Online Group-Structured Dictionary Learning*

Zoltán Szabó[1] Barnabás Póczos[2] András Lőrincz[1]

[1]School of Computer Science, Eötvös Loránd University, Pázmány Péter sétány 1/C, H-1117 Budapest, Hungary

[2]School of Computer Science, Carnegie Mellon University, 5000 Forbes Ave, 15213, Pittsburgh, PA, USA

szzoli@cs.elte.hu, bapoczos@cs.cmu.edu, andras.lorincz@elte.hu

Abstract

We develop a dictionary learning method which is (i) online, (ii) enables overlapping group structures with (iii) non-convex sparsity-inducing regularization and (iv) handles the partially observable case. Structured sparsity and the related group norms have recently gained widespread attention in group-sparsity regularized problems in the case when the dictionary is assumed to be known and fixed. However, when the dictionary also needs to be learned, the problem is much more difficult. Only a few methods have been proposed to solve this problem, and they can handle two of these four desirable properties at most. To the best of our knowledge, our proposed method is the first one that possesses all of these properties. We investigate several interesting special cases of our framework, such as the online, structured, sparse non-negative matrix factorization, and demonstrate the efficiency of our algorithm with several numerical experiments.

1. Introduction

Sparse signal representation and signal processing are in the focus of machine learning research. In the *sparse coding* framework one approximates the observations with the linear combination of a few vectors (basis elements) from a fixed dictionary. This principle has been successful in a number of applications including the processing of natural images, bio-informatics and many others. For a recent review see [29].

The general task, namely the ℓ_0-norm solution that searches for the least number of basis elements is NP-hard, and thus one often considers the relaxed and convex ℓ_1 variant of this task, the so-called Lasso problem [28]. The ℓ_1-norm based approach leads to sparse models, but it does not take into account any prior information about the structure of hidden representation (also called covariates, or code), for example that certain covariate groups are selected jointly. Numerous works point to the advantages if such structure could be taken into account. The Lasso formulation is improved from this point of view in the group Lasso framework using *group* $\ell_{1,2}$-norm, where the coordinates of the hidden representation may form distinct groups [33]. Recently, [9] presented a general theoretical framework underpinning the advantages of such a group based Lasso assumption. Among the broad spectrum of successful applications of group norms, one finds multi-task feature learning [2], joint subspace/covariate selection for classification [22], and structure learning in log-linear models [26], too.

Recent research on *structured-sparsity* has shown that more general structures than sparse disjunct groups, such as trees or general groups with possible overlaps may help in many applications, *e.g.*, in multiple kernel learning and multi-task regression [15]. For more information on tree-structured group Lasso, and structured sparsity regularization see [18, 11, 24, 21, 34].

All the above Lasso-like problems assume, however, that the dictionary is fixed and known. This is not the case in many tasks, and learning a dictionary that leads to sparse codes can be important. This is the *dictionary learning* task [32] (also called matrix factorization [31]), which can be traced back to [23]. Dictionary learning is a general problem class that contains, *e.g.*, (sparse) Principal Component Analysis (PCA) [36], Independent Component Analysis (ICA) [10], and (sparse) Non-negative Matrix Factorization (NMF) [17, 27, 8], among many others. Considerable research efforts have been devoted to these problems and led to state-of-the-art methods, see, *e.g.*, the image processing application in [1].

Although both dictionary learning and structured sparse coding (when the dictionary is given) are very popular, interestingly, very few works focused on the combination of these two tasks, *i.e.*, on the learning of *structured dictionaries* by pre-assuming certain structures on the representation. We list a few notable exceptions. Groups are considered on the observations in [4] with alternating minimization of the

*©2011 IEEE. IEEE Computer Vision and Pattern Recognition (CVPR 2011), pages 2865-2872, Colorado Springs, CO, USA. http://dx.doi.org/10.1109/CVPR.2011.5995712.

dictionary and the hidden representations subject to group $\ell_{1,2}$ and group $\ell_{1,1}$ or $\ell_{1,2}$ regularization, respectively. Tree based group structure is assumed in [12], and dictionary learning is accomplished by means of the so-called proximal methods [6]. The efficiency of *non-convex sparsity-inducing norms* on the dictionary has recently been demonstrated in structured sparse PCA [13]. General *group-structured*, but convex sparsity-inducing regularizer is applied in [20] for the learning of the dictionary by taking advantage of network flow algorithms. In [25], the authors take partition (special group structure) on the hidden covariates and explicitly limit the number of non-zero elements in each group in the dictionary learning problem.

All the cited algorithms above work *off-line*. However, online methods fit large or slowly varying systems better. The cost function based on structure inducing regularization in [14] is a special case of [13]. However, as opposed to the previous works, here in [14] the presented dictionary learning approach is *online*. Lasso and certain convex regularizers are used for online dictionary learning in [19] allowing a continuous flow of observations, but group structures are not considered.

All of these methods deal with the fully observable case. By contrast, [3] develops an online dictionary learning technique for PCA subject to *missing observations*, but without group structures.

Our goal is to develop a dictionary learning method exhibiting all the four properties at a time, *i.e.*, it (i) is online, (ii) enables general overlapping group structures, (iii) applies non-convex sparsity inducing regularization, and (iv) can deal with missing information. The above methods can exhibit two of these features at most. We will see that the derivation of such an algorithm is far from being trivial. We will reduce the optimization of dictionary learning to convex subtasks and derive online rules for the update of the dictionary using block-coordinate descent method. This is the contribution of our current work.

The paper is built as follows: We define the *online group-structured dictionary learning* (OSDL) task in Section 2. Section 3 is dedicated to our optimization scheme solving the OSDL problem. Numerical examples are shown in Section 4. Conclusions are drawn in Section 5.

Notations. Vectors have bold faces ($\mathbf{a}$), matrices are written by capital letters ($\mathbf{A}$). The i^{th} coordinate of vector $\mathbf{a}$ is a_i, $diag(\mathbf{a})$ denotes the diagonal matrix formed from vector $\mathbf{a}$. For a set (number), $|\cdot|$ denotes the number of elements in the set, (the absolute value of the number). For $\mathbf{a} \in \mathbb{R}^d, \mathbf{A} \in \mathbb{R}^{d\times D}$ and for set $O \subseteq \{1,\dots,d\}$, $\mathbf{a}_O \in \mathbb{R}^{|O|}$ denotes the coordinates of vector $\mathbf{a}$ in O, whereas $\mathbf{A}_O \in \mathbb{R}^{|O|\times D}$ contains the rows of matrix $\mathbf{A}$ in O. $\mathbf{A}^T$ is the transposed of matrix $\mathbf{A}$. $\mathbf{I}$ and $\mathbf{0}$ stand for the identity and the null matrices, respectively. Operation $\max$ acts component-wise on vectors. For positive numbers p, q, (i) (quasi-)norm ℓ_q of vector $\mathbf{a} \in \mathbb{R}^d$ is $\|\mathbf{a}\|_q = (\sum_{i=1}^d |a_i|^q)^{\frac{1}{q}}$, (ii) $\ell_{p,q}$-norm of the same vector is $\|\mathbf{a}\|_{p,q} = \|[\|\mathbf{a}_{P_1}\|_q,\dots,\|\mathbf{a}_{P_K}\|_q]\|_p$, where $\{P_i\}_{i=1}^K$ is a partition of the set $\{1,\dots,d\}$. $S_p^d = \{\mathbf{a} \in \mathbb{R}^d : \|\mathbf{a}\|_p \le 1\}$ is the unit sphere associated with ℓ_p in $\mathbb{R}^d$. Pointwise product of vectors $\mathbf{a}, \mathbf{b} \in \mathbb{R}^d$ is denoted by $\mathbf{a} \circ \mathbf{b} = [a_1 b_1; \dots; a_d b_d]$. For any given set system $\mathcal{G}$, elements of vector $\mathbf{a} \in \mathbb{R}^{|\mathcal{G}|}$ are denoted by a^G, where $G \in \mathcal{G}$, that is $\mathbf{a} = (a^G)_{G\in\mathcal{G}}$. $\Pi_{\mathcal{C}}(\mathbf{x}) = \operatorname{argmin}_{\mathbf{c}\in\mathcal{C}} \|\mathbf{x} - \mathbf{c}\|_2$ denotes the orthogonal projection to the closed and convex set $\mathcal{C} \subseteq \mathbb{R}^d$, where $\mathbf{x} \in \mathbb{R}^d$. Partial derivative of function g with respect to variable $\mathbf{x}$ at point $\mathbf{x}_0$ is $\frac{\partial g}{\partial \mathbf{x}}(\mathbf{x}_0)$. $\mathbb{R}^d_+ = \{\mathbf{x} \in \mathbb{R}^d : x_i \ge 0\, (\forall i)\}$ stands for the non-negative ortant in $\mathbb{R}^d$.

2. Formal problem definition

Let us define the online group-structured dictionary learning task starting from the fully observed case. Our goal in dictionary learning is to find a dictionary matrix $\mathbf{D} \in \mathbb{R}^{d_x \times d_\alpha}$ that can approximate observations $\mathbf{x}_i \in \mathbb{R}^{d_x}$ by the linear combinations of its columns. We assume that dictionary $\mathbf{D}$ belongs to a closed, convex and bounded set $\mathcal{D}$ ($\mathbf{D} \in \mathcal{D}$), which is defined by the direct product of constraints $\mathcal{D}_i$ of columns $\mathbf{d}_i$ (atoms, basis elements) of matrix $\mathbf{D}$ ($\mathcal{D} = \times_{i=1}^{d_\alpha} \mathcal{D}_i$). We assume further that the hidden representation (coefficients) $\boldsymbol{\alpha}_i \in \mathbb{R}^{d_\alpha}$ of observation $\mathbf{x}_i$ belongs to a convex, closed set $\mathcal{A}$ ($\boldsymbol{\alpha}_i \in \mathcal{A}$) subject to certain structural constraints. Namely, we assume that (i) a group structure $\mathcal{G}$ is given for the hidden representation, that is, a subset of the power set of $\{1,\dots,d_\alpha\}$ for which $\cup_{G\in\mathcal{G}} G = \{1,\dots,d_\alpha\}$, and (ii) weight vectors $\mathbf{d}^G \in \mathbb{R}^{d_\alpha}$ ($G \in \mathcal{G}$) are also given. For a given weight vector $\mathbf{d}^G$, the coefficients belonging to G are positive, and the coefficients not in G are zeros. For *fixed* $\mathbf{D}$ and $\mathbf{x}$, we define the representation $\boldsymbol{\alpha}$ of $\mathbf{x}$ to be the vector in $\mathcal{A}$ that minimizes the following structured sparse representation task

$$l(\mathbf{x},\mathbf{D}) = l_{\kappa,\eta,\mathcal{G},\{\mathbf{d}^G\}_{G\in\mathcal{G}}}(\mathbf{x},\mathbf{D}) \quad (1)$$

$$= \min_{\boldsymbol{\alpha}\in\mathcal{A}} \left[\frac{1}{2}\|\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}\|_2^2 + \kappa\Omega(\boldsymbol{\alpha})\right], \quad (2)$$

where $\kappa > 0$, $\eta \in (0,1]$, and

$$\Omega(\mathbf{y}) = \Omega_{\eta,\mathcal{G},\{\mathbf{d}^G\}_{G\in\mathcal{G}}}(\mathbf{y}) = \|(\|\mathbf{d}^G \circ \mathbf{y}\|_2)_{G\in\mathcal{G}}\|_\eta \quad (3)$$

is the structured regularizer for groups G in $\mathcal{G}$ and for weights $\mathbf{d}^G$.

Let $\mathbf{x}_i \in \mathbb{R}^{d_x}$ $(i = 1,2,\dots)$ be a sequence of i.i.d. (independent identically distributed) observations. The online group-structured dictionary learning (OSDL) problem is defined as the minimization of the following cost function:

$$\min_{\mathbf{D}\in\mathcal{D}} f_t(\mathbf{D}) := \frac{1}{\sum_{j=1}^t (j/t)^\rho} \sum_{i=1}^t \left(\frac{i}{t}\right)^\rho l(\mathbf{x}_i,\mathbf{D}), \quad (4)$$

where ρ is a non-negative forgetting factor. For the case of $\rho = 0$ in (4), $f_t(\mathbf{D}) = \frac{1}{t}\sum_{i=1}^{t} l(\mathbf{x}_i, \mathbf{D})$ reduces to the empirical average. Note that here the OSDL task is defined via the sparsity-inducing regularizer Ω [13] aiming to eliminate the terms $\|\mathbf{d}^G \circ \mathbf{y}\|_2$ ($G \in \mathcal{G}$) by means of $\|\cdot\|_\eta$. An alternative sparsity inducing solution (for fixed $\mathbf{D}$) is provided in [11, 24], it searches for non-zero elements of $\boldsymbol{\alpha}$ on the union of groups in $\mathcal{G}$.

Let us now define the OSDL problem for the partially observable case. Now, at time instant i we can access only a certain subset $O_i \subseteq \{1, \ldots, d_x\}$ of $\mathbf{x}_i$. We modify l in (2) by applying the approach of [31, 3], that is, we use the error on the observed coordinates:

$$l(\mathbf{x}_{O_i}, \mathbf{D}_{O_i}) = \min_{\boldsymbol{\alpha}\in\mathcal{A}} \left[\frac{1}{2}\|\mathbf{x}_{O_i} - \mathbf{D}_{O_i}\boldsymbol{\alpha}\|_2^2 + \kappa\Omega(\boldsymbol{\alpha})\right], \tag{5}$$

and we also change $l(\mathbf{x}_i, \mathbf{D})$ to $l(\mathbf{x}_{O_i}, \mathbf{D}_{O_i})$ in optimization (4). In turn, our goal is to solve the following minimization

$$\min_{\mathbf{D}\in\mathcal{D}} f_t(\mathbf{D}) := \frac{1}{\sum_{j=1}^{t}(j/t)^\rho}\sum_{i=1}^{t}\left(\frac{i}{t}\right)^\rho l(\mathbf{x}_{O_i}, \mathbf{D}_{O_i}). \tag{6}$$

2.1. Interesting special cases

For $O_i = \{1, \ldots, d_x\}$ $(\forall i)$ the fully observed OSDL task is recovered. Further special cases of the OSDL model include the following:

Special cases for $\mathcal{G}$:

- If $|\mathcal{G}| = d_\alpha$ and $\mathcal{G} = \{\{1\}, \{2\}, \ldots, \{d_\alpha\}\}$, then we assume no dependence between coordinates α_i, and the problem reduces to the classical task of learning 'sparse representation dictionaries'. A further specialization is when $\mathbf{D}$ is given, $\rho = 0$, $\eta = 1$, $\mathbf{d}^i = \mathbf{e}_i$, where $\mathbf{e}_i$ is the i^{th} canonical basis vector. This corresponds to the Lasso task.
- If $|\mathcal{G}| = d_\alpha$, coordinates α_i make the nodes of a tree, and $\mathcal{G} = \{descendants_1, \ldots, descendants_{d_\alpha}\}$, where $descendants_i$ stands for the i^{th} node and its descendants, then we have a tree-structured, hierarchial representation.
- If $|\mathcal{G}| = d_\alpha$, coordinates α_i make the nodes of a grid, and $\mathcal{G} = \{NN_1, \ldots, NN_{d_\alpha}\}$, where NN_i denotes the neighbors of the i^{th} point in radius r on the grid, then we obtain a grid representation.
- If $\mathcal{G} = \{P_1, \ldots, P_K\}$, where $\{P_k\}_{k=1}^K$ is a partition of $\{1, \ldots, d_\alpha\}$, then non-overlapping group structure is obtained.

Special cases for $\mathcal{D}, \mathcal{A}$:

- $\mathcal{D}_i = S_2^{d_x}$ $(\forall i)$, $\mathcal{A} = \mathbb{R}^{d_\alpha}$: columns of dictionary $\mathbf{D}$ are constrained to the Euclidean unit sphere.
- $\mathcal{D}_i = S_2^{d_x} \cap \mathbb{R}_+^{d_x}$ $(\forall i)$, $\mathcal{A} = \mathbb{R}_+^{d_\alpha}$: columns of dictionary $\mathbf{D}$ are constrained to the non-negative ℓ_2 unit sphere, α_is are non-negative and $\mathcal{G}$ can arbitrary. This is the structured NMF model.
- $\mathcal{D}_i = S_1^{d_x} \cap \mathbb{R}_+^{d_x}$ $(\forall i)$, $\mathcal{A} = \mathbb{R}_+^{d_\alpha}$: columns of dictionary $\mathbf{D}$ are constrained to the non-negative ℓ_1-sphere, α_is are non-negative and $\mathcal{G}$ can arbitrary. This is the structured mixture-of-topics model.

3. Optimization

We consider the optimization of cost function (6), which is equivalent to the joined optimization of dictionary $\mathbf{D}$ and coefficients $\{\boldsymbol{\alpha}_i\}_{i=1}^t$:

$$\operatorname*{arg\,min}_{\mathbf{D}\in\mathcal{D}, \{\boldsymbol{\alpha}_i\in\mathcal{A}\}_{i=1}^t} f_t(\mathbf{D}, \{\boldsymbol{\alpha}_i\}_{i=1}^t), \tag{7}$$

where

$$f_t = \frac{1}{\sum_{j=1}^{t}(j/t)^\rho}\sum_{i=1}^{t}\left(\frac{i}{t}\right)^\rho \left[\frac{1}{2}\|\mathbf{x}_{O_i} - \mathbf{D}_{O_i}\boldsymbol{\alpha}_i\|_2^2 + \kappa\Omega(\boldsymbol{\alpha}_i)\right]. \tag{8}$$

Assume that our samples $\mathbf{x}_i$ are emitted from an i.i.d. source $p(\mathbf{x})$, and we can observe $\mathbf{x}_{O_i}$. We execute the online optimization of dictionary $\mathbf{D}$ (*i.e.*, the minimization of (7)) through alternations:

1. For the actual sample $\mathbf{x}_{O_t}$ we optimize hidden representation $\boldsymbol{\alpha}_t$ belonging to $\mathbf{x}_{O_t}$ using our estimated dictionary $\mathbf{D}_{t-1}$ and solving the minimization task

$$\boldsymbol{\alpha}_t = \operatorname*{argmin}_{\boldsymbol{\alpha}\in\mathcal{A}} \left[\frac{1}{2}\|\mathbf{x}_{O_t} - (\mathbf{D}_{t-1})_{O_t}\boldsymbol{\alpha}\|_2^2 + \kappa\Omega(\boldsymbol{\alpha})\right]. \tag{9}$$

2. We use hidden representations $\{\boldsymbol{\alpha}_i\}_{i=1}^t$ and update $\mathbf{D}_{t-1}$ by means of quadratic optimization

$$\hat{f}_t(\mathbf{D}_t) = \min_{\mathbf{D}\in\mathcal{D}} f_t(\mathbf{D}, \{\boldsymbol{\alpha}_i\}_{i=1}^t). \tag{10}$$

In the next subsections, we elaborate on the optimization of representation $\boldsymbol{\alpha}$ in (9) and the dictionary $\mathbf{D}$ in (10).

3.1. Representation update ($\boldsymbol{\alpha}$)

Objective function (9) is not convex in $\boldsymbol{\alpha}$. We use a variational method to find a solution: (i) we rewrite the term Ω by introducing an auxiliary variable ($\mathbf{z}$) that converts the expression to a quadratic one in $\boldsymbol{\alpha}$, and then (ii) we use an explicit solution to $\mathbf{z}$ and continue by iteration. Namely, we use Lemma 3.1 of [13]: for any $\mathbf{y} \in \mathbb{R}^d$ and $\eta \in (0, 2)$

$$\|\mathbf{y}\|_\eta = \min_{\mathbf{z}\in\mathbb{R}_+^d} \frac{1}{2}\sum_{i=1}^{d}\frac{y_j^2}{z_j} + \frac{1}{2}\|\mathbf{z}\|_\beta, \tag{11}$$

where $\beta = \frac{\eta}{2-\eta}$, and it takes its minimum value at $z_i^* = |y_i|^{2-\eta}\|\mathbf{y}\|_\eta^{\eta-1}$. We apply this relation for the term Ω in (9) (see Eq. (3)), and have that

$$2\Omega(\boldsymbol{\alpha}) = \min_{\mathbf{z}=[(z^G)_{G\in\mathcal{G}}]\in\mathbb{R}_+^{|\mathcal{G}|}} \left[\sum_{G\in\mathcal{G}} \frac{\left\|\mathbf{d}^G \circ \boldsymbol{\alpha}\right\|_2^2}{z^G} + \|\mathbf{z}\|_\beta\right] = \min_{\mathbf{z}\in\mathbb{R}_+^{|\mathcal{G}|}} \left[\boldsymbol{\alpha}^T diag(\boldsymbol{\zeta})\boldsymbol{\alpha} + \|\mathbf{z}\|_\beta\right], \quad (12)$$

where $\boldsymbol{\zeta} = \boldsymbol{\zeta}(\mathbf{z}) \in \mathbb{R}^{d_\alpha}$, and

$$\zeta_j = \sum_{G\in\mathcal{G}, G\ni j} \frac{\left(d_j^G\right)^2}{z^G}. \quad (13)$$

Inserting (12) into (9) we get the optimization task:

$$\arg\min_{\boldsymbol{\alpha}\in\boldsymbol{\mathcal{A}}, \mathbf{z}\in\mathbb{R}_+^{|\mathcal{G}|}} J(\boldsymbol{\alpha}, \mathbf{z}), \text{ where} \quad (14)$$

$$J(\boldsymbol{\alpha}, \mathbf{z}) = \quad (15)$$

$$= \frac{1}{2}\left\|\mathbf{x}_{O_t} - (\mathbf{D}_{t-1})_{O_t}\boldsymbol{\alpha}\right\|_2^2 + \kappa\frac{1}{2}\left(\boldsymbol{\alpha}^T diag(\boldsymbol{\zeta})\boldsymbol{\alpha} + \|\mathbf{z}\|_\beta\right).$$

One can solve the minimization of $J(\boldsymbol{\alpha}, \mathbf{z})$ by alternations:

1. For given $\mathbf{z}$: we can use least mean square solver for $\boldsymbol{\alpha}$ when $\boldsymbol{\mathcal{A}} = \mathbb{R}^{d_\alpha}$ in (15), and non-negative least square solver when $\boldsymbol{\mathcal{A}} = \mathbb{R}_+^{d_\alpha}$. For the general case, the cost function $J(\boldsymbol{\alpha}, \mathbf{z})$ is quadratic in $\boldsymbol{\alpha}$ and is subject to convex and closed constraints ($\boldsymbol{\alpha} \in \boldsymbol{\mathcal{A}}$). There are standard solvers for this case [16, 5], too.

2. For given $\boldsymbol{\alpha}$: According to (11), the minimum $\mathbf{z} = (z^G)_{G\in\mathcal{G}}$ can be found as

$$z^G = \|\mathbf{d}^G \circ \boldsymbol{\alpha}\|_2^{2-\eta}\|(\|\mathbf{d}^G \circ \boldsymbol{\alpha}\|_2)_{G\in\mathcal{G}}\|_\eta^{\eta-1}. \quad (16)$$

   Note that for numerical stability smoothing, $\mathbf{z} = \max(\mathbf{z}, \varepsilon)$ ($0 < \varepsilon \ll 1$), is suggested in practice.

3.2. Dictionary update ($\mathbf{D}$)

We use block-coordinate descent (BCD) [5] for the optimization of (10). This optimization is not influenced by the regularizer $\Omega(\boldsymbol{\alpha})$, since it is independent of $\mathbf{D}$. Thus the task (10) is *similar* to the fully observable case [19], where for $O_i = \{1, \ldots, d_x\}$ $(\forall i)$ it has been shown that the BCD method can work without storing all of the vectors $\mathbf{x}_i, \boldsymbol{\alpha}_i$ $(i \le t)$. Instead, it is sufficient to keep certain statistics that characterize $\hat{f}_t$, which can be updated online. This way, optimization of $\hat{f}_t$ in (10) becomes online, too. As it will be elaborated below, (i) certain statistics describing $\hat{f}_t$ can also be derived for the partially observed case, which (ii) can be updated online with a single exception, and (iii) a good approximation exists for that exception (see Section 4).

During the BCD optimization, columns of $\mathbf{D}$ are minimized sequentially: other columns than the actually updated $\mathbf{d}_j$ (*i.e.*, $\mathbf{d}_i, i \neq j$) are kept fixed. The function $\hat{f}_t$ is quadratic in $\mathbf{d}_j$. During minimization we search for its minimum (denoted by $\mathbf{u}_j$) and project the result to the constraint set $\mathcal{D}_j$ ($\mathbf{d}_j \leftarrow \Pi_{\mathcal{D}_j}(\mathbf{u}_j)$). To find this $\mathbf{u}_j$, we solve the equation $\frac{\partial \hat{f}_t}{\partial \mathbf{d}_j}(\mathbf{u}_j) = \mathbf{0}$, which leads (as we show it in the supplementary material) to the following linear equation system

$$\mathbf{C}_{j,t}\mathbf{u}_j = \mathbf{b}_{j,t} - \mathbf{e}_{j,t} + \mathbf{C}_{j,t}\mathbf{d}_j, \quad (17)$$

where $\mathbf{C}_{j,t} \in \mathbb{R}^{d_x \times d_x}$ is a diagonal coefficient matrix, and

$$\mathbf{C}_{j,t} = \sum_{i=1}^{t} \left(\frac{i}{t}\right)^\rho \boldsymbol{\Delta}_i \alpha_{i,j}^2, \quad (18)$$

$$\mathbf{B}_t = \sum_{i=1}^{t} \left(\frac{i}{t}\right)^\rho \boldsymbol{\Delta}_i \mathbf{x}_i \boldsymbol{\alpha}_i^T = [\mathbf{b}_{1,t}, \ldots, \mathbf{b}_{d_\alpha,t}], \quad (19)$$

$$\mathbf{e}_{j,t} = \sum_{i=1}^{t} \left(\frac{i}{t}\right)^\rho \boldsymbol{\Delta}_i \mathbf{D}\boldsymbol{\alpha}_i \alpha_{i,j}. \quad (20)$$

Here $\boldsymbol{\Delta}_i$ represents a diagonal matrix corresponding to O_i (element j in the diagonal is 1 if $j \in O_i$, and 0 otherwise). $\mathbf{C}_{j,t}$s $\in \mathbb{R}^{d_x \times d_x}$ and $\mathbf{B}_t \in \mathbb{R}^{d_x \times d_\alpha}$ take the form of $\mathbf{M}_t = \sum_{i=1}^{t} \left(\frac{i}{t}\right)^\rho \mathbf{N}_i$ matrix series/statistics, and thus (as we detail it in the supplementary material) they can be updated as

$$\mathbf{C}_{j,t} = \gamma_t \mathbf{C}_{j,t-1} + \boldsymbol{\Delta}_t \alpha_{tj}^2, \quad (21)$$

$$\mathbf{B}_t = \gamma_t \mathbf{B}_{t-1} + \boldsymbol{\Delta}_t \mathbf{x}_t \boldsymbol{\alpha}_t^T, \quad (22)$$

with initialization $\mathbf{C}_{j,0} = \mathbf{0}$, $\mathbf{B}_0 = \mathbf{0}$ for the case of $\rho = 0$, and with arbitrary initialization for $\rho > 0$, where $\gamma_t = \left(1 - \frac{1}{t}\right)^\rho$. For the fully observed case ($\boldsymbol{\Delta}_i = \mathbf{I}$, $\forall i$), one can pull out $\mathbf{D}$ from $\mathbf{e}_{j,t} \in \mathbb{R}^{d_x}$, the remaining part is of the form $\mathbf{M}_t$, and thus it can be updated online giving rise to the update rules in [19], see the supplementary material. In the general case this procedure can not be applied (matrix $\mathbf{D}$ changes during the BCD updates). According to our numerical experiences (see Section 4) an efficient online approximation for $\mathbf{e}_{j,t}$ is

$$\mathbf{e}_{j,t} = \gamma_t \mathbf{e}_{j,t-1} + \boldsymbol{\Delta}_t \mathbf{D}_t \boldsymbol{\alpha}_t \alpha_{t,j}, \quad (23)$$

with the actual estimation for $\mathbf{D}_t$ and with initialization $\mathbf{e}_{j,0} = \mathbf{0}$ $(\forall j)$. We note that

1. convergence is often speeded up if the updates of statistics $\{\{\mathbf{C}_{j,t}\}_{j=1}^{d_\alpha}, \mathbf{B}_t, \{\mathbf{e}_{j,t}\}_{j=1}^{d_\alpha}\}$ are made in batches of R samples $\mathbf{x}_{O_{t,1}}, \ldots, \mathbf{x}_{O_{t,R}}$ (in R-tuple mini-batches). The pseudocode of this OSDL method is presented in the supplementary material.

2. Projections to $\mathcal{A}$ and $\mathcal{D}_j$: For many convex and closed sets $\mathcal{C}$ ($\mathcal{A}$ or $\mathcal{D}_j$), the computation of projection $\Pi_{\mathcal{C}}$ can be done efficiently. For example, for $\mathcal{C} = \mathbb{R}^d_+$, $\Pi_{\mathcal{C}}(\mathbf{x}) = \max(\mathbf{x}, \mathbf{0})$, whereas for the $\mathcal{C} = S_2^d$, $\Pi_{\mathcal{C}}(\mathbf{x}) = \frac{\mathbf{x}}{\max(\|\mathbf{x}\|_2, 1)}$. Similarly, the projection to the ℓ_1-sphere (S_1^d) can be done easily, even when we have extra non-negativity constraints, too. Other famous examples where this projection can be done efficiently include the elastic net constraints, the fused Lasso constraints, and the group ℓ_1-sphere as well. For more details, see, *e.g.*, [7, 30, 19] and references therein. We note that since group norm projections can be computed efficiently, by choosing $\mathcal{D}_j$ to a group-norm sphere, one can obtain a double-structured (group structure on $\boldsymbol{\alpha}$ *and* $\mathbf{D}$) dictionary learning scheme as a special case of our presented OSDL framework.

3. The trick in the representation update (Section 3.1) was that the auxiliary variable $\mathbf{z}$ 'replaced' the Ω term with a quadratic one in $\boldsymbol{\alpha}$. One could use further $g(\boldsymbol{\alpha})$ regularizers augmenting Ω in (8) provided that the corresponding $J(\boldsymbol{\alpha}, \mathbf{z}) + g(\boldsymbol{\alpha})$ cost function (see Eq. (15)) can be efficiently optimized in $\boldsymbol{\alpha} \in \mathcal{A}$.

4. Numerical experiments

We illustrate our OSDL method on inpainting of natural images (Section 4.1) and on structured non-negative matrix factorization of faces (Section 4.2).

4.1. Inpainting of natural images

We studied the following issues on natural images:

1. Is structured dictionary $\mathbf{D}$ beneficial for inpainting of patches of natural images, and how does it compare to the dictionary of classical sparse representation? During learning of $\mathbf{D}$, training samples $\mathbf{x}_i$ were fully observed (*i.e.*, $\boldsymbol{\Delta}_i = \mathbf{I}$).
2. In this image patches inpainting problem, we also studied the case when the training samples $\mathbf{x}_i$ were partially observed ($\boldsymbol{\Delta}_i \neq \mathbf{I}$).
3. We also show results for inpainting of *full images* using a dictionary learned from partially observed ($\boldsymbol{\Delta}_i \neq \mathbf{I}$) patches.

In our numerical experiments we used $\mathcal{D}_i = S_2^{d_x}$ ($\forall i$), $\mathcal{A} = \mathbb{R}^{d_\alpha}$ without additional weighing ($\mathbf{d}^G = \chi_G, \forall G \in \mathcal{G}$, where χ is the indicator function). Group structure $\mathcal{G}$ of vector $\boldsymbol{\alpha}$ was realized on a 16×16 torus ($d_\alpha = 256$) with $|\mathcal{G}| = d_\alpha$ applying $r = 0, 1, 2$, or 3 neighbors to define $\mathcal{G}$. For $r = 0$ ($\mathcal{G} = \{\{1\}, \ldots, \{d_\alpha\}\}$) the classical sparse representation is recovered. Our test database was the ICA natural image database.[1] We chose 12 of the 13 images

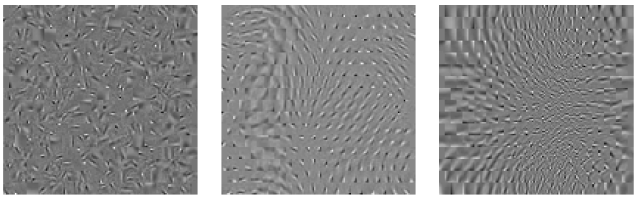
[1] See http://www.cis.hut.fi/projects/ica/data/images/.

(a) (b)

Figure 1: Illustration of the used natural image dataset. (a): 12 images of similar kind were used to select training $\mathbf{X}_{tr}$, validation $\mathbf{X}_{val}$, and test $\mathbf{X}_{test}$ sets. (b): test image used for the illustration of full image inpainting.

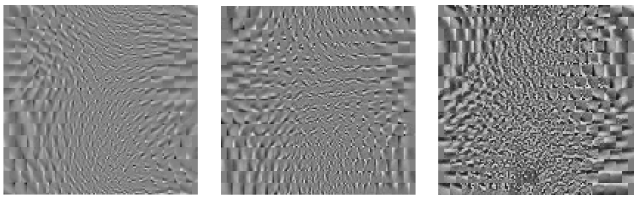
of the dataset to study the first two questions above (see Fig. 1(a)), and used the 13^{th} picture for studying the third question (Fig. 1(b)). For each of the 12 images, we sampled $131072 = 2^{17}$ pieces of 8×8 disjunct image patches randomly (without replacement). This patch set was divided to a training set $\mathbf{X}_{tr}$ made of 65536 pieces, and to a validation ($\mathbf{X}_{val}$) and test ($\mathbf{X}_{test}$) set with set sizes 32768. Each patch was normalized to zero average and unit ℓ_2-norm.

In the **first experiment** $\mathbf{x}_i$s were fully observed ($\boldsymbol{\Delta}_i = \mathbf{I}$) and thus the update of their statistics was precise. This is called the BCD case in the figures. Matrix $\mathbf{D}$ was learned on the set $\mathbf{X}_{tr}$, columns $\mathbf{d}_j$ were initialized by using a uniform distribution on the surface of the ℓ_2-sphere. Pixels of the $\mathbf{x}$ patches in the validation and test sets were removed with probability p_{test}^{val}. For a given noise-free image patch $\mathbf{x}$, let $\mathbf{x}_O$ denote its observed version, where O stands for the indices of the available coordinates. The task was the inpainting of the missing pixels of $\mathbf{x}$ by means of the pixels present ($\mathbf{x}_O$) and by the learned matrix $\mathbf{D}$. After removing the rows of $\mathbf{D}$ corresponding to missing pixels of $\mathbf{x}$, the resulting $\mathbf{D}_O$ and $\mathbf{x}_O$ were used to estimate $\boldsymbol{\alpha}$. The final estimation of $\mathbf{x}$ was $\hat{\mathbf{x}} = \mathbf{D}\boldsymbol{\alpha}$. According to preliminary experiments, learning rate ρ and mini-batch size R were set to 32 and 64 respectively (the estimation was robust as a function of ρ and R). In the updates of $\mathbf{z}$ and $\boldsymbol{\alpha}$ (14) only minor changes were experienced after 2-3 iterations, thus the number of iterations T_α was set to 5. Concerning the other parameters, we used $\eta = 0.5$, and $\kappa \in \{2^{-19}, 2^{-18}, \ldots, 2^{-10}\}$. The ϵ smoothing parameter was 10^{-5}, and the iteration number for the update of $\mathbf{D}$ was $T_D = 5$. Values of p_{test}^{val} were chosen from set $\{0.3, 0.5, 0.7, 0.9\}$, so for the case of $p_{test}^{val} = 0.9$, only 10% of the pixels of $\mathbf{x}$ were observed. For each fixed neighborhood size r and parameter p_{test}^{val}, κ was chosen as the minimum of mean squared error (MSE) using $\mathbf{D}$ trained on patch set $\mathbf{X}_{tr}$ and evaluated on $\mathbf{X}_{val}$. Having found this optimal κ on the validation set, we used its value to compute the MSE on $\mathbf{X}_{test}$. Then we changed the roles of $\mathbf{X}_{val}$ and $\mathbf{X}_{test}$, that is, validated on $\mathbf{X}_{test}$, and tested on $\mathbf{X}_{val}$. This procedure was repeated for four random initializations ($\mathbf{D}_0$) and different corruptions ($\mathbf{X}_{val}$, $\mathbf{X}_{test}$). The



(a) (b) (c)

Figure 2: Illustration of the online learned group-structured $\mathbf{D}$ dictionaries with the BCD technique and MSE closest to the average (see Table 1) and $p_{test}^{val} = 0.7$. (a): $r = 0$, (b): $r = 2$, (c): $r = 3$.



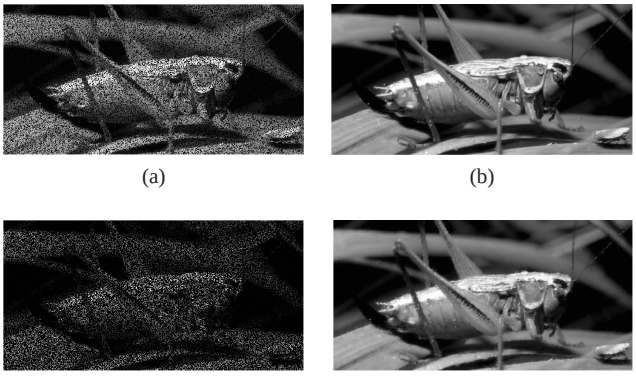
(a) (b) (c)

Figure 3: Illustration of the online learned group-structured $\mathbf{D}$ dictionaries for the BCDA technique with MSE closest to the average (see Table 2) and $p_{test}^{val} = 0.7$. (a): $p_{tr} = 0$, (b): $p_{tr} = 0.1$, (c): $p_{tr} = 0.5$.

average MSE values (multiplied by 100) and their standard deviations for different neighbor sizes r and corruption rates p_{test}^{val} are summarized in Table 1. This table shows that (i) the inpainting error grows with the corruption rate p_{test}^{val}, (ii) compared to sparse representation ($r = 0$) small neighborhood size $r = 1$ gives rise to similar results, $r = 2$ is better and $r = 3$ seems to be the best for all cases with $13 - 19\%$ improvement in precision for MSE. Learned and average quality dictionaries $\mathbf{D}$ can be seen in Fig. 2 ($r = 0$ no structure, $r = 2, 3$ with torus structure). Based on this experiment we can conclude that the structured algorithm gives rise to better results than ordinary sparse representations.

In the **second experiment**, the size of the neighborhood was fixed, set to $r = 3$. We learned dictionary $\mathbf{D}$ on *partially observed* patches ($\mathbf{\Delta}_i \neq \mathbf{I}$). The probability p_{tr} of missing any pixel from the observations in the training set assumed values from the set $\{0, 0.1, 0.3, 0.5, 0.7, 0.9\}$. In this case, we updated $\mathbf{e}$ using the approximation Eq. (23), hence we call this method Approximate-BCD (or BCDA, for short). The other experimental details were identical to the previous case (*i.e.*, when $\mathbf{\Delta}_i = \mathbf{I}$). Results and statistics for MSE are provided for a smaller (0.3) and for a larger (0.7) value of p_{test}^{val} in Table 2 for different probability values p_{tr}. We found that increasing p_{tr} up to $p_{tr} = 0.7$ MSE values grow slowly. Note that we kept the number of samples $\mathbf{x}_i$ at 65536 identical to the previous case ($\mathbf{\Delta}_i = \mathbf{I}$), and thus by increasing p_{tr} the effective number of observations/coordinates decreases. Learned average quality dictionaries $\mathbf{D}$ are shown in Fig. 3 for $p_{test}^{val} = 0.7$. Note that the MSE values are still relatively small for missing pixel probability $p_{tr} = 0.9$ ($100\times$ MSE maximum is about 0.96), thus our proposed method is still efficient in this case. Reconstruction with value 0.92 ($100\times$ MSE) is shown in Fig. 4.

In our **third illustration** we show full image inpainting using dictionary $\mathbf{D}$ learned with $p_{tr} = 0.5$ and using the 13^{th} image ($\mathbf{X}$) shown in Fig. 1(b). We executed inpainting consecutively on all 8×8 patches of image $\mathbf{X}$ and for each pixel of image $\mathbf{X}$, we averaged all estimations $\hat{\mathbf{x}}_i$ from all 8×8 patches that contained the pixel. Results are shown in



(a) (b) (c) (d)

Figure 4: Inpainting illustration using the online learned group-structured $\mathbf{D}$ dictionaries for the BCDA technique with MSE closest to the average (see Table 2) and $p_{tr} = 0.5$. (a): measured, (b): estimated, PSNR = 36 dB. (a)-(b): $p_{test}^{val} = 0.3$. (c)-(d): the same as (a)-(b), but with $p_{test}^{val} = 0.7$, in (d) PSNR = 29 dB.

Fig. 4 for $p_{test}^{val} = 0.3$ and 0.7 values. We also provide the PSNR (peak signal-to-noise ratio) values of our estimations. This measure for vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^d$ (*i.e.*, for vectors formed from the pixels of the image) is defined as

$$PSNR(\mathbf{u}, \mathbf{v}) = 10 \log_{10} \left[\frac{\left(\max(\max_i |u_i|, \max_j |v_j|)\right)^2}{\frac{1}{d} \|\mathbf{u} - \mathbf{v}\|_2^2} \right], \tag{24}$$

where the higher value is the better. Acceptable values in wireless transmission (lossy image and video compression) are around $20 - 25$ dB (30 dB). By means of $\mathbf{D}$ and for missing probability $p_{test}^{val} = 0.3$ we achieved 36 dB PSNR, whereas for missing probability $p_{test}^{val} = 0.7$ we still have 29 dB PSNR, underlining the efficiency of our method.

	$p_{test}^{val} = 0.3$	$p_{test}^{val} = 0.5$	$p_{test}^{val} = 0.7$	$p_{test}^{val} = 0.9$
$r = 0$	0.65 ($\pm$0.002)	0.83 ($\pm$0.003)	1.10 ($\pm$0.002)	1.49 ($\pm$0.006)
$r = 1$	0.60 ($\pm$0.005; +6.78%)	0.85 ($\pm$0.017; −2.25%)	1.10 ($\pm$0.029; +0.27%)	1.45 ($\pm$0.004; +2.96%)
$r = 2$	0.59 ($\pm$0.005; +10.39%)	0.81 ($\pm$0.008; +2.67%)	1.12 ($\pm$0.029; −1.09%)	1.46 ($\pm$0.029; +2.51%)
$r = 3$	**0.56** ($\pm$0.002; **+16.38%**)	**0.71** ($\pm$0.002; **+16.01%**)	**0.93** ($\pm$0.001; **+18.93%**)	**1.31** ($\pm$0.002; **+13.87%**)

Table 1: BCD: 100× the MSE average ($\pm$ std) as a function of neighbors ($r = 0$: sparse representation, no structure) for different p_{test}^{val} corruption rates.

	$p_{tr} = 0$	$p_{tr} = 0.1$	$p_{tr} = 0.3$	$p_{tr} = 0.5$	$p_{tr} = 0.7$	$p_{tr} = 0.9$
$p_{test}^{val} = 0.3$	**0.55** ($\pm$0.003)	0.56 ($\pm$0.001)	0.57 ($\pm$0.003)	0.59 ($\pm$0.001)	0.61 ($\pm$0.002)	0.71 ($\pm$0.007)
$p_{test}^{val} = 0.7$	**0.91** ($\pm$0.002)	0.91 ($\pm$0.002)	0.91 ($\pm$0.002)	0.92 ($\pm$0.003)	0.93 ($\pm$0.002)	0.96 ($\pm$0.003)

Table 2: BCDA ($r = 3$): 100× the MSE average ($\pm$ std) for different for different p_{test}^{val} and p_{tr} corruption rates.

4.2. Online structured non-negative matrix factorization on faces

It has been shown on the CBCL database that dictionary vectors ($\mathbf{d}_i$) of the offline NMF method can be interpreted as face components [17]. However, to the best of our knowledge, there is no existing NMF algorithm as of yet, which could handle general $\mathcal{G}$ group structures in an online fashion. Our OSDL method is able to do that, can also cope with only partially observed inputs, and can be extended with non-convex sparsity-inducing norms. We illustrate our approach on the color FERET[2] dataset: we set $\mathcal{D}_i = S_2^{d_x} \cap \mathbb{R}_+^{d_x}$ ($\forall i$), $\mathcal{A} = \mathbb{R}_+^{d_\alpha}$, $\mathbf{\Delta}_i = \mathbf{I}$ and $\eta = 0.5$. We selected 1736 facial pictures from this dataset. Using affine transformations we positioned the noses and eyes to the same pixel coordinates, reduced the image sizes to 140×120, and set their l_2 norms to be one. These images were the observations for our ODSL method ($\mathbf{x}_i$, $d_x = 49140 = 140 \times 120 \times 3$ minus some masking). The group structure $\mathcal{G}$ was chosen to be hierarchical; we applied a full, 8-level binary tree. Each node with its corresponding descendants formed the sets of $G \in \mathcal{G}$ ($d_\alpha = 255$). According to our experiments, the learned dictionary $\mathbf{D}$ was influenced mostly by the constant κ, and similarly to Section 4.1, it proved to be quite insensitive to the value of the learning factor ρ, and to the size of the mini-batches (R). Fig. 5 shows a few elements from the online estimated structured NMF dictionary (using $\kappa = \frac{1}{2^{10.5}}$, $\rho = 32$, $R = 8$, $\mathbf{d}^G = \chi_G$ ($\forall G \in \mathcal{G}$), $T_\alpha = 5$, $T_D = 5$ and $\varepsilon = 10^{-5}$). We can observe that the proposed algorithm is able to naturally develop and hierarchically organize the elements of the dictionary: towards the leaves the learned filters reveal more and more details. We can also notice that the colors are separated as well. This example demonstrates that our method can be used for large problems where the dimension of the observations is about 50000.

[2] See http://face.nist.gov/colorferet/.

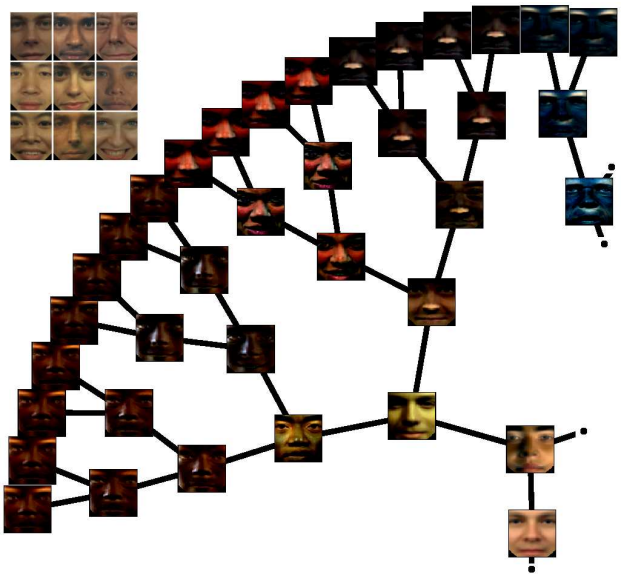


Figure 5: Illustration of the online learned structured NMF dictionary. Upper left corner: training samples.

5. Conclusions

In this paper we proposed a new dictionary learning method, which is (i) online, (ii) enables overlapping group structures on the hidden representation/dictionary, (iii) applies non-convex, sparsity inducing regularization, and (iv) can handle the partially observable case, too. We reduced the optimization problem of dictionary learning to convex subtasks, and using a block-coordinate descent approach and a variational method we derived online update rules for the statistics of the cost of the dictionary. The efficiency of our algorithm was demonstrated by several numerical experiments. We have shown that in the inpainting problem our method can perform better than the traditional sparse methods. As a special case, we have also shown that our ap-

proach can be used for the online structured NMF problem, too, and it is able to hierarchically organize the elements of the dictionary.

One possible extension of our online group-structured dictionary learning framework may touch the nonparametric Bayesian dictionary learning approach [35], recently introduced for the (traditional, unstructured) sparse dictionary learning problem.

Acknowledgments. The research was partly supported by the Department of Energy (grant number DESC0002607). The European Union and the European Social Fund have provided financial support to the project under the grant agreement no. TÁMOP 4.2.1./B-09/1/KMR-2010-0003.

References

[1] M. Aharon, M. Elad, and A. Bruckstein. K-SVD: An algorithm for designing of overcomplete dictionaries for sparse representation. *IEEE Trans. Signal Process.*, 54(11), 2006.

[2] A. Argyriou, T. Evgeniou, and M. Pontil. Convex multi-task feature learning. *Mach. Learn.*, 73(3):243–272, 2008.

[3] L. Balzano, R. Nowak, and B. Recht. Online identification and tracking of subspaces from highly incomplete information. In *48th Annual Allerton Conference*, 2010.

[4] S. Bengio, F. Pereira, Y. Singer, and D. Strelow. Group sparse coding. In *NIPS 2009*, pages 82–89.

[5] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific Belmont, 1999.

[6] P. L. Combettes and J.-C. Pesquet. *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, chapter Proximal splitting methods in signal processing. Springer, 2010.

[7] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. Efficient projections onto the l_1-ball for learning in high dimensions. In *ICML 2008*, pages 272–279.

[8] P. Hoyer. Non-negative matrix factorization with sparseness constraints. *J. Mach. Learn. Res.*, 5:1457–1469, 2004.

[9] J. Huang and T. Zhang. The benefit of group sparsity. *Ann. Stat.*, 38(4):1978–2004, 2010.

[10] A. Hyvärinen, J. Karhunen, and E. Oja. *Independent Component Analysis*. John Wiley & Sons, 2001.

[11] L. Jacob, G. Obozinski, and J.-P. Vert. Group Lasso with overlap and graph Lasso. In *ICML 2009*, pages 433–440.

[12] R. Jenatton, J. Mairal, G. Obozinski, and F. Bach. Proximal methods for sparse hierarchical dictionary learning. In *ICML 2010*, pages 487–494.

[13] R. Jenatton, G. Obozinski, and F. Bach. Structured sparse principal component analysis. *J. Mach. Learn. Res.:W&CP*, 9:366–373.

[14] K. Kavukcuoglu, M. Ranzato, R. Fergus, and Y. LeCun. Learning invariant features through topographic filter maps. In *CVPR 2009*, pages 1605–1612.

[15] S. Kim and E. P. Xing. Tree-guided group Lasso for multi-task regression with structured sparsity. In *ICML 2010*, pages 543–550.

[16] C. L. Lawson and R. J. Hanson. *Solving Least Squares Problems*. Prentice-Hall, 1974.

[17] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *NIPS 2000*, pages 556–562.

[18] J. Liu and J. Ye. Moreau-Yosida regularization for grouped tree structure learning. In *NIPS 2010*, pages 1459–1467.

[19] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online learning for matrix factorization and sparse coding. *J. Mach. Learn. Res.*, 11:10–60, 2010.

[20] J. Mairal, R. Jenatton, G. Obozinski, and F. Bach. Network flow algorithms for structured sparsity. In *NIPS 2010*, pages 1558–1566.

[21] S. Mosci, L. Rosasco, M. Santoro, A. Verri, and S. Villa. Solving structured sparsity regularization with proximal methods. In *ECML 2010*, pages 418–433.

[22] G. Obozinski, B. Taskar, and M. I. Jordan. Joint covariate selection and joint subspace selection for multiple classification problems. *Stat. Comput.*, 20:231–252, 2010.

[23] B. A. Olshausen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:607–609, 1996.

[24] L. Rosasco, S. Mosci, S. Villa, and A. Verri. A primal-dual algorithm for group sparse regularization with overlapping groups. In *NIPS 2010*, pages 2604–2612.

[25] K. Rosenblum, L. Zelnik-Manor, and Y. Eldar. Dictionary optimization for block-sparse representations. In *AAAI Fall 2010 Symposium on Manifold Learning*.

[26] M. Schmidt and K. Murphy. Convex structure learning in log-linear models: Beyond pairwise potentials. *J. Mach. Learn. Res.:W&CP*, 9:709–716, 2010.

[27] B. Szatmáry, B. Póczos, J. Eggert, E. Körner, and A. Lőrincz. Non-negative matrix factorization extended by sparse code shrinkage and by weight sparsification. In *ECAI 2002*, pages 503–507.

[28] R. Tibshirani. Regression shrinkage and selection via the Lasso. *J. Roy. Stat. Soc. B. Met.*, 58(1):267–288, 1996.

[29] J. A. Tropp and S. J. Wright. Computational methods for sparse solution of linear inverse problems. *Proc. of the IEEE special issue on Applications of sparse representation and compressive sensing*, 98(6):948–958, 2010.

[30] E. van den Berg, M. Schmidt, M. P. Friedlander, and K. Murphy. Group sparsity via linear-time projection. Technical report, 2008.

[31] D. M. Witten, R. Tibshirani, and T. Hastie. A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis. *Biostatistics*, 10(3):515–534, 2009.

[32] M. Yaghoobi, T. Blumensath, and M. Davies. Dictionary learning for sparse approximations with the majorization method. *IEEE Trans. Signal Process.*, 57(6):2178–2191, 2009.

[33] M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *J. Roy. Stat. Soc. B Met.*, 68, 2006.

[34] P. Zhao, G. Rocha, and B. Yu. The composite absolute penalties family for grouped and hierarchical variable selection. *Ann. Stat.*, 37(6A):3468–3497, 2009.

[35] M. Zhou, H. Chen, J. Paisley, L. Ren, G. Sapiro, and L. Carin. Non-parametric Bayesian dictionary learning for sparse image representations. In *NIPS 2009*, pages 2295–2303.

[36] H. Zou, T. Hastie, and R. Tibshirani. Sparse principal component analysis. *J. Comput. Graph. Stat.*, 15(2):265–286, 2006.

Online Group-Structured Dictionary Learning
Supplementary Material

Zoltán Szabó[1] Barnabás Póczos[2] András Lőrincz[1]
[1]School of Computer Science, Eötvös Loránd University,
Pázmány Péter sétány 1/C, H-1117 Budapest, Hungary
web: http://nipg.inf.elte.hu
email: szzoli@cs.elte.hu, andras.lorincz@elte.hu
[2]School of Computer Science, Carnegie Mellon University,
5000 Forbes Ave, 15213, Pittsburgh, PA, USA
web: http://www.autonlab.org
email: bapoczos@cs.cmu.edu

In this note we will derive the update equations for the statistics describing the minimum point of $\hat{f}_t$ (Section 2). During the derivation we will need an auxiliary lemma concerning the behavior of certain matrix series. We will introduce this lemma in Section 1. The pseudocode of our OSDL method is provided in Section 3.

1 The forgetting factor in matrix recursions

Let $\mathbf{N}_t \in \mathbb{R}^{L_1 \times L_2}$ $(t = 1, 2, \ldots)$ be a given matrix series, and let $\gamma_t = \left(1 - \frac{1}{t}\right)^\rho$, $\rho \geq 0$. Define the following matrix series with the help of these quantities:

$$\mathbf{M}_t = \gamma_t \mathbf{M}_{t-1} + \mathbf{N}_t \in \mathbb{R}^{L_1 \times L_2} \quad (t = 1, 2, \ldots), \tag{1}$$

$$\mathbf{M}'_t = \sum_{i=1}^{t} \left(\frac{i}{t}\right)^\rho \mathbf{N}_i \in \mathbb{R}^{L_1 \times L_2} \quad (t = 1, 2, \ldots). \tag{2}$$

Lemma 1. *If $\rho = 0$, then $\mathbf{M}_t = \mathbf{M}_0 + \mathbf{M}'_t$ $(\forall t \geq 1)$. When $\rho > 0$, then $\mathbf{M}_t = \mathbf{M}'_t$ $(\forall t \geq 1)$.*

Proof.

1. Case $\rho = 0$: Since $\gamma_t = 1$ $(\forall t \geq 1)$, thus $\mathbf{M}_t = \mathbf{M}_0 + \sum_{i=1}^{t} \mathbf{N}_i$. We also have that $\left(\frac{i}{t}\right)^0 = 1$ $(\forall i \geq 1)$, and therefore $\mathbf{M}'_t = \sum_{i=1}^{t} \mathbf{N}_i$, which completes the proof.
2. Case $\rho > 0$: The proof proceeds by induction.
   - $t = 1$: In this case $\gamma_1 = 0$, $\mathbf{M}_1 = 0 \times \mathbf{M}_0 + \mathbf{N}_1 = \mathbf{N}_1$ and $\mathbf{M}'_1 = \mathbf{N}_1$, which proves that $\mathbf{M}_1 = \mathbf{M}'_1$.
   - $t > 1$: Using the definitions of $\mathbf{M}_t$ and $\mathbf{M}'_t$, and exploiting the fact that $\mathbf{M}_{t-1} = \mathbf{M}'_{t-1}$ by induction, after some calculation we have that:

$$\mathbf{M}_t = \gamma_t \mathbf{M}_{t-1} + \mathbf{N}_t = \left(1 - \frac{1}{t}\right)^\rho \left[\sum_{i=1}^{t-1} \left(\frac{i}{t-1}\right)^\rho \mathbf{N}_i\right] + \mathbf{N}_t \tag{3}$$

$$= \left(\frac{t-1}{t}\right)^\rho \left[\sum_{i=1}^{t-1} \left(\frac{i}{t-1}\right)^\rho \mathbf{N}_i\right] + \left(\frac{t}{t}\right)^\rho \mathbf{N}_t = \sum_{i=1}^{t} \left(\frac{i}{t}\right)^\rho \mathbf{N}_i = \mathbf{M}'_t. \tag{4}$$

□

2 Online update equations for the minimum point of $\hat{f}_t$

Our goals are (i) to find the minimum of

$$\hat{f}_t(\mathbf{D}) = \frac{1}{\sum_{j=1}^{t}(j/t)^{\rho}} \sum_{i=1}^{t} \left(\frac{i}{t}\right)^{\rho} \left[\frac{1}{2}\left\|\mathbf{x}_{O_i} - \mathbf{D}_{O_i}\boldsymbol{\alpha}_i\right\|_2^2 + \kappa\Omega(\boldsymbol{\alpha}_i)\right] \tag{5}$$

in $\mathbf{d}_j$ while the other column vectors of $\mathbf{D}$ ($\mathbf{d}_i$ $(i \neq j)$) are being fixed, and (ii) to derive online update rules for the statistics of $\hat{f}_t$ describing this minimum point. $\hat{f}_t$ is quadratic in $\mathbf{d}_j$, hence in order to find its minimum, we simply have to solve the following equation:

$$\frac{\partial \hat{f}_t}{\partial \mathbf{d}_j}(\mathbf{u}_j) = \mathbf{0}, \tag{6}$$

where $\mathbf{u}_j$ denotes the optimal solution. We can treat the Ω, and the $\frac{1}{\sum_{j=1}^{t}(j/t)^{\rho}}$ terms in (5) as constants, since they do not depend on $\mathbf{d}_j$. Let $\mathbf{D}_{-j}$ denote the slightly modified version of matrix $\mathbf{D}$; its j^{th} column is set to zero. Similarly, let $\boldsymbol{\alpha}_{i,-j}$ denote the vector $\boldsymbol{\alpha}_i$ where its j^{th} coordinate is set to zero. Now, we have that

$$\mathbf{0} = \frac{\partial \hat{f}_t}{\partial \mathbf{d}_j} = \frac{\partial}{\partial \mathbf{d}_j}\left[\sum_{i=1}^{t}\left(\frac{i}{t}\right)^{\rho}\left\|\boldsymbol{\Delta}_i(\mathbf{x}_i - \mathbf{D}\boldsymbol{\alpha}_i)\right\|_2^2\right] \tag{7}$$

$$= \frac{\partial}{\partial \mathbf{d}_j}\left[\sum_{i=1}^{t}\left(\frac{i}{t}\right)^{\rho}\left\|\boldsymbol{\Delta}_i[(\mathbf{x}_i - \mathbf{D}_{-j}\boldsymbol{\alpha}_{i,-j}) - \mathbf{d}_j\alpha_{i,j}]\right\|_2^2\right] \tag{8}$$

$$= \frac{\partial}{\partial \mathbf{d}_j}\left[\sum_{i=1}^{t}\left(\frac{i}{t}\right)^{\rho}\left\|(\boldsymbol{\Delta}_i\alpha_{i,j})\mathbf{d}_j - \boldsymbol{\Delta}_i(\mathbf{x}_i - \mathbf{D}_{-j}\boldsymbol{\alpha}_{i,-j})\right\|_2^2\right] \tag{9}$$

$$= 2\sum_{i=1}^{t}\left(\frac{i}{t}\right)^{\rho}\boldsymbol{\Delta}_i\alpha_{i,j}[(\boldsymbol{\Delta}_i\alpha_{i,j})\mathbf{d}_j - \boldsymbol{\Delta}_i(\mathbf{x}_i - \mathbf{D}_{-j}\boldsymbol{\alpha}_{i,-j})] \tag{10}$$

$$= 2\sum_{i=1}^{t}\left(\frac{i}{t}\right)^{\rho}\boldsymbol{\Delta}_i\alpha_{i,j}^2\mathbf{d}_j - 2\sum_{i=1}^{t}\left(\frac{i}{t}\right)^{\rho}\boldsymbol{\Delta}_i\alpha_{i,j}(\mathbf{x}_i - \mathbf{D}_{-j}\boldsymbol{\alpha}_{i,-j}), \tag{11}$$

where we used the facts that

$$\mathbf{x}_{O_i} - \mathbf{D}_{O_i}\boldsymbol{\alpha}_i = \boldsymbol{\Delta}_i(\mathbf{x}_i - \mathbf{D}\boldsymbol{\alpha}_i), \tag{12}$$

$$\frac{\partial\left\|\mathbf{A}\mathbf{y} - \mathbf{b}\right\|_2^2}{\partial \mathbf{y}} = 2\mathbf{A}^T(\mathbf{A}\mathbf{y} - \mathbf{b}), \tag{13}$$

$$\boldsymbol{\Delta}_i = \boldsymbol{\Delta}_i^T = (\boldsymbol{\Delta}_i)^2. \tag{14}$$

After rearranging the terms in (11), we have that

$$\left(\sum_{i=1}^{t}\left(\frac{i}{t}\right)^{\rho}\boldsymbol{\Delta}_i\alpha_{i,j}^2\right)\mathbf{u}_j = \sum_{i=1}^{t}\left(\frac{i}{t}\right)^{\rho}\boldsymbol{\Delta}_i\alpha_{i,j}(\mathbf{x}_i - \mathbf{D}_{-j}\boldsymbol{\alpha}_{i,-j}) \tag{15}$$

$$= \sum_{i=1}^{t}\left(\frac{i}{t}\right)^{\rho}\boldsymbol{\Delta}_i\mathbf{x}_i\alpha_{i,j} - \sum_{i=1}^{t}\left(\frac{i}{t}\right)^{\rho}\boldsymbol{\Delta}_i\mathbf{D}_{-j}\boldsymbol{\alpha}_{i,-j}\alpha_{i,j} \tag{16}$$

$$= \sum_{i=1}^{t}\left(\frac{i}{t}\right)^{\rho}\boldsymbol{\Delta}_i\mathbf{x}_i\alpha_{i,j} - \sum_{i=1}^{t}\left(\frac{i}{t}\right)^{\rho}\boldsymbol{\Delta}_i(\mathbf{D}_{-j}\boldsymbol{\alpha}_{i,-j} + \mathbf{d}_j\alpha_{i,j} - \mathbf{d}_j\alpha_{i,j})\alpha_{i,j} \tag{17}$$

$$= \sum_{i=1}^{t}\left(\frac{i}{t}\right)^{\rho}\boldsymbol{\Delta}_i\mathbf{x}_i\alpha_{i,j} - \sum_{i=1}^{t}\left(\frac{i}{t}\right)^{\rho}\boldsymbol{\Delta}_i\mathbf{D}\boldsymbol{\alpha}_i\alpha_{i,j} + \left(\sum_{i=1}^{t}\left(\frac{i}{t}\right)^{\rho}\boldsymbol{\Delta}_i\alpha_{i,j}^2\right)\mathbf{d}_j. \tag{18}$$

We note that (16) is a system of linear equations, and its solution $\mathbf{u}_j$ does not depend on $\mathbf{d}_j$. We have introduced the '$\mathbf{d}_j\alpha_{ij} - \mathbf{d}_j\alpha_{ij}$' term only for one purpose; it can help us with deriving the recursive updates for $\mathbf{u}_j$ in a simple form. Define the following quantities

$$\mathbf{C}_{j,t} = \sum_{i=1}^{t} \left(\frac{i}{t}\right)^{\rho} \boldsymbol{\Delta}_i \alpha_{i,j}^2 \in \mathbb{R}^{d_x \times d_x} \quad (j = 1, \ldots, d_\alpha), \tag{19}$$

$$\mathbf{B}_t = \sum_{i=1}^{t} \left(\frac{i}{t}\right)^{\rho} \boldsymbol{\Delta}_i \mathbf{x}_i \boldsymbol{\alpha}_i^T = [\mathbf{b}_{1,t}, \ldots, \mathbf{b}_{d_\alpha,t}] \in \mathbb{R}^{d_x \times d_\alpha}, \tag{20}$$

$$\mathbf{e}_{j,t} = \sum_{i=1}^{t} \left(\frac{i}{t}\right)^{\rho} \boldsymbol{\Delta}_i \mathbf{D}\boldsymbol{\alpha}_i \alpha_{i,j} \in \mathbb{R}^{d_x} \quad (j = 1, \ldots, d_\alpha). \tag{21}$$

Here (i) $\mathbf{C}_{j,t}$s are diagonal matrices and (ii) the update rule of $\mathbf{B}_t$ contains the quantity $\boldsymbol{\Delta}_i \mathbf{x}_i$, which is $\mathbf{x}_{O_i}$ extended by zeros at the non-observable ($\{1, \ldots, d_x\} \setminus O_i$) coordinates. By using these notations and (18), we obtain that $\mathbf{u}_j$ satisfies the following equation:

$$\mathbf{C}_{j,t}\mathbf{u}_j = \mathbf{b}_{j,t} - \mathbf{e}_{j,t} + \mathbf{C}_{j,t}\mathbf{d}_j. \tag{22}$$

Now, according to Lemma 1, we can see that (i) when $\rho = 0$ and $\mathbf{C}_{j,0} = \mathbf{0}$, $\mathbf{B}_0 = \mathbf{0}$, or (ii) $\rho > 0$ and $\mathbf{C}_{j,0}$, $\mathbf{B}_0$ are arbitrary, then the $\mathbf{C}_{j,t}$ and $\mathbf{B}_t$ quantities can be updated online with the following recursions:

$$\mathbf{C}_{j,t} = \gamma_t \mathbf{C}_{j,t-1} + \boldsymbol{\Delta}_t \alpha_{t,j}^2, \tag{23}$$

$$\mathbf{B}_t = \gamma_t \mathbf{B}_{t-1} + \boldsymbol{\Delta}_t \mathbf{x}_t \boldsymbol{\alpha}_t^T, \tag{24}$$

where $\gamma_t = \left(1 - \frac{1}{t}\right)^{\rho}$. We use the following online approximation for $\mathbf{e}_{j,t}$:

$$\mathbf{e}_{j,t} = \gamma_t \mathbf{e}_{j,t-1} + \boldsymbol{\Delta}_t \mathbf{D}\boldsymbol{\alpha}_t \alpha_{t,j}, \tag{25}$$

with initialization $\mathbf{e}_{j,0} = \mathbf{0}$ $(\forall j)$, and $\mathbf{D}$ is the *actual* estimation for the dictionary. This choice seems to be efficient according to our numerical experiences.

Note. *In the fully observable special case (*i.e., *when* $\boldsymbol{\Delta}_i = \mathbf{I}$, $\forall i$*) the* (19)-(21) *equations have the following simpler form:*

$$\mathbf{C}_{j,t} = \mathbf{I} \sum_{i=1}^{t} \left(\frac{i}{t}\right)^{\rho} \alpha_{i,j}^2, \tag{26}$$

$$\mathbf{B}_t = \sum_{i=1}^{t} \left(\frac{i}{t}\right)^{\rho} \mathbf{x}_i \boldsymbol{\alpha}_i^T, \tag{27}$$

$$\mathbf{e}_{j,t} = \sum_{i=1}^{t} \left(\frac{i}{t}\right)^{\rho} \mathbf{D}\boldsymbol{\alpha}_i \alpha_{i,j} = \mathbf{D} \sum_{i=1}^{t} \left(\frac{i}{t}\right)^{\rho} \boldsymbol{\alpha}_i \alpha_{i,j}. \tag{28}$$

Define the following term:

$$\mathbf{A}_t = \sum_{i=1}^{t} \left(\frac{i}{t}\right)^{\rho} \boldsymbol{\alpha}_i \boldsymbol{\alpha}_i^T \in \mathbb{R}^{d_\alpha \times d_\alpha}, \tag{29}$$

and let $\mathbf{a}_{j,t}$ *denote the* j^{th} *column of* $\mathbf{A}_t$*. Now,* (28) *can be rewritten as*

$$\mathbf{e}_{j,t} = \mathbf{D}\mathbf{a}_{j,t}, \tag{30}$$

and thus (22) *has the following simpler form:*

$$(\mathbf{A}_t)_{j,j}\mathbf{u}_j = \mathbf{b}_{j,t} - \mathbf{D}\mathbf{a}_{j,t} + (\mathbf{A}_t)_{j,j}\mathbf{d}_j. \tag{31}$$

Algorithm (Online Group-Structured Dictionary Learning)

Input of the algorithm

$\mathbf{x}_{t,r} \sim p(\mathbf{x})$, (observation: $\mathbf{x}_{O_{t,r}}$, observed positions: $O_{t,r}$), $\mathbf{D}_0$ (initial dictionary),
T (number of mini-batches), R (size of the mini-batches), $\mathcal{G}$ (group structure),
ρ (≥ 0 forgetting factor), κ (> 0 tradeoff-), η ($\in (0, 1]$ regularization constant),
$\{\mathbf{d}^G\}_{G\in\mathcal{G}}$ ($\geq \mathbf{0}$, weights), $\boldsymbol{\mathcal{A}}$ (constraint set for $\boldsymbol{\alpha}$), $\boldsymbol{\mathcal{D}} = \times_{i=1}^{d_\alpha} \boldsymbol{\mathcal{D}}_i$ (constraint set for $\mathbf{D}$),
inner loop constants: ϵ (smoothing), T_α, T_D (number of iterations).

Initialization

$\mathbf{C}_{j,0} = \mathbf{0} \in \mathbb{R}^{d_x}$ $(j = 1, \ldots, d_\alpha)$, $\mathbf{B}_0 = \mathbf{0} \in \mathbb{R}^{d_x \times d_\alpha}$, $\mathbf{e}_{j,0} = \mathbf{0} \in \mathbb{R}^{d_x}$ $(j = 1, \ldots, d_\alpha)$.

Optimization

for $t = 1 : T$

Draw samples for mini-batch from $p(\mathbf{x})$: $\{\mathbf{x}_{O_{t,1}}, \ldots, \mathbf{x}_{O_{t,R}}\}$.
Compute the $\{\boldsymbol{\alpha}_{t,1} \ldots, \boldsymbol{\alpha}_{t,R}\}$ representations:
$\boldsymbol{\alpha}_{t,r}$=Representation($\mathbf{x}_{O_{t,r}}, (\mathbf{D}_{t-1})_{O_{t,r}}, \mathcal{G}, \{\mathbf{d}^G\}_{G\in\mathcal{G}}, \kappa, \eta, \boldsymbol{\mathcal{A}}, \epsilon, T_\alpha$), $r = 1, \ldots, R$.
Update the statistics of the cost function:
$\gamma_t = \left(1 - \frac{1}{t}\right)^\rho$,
$\mathbf{C}_{j,t} = \gamma_t \mathbf{C}_{j,t-1} + \frac{1}{R}\sum_{r=1}^{R} \boldsymbol{\Delta}_{t,r} \alpha_{t,r,j}^2$, $j = 1, \ldots, d_\alpha$,
$\mathbf{B}_t = \gamma_t \mathbf{B}_{t-1} + \frac{1}{R}\sum_{r=1}^{R} \boldsymbol{\Delta}_{t,r} \mathbf{x}_{t,r} \boldsymbol{\alpha}_{t,r}^T$,
$\mathbf{e}_{j,t} = \gamma_t \mathbf{e}_{j,t-1}$, $j = 1, \ldots, d_\alpha$. %(part-1)
Compute $\mathbf{D}_t$ using BCD:
$\mathbf{D}_t$=Dictionary($\{\mathbf{C}_{j,t}\}_{j=1}^{d_\alpha}, \mathbf{B}_t, \{\mathbf{e}_{j,t}\}_{j=1}^{d_\alpha}, \boldsymbol{\mathcal{D}}, T_D, \{O_{t,r}\}_{r=1}^{R}, \{\boldsymbol{\alpha}_{t,r}\}_{r=1}^{R}$).
Finish the update of $\{\mathbf{e}_{j,t}\}_{j=1}^{d_\alpha}$-s: %(part-2)
$\mathbf{e}_{j,t} = \mathbf{e}_{j,t} + \frac{1}{R}\sum_{r=1}^{R} \boldsymbol{\Delta}_{t,r} \mathbf{D}_t \boldsymbol{\alpha}_{t,r} \alpha_{t,r,j}$, $j = 1, \ldots, d_\alpha$.

end

Output of the algorithm

$\mathbf{D}_T$ (learned dictionary).

Table 1: Pseudocode: Online Group-Structured Dictionary Learning.

Here $(\cdot)_{j,j}$ stands for the $(j,j)^{th}$ entry of its argument. By applying again Lemma 1 for (29)*, we have that when (i) $\rho = 0$ and $\mathbf{A}_0 = \mathbf{0}$, or (ii) $\rho > 0$ and $\mathbf{A}_0$ is arbitrary, then $\mathbf{A}_t$ can be updated online with the following recursion:*

$$\mathbf{A}_t = \gamma_t \mathbf{A}_{t-1} + \boldsymbol{\alpha}_t \boldsymbol{\alpha}_t^T. \tag{32}$$

We also note that in the fully observable case (24) *reduces to*

$$\mathbf{B}_t = \gamma_t \mathbf{B}_{t-1} + \mathbf{x}_t \boldsymbol{\alpha}_t^T, \tag{33}$$

and thus [1] is indeed a special case of our model:

- *We calculate $\mathbf{u}_j$ by* (31)*.*
- *To optimize $\hat{f}_t$, it is enough to keep track of $\mathbf{A}_t$ and $\mathbf{B}_t$ instead of $\{\mathbf{C}_{j,t}\}_{j=1}^{d_\alpha}, \mathbf{B}_t, \{\mathbf{e}_{j,t}\}_{j=1}^{d_\alpha}$.*
- *The quantities $\mathbf{A}_t$ and $\mathbf{B}_t$ can be updated online by* (32) *and* (33)*.*

3 Pseudocode

The pseudocode of the OSDL method with mini-batches is presented in Table 1-3. Table 2 calculates the representation for a fixed dictionary, and Table 3 learns the dictionary using fixed representations. Table 1 invokes both of these subroutines.

Algorithm (Representation)

Input of the algorithm

$\mathbf{x}$ (observation), $\mathbf{D} = [\mathbf{d}_1, \ldots, \mathbf{d}_{d_\alpha}]$ (dictionary),
$\mathcal{G}$ (group structure), $\{\mathbf{d}^G\}_{G\in\mathcal{G}}$ (weights), κ (tradeoff-), η (regularization constant),
$\mathcal{A}$ (constraint set for $\boldsymbol{\alpha}$), ϵ (smoothing), T_α (number of iterations).

Initialization

$\boldsymbol{\alpha} \in \mathbb{R}^{d_\alpha}$.

Optimization

for $t = 1 : T_\alpha$

Compute $\mathbf{z}$: $z^G = \max\left(\left\|\mathbf{d}^G \circ \boldsymbol{\alpha}\right\|_2^{2-\eta} \left\|\left(\left\|\mathbf{d}^G \circ \boldsymbol{\alpha}\right\|_2\right)_{G\in\mathcal{G}}\right\|_\eta^{\eta-1}, \epsilon\right), G \in \mathcal{G}.$

Compute $\boldsymbol{\alpha}$:

compute $\boldsymbol{\zeta}$: $\zeta_j = \sum_{G\in\mathcal{G}, G\ni j} \frac{\left(d_j^G\right)^2}{z^G}, j = 1, \ldots, d_\alpha,$

$\boldsymbol{\alpha} = \underset{\boldsymbol{\alpha}\in\mathcal{A}}{\operatorname{argmin}} \left[\|\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}\|_2^2 + \kappa\boldsymbol{\alpha}^T diag(\boldsymbol{\zeta})\boldsymbol{\alpha}\right].$

end

Output of the algorithm

$\boldsymbol{\alpha}$ (estimated representation).

Table 2: Pseudocode for *representation* estimation using fixed dictionary.

Algorithm (Dictionary)

Input of the algorithm

$\{\mathbf{C}_j\}_{j=1}^{d_\alpha}, \mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_{d_\alpha}], \{\mathbf{e}_j\}_{j=1}^{d_\alpha}$ (statistics of the cost function),
$\mathcal{D} = \times_{i=1}^{d_\alpha} \mathcal{D}_i$ (constraint set for $\mathbf{D}$), T_D (number of $\mathbf{D}$ iterations),
$\{O_r\}_{r=1}^R$(equivalent to $\{\boldsymbol{\Delta}_r\}_{r=1}^R$), $\{\boldsymbol{\alpha}_r\}_{r=1}^R$ (observed positions, estimated representations).

Initialization

$\mathbf{D} = [\mathbf{d}_1, \ldots, \mathbf{d}_{d_\alpha}]$.

Optimization

for $t = 1 : T_D$

for $j = 1 : d_\alpha$ %update the j^{th} column of $\mathbf{D}$

Compute '$\{\mathbf{e}_j\}_{j=1}^{d_\alpha}$'-s:

$\mathbf{e}_j^{temp} = \mathbf{e}_j + \frac{1}{R}\sum_{r=1}^R \boldsymbol{\Delta}_r \mathbf{D}\boldsymbol{\alpha}_r \alpha_{r,j}.$

Compute $\mathbf{u}_j$ solving the linear equation system:

$\mathbf{C}_j\mathbf{u}_j = \mathbf{b}_j - \mathbf{e}_j^{temp} + \mathbf{C}_j\mathbf{d}_j.$

Project $\mathbf{u}_j$ to the constraint set:

$\mathbf{d}_j = \Pi_{\mathcal{D}_j}(\mathbf{u}_j).$

end

end

Output of the algorithm

$\mathbf{D}$ (estimated dictionary).

Table 3: Pseudocode for *dictionary* estimation using fixed representations.

References

[1] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research*, 11:10–60, 2010.