Structured Outlier Models for Robust Dictionary Learning

Pedro A. Forero, Scott Shafer, and Josh Harguess
SPAWAR Systems Center Pacific, San Diego, CA 92152, United States
Email: {pedro.a.forero;scott.shafer;joshua.harguess}@navy.mil

***Abstract*—Robust dictionary learning algorithms seek to learn a dictionary while being robust to the presence of outliers in the training set. Often, the elements of the training set have an underlying structure due to, for example, their spatial relation or their similarity. When outliers are present as elements of the training set, they often inherit the underlying structure of the training set. This work capitalizes on such structure, encoded as an undirected graph connecting elements of the training set, and on sparsity-aware outlier modeling tools to develop robust dictionary learning algorithms. Not only do these algorithms yield a robust dictionary, but they also identify the outliers in the training set. Computationally efficient algorithms based on block coordinate descent and proximal gradient methods are developed.**

***Index Terms*—Dictionary learning, proximal gradient algorithms, Laplacian regularization**

I. Introduction

Succinct representations of high-dimensional data are paramount to promptly extract actionable information and reduce the storage and communication requirements related to sharing and archiving data. A fundamental observation, first made in the image processing community, is that a more efficient signal representation is possible if one uses an overcomplete dictionary *learned* from the signals themselves rather than a fixed basis [11], [15]. Given a large number of signal samples, the dictionary learning (DL) problem aims to construct a dictionary that can be used to *efficiently* represent the data as a linear combination of its columns, also known as *atoms*. In this context, a more efficient data representation is one that uses a smaller number of atoms to achieve the desired signal reconstruction quality [4].

Usually, the dictionary is learned by fitting the reconstructed signal to the corresponding sample signal via a regularized least-squares (LS) criterion [4]. Here, the regularization is used to encourage a sparse structure in the vectors of regression coefficients. Naturally, the samples used for learning the dictionary are assumed to adhere to the *nominal* process that caused the signal. However, in many cases it is not possible to screen all data samples to guarantee that no datum behaves as an *outlier*, that is, deviates significantly from the remaining data [17]. Moreover, outliers alone may be more informative than the remaining data, in which case one would like to quickly identify them and possibly remove them from the dataset. It is known that the LS criterion is sensitive to the presence of outliers [17]. DL problem formulations based on the LS cost inherit such sensitivity and are, thus, sensitive to outliers, which can bias the dictionary estimate and reduce the efficiency of DL as data compression tool.

This work was funded by the Naval Innovative Science and Engineering Program at Space and Naval Warfare Systems Center Pacific.

Robust DL approaches leveraging tools developed in the area of robust statistics have been recently proposed [12], [16]. Instead of using the LS criterion as an error fitting terms, they use criteria such as the ℓ_1-error and the Huber loss to induce robustness to outliers. Unfortunately, some of these tools do not lend themselves for constructing efficient solvers, and in many cases they do not directly identify the outliers. Robust analysis methods for data capitalizing on modern signal processing tools have been recently studied by the signal processing community [9], [5]. These approaches leverage a data model that captures the presence of outliers, and the fact that by definition outliers are rare, to develop efficient tools for identifiying and tapering the effects of outliers. As outlined by Mateos et al. [10], sparsity-aware outlier models and the corresponding solvers associated to them can be extended to develop robust DL algorithms as was done by Chen et al. [3].

When there is a structural relationship among elements of the training set, outliers can inherit such a structure. For instance, when training a dictionary using image patches contaminated by outliers, the spatial structure of the patches (size and location) induces a similar structure in the outliers. None of the aforementioned robust DL approaches leverages outlier structure to enhance their robustness and ability to identify outliers. Our contribution in this work is to develop a robust DL framework that capitalizes on sparsity-aware modeling tools and the predefined structure among outliers. The specific outlier structure is defined via an undirected graph subsuming the *similarity* among elements of the training set. This information is introduced in the robust DL problem via a Laplacian regularization term that now couples all outlier vector representations. A block coordinate descent (BCD) algorithm coupled with proximal gradient (PG) methods is developed for training the dictionary and identifying the outliers.

II. Preliminaries

Consider a training set of M-dimensional real-valued vectors $\mathcal{Y} := \{\mathbf{y}_n\}_{n=1}^N$ with cardinality $N := |\mathcal{Y}|$. Let $\mathbf{D} := [\mathbf{d}_1, \ldots, \mathbf{d}_Q] \in \mathbb{R}^{M \times Q}$ denote a dictionary with Q *atoms*, where an atom corresponds to a column of $\mathbf{D}$. It is assumed that each $\mathbf{y}_n$ can be represented as a linear combination of *atoms* from $\mathbf{D}$. Whenever $Q > M$, $\mathbf{D}$ is called *overcomplete*.

If $\mathbf{D}$ is overcomplete, vectors $\{\mathbf{d}_q\}_{q=1}^Q$ are linearly dependent. Then, the vector of expansion coefficients $\mathbf{s}_n$ in $\mathbf{y}_n = \mathbf{D}\mathbf{s}_n$ is not unique for any n. By requiring each $\mathbf{s}_n$ to be *sparse*, a balance can be stricken between the quality of the approximation and the stability of the representation of each $\mathbf{y}_n$ [4].

Given $\mathcal{Y}$, the goal of the *classical* DL problem is to find both $\mathbf{D}$ and $\{\mathbf{s}_n\}_{n=1}^N$ as the minimizers of

$$\min_{\mathbf{D}\in\mathcal{D},\{\mathbf{s}_n\}_{n=1}^N} \frac{1}{2}\sum_{n=1}^N \|\mathbf{y}_n - \mathbf{D}\mathbf{s}_n\|_2^2 + \eta\sum_{n=1}^N \|\mathbf{s}_n\|_1 \quad (1)$$

where $\mathcal{D} := \{\mathbf{D}\in\mathbb{R}^{M\times Q} : \|\mathbf{d}_q\|_2 \leq 1, \forall q\}$, $\eta > 0$ is a tuning parameter, and $\|\mathbf{s}_n\|_1 := \sum_{q=1}^Q |s_{n,q}|$ represents the ℓ_1-norm of $\mathbf{s}_n$ with $s_{n,q} := [\mathbf{s}_n]_q$. The ℓ_1-norm regularizer encourages sparsity on the $\mathbf{s}_n$'s and η controls the sparsity level of the resulting $\mathbf{s}_n$'s. The constraint on the size of the columns of $\mathbf{D}$ defined in $\mathcal{D}$ are necessary to avoid dictionary atoms to grow unbounded and the corresponding entries of each $\mathbf{s}_n$ approach zero.

Problem (1) is non-convex and usually solved via a BCD approach [4]. The BCD-based solver iteratively updates $\mathbf{D}$ with $\{\mathbf{s}_n\}_{n=1}^N$ fixed, and then updates all $\{\mathbf{s}_n\}_{n=1}^N$ with $\mathbf{D}$ fixed. This BCD solver must solve a constrained LS optimization problem to update the dictionary, and is thus sensitive to the presence of outliers in $\mathcal{Y}$ [17]. Note that in the aforementioned BCD solver the updates for the $\mathbf{s}_n$'s are also adversely impacted by outliers both through the outlier-sensitive $\mathbf{D}$ and the optimization criterion used.

III. Sparse and structured outlier modeling

In this section, a model for each $\mathbf{y}_n \in \mathcal{Y}$ that explicitly captures the presence of outliers in the training data set $\mathcal{Y}$ is introduced. To this end, each $\mathbf{y}_n \in \mathcal{Y}$ is modeled as

$$\mathbf{y}_n = \mathbf{D}\mathbf{s}_n + \mathbf{o}_n + \boldsymbol{\epsilon}_n,\ n = 1,\ldots,N \quad (2)$$

where $\mathbf{o}_n$ denotes a *deterministic* outlier vector, and $\boldsymbol{\epsilon}_n$ denotes an inlier noise vector affecting $\mathbf{y}_n$. If $\mathbf{y}_n$ is an outlier, then $\mathbf{o}_n$ takes a nonzero value and lessens the effect of $\mathbf{y}_n$ on $\mathbf{D}$; otherwise, $\mathbf{o}_n = \mathbf{0}_N$, where $\mathbf{0}_N$ is an $N\times 1$ vector of zeros. Solving for the tuple $(\mathbf{D}, \{\mathbf{s}_n, \mathbf{o}_n\}_{n=1}^N)$ using the set of equations in (2) entails solving a *nonlinear* and *underdetermined* system of P equations with $N(M+Q)+MQ$ unknowns. Thus, auxiliary information about the structure of each $\mathbf{s}_n$ and the $\{\mathbf{o}_n\}_{n=1}^N$, and on the proportion of outliers present in $\mathcal{Y}$ becomes essential to obtain an stable estimate for $(\mathbf{D}, \{\mathbf{s}_n, \mathbf{o}_n\}_{n=1}^N)$ and avoid overfitting the model in (2).

The structural information about the outliers is summarized by an auxiliary weighted graph $\mathcal{G}(V,E)$. The set V defines the vertex set, with cardinality $|V| = N$, where each vertex is associated with an $\mathbf{o}_n$. The set E defines the edge set whose elements define the structure of the $\mathbf{o}_n$'s and the *similarity* of each pair of $\mathbf{o}_n$'s. The connectivity of $\mathcal{G}$ is compactly summarized by the adjacency matrix $\mathbf{W}\in\mathbb{R}^{N\times N}$, where an entry $w_{n,n'} := [\mathbf{W}]_{n,n'} > 0$ if nodes $v_n, v_{n'} \in V$ are connected in $\mathcal{G}$, and $w_{n,n'} = 0$ otherwise. The topology of $\mathcal{G}$, and thus the underlying structure associated with the outliers, is encapsulated in the Laplacian matrix $\mathbf{L} := \text{diag}(\mathbf{W}\mathbf{1}_N) - \mathbf{W}$, where $\mathbf{1}_N$ denotes an $N\times 1$ vector of ones and $\text{diag}(\mathbf{y}_n)$ a diagonal matrix with $\mathbf{y}_n$ on its main diagonal. Note that $\mathcal{G}$ does not have to be *connected*.

Usually DL seeks sparse expansion coefficient vectors $\{\mathbf{s}_n\}_{n=1}^N$. Moreover, outliers by definition are rare and, thus, few of them are assumed to be present in $\mathcal{Y}$. The sparse presence of outliers in (2) can be quantified through the support of the vector $[\|\mathbf{o}_1\|_2,\ldots,\|\mathbf{o}_N\|_2]$. Let $\mathbf{O} := [\mathbf{o}_1,\ldots,\mathbf{o}_N] \in \mathbb{R}^{M\times N}$ denote a matrix of outlier vectors, $\mathbf{S} := [\mathbf{s}_1,\ldots,\mathbf{s}_N] \in \mathbb{R}^{Q\times N}$ a matrix of expansion coefficient vectors, and $\|\mathbf{O}\|_{2,p} := \|[\|\mathbf{o}_1\|_2,\ldots,\|\mathbf{o}_N\|_2]\|_p$, where $\|\mathbf{o}_n\|_p := (\sum_{m=1}^M |o_{m,n}|^p)^{1/p}$ represents the ℓ_p-norm of $\mathbf{o}_n$ with $o_{m,n} := [\mathbf{o}_n]_m$ and $p\in[1,\infty)$. An estimator for $(\mathbf{D},\mathbf{S},\mathbf{O})$ that captures the spatial structure of the outliers is given by the solution of

$$\min_{\mathbf{D}\in\mathcal{D},\mathbf{S},\mathbf{O}} \frac{1}{2}\sum_{n=1}^N \|\mathbf{y}_n - \mathbf{D}\mathbf{s}_n - \mathbf{o}_n\|_2^2 + \eta\sum_{n=1}^N \|\mathbf{s}_n\|_1 + \lambda\|\mathbf{O}\|_{2,1} + \frac{\mu}{2}\text{Tr}(\mathbf{O}\mathbf{L}\mathbf{O}') \quad (3)$$

where $\text{Tr}(\cdot)$ denotes the matrix trace operator, $\eta > 0$ is a tuning parameter that controls the sparsity of the $\mathbf{s}_n$'s $\lambda > 0$ a tuning parameter that controls the number vectors in $\mathcal{Y}$ identified as outliers, and $\mu > 0$ a tuning parameter that controls how much emphasis is placed on the structural relation among outliers. The group Lasso regularizer in (3) encourages entire $\mathbf{o}_n$ vectors to be set to zero. The Laplacian regularization term in (3) can be explicitly written as

$$\text{Tr}(\mathbf{O}\mathbf{L}\mathbf{O}') = \frac{1}{2}\sum_{m=1}^M \left[\sum_{n=1}^N\sum_{n'=1}^N w_{n,n'}(o_{m,n} - o_{m,n'})^2\right]. \quad (4)$$

From (4), it follows that the Laplacian regularization term in (3) encourages $\mathbf{o}_n$'s to be similar to each other if they are connected in $\mathcal{G}$.

In general, (3) is a non convex optimization problem which is difficult to solve. Nevertheless and as argued in the ensuing section, when optimizing with respect to any one of $\mathbf{D}$, $\mathbf{S}$, and $\mathbf{O}$ with the other two fixed, the resulting optimization problems can be solved efficiently.

Remark 1 (Extensions to other robust DL formulations): DL-alternate formulations that rely on a constrained version of (3), where the ℓ_0-"norm" is used in lieu of the ℓ_1-norm, and bestow structure on the dictionary's atoms are commonly used [4], [15]. There are no general rules for choosing between any of these DL formulations and the selection often depends on the specific problem at hand. Our outlier model in (2) and the Laplacian regularizer in (4) can be used to develop robust versions of these DL approaches. The resulting robust and structure DL approaches are able to capture the underlying structure of the outliers after appropriately modifying the algorithmic framework outlined next.

IV. Solvers based on BCD and PG

In this section, a solver for (3) based on BCD is developed. This solver capitalizes on the observation that efficient solvers

Algorithm 1 BCD algorithm for dictionary update in (6a)

Require: $\mathcal{Y}$ and tuple $(\hat{\mathbf{D}}^{(k-1)}, \hat{\mathbf{S}}^{(k-1)}, \hat{\mathbf{O}}^{(k-1)})$.

Let $\mathbf{D}^{(\upsilon)} := [\mathbf{d}_1^{(\upsilon)}, \ldots, \mathbf{d}_Q^{(\upsilon)}]$ and $\hat{s}_{q,n}^{(k-1)} := [\hat{\mathbf{S}}^{(k-1)}]_{q,n}$.

Use υ as BCD iteration index and set $\mathbf{D}^{(0)} = \hat{\mathbf{D}}^{(k-1)}$.

for $\upsilon = 1, 2, \ldots, \Upsilon$ **do**

  for $q = 1, \ldots, Q$ **do**

    Update each atom in $\mathbf{D}^{(\upsilon)}$ as

$$\mathbf{r}_{q,n}^{(\upsilon,k-1)} = \mathbf{y}_n - \mathbf{o}_n^{(k-1)} - \sum_{\substack{q'=1 \\ q' \neq q}}^{Q} \hat{s}_{q',n}^{(k-1)} \mathbf{d}_{q'}^{(\upsilon-1)} \tag{5a}$$

$$\mathbf{u}_q^{(\upsilon)} = \left[\sum_{n=1}^{N} \left(\hat{s}_{q,n}^{(k-1)} \right)^2 \right]^{-1} \left[\sum_{n=1}^{N} \hat{s}_{q,n}^{(k-1)} \mathbf{r}_{q,n}^{(\upsilon,k-1)} \right] \tag{5b}$$

$$\mathbf{d}_q^{(\upsilon)} = \frac{\mathbf{u}_q^{(\upsilon)}}{\min\{1, \|\mathbf{u}_q^{(\upsilon)}\|_2\}} \tag{5c}$$

  end for

end for

return $\hat{\mathbf{D}}^{(k)} = \mathbf{D}^{(\Upsilon)}$.

can be realized for updating either $\mathbf{D}$, $\mathbf{S}$, or $\mathbf{O}$ when the other two optimization variables are fixed. In fact, (3) is convex with respect to $\mathbf{D}$ and $\mathbf{O}$ individually. With $f(\mathbf{D}, \mathbf{S}, \mathbf{O})$ denoting the objective function in (3), the proposed BCD algorithm minimizes $f(\mathbf{D}, \mathbf{S}, \mathbf{O})$ with respect to each optimization block, namely $\mathbf{D}$, $\mathbf{S}$, and $\mathbf{O}$, one at a time. Let k denote the BCD-iteration index. Our robust DL algorithm is summarized by the following iterations

$$\hat{\mathbf{D}}^{(k)} = \arg\min_{\mathbf{D} \in \mathcal{D}} \; f(\mathbf{D}, \hat{\mathbf{S}}^{(k-1)}, \hat{\mathbf{O}}^{(k-1)}) \tag{6a}$$

$$\hat{\mathbf{S}}^{(k)} = \arg\min_{\mathbf{S}} \; f(\hat{\mathbf{D}}^{(k)}, \mathbf{S}, \hat{\mathbf{O}}^{(k-1)}) \tag{6b}$$

$$\hat{\mathbf{O}}^{(k)} = \arg\min_{\mathbf{O}} \; f(\hat{\mathbf{D}}^{(k)}, \hat{\mathbf{S}}^{(k)}, \mathbf{O}) \tag{6c}$$

where $\hat{\mathbf{D}}^{(k)}$, $\hat{\mathbf{S}}^{(k)}$, and $\hat{\mathbf{O}}^{(k)}$ denote estimates for $\mathbf{D}$, $\mathbf{S}$, and $\mathbf{O}$ at iteration k, respectively.

Updating $\mathbf{D}^{(k)}$ in (6a) is done via a BCD algorithm iterating over the columns of $\mathbf{D}$. Per BCD iteration, each atom of $\mathbf{D}$ is updated with all other atoms fixed by solving the resulting unconstrained LS optimization problem. This BCD algorithm is summarized as Algorithm 1. Note that for (5b) to be well-defined, each atom in $\hat{\mathbf{D}}^{(k-1)}$ must be used by at least one $\mathbf{s}_n$. In practice, every unused atom of the dictionary is updated to a randomly chosen element of $\mathcal{Y}$ in before running Algorithm 1 and (5b) is skipped for such an atom when updating the dictionary. If there are no unused atoms, Algorithm 1 is guaranteed to achieve the minimum of the cost in (6a) [2, Prop. 2.7.1].

The update in (6b) decouples per column of $\mathbf{S}$. The resulting set of optimization problem are convex and correspond to the Lagrangian version of the least-absolute shrinkage and selection operator [14]. These optimization problem can be efficiently solved via, e.g, BCD or the alternating direction method of multipliers [2], [4]. The update in (6c) entails solving a convex optimization problem. However, tackling (6c) in its current form leads to solvers whose computational complexity can be high due to the coupling across $\mathbf{o}_n$'s caused by the Laplacian regularizer.

Algorithm 2 PG algorithm for solving (6c)

Require: $\mathcal{Y}$, $(\hat{\mathbf{D}}^{(k)}, \hat{\mathbf{S}}^{(k)}, \hat{\mathbf{O}}^{(k-1)})$, and $\mu, \lambda, \Lambda > 0$.

Let $\mathbf{O}^{(\tau)} := [\mathbf{o}_1^{(\tau)}, \ldots, \mathbf{o}_N^{(\tau)}]$.

Use τ as PG iteration index and set $\mathbf{O}^{(1)} = \hat{\mathbf{O}}^{(k-1)}$.

for $\tau = 1, 2, \ldots, T$ **do**

  for $n = 1, \ldots, N$ **do**

    Compute $\boldsymbol{\omega}_n(\mathbf{O}^{(\tau)})$ via (7a).

    Compute $h_n(\mathbf{O}^{(\tau)})$ via (7b).

    Update $\mathbf{o}_n^{(\tau)}$ via (9).

  end for

end for

return $\hat{\mathbf{O}}^{(k)} = \mathbf{O}^{(T)}$.

We propose to solve (6c) via a PG algorithm [1]. Although the resulting solver for (6c) is iterative in nature, it features fast convergence when compared to other first order optimization methods such as gradient descent and BCD. Moreover, the updates are decomposable across each $\mathbf{o}_n$. Hence, they can be parallelized. The PG algorithm for updating (6c) at iteration k relies on

$$\boldsymbol{\omega}_n(\mathbf{O}^{(\tau)}) := -(\mathbf{y}_n - \hat{\mathbf{D}}^{(k)} \hat{\mathbf{s}}_n^{(k)} - \mathbf{o}_n^{(\tau)}) + \mu \mathbf{O}^{(\tau)} \mathbf{a}_n \tag{7a}$$

$$h_n^{(k)}(\mathbf{O}^{(\tau)}) := \mathbf{o}_n^{(\tau)} - \boldsymbol{\omega}_n^{(k)}(\mathbf{O}^{(\tau)})/\Lambda \tag{7b}$$

$$g^{(k)}(\mathbf{O}; \mathbf{O}^{(\tau-1)}) := \sum_{n=1}^{N} \left[\frac{1}{2} \left\| \mathbf{o}_n - h_n^{(k)}(\mathbf{O}^{(\tau-1)}) \right\|_2^2 + \frac{\lambda}{\Lambda} \|\mathbf{o}_n\|_2 \right] \tag{7c}$$

where τ denotes the proximal method iteration index, $\boldsymbol{\omega}_n(\mathbf{O}^{(\tau)})$ denotes the gradient of the differentiable part of f with respect to $\mathbf{o}_n$ evaluated at $\mathbf{O}^{(\tau)}$, $\mathbf{a}_n$ the n-th column of $\mathbf{L}$, and Λ the Lipschitz constant for the gradient (with respect to $\mathbf{O}$) of the differentiable part of f, which corresponds to the largest singular value of $\mathbf{I} + \mu \mathbf{L}$. Our iterative update for computing (6c) is summarized as

$$\mathbf{O}^{(\tau)} = \arg\min_{\mathbf{O}} g^{(k)}(\mathbf{O}; \mathbf{O}^{(\tau-1)}). \tag{8}$$

This update is separable across $\mathbf{o}_n$'s. Computing each $\mathbf{o}_n^{(\tau)}$ is then done in closed form as

$$\mathbf{o}_n^{(\tau)} = h_n^{(k)}(\mathbf{O}^{(\tau-1)}) \left(1 - \frac{\lambda}{\Lambda \|h_n^{(k)}(\mathbf{O}^{(\tau-1)})\|_2} \right)_+ \tag{9}$$

where $(\cdot)_+ := \max\{0, \cdot\}$. Note that $1/\Lambda$ in (9) can be understood as the step size α of the PG algorithm [13, Sec. 4.2]. From this vantage point, it has been shown that the PG algorithm is guaranteed to converge for any $\alpha \in (0, 1/\Lambda]$. It is also possible to choose α at each PG iteration via a line search (see [13] and reference therein).

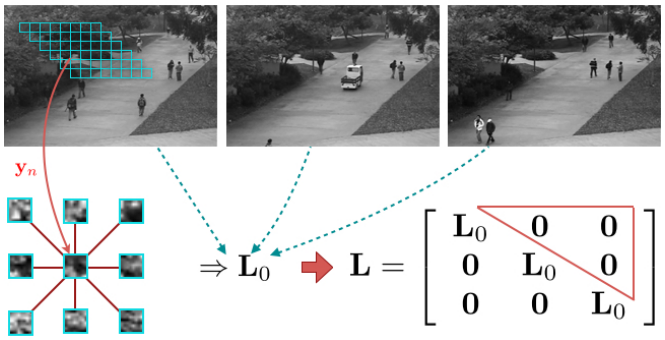


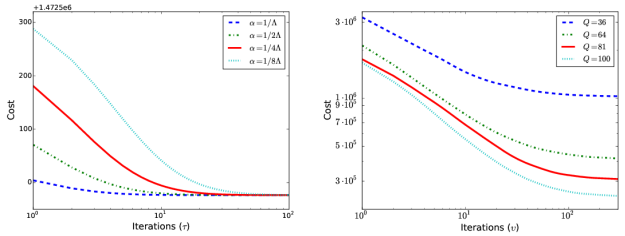
Figure 1. Images from the UCSD Anomaly Detection Dataset used to train the dictionary. Each image was partitioned to multiple blocks and their relative position with respect to each other was used to define an auxiliary graph whose structure is captured by the Laplacian matrix $\mathbf{L}_0$. This spatial structure is illustrated for the n-th block (the one corresponding to $\mathbf{y}_n$) on the left side of the figure. The Laplacian matrix that captures the structure of the entire dataset $\mathcal{Y}$ is illustrated on the right side of the figure, where $\mathbf{0}$ denotes a 551×551 matrix of zeros. The portion of $\mathbf{L}$ enclosed by the red triangle underscores the fact that $\mathbf{L}$ does not capture block structures across images.

The PG algorithm for solving (6c) is summarized as Algorithm 2. Each iteration of this PG algorithm can be computed in $O(MNQ + MN^2)$ run time, where the quadratic dependency on N stems from the second term in (7a) that defines an operation to be performed for each n. The computational complexity of Algorithm 2 can be reduced after noticing that $\mathbf{O}^{(\tau)}$ is a sparse matrix and often $\mathbf{L}$ is a sparse matrix as well. In this case, specialized sparse matrix multiplication algorithms can be used to reduce the run time of (7a) for each n. The PG method outlined by Algorithm 2 converges to the solution of (6c) and features a worst-case convergence rate of $O(1/\tau)$ [1], [13].

Remark 2 (Improved convergence rate for the PG iterations): Recent works have shown that it is possible to enhance the suboptimal convergence rate of the PG algorithms while maintaining their computational simplicity [1], [13]. From that vantage point, it becomes possible to develop an accelerated PG (APG) algorithm for solving (6c) which features worst-case convergence rate of $O(1/\tau^2)$ to the solution of (6c), see [1] and references therein. The resulting APG solver requires a copy of both $\mathbf{O}^{(\tau)}$ and $\mathbf{O}^{(\tau-1)}$. Maintaining a copy of $\mathbf{O}^{(\tau-1)}$ does not add excessive data storage requirements since $\mathbf{O}^{(\tau-1)}$ is column-wise sparse. Similarly to the PG algorithm, each iteration of the APG algorithm can be computed in $O(MNQ + MN^2)$ run time and benefits from specialized sparse matrix multiplication algorithms for updating each $\boldsymbol{\omega}_n$.

V. Numerical Examples

In this section the numerical performance of the robust DL algorithm defined in (6) is illustrated via numerical tests on two real datasets. For these tests all algorithms were implemented in Python 2.7. Numerical tests were performed on the University of California San Diego (UCSD) Anomaly Detection Dataset [8] and on the Resampled United States Postal Service (USPS) dataset [6].



(a) Iterations of Algorithm 2. (b) BCD iterations in (6).

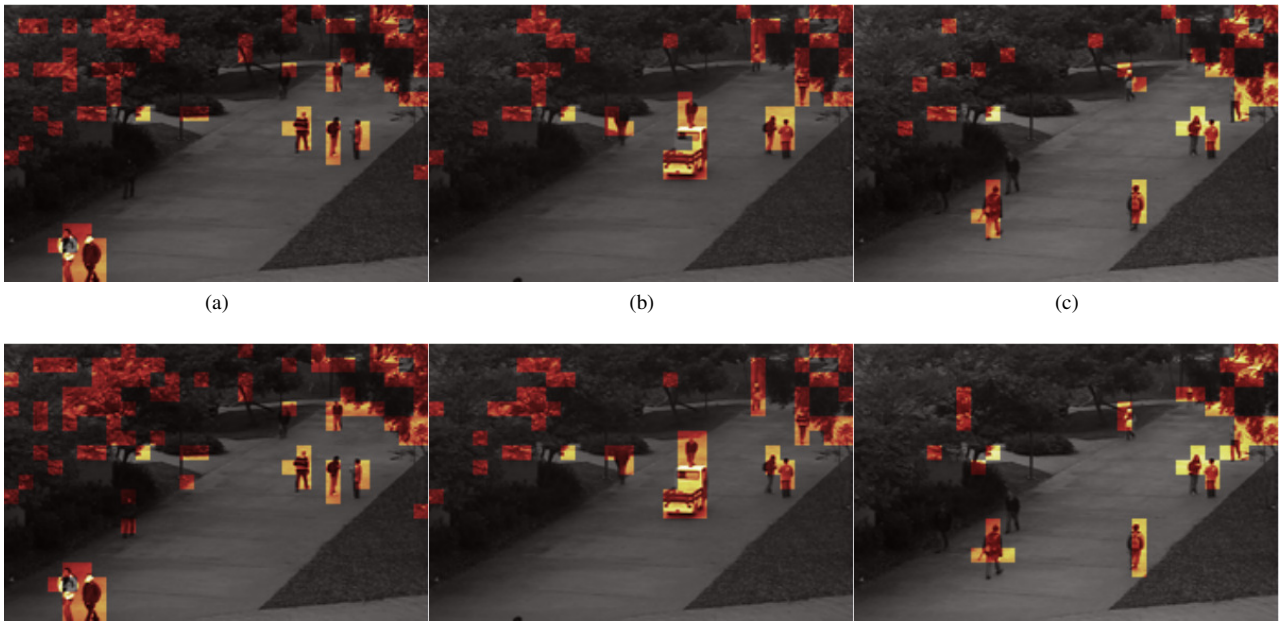
Figure 2. Illustration of the evolution of the cost in (3) during the execution of (6c) via Algorithm 2 (a), and during the execution of the robust DL algorithm defined by (6) (b). *Subfig. a*: evolution of the cost in (3) when updating $\mathbf{O}$ via Algorithm 2. These curves were obtained for an early BCD iteration of (6) and various choices of step size α. Note that the converge rate of Algorithm 2 was not significantly impacted by the choice of μ. *Subfig. b*: evolution of the cost in (3) during the execution of the BCD algorithm in (6) for various values of the dictionary size Q.

A. UCSD Anomaly Detection Dataset

The UCSD Anomaly Detection Dataset dataset features video of a pedestrian walkway acquired with an stationary camera and a fixed background. Several works have used this dataset for testing anomaly detection algorithms in crowded scenes by capitalizing on both temporal and spatial structure. [8], [7]. With few exceptions, most portions of the video contain only pedestrians moving along the walkway. Abnormal events include the presence of bikers, skates and small carts. Three images from this dataset were used for this test, see Fig. 1. Each image was chosen so that the crowd density in the image was low. Note that no artificial anomalies were introduced in these images.

Each image was resized from 158×258 pixels to 152×252 and then partitioned into 8×8 pixel blocks. The resulting $N = 1,653$ blocks (551 per image), were used to construct $\mathcal{Y} \subset \mathbb{R}^M$, with $M = 64$. Each 64-dimensional $\mathbf{y}_n \in \mathcal{Y}$ was constructed by vertically stacking the columns of each block. No further preprocessing was performed on the elements of the training set. In order to capture the spatial structure among blocks per image, an auxiliary graph with $N_0 = 551$ nodes was constructed as follows. Each $\mathbf{y}_n$ was assigned to a node of the graph. The edges for each node were chosen so that each $\mathbf{y}_n$ is connected to its spatial neighboring blocks only, as shown in Fig. 1, and their corresponding weights were set to unity. Thus, in the resulting graph the degree of each node is either 3 for corner blocks, 5 for border blocks that are not corners, and 8 for all other ones. The connectivity of the graph is summarized by the Laplacian matrix $\mathbf{L}_0$. Finally, $\mathbf{L}$ was defined as a block diagonal matrix containing three copies of $\mathbf{L}_0$ on its main block diagonal (see Fig. 1). Note that $\mathbf{L}$ does not capture block structures across images. With the current form of $\mathbf{L}$ and different from [8], [7], our method does not rely on any form of temporal information. Such temporal information can, however, be included on the off-diagonal entries of $\mathbf{L}$.

Fig. 2a shows the evolution of the cost function in (3) as the PG algorithm iterates for solving (6c). For illustration purposes



(a) (b) (c)

(d) (e) (f)

Figure 3. Illustration of blocks identified as outliers. Subfigs. (a), (b), and (c) were obtained by setting $(\lambda, \mu) = (37.5, 0)$. In this case, 191 blocks (approximately 12% of the training blocks) were identified as outliers and on average 57% of the entries of each $\mathbf{s}_n$ were set to zero. Subfigs. (d), (e), and (f) were obtained by setting $(\lambda, \mu) = (36.0, 10)$. In this case, 207 blocks (approximately 12.5% of the training blocks) were identified as outliers and on average 55% of the entries of each $\mathbf{s}_n$ were set to zero.

only, we used a modified version of Algorithm 2 where a tunable step size α was used in lieu of the factor $1/\Lambda$. The PG algorithm is guaranteed to converge for any $\alpha \in (0, 1/\Lambda]$ [13]. However, the best convergence rate is achieved by setting $\alpha = 1/\Lambda$ as in Algorithm 2. In practice it was observed that during the first iteration of the BCD algorithm the PG algorithm took less than 20 iterations. As the BCD iterations progressed, the average number of PG iterations required for convergence decreased to around 3 - 5. Fig. 2b illustrates the convergence rate of the BCD algorithm defined by the updates in (6) for various values of Q.

Fig. 3 shows the blocks identified as outliers in the images shown in Fig. 1. For this experiment, a dictionary with $Q = 64$ atoms was used. Parameter η was set to $\eta = 3.7$ so that on average 36 nonzero entries were chosen for each $\mathbf{s}_n$. Interestingly, some of the blocks identified as outliers correspond to the portions of the images displaying pedestrians and the service cart. It was observed that setting $\mu > 0$ encouraged outlier blocks to be chosen according to the structure defined by the $\mathbf{L}$. For instance, in Fig. 3e the outlier blocks corresponding to some of the pedestrians seem to follow more rigidly the spatial structure defined for the elements of $\mathcal{Y}$. Finally, several blocks corresponding to the foliage in the background are picked up as outliers. This can in part be due to the texture richness inherent to foliage and other natural texture images. Using a dictionary with $Q > 64$ and temporal information in $\mathbf{L}$ can help alleviate the sensitivity of our method to the background texture diversity.

B. USPS Dataset

The USPS resampled data set is a randomly shuffled version of the USPS handwritten numeral digital data, which had many more training digits than testing digits. This dataset includes 4,696 16×16 training digits, which are stored as 256-dimensional column vectors whose entries range over the interval $[-1, 1]$. In this test, $N = 1,000$ digits were used for training and artificial anomalies were introduced by coloring a fraction of randomly chosen digits. If a digit is deemed as an anomaly, its numeral is colored red and a blue stripe is placed over the center of the image. Otherwise, the digit is left as a grayscale image. In order to emulate the RGB model for each digit, three copies of each training digit were created, one for each channel of the RGB color model. Per digit, each RGB-color-channel image was vectorized and vertically stacked to form a 768×1 training vector $\mathbf{y}_n \in \mathcal{Y}$.

The goal of this test was to find the injected anomalies along with other "odd" looking digits in $\mathcal{Y}$. The size of the dictionary was set to $Q = 100$ and it was initialized with randomly chosen elements form $\mathcal{Y}$. The spatial structure of the digits was captured via an auxiliary graph with $N = 1,000$ nodes. For each node, the Cosine similarities between its image and all other images corresponding to all other nodes were computed. Per node n, a set of K weighted edges was added to the graph, where the chosen edges corresponded to the node pairs $\{(n, m)\}_{m \neq n}$ with the K-largest similarity score. In this experiment, the constrained DL formulation that uses the ℓ_0-"norm" was used in lieu of (3) [15]. Thus, the solvers used for (6a) and (6b) were modified accordingly. Specifically,

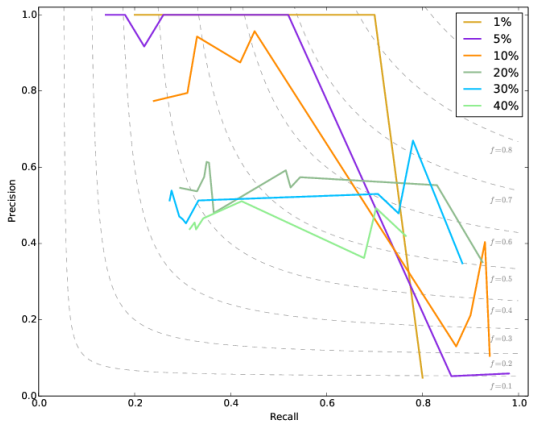


Figure 4. Precision-Recall curves obtained by the robust DL algorithm on the USPS dataset with manually-injected color anomalies. Dashed light-gray lines denote the isolevel lines for various F-score levels, which is defined as the harmonic mean of the precision and recall scores, indexed by f.

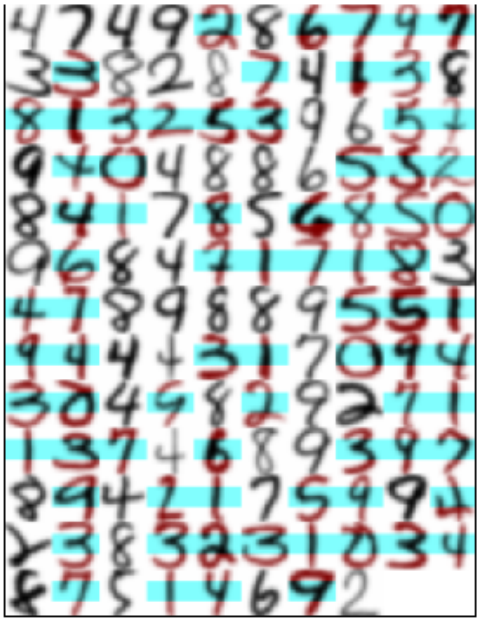


Figure 5. Anomalies detected on the artificially-modified Resampled USPS dataset using the robust DL algorithm with parameters $\lambda = 1.025$, $\mu = 480$, and $K = 100$. Artificially injected anomalies are shown in color.

orthogonal matching pursuit (OMP) was used to solve (6b) and the constrained set $\mathcal{D}$ was dropped from (6a). The desired set of nonzero coefficients used by OMP was set to $S_0 = 40$.

Various percentages of anomalous images ranging from 1% to 40% were injected into the training set. The parameters λ, μ, and K were tuned so as to try to find all the artificially-added anomalous digits. Fig. 4 shows Precision-Recall curves for the various percentages of injected anomalies. These curves show that precision scores are higher for a broad range of recall scores when a smaller percentage of anomalies are injected. An example of the anomalies detected is shown in Fig. 5. In this case, 100 images were colored (10% anomalies injected) and 78 were correctly identified yielding a precision score of 78.00% and recall score of 60.94%.

VI. Conclusions

In this paper, a robust DL algorithm that leveraged known structure among the elements of the training set was developed. The training set structure was summarized by an auxiliary graph whose connectivity was defined according to how similar elements of the training set were. A BCD solver was developed for learning the dictionary and identifying the outliers. Numerical tests on two real datasets were used to illustrate the performance of the proposed robust DL algorithm.

References

[1] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM Journal on Imaging Sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[2] D. P. Bertsekas, *Nonlinear Programming*. Athena Scientific, 2nd ed., 1999.

[3] Z. Chen and Y. Wu, "Robust dictionary learning by error source decomposition," in *Proc. of IEEE International Conference on Computer Vision*, Dec. 1-8, Sydney, Australia 2013, pp. 2216–2223.

[4] M. Elad, *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing*. New York: Springer, 2010.

[5] P. A. Forero and G. B. Giannakis, "Sparsity-exploiting robust multidimensional scaling," *IEEE Transactions on Signal Processing*, vol. 60, no. 8, pp. 4118–4134, Aug. 2012.

[6] J. J. Hull, "A database for handwritten text recognition research," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 5, pp. 550–554, May 1994.

[7] W. Li, V. Mahadevan, and N. Vasconcelos, "Anomaly detection and localization in crowded scenes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 1, pp. 18–32, Jan. 2014.

[8] V. Mahadevan, W. Li, V. Bhalodia, and N. Vasconcelos, "Anomaly detection in crowded scenes," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 13-18, San Francisco, CA, United States 2010, pp. 1975–1981.

[9] G. Mateos and G. B. Giannakis, "Robust nonparametric regression via sparsity control with application to load curve data cleansing," *IEEE Transactions on Signal Processing*, vol. 60, no. 4, pp. 1571–1584, Apr. 2012.

[10] ——, "Robust PCA as bilinear decomposition with outlier-sparsity regularization," *IEEE Transactions on Signal Processing*, vol. 60, no. 10, pp. 5176–5190, Oct. 2012.

[11] B. A. Olshausen and D. J. Field, "Sparse coding with an overcomplete basis set: A strategy employed by V1?" *Vision Research*, vol. 37, no. 23, pp. 3311 –3325, 1997.

[12] Q. Pan, D. Kong, C. Ding, and B. Luo, "Robust non-negative dictionary learning," in *Proc of 28th AAAI Conference on Artificial Intelligence*, Jul. 27- 31, 2014, Quebec, Canada 2014.

[13] N. Parikh and S. Boyd, "Proximal algorithms," *Found. Trends Optimization*, vol. 1, pp. 123–231, 2013.

[14] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[15] I. Tošić and P. Frossard, "Dictionary learning," *IEEE Signal Process. Mag.*, vol. 28, no. 2, pp. 27–38, Mar. 2011.

[16] N. Wang, J. Wang, and D.-Y. Yeung, "Online robust non-negative dictionary learning for visual tracking," in *Proc. of IEEE International Conference on Computer Vision*, Dec. 1-8, Sydney, Australia 2013, pp. 657–664.

[17] A. M. Zoubir, V. Koivunen, Y. Chakhchoukh, and M. Muma, "Robust estimation in signal processing: A tutorial-style treatment of fundamental concepts," *IEEE Signal Processing Magazine*, vol. 29, no. 4, pp. 61–80, Jul. 2012.