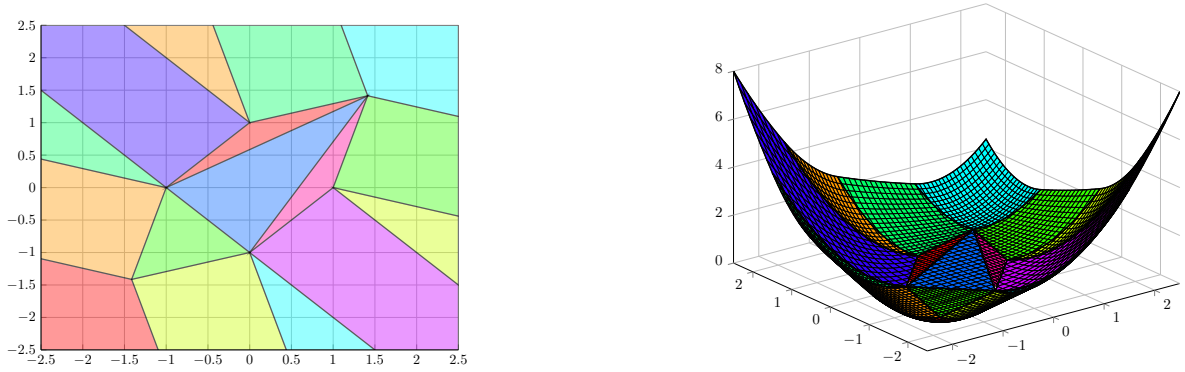


Figure 2. Feasible domain and degenerate cost surface

the unconstrained optimum (the one for which $T\xi - y = 0$) is projected on the feasible domain, thus leading to the constrained optimum (i.e,, the solution $\xi^\star$). This is done by inflating the sub-level cost sets (those for which the cost remains constant on their boundary) until there is an intersection with the feasible domain. Since, in the particular case of (10), the quadratic cost depends on a semi-definite matrix ($T^\top T$ is not full-rank) it means that the optimum cost lies in the null subspace of matrix $T^\top T$ and the sub-level sets correspond to a degenerate ellipsoid (the blue-and-dotted cylinder shown in Figure 2). Note that in (14) and its subsequent variations we made a change of variable which increases the search space from $\mathbb{R}^3$ to $\mathbb{R}^6$, which is of course, not representable in 3D.

Continuing with the geometric interpretation: the sub-level surface is inflated and touches the feasible domain at some point on its boundary. As long as this point is described by the same combination of active constraints (this means in Figure 2 that it is described by the same combination of vertices), we have that the solution $\alpha(\boldsymbol{y})$ is given as in (24) for a fixed index i. Moreover, we identify the region $\mathcal{R}^i$ in which $\boldsymbol{y}$ stays. For this numerical example, the 16 regions $\mathcal{R}^i$ are shown in Figure 3a and the associated cost value in 3b. We have not illustrated the actual solution $\alpha^\star(\boldsymbol{y})$ (due to its



(a) parameter domain partition **(b)** piecewise cost value

Figure 3. Representation of the multi-parametric interpretation.

dimension). We can however note that, as expected, no more than $m + 1 = 3$ vertices are active for any of the critical regions.

4. Integration of the multi-parametric formulation in the representation problem

The sparse representation stage of (4) is a repeated computation of (2), parametrized after signals $\boldsymbol{y} \in \boldsymbol{Y}$. Thus, we may re-interpret (2) as a multi-parametric problem. This means that an explicit representation of the optimal solution can be obtained as a piecewise function, linear in parameter $\boldsymbol{y}$: $\boldsymbol{x}^\star(\boldsymbol{y})$. Thus, the offline part consists in computing the critical regions and their associated laws whereas the online part is simply a lookup procedure which identifies and subsequently applies the law associated to the current value of parameter $\boldsymbol{y}$.

In what follows we proceed to show in Section 4.1 how Section 3 applies to the formulations of Section 2 and to highlight some of the remaining difficulties in Sections 4.2 and 4.3.

4.1. Multi-parametric formulations for the sparse and cosparse representations

In what follows we particularize the results of Section 3 for the sparse and cosparse dictionary representations from Section 2.

sparse representation (2)	form (14)	cosparse representation (8)
$\boldsymbol{x} \in \mathbb{R}^n$	$\boldsymbol{\xi}$	$\boldsymbol{z} \in \mathbb{R}^m$
$\boldsymbol{y} \in \mathbb{R}^m$	$\boldsymbol{y}$	$\boldsymbol{y} \in \mathbb{R}^m$
$\boldsymbol{D} \in \mathbb{R}^{m\times n}$	$\boldsymbol{T}$	$\boldsymbol{I} \in \mathbb{R}^m$
$\boldsymbol{I} \in \mathbb{R}^{n\times n}, \boldsymbol{1} \in \mathbb{R}^n$	$\boldsymbol{\Delta}, \boldsymbol{\delta}$	$\boldsymbol{\Omega} \in \mathbb{R}^{n\times m}, \boldsymbol{1} \in \mathbb{R}^n$
$\begin{bmatrix}\boldsymbol{D} & -\boldsymbol{D}\end{bmatrix} \in \mathbb{R}^{m\times 2n}$	$\boldsymbol{TF}$	$\begin{bmatrix}\boldsymbol{\Omega} & -\boldsymbol{\Omega}\end{bmatrix} \in \mathbb{R}^{n\times 2m}$

Table 1. Equivalences of the sparse and cosparse representations wrt form (14).

For compactness, in Section 3 we have dealt with a somewhat generic constrained optimization problem (particularized in the fact that we start with cross-polytopic constraints and with a rank-deficient quadratic cost). This is justified by Table 1 which shows that both the sparse (2) and the cosparse representation (8) are particular instances of (14). A few details are in order.

The sparse representation (2) meshes easily onto (14) with the observation that the constraints describe a canonical cross-polytope (not scaled or projected in a lower-space). Thus, the application of the multi-parametric proceeds directly and requires no further clarification.

The cosparse representation can as well be written in form (14) with some caveats. Foremost, since it is natural to take $m \ll n$ we may consequently assume that $m \leq 2n$. In this case $\boldsymbol{TF}$ is full column-rank and thus $(\boldsymbol{TF})^\top (\boldsymbol{TF}) \in \mathbb{R}^{2m\times 2m}$ is full-rank, which simplifies the reasoning around (19)–(20). On the other hand, the constraint representation is more challenging. In (8) appears Ω_Λ which is the restriction of Ω by the elimination of $m - s$ columns. This is done in (8) through the addition of a rank condition. Within the framework of Section 3 we can instead restrict the enumeration to those sets of active constraints which verify the rank condition.

4.2. The enumeration roadblock

The "dirty" secret in multi-parametric programming is that the number of critical regions explodes exponentially with the parameter space size and with the complexity of the feasible domain [9]. Moreover, the number of feasible (non-empty) regions is not usually known a priori [6].

In this particular case, the cross-polytope's specific structure means that we can bound a priori the number of critical regions and enumerate them constructively. To do this, we recall [11] that for each pair of non-opposite vertices, there is an edge joining them. More generally, each set of $k+1$ orthogonal vertices corresponds to a distinct k-dimensional component which contains them. The number of k-dimensional components in an n-dimensional cross-polytope is therefore

$$2^{k+1}\binom{n}{k+1}.$$

As shown in Section 3, each critical region is uniquely characterized by a combination of, at least $n-m-1$, active constraints. In the context of (14) and with the notation from Section 3, this means that to each critical region $\mathcal{R}^i$ correspond $\tilde{m}+1 \leq m+1$ non-zero elements from α_i^*. Since, by definition, $\zeta_i^\star = \boldsymbol{F}\alpha_i^\star$ we have that the solution associated with $\mathcal{R}^i$ lies in the facet determined by combination of at most $m+1$ vertices[7] from the cross-polytope C_n. Thus, the total number of feasible combinations of active constraints is:

$$\sum_{k=0}^{m} 2^{k+1} \binom{n}{k+1}. \tag{25}$$

Unfortunately, depending on the values of m, n, the number of regions may prove to be prohibitively large (both from a computational and from a memory footprint/retrieval viewpoint).

While there are multiple approaches which attempt to alleviate the issue [14], there is no panacea. In what follows, we consider a simple approach: regions $\mathcal{R}^i$ are added to the partition only on an "as-needed" basis. More precisely, $\boldsymbol{y}$ is searched in the list of already-computed regions, if found, the affine law (24) is applied, if not, (20) is solved for the current parameter $\boldsymbol{y}$ in order to identify the optimal solution and the critical region over which it is valid. Once found, these are stored for further use.

Algorithm 2: Multi-parametric implementation of the sparse problem.

Data: signals set $\boldsymbol{Y} \in \mathbb{R}^{m \times N}$
collection of critical regions $\mathcal{P} = \{\emptyset\} \subset \mathbb{R}^m$
piecewise affine law $(\mathcal{M} = \{\emptyset\}, \mathcal{N} = \{\emptyset\}) \subset \mathbb{R}^{n \times m} \times \mathbb{R}^n$

1 **for** $k = 1 : N$ **do**
2 Find index of the region containing the current value of parameter $\boldsymbol{y}$:
3 $$i = \arg \min_{i, \mathcal{R}^i \in \mathcal{CR}} \boldsymbol{y} \in \mathcal{R}^i;$$
4 **if** $i = \{\emptyset\}$ **then**
5 solve (20) for the current value of $\boldsymbol{y}$ obtaining the affine law (M_α, N_α) as in (22) and the critical region $\mathcal{R}$, as in (23);
6 update the partition and the affine law collections: $\mathcal{P} = \mathcal{P} \cup \{\mathcal{R}$, $\mathcal{M} = \mathcal{M} \cup M_\alpha$ and $\mathcal{N} = \mathcal{N} \cup N_\alpha$;
7 change the index to point to the last collection element: $i = \#\mathcal{P}$;
8 apply the selected law to the current parameter: $\alpha = M_\alpha^i \boldsymbol{y} + N_\alpha^i$, $x = F\alpha$;

A sketch of the approach is shown in Algorithm 2 (for simplicity's sake, only for the sparse representation case; the cosparse case is similar and can be deduced easily). Some remarks are in order.

Remark 5. *Algorithm 2 only provides an alternative for step 2 of Algorithm 1. Still, the same reasoning can be extended to the dictionary learning as it is, a relatively similar optimization problem.*

Remark 6. *Completing online the partition of critical regions (as is done in step 6 of Algorithm 1) may still lead to an exorbitant number of regions (as it may happen that solving for each $\boldsymbol{y}$ requires to describe a significant part of the entire partitioning). We expect this to happen when the set of signals $\boldsymbol{Y}$ is chosen randomly. Fortunately, as stated earlier, $\boldsymbol{Y}$ is a collection of corrupted signals coming from a smaller sub-space. We expect thus that the number of regions actually described is much less than the total number of feasible ones.*

[7] Due to the structure of the cross-polytope, any feasible combination of d vertices describes a $n-d-1$-order facet, e.g., 2 vertices describe an edge.

Remark 7. *Another approach, not described here is to merge the explicit and implicit approaches [14]. Instead of enumerating all sets of active constraints for a relatively large parameter dimension we may choose to consider a reduced number of constraints. Thus, in the offline stage we compute and store a reasonable number of regions and in the online stage we solve a reduced-size optimization problem (based on information retrieved from the stored lookup tables).*

4.3. On the multi-parametric interpretation of the 'given approximation quality' case

For simplicity we discuss only the sparse representation case shown in (7). The cosparse case (9) can be treated similarly and is not detailed here.

Recall that, as stated in Section 2, the "true" DL formulations are those shown in (1) and (7) where the ℓ_0 norm appears (either explicitly in the first case or implicitly in the second case). Thus, all the other formulations appearing in Section 2 are relaxations[8] which aim to simplify the computational effort.

Until here we have concentrated on these forms where the ℓ_1 norm appears in the constraint part of the optimization problem. Now we mention some of the drawbacks and possible solutions for the other class of relaxations (those for which the ℓ_1 norm appears in the cost). First, we consider an over-approximation of the quadratic constraint. Any ellipsoid (and therefore the ball $\|\boldsymbol{y} - \boldsymbol{Dx}\|_2^2 \leq \epsilon$) is a convex set, which by definition means that it can be approximated arbitrarily well by a polyhedral set [18], say, a construction like

$$\{x:\ F(Dx-y) \leq \theta\} \subset \{x:\ \|\boldsymbol{y} - \boldsymbol{Dx}\|_2^2 \leq \epsilon\} \subset \{x:\ F(Dx-y) \leq (1+\epsilon)\theta\}, \tag{26}$$

with F, θ appropriately chosen.

Proceeding as in Section 3 we arrive at the dual (KKT) form

$$\begin{aligned}
\text{stationarity:} \quad & 0 \in \partial\left(||x||_1\right) + \nabla(FDx - Fy - \theta)\cdot\lambda, && \text{(27a)}\\
\text{primal feasibility:} \quad & FDx - Fy - \theta \leq 0, && \text{(27b)}\\
\text{dual feasibility:} \quad & \lambda \geq 0, && \text{(27c)}\\
\text{complementarity:} \quad & \lambda \times (FDx - Fy - \theta) = 0. && \text{(27d)}
\end{aligned}$$

There are several issues which make (27) challenging to solve as in Section 3:

i) the gradient is not well defined in inflexion points (where at least one of the vector's components is zero); this requires the use of the 'sub-gradient' notion which is set-valued ((27a) becomes '$0 \in \text{sign}(x) + (FD)^\top \cdot \lambda$' where

$$\text{sign}(x_i) = \begin{cases} 1, & x_i > 0 \\ -1, & x_i < 0 \\ [-1,1], & x_i = 0 \end{cases}.$$

ii) the problem may be relatively large (depending on the number of inequalities used to over-approximate the initial quadratic constraint).

Remark 8. *An alternative to item ii) is to consider a zonotopic representation of the quadratic constraints. The ellipsoid they describe is in fact a zonoid (a set which can be approximated arbitrarily well by a zonotope [19]). The advantage of this method is that, in generator representation, a zonotope has a much more compact form (which directly leads to a reduced KKT form).*

[8] Which exploit the sparsity induced by the ℓ_1 norm, appearing either in the cost or as a constraint.

5. Results

In our experiments we use synthetic signals. First we generate a random dictionary of $m \times n$ elements drawn from a zero mean and unit variance Gaussian distribution. After normalizing the dictionary, we randomly choose s atoms to generate sparse signals whose coefficients are also picked at random. Finally we perturb the resulting signal with additive Gaussian

$$\boldsymbol{y} = \boldsymbol{D}_0\boldsymbol{x} + \boldsymbol{v} \tag{28}$$

where $\boldsymbol{D}_0$ is the random dictionary, $\boldsymbol{x}$ the sparse signal and $\boldsymbol{v}$ the additive noise. Signal $\boldsymbol{y}$ is thus a dense signal; all its elements are non-zero. We repeat (28) N times to generate the synthetic signal set $\boldsymbol{Y}$ which is thus known to have a sparse structure.

In the following we generate $N = 1500$ signals, of $m = 20$ elements each, built as the linear combination of $s = \{3, 4, 5\}$ vectors from a dictionary with $n = 50$ atoms.

We proceed with the construction shown in Algorithm 2 where the critical regions and associated optimal laws are computed as-needed. As stated earlier, this is justified by the excessive number of potentially feasible sets of active constraints (25).

Note that we use the real dictionary D_0. For a full analysis (not the object of this paper), we may repeat the procedure for all the intermediate dictionary obtained along the dictionary learning procedure (step 4 in Algorithm 1).

For the $N = 1500$ signals taken in this example we observe that a relatively small number of critical regions are repeating themselves. E.g., taking a signal-to-noise ratio of 1 dB we arrive at 1489 unique regions out of 1500. Repeating for various combinations of signals (with varying degrees of added noise and/or repeating them) shows variation in the number of computed critical regions but without a clear evolution towards the reduction of their total number. We assume, and it remains the subject of further study, that numerical issues lead to improperly computed polyhedral sets (a challenge increased by the output dimension) and that the number and distribution of signals plays a discordantly large role in the distribution of the critical region within the parameter space.

The scripts have been written in Matlab 2018b, using the MPT3 toolbox [20] and Yalmip [21].

6. Conclusions and future directions

In this article we proposed a novel sparse representation method for both the spare and cosparse models by looking at the problem as a multi-parametric formulation and solving it through the Karush-Kuhn-Tucker conditions. Thus we solve a large optimization problem once in the beginning, and then, for each signal, we simply enumerate through a small set of possible regions where the solution might lie. The advantage of this approach is the significant reduction in sparse representation execution times: if the trained dictionary is distributed together with the associated collection of regions, representation reduces to performing a few matrix-vector computations.

Our model is a very good fit for online representation scenarios which is why we plan on investigating in the future its adaptation to online dictionary learning [22], where the dictionary is also updated with each new signal that is sparsely represented. Existing methods have shown that the dictionary update problem can be reduced to a simple rank-1 update $\boldsymbol{D} \leftarrow \boldsymbol{D} + \boldsymbol{\Gamma}$ [15, Chapter 5] and we are currently working on changing $\boldsymbol{\Gamma}$ in order to exploit the existing multi-parametric model of the existing dictionary.

Funding: Paul Irofti was supported by a grant of the Romanian Ministry of Research and Innovation, CCCDI-UEFISCDI, project number 17PCCDI/2018 within PNCDI III.

References

1. Elad, M. *Sparse and redundant representations: from theory to applications in signal and image processing*; Springer Science & Business Media, 2010.
2. Fletcher, R. *Practical methods of optimization*; John Wiley & Sons, 2013.
3. Li, Z.; Ierapetritou, M.G. A method for solving the general parametric linear complementarity problem. *Annals of operations Research* **2010**, *181*, 485–501.
4. Mönnigmann, M.; Jost, M. Vertex based calculation of explicit MPC laws. 2012 American Control Conference (ACC). IEEE, 2012, pp. 423–428.
5. Herceg, M.; Jones, C.N.; Kvasnica, M.; Morari, M. Enumeration-based approach to solving parametric linear complementarity problems. *Automatica* **2015**, *62*, 243–248.
6. TøNdel, P.; Johansen, T.A.; Bemporad, A. An algorithm for multi-parametric quadratic programming and explicit MPC solutions. *Automatica* **2003**, *39*, 489–497.
7. Tøndel, P.; Johansen, T.A.; Bemporad, A. Evaluation of piecewise affine control via binary search tree. *Automatica* **2003**, *39*, 945–950.
8. Bemporad, A. A multiparametric quadratic programming algorithm with polyhedral computations based on nonnegative least squares. *IEEE Transactions on Automatic Control* **2015**, *60*, 2892–2903.
9. Alessio, A.; Bemporad, A. A survey on explicit model predictive control. In *Nonlinear model predictive control*; Springer, 2009; pp. 345–369.
10. Jones, C.N.; Morrari, M. Multiparametric linear complementarity problems. Proceedings of the 45th IEEE Conference on Decision and Control. IEEE, 2006, pp. 5687–5692.
11. Coxeter, H.S.M. *Regular polytopes*; Courier Corporation, 1973.
12. Ahmadi-Moshkenani, P.; Johansen, T.A.; Olaru, S. On degeneracy in exploration of combinatorial tree in multi-parametric quadratic programming. 2016 IEEE 55th Conference on Decision and Control (CDC). IEEE, 2016, pp. 2320–2326.
13. Henk, M.; Richter-Gebert, J.; Ziegler, G.M. 16 basic properties of convex polytopes. In *Handbook of discrete and computational geometry*; CRC Press, 2004; pp. 255–382.
14. Jost, M. Accelerating the calculation of model predictive control laws for constrained linear systems **2016**.
15. Dumitrescu, B.; Irofti, P. *Dictionary Learning Algorithms and Applications*; Springer, 2018; pp. XIV, 284. doi:10.1007/978-3-319-78674-2.
16. Tibshirani, R. Regression Shrinkage and Selection via the Lasso. *J. R. Statist. Soc. B* **1996**, *58*, 267–288.
17. Nam, S.; Davies, M.; Elad, M.; Gribonval, R. The cosparse analysis model and algorithms. *Applied and Computational Harmonic Analysis* **2013**, *34*, 30–56.
18. Ziegler, G.M. *Lectures on polytopes*; Vol. 152, Springer Science & Business Media, 2012.
19. Bourgain, J.; Lindenstrauss, J.; Milman, V. Approximation of zonoids by zonotopes. *Acta mathematica* **1989**, *162*, 73–141.
20. Herceg, M.; Kvasnica, M.; Jones, C.; Morari, M. Multi-Parametric Toolbox 3.0. Proc. of the European Control Conference; , 2013; pp. 502–510. http://control.ee.ethz.ch/~mpt.
21. Löfberg, J. YALMIP : A Toolbox for Modeling and Optimization in MATLAB. Proceedings of the CACSD Conference; , 2004.
22. Skretting, K.; Engan, K. Recursive least squares dictionary learning. *IEEE Trans. Signal Proc.* **2010**, *58*, 2121–2130.