Online Orthogonal Dictionary Learning Based on Frank–Wolfe Method

Ye Xue\textsuperscript{\textcopyright}, Graduate Student Member, IEEE, and Vincent K. N. Lau\textsuperscript{\textcopyright}, Fellow, IEEE

Abstract—Dictionary learning is a widely used unsupervised learning method in signal processing and machine learning. Most existing works on dictionary learning adopt an off-line approach, and there are two main off-line ways of conducting it. One is to alternately optimize both the dictionary and the sparse code, while the other is to optimize the dictionary by restricting it over the orthogonal group. The latter, called orthogonal dictionary learning (ODL), has a lower implementation complexity and, hence, is more favorable for low-cost devices. However, existing schemes for ODL only work with batch data and cannot be implemented online, making them inapplicable for real-time applications. This article, thus, proposes a novel online orthogonal dictionary scheme to dynamically learn the dictionary from streaming data, without storing the historical data. The proposed scheme includes a novel problem formulation and an efficient online algorithm design with convergence analysis. In the problem formulation, we relax the orthogonal constraint to enable an efficient online algorithm. We then propose the design of a new Frank–Wolfe-based online algorithm with a convergence rate of $O(\ln t/t^{1/4})$. The convergence rate in terms of key system parameters is also derived. Experiments with synthetic data and real-world Internet of things (IoT) sensor readings demonstrate the effectiveness and efficiency of the proposed online ODL scheme.

Index Terms—Convergence analysis, dictionary learning, Frank–Wolfe, online learning, sensor.

I. INTRODUCTION

Sparse representation of data has been widely used in signal processing, machine learning, and data analysis, and shows a highly expressive and effective representational ability [1]–[4]. It represents the data $y \in \mathbb{R}^N$ by a linear combination $y \approx D_{\text{true}}^{\text{true}}x$, where $x \in \mathbb{R}^M$ is a sparse code (i.e., the number of nonzero entries of $x$ is much smaller than $M$) and $D^{\text{true}} \in \mathbb{R}^{N \times M}$ is the dictionary that contains the compact information of $y$. At the initial stage in this line of research, predefined dictionaries based on a Fourier basis and various types of wavelets were successfully used for signal processing. However, using a learned dictionary instead of a generic one has been shown to dramatically improve the performance on various tasks, e.g., image denoising and classification.

One method to learn a dictionary is to alternately optimize (AO) problems with both the dictionary and the sparse code as the variables [1]–[5]. In this approach, the dictionary usually has no constraints or has a bounded norm constraint on.

Notations

| Symbol | Description |
|--------|-------------|
| $x, X$ | Column vector and matrix. |
| $X_{a,:}, X_{:,i}$ | The $a$-row and the $i$th column of matrix $X$. |
| $x_i, X_{i,j}$ | The $i$th element of vector $x$ and the element in the $i$th row and the $j$th column of matrix $X$. |
| $e_i$ | Standard basis vector with a 1 in the $i$th coordinates and 0 elsewhere. |
| $\mathbb{O}(N, \mathbb{R})$ | $N$-dimensional orthogonal group with real-valued entries. |
| $\mathbb{B}_p(N, \mathbb{R})$ | $N$-dimensional (closed) unit spectral ball. |
| $(\cdot)^T, \text{vec}(\cdot)$ | Transpose and vectorization. |
| diag$(x)$ | Diagonal matrix with vector $x$ on its diagonal. |
| $[\cdot], [\cdot, \cdot], | \cdot |$ | Elementwise ceiling operator, elementwise floor operator and taking elementwise absolute value. |

Manuscript received 8 February 2021; revised 3 August 2021 and 21 October 2021; accepted 24 November 2021. Date of publication 8 December 2021; date of current version 1 September 2023. (Corresponding author: Ye Xue.)

The authors are with the Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology, Hong Kong (e-mail: yxueaf@ust.hk; eeknlau@ust.hk).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2021.3131181.

Digital Object Identifier 10.1109/TNNLS.2021.3131181
each atom\(^1\) of the dictionary to prevent trivial solutions [3], [4]. Another method is to restrict the dictionary over the orthogonal group \(\mathbb{O}(N, \mathbb{R})\) and solve the following optimization problem only for the dictionary [6]–[8]

\[
\min_{D \in \mathbb{O}(N, \mathbb{R})} \frac{1}{T} \sum_{t=1}^{T} S_p(D^T y_t)
\]

(1)

where \(S_p(\cdot)\) is a sparsity-promoting function. This formulation is motivated by the fact that the sparse code can be obtained as \(x_t = (D^\text{true})^T D^\text{true} x_t \approx (D^\text{true})^T y_t\) if the dictionary \(D^\text{true}\) is orthogonal.

To enable efficient data processing, the latter method, called orthogonal dictionary learning (ODL), is more favorable than the AO method for the following reasons. First, ODL has a lower computational complexity. It updates only the dictionary at each iteration, while the AO method requires solving two subproblems, respectively, for the dictionary and the sparse code. Second, ODL has a lower sample complexity since it restricts the dictionary over a smaller optimization space.\(^2\)

Third, ODL allows efficient transmission of the dictionary. This is because an \(N \times N\) orthogonal matrix can be represented by \((N(N - 1)/2)\) statistically independent angles via the Givens rotation,\(^3\) and the angles can be quantized efficiently to achieve a minimum quantization loss [11]. This angle-based transmission has been adopted in many wireless communication standards [12].

Despite the benefits of ODL, however, the existing ODL methods only work with batch data. In other words, they require the whole dataset to run the algorithm. Hence, they are not applicable in many real-time applications, including real-time network monitoring [13], sensor networks [14], and Twitter analysis [15], where data arrive continuously in rapid, unpredictable, and unbounded streams. To deal with streaming data, we propose an online ODL approach that processes the data in a single sample or in a minibatch. The online ODL problem can be regarded as taking \(T \to \infty\) in Problem (1) and is formally formulated as

\[
\min_{D \in \mathbb{O}(N, \mathbb{R})} F(D) = \mathbb{E}_{y \sim P}[S_p(D^T y)]
\]

(2)

where \(y\) is the realization of the random variable \(Y\) drawn from a distribution \(P\). Since the distribution \(P\) is usually unknown and the cost of computing the expectation is prohibitive, the main challenge is to solve Problem (2) without the accessibility of \(F(D)\) or its gradient \(\nabla F(D)\). In this case, we can only rely on the sampled data to calculate the approximation of the objective function or the gradient. These approximations will jeopardize the performance and the convergence of the algorithms compared to the case where the exact objective value and gradient are available [16].

Problem (2) is an online constraint optimization problem, and many general algorithms are available for solving this type of problem, such as regularized dual averaging (RDA) [17] and stochastic mirror descent (SMD) [18]. However, none can be directly applied to Problem (2) due to the nonconvex constraint set and possibly nonconvex sparsity-promoting function, e.g., \(S_p(\cdot) = -\|\cdot\|^p_2, (p \in \mathbb{N}, p > 2)\) [8], [19], [20].

In this work, to enable an efficient online algorithm with a convergence guarantee, we propose a novel online ODL solution with a convex relaxation of the orthogonal constraint in Problem (2). After the relaxation, the problem becomes a nonconvex optimization over a convex set. One could transform this relaxed problem into an unconstrained problem with a composite nonconvex objective \(\tilde{F}(D)\) (see Example 3 in [17, Section I.1]) and solve the transformed problem by ProxSGD [21]. However, the use of ProxSGD gives rise to issues: first, the proximal operator in ProxSGD creates a high per-iteration computational complexity; second, ProxSGD can only be guaranteed to converge to an \(\epsilon\)-stationary point (a point \(\vec{D}^*\), such that \(\mathbb{E}[\|\nabla \tilde{F}(\vec{D}^*)\|^2] \leq \epsilon\)) when the minibatch size is increasing by \(1/\epsilon\) [21]. To achieve small error, the minibatch size needs to be large, which is not suitable for most online processors with limited memory.

In this work, we propose a Frank–Wolfe-based [22] algorithm, the Nonconvex Stochastic Frank–Wolfe (NoncvxSFW) method, to solve the relaxed problem directly without the problem transformation. NoncvxSFW can achieve low-complexity per-iteration computation due to the linear minimization oracle (LMO) in the Frank–Wolfe method. We also prove that the proposed algorithm with a single sample or a fixed minibatch size can be guaranteed to converge to a stationary point of the relaxed online ODL problem. The main contributions are summarized as follows.

1) Novel Online ODL Formulation: We propose an online ODL problem with an \(\ell_1\)-norm-based sparsity-promoting function and a convex relaxation of the orthogonal constraint. We prove that all the optimal solutions of the original problem are also the optimal solutions of the relaxed problem, which enables an efficient online algorithm with guaranteed convergence.

2) Online Frank–Wolfe-Based Algorithm: We develop an online algorithm, the NoncvxSFW method, to solve the relaxed optimization problem with a single sample or a fixed minibatch size. The convergence is analyzed, and its rate is shown to be \(O((\ln t)/t^{1/4})\), where \(t\) is the number of iterations. As far as we are aware, this is the first nonasymptotic convergence rate for online nonconvex optimization using the Frank–Wolfe-based method. The proposed algorithm and the corresponding theoretical results can also be generalized into general online nonconvex problems with convex constraints.

3) Effective and Efficient Application on IoT Sensor Data Compression: We provide extensive simulations with both synthetic data and a real-world IoT sensor dataset. The simulation results demonstrate the effectiveness and efficiency of our proposed online ODL method. They also verify the correctness of our theoretical results. For the synthetic data, the proposed scheme can
achieve superb performance in terms of the convergence rate and the recovery error, while, for the real-world sensor data, it can achieve a better root-mean-square error (RMSE) with a higher compression ratio for sensor data compression compared to the state-of-the-art baselines [4], [8], [23]–[25].

The rest of this article is organized as follows. In Section II, we illustrate the signal model and present the problem formulation for the online ODL. In Section III, we present the Frank–Wolfe-based online algorithm with nonasymptotic convergence analysis. Application examples and numerical simulation results are provided in Section IV and Section V, respectively. Finally, Section VI summarizes the work. The notations used throughout this article are listed as follows.

II. SIGNAL MODEL AND PROBLEM FORMULATION

In this section, we introduce the signal model and the proposed online ODL problem formulation.

A. Signal Model

In the online ODL, we consider that the data samples arrive in streams. At time $t$, there is one minibatch of samples $Y_t = [y_t^1, \ldots, y_t^M] \in \mathbb{R}^{N \times M}$, arriving at the processor, where $M_t$ is the minibatch size at time $t$ and $y_t^j$ is the $j$th sample in the $t$th minibatch. Each sample is assumed to be generated by

$$y_t^j = D^\text{true}x_t^j \quad \forall j = 1, \ldots, M_t \quad \forall t$$

(3)

where $D^\text{true}$ is the orthogonal dictionary and $x_t^j$ is a realization of the random variable $X$ drawn from some distribution that induces sparsity. The basic goal of the online ODL processor is to dynamically learn the dictionary in an online manner without storing all the historical samples. That is to say, the processor updates the dictionary $D_t$ at time $t$ only according to the samples arriving at that time, i.e., $Y_t$.

B. $\ell_3$-Norm-Based Formulation

For the online ODL scheme, the dynamic updating of the dictionary can be done by solving the generic Problem (2) in an online manner. In Problem (2), the sparsity-promoting function $S\rho(\cdot)$ needs to be carefully designed since it determines the performance of the online ODL and the complexity of the algorithm. In this work, we use $S\rho(\cdot) = -\|\cdot\|_3^p$, which results in the following optimization problem:

$$\min_{D \in \mathbb{O}(N, \mathbb{R})} F(D) = E_{\mathcal{Y}} - p\left[\|D^T y\|_3^p\right].$$

(4)

The choice of $-\|\cdot\|_3^p$ is inspired by the recent result that minimizing the negative $p$th power of the $\ell_p$-norm ($p \in \mathbb{N}, p > 2$) with the unit $\ell_2$-norm constraint leads to sparse (or spiky) solutions [8], [19], [26]. An illustration is given in Fig. 1. Compared to the widely used $\ell_1$-norm, the negative $\ell_p$-norm formulation allows a convex relaxation of the orthogonal constraint. Hence, it provides flexibility for the algorithm design, as we will illustrate in Section II-C. Also, the differentiability of this formulation enables a faster convergence of algorithms. In this article, we choose $p = 3$ since the sample complexity and the total computation complexity achieve the minimum when $p = 3$ among all the choices of $p (p \in \mathbb{N}, p > 2)$ for maximizing the $\ell_p$-norm over the orthogonal group [19].

C. Orthogonal Constraint Relaxation

After determining the sparsity-promoting function, to facilitate an efficient online solver with a convergence guarantee, we propose a convex relaxation of the orthogonal constraint in Problem (4) based on Lemma 1.

**Lemma 1** ([27, 3.4] Convex Hull of the Orthogonal Group): The convex hull of the orthogonal group is the (closed) unit spectral ball

$$\text{conv} (\mathbb{O}(N, \mathbb{R})) = B_{SP}(N, \mathbb{R})$$

where $B_{SP}(N, \mathbb{R}) := \{X \in \mathbb{R}^{N \times N} : \|X\|_F \leq 1\}$ is the unit spectral ball.

From Lemma 1, we know that the unit spectral ball is the minimal convex set containing the orthogonal group. Then, we propose the following relaxed problem for the online ODL:

$$\mathcal{P} : \min_{D \in B_{SP}(N, \mathbb{R})} F(D) = E_{\mathcal{Y}} - p\left[\|D^T y\|_3^p\right].$$

(5)

Problem $\mathcal{P}$ is a proper relaxation of Problem (4) since all the optimal solutions of Problem (4) belong to the set of the optimal solutions of Problem $\mathcal{P}$ under some general statistical model for $y$. We will show this relationship formally in the following.

We first introduce the following definition of the sign-permutation matrix for characterizing the optimal solutions.

**Definition 1** (Sign-Permutation Matrix): The $N$-dimensional sign-permutation matrix $\Sigma \in \mathbb{R}^{N \times N}$ is defined as

$$\Sigma = \Sigma \Pi$$

(6)

where $\Sigma = \text{diag}(\pm e_N)$ with $e_N$ the $N$-dimensional all-one vector, and $\Pi = [e_{\pi(1)}, e_{\pi(2)}, \ldots, e_{\pi(N)}]$, with $e_n$ being a standard basis vector and $[\pi(1), \pi(2), \ldots, \pi(N)]$ being any permutations of the $N$ elements.

Next, the relationship between the optimal solutions of Problem (4) and Problem $\mathcal{P}$ is formally presented in Theorem 1.

**Theorem 1** (Consistency of the Relaxation): If $y$ follows the distribution $P$ such that $y = D^\text{true}x$ with $D^\text{true} \in \mathbb{O}(N, \mathbb{R})$ and the entries of $x$ being i.i.d Bernoulli Gaussian,\(^4\)

\(^4\)The Bernoulli Gaussian is a typical statistical model for the sparse coefficient, and it is widely used for the analysis of dictionary learning [6]–[8].
Problem \( \mathcal{P} \) maintains the optimality of the minimizers of Problem (4), but the minimizers of Problem \( \mathcal{P} \) are not necessarily the minimizers of Problem (4). However, we can show that, if the dictionary in Problem \( \mathcal{P} \) is further restricted to be full rank, then the minimizers of Problem \( \mathcal{P} \) are also the minimizers of Problem (4) (see Appendix A). Fortunately, the full-rank condition holds, as shown from the extensive experiments in Section V. We believe the probability of encountering a rank-deficient instance is very low when adopting algorithms with full-rank dictionary initialization and randomness in updating the dictionary, e.g., the proposed Algorithm 2.

After the convex relaxation, we then focus on solving Problem \( \mathcal{P} \), which is a nonconvex optimization problem over a convex set. The convex set has a key property: it contains the convex combination of any two points in the set. That is to say, if we have \( A, B \in \mathbb{B}_{\text{sp}}(N, \mathbb{R}) \), then
\[
\eta A + (1 - \eta)B \in \mathbb{B}_{\text{sp}}(N, \mathbb{R}), \quad \eta \in (0, 1).
\]
This property enables an efficient online algorithm with a convergence guarantee, as we will illustrate in Section III.

### III. Online Nonconvex Frank–Wolfe-Based Algorithm

In this section, we first outline the proposed Frank–Wolfe-based algorithm, NoncvxSFW, for general online convex-constraint nonconvex problems, and then specialize it to solve Problem \( \mathcal{P} \).

#### A. NoncvxSFW for General Online Nonconvex Optimization

To solve a general online convex-constraint nonconvex problem
\[
\min_{x \in \mathcal{C}} F_{\text{gen}}(X) = \mathbb{E}_{y \sim P} \left[ f(X, y) \right]_{\text{nonconvex in } X}
\]
we require the algorithm to have the following properties.

1) **Computational Efficiency**: Efficient per-iteration computation.
2) **Theoretical Effectiveness**: Theoretical guarantee of convergence to a stationary point.

### Algorithm 1 NoncvxSFW for the General Nonconvex Problem

**Data**: \( \{Y_t\}_{t=1}^{\infty} \) with \( Y_t = [y_t^1, \ldots, y_t^M] \)

**Result**: \( \{X_t\}_{t=1}^{\infty} \)

**Initialization**: \( G_0 = 0 \) and random \( X_0 \in \mathcal{C} \)

for \( t = 1, 2, \ldots \)
do
1. **Gradient Approximation**: \( \rho_t = 4(t + 1)^{-1/2}, \gamma_t = (t + 2)^{-3/4} \)
2. **LMO**: \( S_t = \arg \min_{S \in \mathcal{C}} \langle G_t, S \rangle \)
3. **Variable Update**: \( X_t = \mathcal{P}((1 - \gamma_t)X_{t-1} + \gamma_t S_t) \)
done

To fulfill the above properties, we propose NoncvxSFW, as shown in Algorithm 1, which is a variant of the stochastic Frank–Wolfe method (SFW) in [23]. The SFW method and the corresponding analysis can only be applied to solve convex problems. However, NoncvxSFW and the analysis that we propose in this article are also applicable to nonconvex problems. In the following, we will elaborate on how the proposed NoncvxSFW algorithm satisfies the required properties.

1) **Computational Efficiency of General Nonconvex Optimization**: Algorithm 1 comprises three main steps.

1. **Step 1 (Gradient Approximation)**: It approximates the true gradient \( \nabla F_{\text{gen}}(X) \) with \( G_t \) in a recursive way. In the calculation, \( M_t \) can be fixed along all \( t \). Hence, compared to the methods in [28] and [29] that require an increasing number of stochastic gradient evaluations as the number of iterations \( t \) grows, NoncvxSFW is more computationally efficient.

2. **Step 2 (LMO)**: It is a procedure to handle the constraint. It can be regarded as solving a linear approximation of the objective function over the constraint set \( \mathcal{C} \) using the approximated gradient produced by Step 1. Compared to the quadratic minimization oracle (QMO) in the proximal-based methods, e.g., ProxSGD [21], the LMO can be more computationally efficient for many constraint sets, such as the trace norm and the \( \ell_p \) balls [30].

3. **Step 3 (Variable Update)**: It updates the variable \( X_t \) by a simple convex combination of \( S_t \in \mathcal{C} \) and \( X_{t-1} \in \mathcal{C} \), \( \mathcal{P}[X_t = (1 - \gamma_t)X_{t-1} + \gamma_t S_t] \), where \( \mathcal{P}[X] \) is any operation that satisfies \( \mathcal{P}[X] \in \mathcal{C} \) and \( F_{\text{gen}}(\mathcal{P}[X]) \leq F_{\text{gen}}(X) \). According to (8), we have \( X_t = (1 - \gamma_t)X_{t-1} + \gamma_t S_t \in \mathcal{C} \). This step only requires the output from Step 2, and the variable of the last iteration and automatically ensures the feasibility of the output \( X_t \).

2) **Theoretical Effectiveness of General Nonconvex Optimization**: To show the convergence of the NoncvxSFW, we first introduce the following Frank–Wolfe gap as the measure for the first-order stationarity.

**Definition 2 (Frank–Wolfe Gap)**: The Frank–Wolfe gap at the \( t \)th iteration, \( g^\text{gen}_{t} \), is defined as
\[
g^\text{gen}_{t} := \max_{S \in \mathcal{C}} \langle -\nabla F_{\text{gen}}(X_{t-1}), S - X_{t-1} \rangle.
\]
Lemma 2 (Frank–Wolfe Gap is a Measure for Stationarity): A point \( x_{t-1} \) is a stationary point for the optimization problem (9) if and only if \( g_t^{\text{gen}} = 0 \).

Proof: See Appendix B.

Then, we assume the following conditions hold for the general problem (9).

Assumption 1:

1) (Bounded Constraint Set): The constraint set \( C \) is bounded with diameter \( \text{diam}(C) \) in terms of the Frobenius norm for matrices, that is,

\[
\|X - Y\|_F \leq \text{diam}(C) \quad \forall X, Y \in C.
\] (11)

2) (Lipschitz Smoothness): \( F_{\text{gen}}(X) \) is L-smooth over the set \( C \), that is,

\[
\|\nabla F_{\text{gen}}(X) - \nabla F_{\text{gen}}(Y)\|_F \leq L\|X - Y\|_F
\] \quad \forall X, Y \in C. (12)

3) (Unbiased Minibatch Gradient): The minibatch gradient \( 1/M_t\sum_{j \in [M_t]} \nabla f(x_{t-1}, y_j^t) \) is an unbiased estimation of the true gradient \( \nabla F_{\text{gen}}(X) \), that is,

\[
\mathbb{E}\left[ \frac{1}{M_t}\sum_{j \in [M_t]} \nabla f(x_{t-1}, y_j^t) \right] = \nabla F_{\text{gen}}(X) \quad \forall t. (13)
\]

4) (Bounded Variance of the Minibatch Gradient): The variance of the minibatch gradient \( 1/M_t\sum_{j \in [M_t]} \nabla f(x_{t-1}, y_j^t) \) is bounded; i.e., for all \( t \), we have

\[
\mathbb{E}\left[ \left\| \frac{1}{M_t}\sum_{j \in [M_t]} \nabla f(x_{t-1}, y_j^t) - \nabla F_{\text{gen}}(X) \right\|_F^2 \right] \leq \frac{V}{M_t}.
\] (14)

The above assumptions ensure the convergence of NoncvxSFW, which is formally stated in Theorem 2.

Theorem 2 (Convergence of NoncvxSFW for the General Nonconvex Problem): If the conditions in Assumption 1 hold, using Algorithm 1 to solve the general problem (9), we have that the expected Frank–Wolfe gap converges to zero, in the sense that

\[
\inf_{1 \leq t \leq T} \mathbb{E}[g_t^{\text{gen}}] \leq c_1(\sqrt{\max\{C_0, C_1\}}\text{diam}(C) + L\text{diam}(C)^2)\ln(t + 2) + (t + 3)^{\frac{1}{2}}
\] (15)

where \( C_0 = \|\nabla F_{\text{gen}}(X_0)\|_F^2 \), \( C_1 = (4V/\min\{M_t\}) + 2L^2\text{diam}(C)^2 \), and \( c_1 \) is some positive constant. In other words, Algorithm 1 is guaranteed to converge to a stationary point of the general problem (9) at a rate of \( O(\ln(t)/t^{1/4}) \) in expectation.

Proof: See Appendix C.

B. NoncvxSFW for the Proposed Online ODL Problem

In this section, we will apply the proposed NoncvxSFW to solve the proposed online ODL Problem \( \mathcal{P} \). The algorithm is summarized in Algorithm 2. Similar to Section III-A, we will illustrate the NoncvxSFW for the proposed Online ODL problem from the computational and theoretical aspects.

Algorithm 2 NoncvxSFW for the Proposed Online ODL Problem

Data: \( \{Y_t\}_{t=1}^\infty \) with \( Y_t = \{y_1^t, \ldots, y_M^t\} \)

Result: \( \{D_t\}_{t=1}^\infty \)

Initialization: \( G_0 = 0 \) and random \( D_0 \in \mathcal{O}(N, \mathbb{R}) \subset \mathbb{R}_p(N, \mathbb{R}) \)

for \( t = 1, 2, \ldots \) do

1. Gradient Approximation:

\[
G_t = (1 - \rho_t)G_{t-1} + \frac{\rho_t}{M_t} \sum_{j \in [M_t]} -\nabla\|D_{t-1}^T y_j^t\|_3^3
\]

2. LMO: \( U, \Sigma, V^T = \text{SVD}(-G_t) \)

\[
S_t = UV^T
\]

3. Variable Update:

\[
D_t = \text{Polar}((1 - \gamma_t)D_{t-1} + \gamma_tS_t).
\]

1) Computational Efficiency of the Proposed Online ODL Problem: When adopting NoncvxSFW for Problem \( \mathcal{P} \), we can obtain a computationally efficient ODL algorithm whose complexity will not increase as \( t \) grows.

Step 1 (Gradient Approximation): It remains unchanged in Algorithm 1. The sampled gradient can be expressed as \( -\nabla\|D^T y_j^t\|_3^3 = -y_j^t((D^{(t-1)})^T y_j^t) \odot (D^{(t-1)}y_j^t)^T \). Hence, at each iteration, the time complexity and memory complexity of Step 1 are \( O(N^2M_t) \) (\( M_t \) can be fixed along all \( t \)) and \( O(N^2) \), respectively.

Step 2 (LMO): It is calculated based on Lemma 3.

Lemma 3 (LMO for the Unit Spectral Ball): The minimum value of \( (G, S), \forall S \in \mathcal{B}_p(N, \mathbb{R}) \) is the nuclear norm of \( -G \), that is,

\[
\min_{S \in \mathcal{B}_p(N, \mathbb{R})} (G, S) = \| -G \|_*.
\] (16)

The minimum is achieved when \( S \) belongs to the subdifferential of \( \| -G \|_* \), that is,

\[
S^* = \arg\min_{S \in \mathcal{B}_p(N, \mathbb{R})} (G, S) = UV^T \in \partial \| -G \|_*.
\] (17)

where \( U, \Sigma, V^T = \text{SVD}(-G) \).

Proof: We can prove Lemma 3 by simply using the definition of the dual norm and the subdifferential of the norm.

Using the LMO to deal with the constraint is much more computationally friendly than using the QMO in the proximal-based method. In the QMO, the proximal operator for the matrix spectral norm requires a proximal operator for the \( \ell_\infty \)-norm of the singular vector, which has no closed-form solution [31]. The calculation of the LMO can be simplified via the fact that \( -G = U\Sigma V^T = UV^T V^T = S^* V^T \).

This indicates that \( S^* \) can be calculated directly from the polar decomposition of \( -G \), which has many efficient calculations [32]. Using Coppersmith–Winograd matrix multiplication [4] in the calculation of polar decomposition, the time complexity and memory complexity of Step 2 are \( O(N^{2.38}) \) and \( O(N^2) \), respectively.

5The detailed deduction can be found in https://stephenhu.github.io/blog/convex-analysis/2014/10/01/subdifferential-of-a-norm.html
Step 3 (Variable Update): We further adopt polar decomposition, inspired by its projection property \cite[Proposition 3.4]{33}, which has \(O(N^2)\) time complexity and \(O(N^2)\) memory complexity. The following lemma shows that this is a valid specification of \(P\) in Algorithm 1.

Lemma 4: (Validation of Variable Update Step) Let \(X \in \mathbb{B}_{sp}(N, \mathbb{R})\). Then, we have Polar\((X) \in \mathbb{B}_{sp}(N, \mathbb{R})\) and \(F(\text{Polar}(X)) \leq F(X)\).

Proof: See Appendix D. \(\square\)

2) Theoretical Effectiveness of the Proposed Online ODL Problem: In this subsection, we will adapt the convergence theory in Theorem 2 to the proposed online ODL problem. We first show in Lemma 5 that the conditions in Assumption 1 are satisfied by Problem \(\mathcal{P}\).

Lemma 5 (The ODL Problem Satisfies the Convergence Condition): If \(y\) follows the distribution \(P\) such that, for all \(t\) and all \(j\), \(y_j^t = D_j^{\text{true}} x_j^t\), with \(D_j^{\text{true}} \in \mathbb{O}(N, \mathbb{R})\) and the entries of \(x_j^t\) being i.i.d Bernoulli Gaussian, \(x_j^t \sim \mathcal{B}\)\(\theta(\theta)\), then Problem \(\mathcal{P}\) satisfies the conditions in Assumption 1. Specifically, we have:

1) (Bounded Constraint Set)
\[
\|D_1 - D_2\|_F \leq \sqrt{2N} \quad \forall D_1, D_2 \in \mathbb{B}_{sp}(N, \mathbb{R}).
\] (18)

2) (Lipschitz Smoothness): \(F(D)\) is \(L\)-smooth over the set \(\mathbb{B}_{sp}(N, \mathbb{R})\), that is,
\[
\|\nabla F(D_1) - \nabla F(D_2)\|_F \leq \sqrt{\frac{2}{\pi}} N^{3/2} (N + 1) \theta \|D_1 - D_2\|_F \\
\forall D_1, D_2 \in \mathbb{B}_{sp}(N, \mathbb{R}).
\] (19)

3) (Unbiased Minibatch Gradient): The minibatch gradient \((1/M_t) \sum_{j \in [M_t]} -\nabla \|D_j^T y_j^t\|_3^3\) is an unbiased estimation of the true gradient \(\nabla F(D)\), that is,
\[
E \left[ \frac{1}{M_t} \sum_{j \in [M_t]} -\nabla \|D_j^T y_j^t\|_3^3 \right] = \nabla F(D) \quad \forall t.
\] (20)

4) (Bounding Variance of the Minibatch Gradient): The variance of the minibatch gradient \((1/M_t) \sum_{j \in [M_t]} -\nabla \|D_j^T y_j^t\|_3^3 - \nabla F(D)\|_F^2\) is bounded, i.e., for all \(t\), we have
\[
E \left[ \frac{1}{M_t} \sum_{j \in [M_t]} -\nabla \|D_j^T y_j^t\|_3^3 - \nabla F(D)\|_F^2 \right] \leq \frac{3\theta^4 N^2}{M_t}.
\] (21)

Proof: See Appendix E. \(\square\)

Based on Lemma 5 and Theorem 2, we have the convergence result for Algorithm 2, as shown in Theorem 3.

Theorem 3 (Convergence of NoncvxSFW for the Proposed Online ODL Problem): Using Algorithm 2 to solve Problem \(\mathcal{P}\), we have that the expected Frank–Wolfe gap
\[
E[g_t] := E \left[ \max_{S \in \mathbb{B}_{sp}(N, \mathbb{R})} (-\nabla F(D_{t-1}) - S - D_{t-1}) \right]
\] converges to zero, in the sense that, \(E[g_t]\), as shown at the bottom of the page, \(C_0 = \|\nabla F(D_0)\|_F^2\) and \(c_2\) is a positive constant.

Proof: The proof can be made by substituting the results in Lemma 5 into Theorem 2. \(\square\)

Remark 2 (Impact of the Key System Parameters): Theorem 3 suggests that a larger value of the smallest minibatch size \(\min|M_t|\), a smaller value of dictionary size \(N\), and a smaller sparsity level \(\theta\) (data become more sparse with a smaller \(\theta\)) will lead to a faster convergence speed. These conclusions are consistent with the simulation results in Section V-B. Though the above theorem only proves the convergence to stationary points, we have observed in experiments that the algorithm actually converges to the global optimum under very broad conditions, as shown in Section V-B. Similar phenomena have also appeared in many other works on off-line ODL \cite{6}–\cite{8}, \cite{19}, \cite{26}.

IV. APPLICATION EXAMPLES

In this section, we give two important application examples where the online ODL method should be adopted.

1) Example 1 (Online Data Compression on Edge Devices in the IoT Network \cite{34}):
Consider an IoT network architecture shown in Fig. 2, where the data are collected from smart objects, such as wearables and industrial sensor devices, and are sent periodically to an edge device using short-range communication protocols (e.g., Wi-Fi and Bluetooth). The edge device is responsible for low-level processing, filtering, and sending the data to the cloud. We assume that an edge device collects sensor data from different smart objects, including wearables, industrial sensor devices, and other IoT devices. The data are then compressed using the online ODL method to reduce the communication load and improve the efficiency of data transmission.
device is connected to a total number of \( N \) geographically distributed IoT sensors. When sensor measurements (temperature, humidity, concentration, and so on) are required by the cloud from the edge for the data analytics, the edge device transmits a compressed version of the data to save communication resources. At the \( t \)th time slot, the \( M_t \) samples of the sensor measurements from all \( N \) sensors, \( Y_t = \{y_{t,j}^1, \ldots, y_{t,j}^M\} \in \mathbb{R}^{N \times M_t} \), are transmitted to an edge device. When the \( j \)th sample from all \( N \) sensors, i.e., \( y_{t,j}^j \), is required by the cloud, the edge data compression is executed using the following steps.

1) \textbf{Preprocessing (Sparse Coding on the Edge):} The edge device calculates the sparse code \( \hat{x}_t^j \) based on the latest edge dictionary \( D_{t-1} \) and the input \( y_t^j \).

2) \textbf{Preprocessing (Transmission Content Decision on the Edge):} Upon obtaining the sparse code \( \hat{x}_t^j \), the edge device calculates the error between \( y_t^j \) and \( D_{t-1} \hat{x}_t^j \) using a certain error metric \( l(y_t^j, D_{t-1} \hat{x}_t^j) \), where \( D_{t-1} \) is a local copy of the latest cloud dictionary in the cloud. Then, the edge decides the content to transmit.
   a) If the error \( l(y_t^j, D_{t-1} \hat{x}_t^j) \) is larger than a predetermined threshold, the edge device updates its local cloud dictionary copy as \( D_{cloud} = D_{t-1} \) and transmits the updated \( D_{cloud} \) to the cloud in a compressed format. It also transmits the sparse code \( \hat{x}_t^j \) to the cloud in a compressed format.
   b) Otherwise, the edge transmits the sparse code \( \hat{x}_t^j \) to the cloud in a compressed format.

3) \textbf{Core Procedure (Online ODL on the Edge):} The edge device runs the online ODL method to produce \( D_t \) using \( D_{t-1} \) and the input \( Y_t \).

4) \textbf{Postprocessing (Sensor Data Recovery on the Cloud):} The cloud recovers the required data \( y_t^j \) by \( \hat{y}_t^j = D_{cloud} \hat{x}_t^j \), where \( \hat{x}_t^j \) is the sparse code received from the edge and \( D_{cloud} \) is the latest dictionary in the cloud.

The proposed ODL module produces the \( D_t, D_{t-1} \) and \( D_{cloud} \), which plays critical roles in the above example of data compression on IoT edge devices.

2) \textbf{Example 2 (Real-Time Novel Document Detection [35]):} Novel document detection can be used to find breaking news or emerging topics on social media. In this application, \( Y_t = \{y_t^1, \ldots, y_t^M\} \in \mathbb{R}^{N \times M_t} \) denotes the minibatch of documents arriving at time \( t \), where each column of \( Y_t \) represents a document at that time, as shown in Fig. 3. Each document is represented by a conventional vector space model, such as TF-IDF [36]. For the minibatch of documents \( Y_t \) arriving at time \( t \), the novel document detector operates using the following steps.

1) \textbf{Preprocessing (Sparse Coding):} For all \( y_t^j \) in \( Y_t \), the detector calculates the sparse code \( \hat{x}_t^j \) based on the latest dictionary \( D_{t-1} \) and the input \( y_t^j \).

2) \textbf{Preprocessing (Novel Document Detection):} For all \( y_t^j \) in \( Y_t \), the detector calculates the error between \( y_t^j \) and \( D_{t-1} \hat{x}_t^j \) with an error metric \( l(y_t^j, D_{t-1} \hat{x}_t^j) \).
   a) If the error \( l(y_t^j, D_{t-1} \hat{x}_t^j) \) is larger than some predefined threshold, the detector marks the document \( y_t^j \) as novel.
   b) Otherwise, the detector marks the document \( y_t^j \) as nonovel.

3) \textbf{Core Procedure (Online ODL):} The detector runs the online ODL method to produce the new dictionary \( D_t \) using \( D_{t-1} \) and the input \( Y_t \).

V. EXPERIMENTS

This section provides experiments demonstrating the effectiveness and the efficiency of our scheme compared to the state-of-the-art prior works. All the experiments are conducted in Python 3.7 with a 3.6-GHz Intel Core i7 processor.

A. List of Baseline Methods

The baseline methods are listed as follows.\footnote{The proposed method and Baselines 1 and 2 have no hyperparameters. For Baselines 3–5, grid search is adopted for tuning the hyperparameters. The grids are drawn around the hyperparameter values given in the baseline articles [4], [24], [25]. In the experiments with real-world sensor data, we extend the grids for \( \lambda \) to \([1,20]\) for Baselines 3 and 4 when \( \eta_0 = 2, 8, 10 \). We pick the hyperparameter with the best performance in hindsight for the online method, Baseline 3, in both the synthetic data and the real-world data experiments. For the off-line methods, Baselines 4 and 5, we adopt the walk-forward validation method [37] with 4593 testing data as the holdout data to pick the hyperparameters in the real-world data experiment.}

1) \textit{Baseline 1 (SFW) [23]:} This baseline adopts the recently proposed SFW algorithm to solve the online ODL problem \( \mathcal{P} \) in (5). We compare the proposed NoncvxSFW to this baseline in terms of the convergence property to show the effectiveness and efficiency of the proposed NoncvxSFW algorithm for solving the online ODL problem \( \mathcal{P} \).

2) \textit{Baseline 2 (\( \ell_4 \)-NoncvxSFW) [8]:} In this baseline, we replace the sparsity-promoting function \(-\|\cdot\|_3^3\) in the ODL problem \( \mathcal{P} \) with \(-\|\cdot\|_4^4\). Then, we solve the problem by the NoncvxSFW algorithm. This baseline is adopted to demonstrate the advantage of the choice of the negative \( \ell_3 \)-norm objective in the online ODL formulation.

3) \textit{Baseline 3 (Online AODL) [24]:} This baseline alternately learns the sparse code and the dictionary by...
solving the following optimization problem in an online manner
\[
\minimize_{\mathbf{D} \in \mathbb{R}^{N \times R}, \{x_t \in \mathbb{R}^R\}} \sum_{t=1}^{T} \|y_t - D x_t\|_F^2 + \lambda \|x_t\|_1. \quad (23)
\]

The online AODL is introduced to show the advantage of the online ODL scheme. The Python SPAMS toolbox is used to implement this baseline.\(^7\)

4) **Baseline 4 (Offline-DeepSTGDL)** [4]: This baseline is a recently proposed off-line dictionary learning method for behind-the-meter load and photovoltaic forecasting. In DeepSTGDL, a deep encoder first transforms the load measurements into a latent space, which captures the spatiotemporal patterns of the data. Then, a spatiotemporal dictionary and a sparse code are alternately learned to capture the significant spatiotemporal patterns for forecasting.

5) **Baseline 5 (Offline-NCBDL)** [25]: This baseline is an off-line nonparametric Bayesian approach in which the dictionary is inferred from a hierarchical Bayesian model based on the beta-Bernoulli process and Gibbs sampling.

**B. Experiments With Synthetic Data**

1) **Experiment Settings:** We evaluate the convergence property of different online dictionary learning methods with synthetic data. For all the experiments, we conduct 100 independent Monte Carlo trials. At the \(t\)th trial, we generate the measurements \(y_t(j) = D_{true}(j)x_t(j)\) \(j \in \{M_t\}, 1 \leq t \leq T\), with the ground-truth dictionary \(D_{true}(j)\) drawn uniformly randomly from the orthogonal group \(O(N, \mathbb{R})\), and with sparse signals \(x_t(j) \in \mathbb{R}^N\) drawn from an i.i.d. Bernoulli-Gaussian distribution, i.e., \(x_t(j) \sim \text{i.i.d.} \mathcal{BG} (\theta)\). For a fair comparison, all the methods share the same random initial point at each trial. Without loss of generality, we set \(M_t = B, \forall t\) and \(T = 3 \times 10^3\) for data generation.

To evaluate the convergence property, the error metric at time index \(t\) is calculated by
\[
\text{Error}_t = \frac{1}{100} \sum_{i=1}^{100} |1 - \frac{D_t^T(D_{true})^2}{N}|. \quad (24)
\]

This metric is an averaged measure for the difference between the dictionary learning result and the ground-truth dictionary since the true dictionary at the \(t\)th trial will be perfectly recovered if \(\left(\frac{(D_t^T(D_{true})^2)}{N}\right) = 1\) [8], [19]. We compare the Error, versus the number of iterations of the proposed method and the online baseline methods as follows.

2) **Convergence Comparison With Different System Parameters:** In Fig. 4, we show the convergence properties under different minibatch sizes \(B\) with a fixed dictionary size of \(N = 10\) and a sparsity level of \(\theta = 0.3\). The minibatch size \(B\) can accelerate the convergence of all methods, and the proposed method with \(B = 10\) can converge to an error at \(10^{-3}\) at around the 1000th time index, which is faster than the baselines.\(^8\)

In Fig. 5, the convergence curves with different dictionary sizes \(N\) are plotted. We fix the minibatch size at \(B = 10\) and the sparsity level at \(\theta = 0.3\). We tune \(\lambda\) for Baseline 3 at \(\lambda = 0.1\). The results show that all the methods have a faster convergence rate under a smaller dictionary size \(N\). In addition, the proposed method can achieve a smaller error with fewer iterations than the baselines.

In Fig. 6, we compare the convergence properties with different sparsity levels \(\theta\). The minibatch size and dictionary

\(^7\)Python code and documents are available at http://spams-devel.gforge.inria.fr/downloads.html

\(^8\)If \(t = \infty\), there should be no gap between the results for Baseline 2 and the proposed method under the same minibatch size. However, in the simulation, \(t = \infty\) is prohibitive. Therefore, when \(t\) is finite, the difference of the curves comes from the fact that the \(\ell_2\)-norm-based formulation has a lower sample complexity than the \(\ell_1\)-norm-based formulation [19]. Hence, when we have a finite number of samples \(t\) (is finite), the optimal point of the proposed formulation will be closer to the ground truth than the formulation in Baseline 2 and, therefore, shows a better performance.
The proposed method and the baselines have a faster convergence. Specifically, we set $\lambda = 0.1$ with $\theta = 0.3$ and $\lambda = 0.05$ with $\theta = 0.5$. The results show that both the proposed method and the baselines have a faster convergence rate with a more sparse $x$ (smaller $\theta$). Moreover, the proposed method achieves $10^{-3}$ error at around the 2000th time index with $\theta = 0.5$, which is faster than the baselines.

### C. Experiments With a Real-World Sensor Dataset

1) **Experiment Dataset**: We evaluate the performance of the IoT sensor data compression task with different dictionary learning methods. The experiments are carried out on the Airly network air quality dataset [38], which records temperature, air pressure, humidity, and the concentrations of particulate matter from 00:00, January 1, 2017, to 00:00, December 25, 2017. The sensor readings are measured by a network of 56 low-cost sensors located in Krakow, Poland, and each sensor has its own location. There are 8593 readings for each item from each sensor sampled per hour. In this work, we use the temperature readings from all the sensors as the input to the dictionary learning schemes. Since there is a missing data issue in the raw data, we replace the missing data with the mean readings over all the sensors at the times that data are missing. Fig. 7 displays the temperature readings that the dictionary learning schemes process.

2) **Performance Metrics**: At each time $t$, $M_t$ readings, $y^j_t \in \mathbb{R}^{56}$ ($j \in [M_t], 1 \leq t \leq T$), are uploaded to the dictionary learning processors and will be approximated by $\hat{y}^j_t$ ($j \in [M_t], 1 \leq t \leq T$) through calculations of the learned dictionary and the sparse code. To show the compression performance of the dictionary learning methods, two performance metrics are calculated under different compression ratios. The compression ratio is defined as

$$\text{compression ratio} = \frac{56}{\eta_0}$$

(25)

where $\eta_0$ is the number of nonzero values in the sparse code.

The first performance metric is the RMSE, which is defined by

$$\text{RMSE} = \sqrt{\frac{\sum_{t=0}^{T-1} \sum_{j=1}^{M_t} \| \hat{y}^j_t - y^j_t \|^2}{\sum_{t=0}^{T-1} \sum_{j=1}^{M_t} \| y^j_t \|^2}}$$

with $\hat{y}^j_t = \phi(\hat{D}^{est}_t, \hat{x}^j_t), \| \hat{x}^j_t \|_0 = \eta_0$

(26)

where $\phi(\cdot)$ is calculations of the dictionary and the sparse code determined by each method, as we will specify in Section V-C3, $\hat{D}^{est}_t$ is the dictionary for compression at time $t$, and $\hat{x}^j_t$ is the sparse code for the $j$th reading at time $t$ with $\eta_0$ nonzero values.

To provide more persuasive results, we have also calculated the HLN-corrected Diebold–Mariano (HLNMD) [39] test results to quantitatively evaluate the compression accuracy of different methods from a statistical point of view. Specifically, the HLNMD statistic is calculated by

$$\text{HLNMD} = \sqrt{\frac{T - 1 - 2h + h(h - 1)}{T}} \frac{\bar{d}}{f_d(0)}$$

(27)

where we set $h = 4$ as the time horizon in the experiments, and $\bar{d} = \sum_{t=1}^{T} d_t = \sum_{t=1}^{T} (\text{RMSE}_t - \text{RMSE}_2^2(t))$ is the average of the distance between the instantaneous RMSE produced by two different methods (method 1 and method 2) at time index $t$. Specifically, we have $d_t = \text{RMSE}_t^2(1) - \text{RMSE}_2^2(t)$ and

$$\text{RMSE}_t(1) = \sqrt{\frac{\sum_{j=1}^{M_t} \| \hat{y}^j_t(1) - y^j_t \|^2}{\sum_{j=1}^{M_t} \| y^j_t \|^2}}$$

(28)

with $\hat{y}^j_t(1) = \phi(\hat{D}^{est}_t(1), \hat{x}^j_t(1)), \| \hat{x}^j_t(1) \|_0 = \eta_0$

where $\hat{y}^j_t(1)$ is the estimated readings calculated from method 1 for the $j$th reading at time $t$. Furthermore, in (27), we have

$$\bar{d} = \sum_{k=-T}^{T-1} \left( \frac{k}{h - 1} \right) \frac{1}{T} \times \sum_{t=|k|+1}^{T} (d_t - \bar{d})(d_t - |k| - \bar{d})$$

(29)

where the indicator function is expressed as

$$\mathbb{I} \left( \frac{k}{h - 1} \right) = \begin{cases} 1, & \frac{k}{h - 1} \leq 1 \\ 0, & \text{otherwise} \end{cases}$$

(30)

To interpret the HLNMD statistic, we define the null hypothesis $H_0 : \mathbb{E}(d_t) = 0, \forall t$, which means that the two methods have the same compression accuracy in terms of statistics. The null hypothesis of no difference will be rejected if the computed HLNMD statistic falls outside the range of $-z_{\alpha/2}$ to $z_{\alpha/2}$, that is,

$$|\text{HLNMD}| > z_{\alpha/2}$$

(31)

where $z_{\alpha/2}$ is the upper (or positive) $z$-value from the standard normal table corresponding to half of the desired $\alpha$ significance level of the test. In the experiment, we set the proposed
method as the reference method 1 to calculate the HLNDM values for the baselines.

3) Experiment Procedures: In the experiments, both online and off-line methods are considered. We first introduce the experiment procedures for the off-line methods (Baseline 4 and Baseline 5).

1) Off-Line Training: The first 4000 temperature readings are used to train the weights and dictionaries in Baseline 4 and Baseline 5. For Baseline 4 [4], the first three terms in the training loss [4, eq.(4)] are considered, a spatiotemporal long short-term memory (ST-LSTM) is trained as the deep encoder \( f_e(x) \), and a four-layer rectified linear unit (ReLU) neural network is trained as the node decoder \( f_n(\cdot) \) with the Adam optimizer using 200 epochs. The edge decoder \( f_e(\cdot) \) follows the expression in [4, eq.(10)]. The hyperparameters are fine-tuned to be \( m = 6, K = 50, d_h = 20, \lambda_e = 0.25, \) and \( \lambda_n = 0.25 \) for better performance. In addition, the choices of \( \lambda \) are listed in Table I. For Baseline 5 [25], the upper bound of the number of atoms for the dictionary is set to \( K = 80 \), and the beta distribution parameters are set to \( a_0 = b_0 = (4000/8) \) for better performance. Other parameters have the same values as those in [25, Sec. V].

2) Compression: The remaining 4593 readings are grouped into 766 minibatches with a minibatch size of \( M_t = 6 \). Then, each minibatch of readings is provided to the offline dictionary learning methods at one specific time index to test the performance of the offline learned dictionary. Specifically, for Baseline 4, the sparse code for the \( j \)th test reading at time \( t \), \( y_j^t (j \in [M_t]) \), is calculated by solving the following sparse coding problem provided in [4]:

\[
\hat{x}_j^t = \arg\min_{x \in \mathbb{R}^n} \| f_n(x) \|_2^2 + \lambda \| x \|_1 \tag{32}
\]

with the sklearn Lasso toolbox.\(^{11}\) The estimated reading \( \hat{y}_j^t \) is calculated by \( \hat{y}_j^t = \phi(D_{\text{ext}}^t \hat{x}_j^t) = f_n(D_{\text{ext}}^t \hat{x}_j^t) \), where \( D_{\text{ext}}^t \) is equal to the off-line learned dictionary for all \( t, f_e(\cdot) \) and \( f_n(\cdot) \) are the off-line learned encoder and decoder, and \( \phi(y_j^t) \) is the graph embedding for \( y_j^t \) given by [4, Section II-B]. \( T_{\text{test}}(a) \) selects \( \eta_0 \) elements in \( a \) with the largest \( \ell_2 \) norm and sets the other elements to zero. For Baseline 5, the sparse code for the \( j \)th test reading at time \( t \), \( y_j^t (j \in [M_t]) \), is calculated by solving the following problem provided by [25]:

\[
\hat{x}_j^t = \arg\min_{x \in \mathbb{R}^n} \| y_j^t - D_{\text{ext}}^t x \|_2^2 \tag{33}
\]

with the sklearn OMP toolbox,\(^{12}\) and the estimated reading \( \hat{y}_j^t \) is calculated by \( \hat{y}_j^t = \phi(D_{\text{ext}}^t \hat{x}_j^t) = D_{\text{ext}}^t \hat{x}_j^t \), where \( D_{\text{ext}}^t \) is equal to the off-line learned dictionary for all \( t \).

3) Performance Metric Calculation: The RMSE values and the HLNDM test results under different compression ratios are calculated for Baseline 4 and Baseline 5 according to (26) and (27). Next, we elaborate on the experiment procedures for the online methods (the proposed method, Baseline 1, Baseline 2, and Baseline 3).

1) Initialization: Since there is no off-line training in the online methods, the last 4593 readings are provided to the online methods. The first 100 readings are utilized to initialize the dictionary \( D_t \) by the batch version of each online method with 20 iterations.

2) Online Learning: The remaining 4493 readings are grouped into 749 minibatches with a minibatch size of \( M_t = 6 \).\(^{13}\) The readings \( y_j^t (j \in [M_t]) \) at time \( t \) are provided to the online methods to produce \( D_t \), sequentially along \( t = 1, \ldots, 749 \).

3) Online Compression: Each reading in the minibatch is compressed by the sparse code after obtaining the dictionary \( D_t \). For the proposed method, Baseline 1, and Baseline 2, the sparse code for the \( j \)th reading at time \( t \), \( y_j^t (j \in [M_t]) \), is calculated using

\[
\hat{x}_j^t = T_{\text{test}}(D_t \hat{y}_j^t). \tag{34}
\]

For Baseline 3, the sparse code is

\[
\hat{x}_j^t = \arg\min_{x \in \mathbb{R}^n} \| y_j^t - D_t x \|_F^2 + \lambda \| x \|_1 \tag{35}
\]

For all online methods, the estimated reading \( \hat{y}_j^t \) is calculated by \( \hat{y}_j^t = \phi(D_{\text{ext}}^t, \hat{x}_j^t) = D_{\text{ext}}^t \hat{x}_j^t \).

4) Performance Metric Calculation: Then, the RMSE values and the HLNDM test results under different compression ratios are calculated for the proposed method, Baseline 1, Baseline 2, and Baseline 3 according to (26) and (27).

4) Performance Comparison and Discussion: The RMSE, HLNDM, and CPU time comparison among different dictionary learning schemes under different compression ratios are given in Table I. The results demonstrate that the proposed online ODL scheme achieves a lower RMSE than the other methods for the data compression task under different compression ratios. The proposed online ODL scheme keeps tracking the input readings and adapts the dictionary learning within a small optimization space. Hence, it is capable of finding a good dictionary with fewer streaming data received at each time index.

To interpret the HLNDM results, we adopt a widely used significance level of \( \alpha = 0.05 \) with \( z_{\alpha/2} = 1.96 \). The HLNDM statistic results and the RMSE results show that the proposed method has better sensor data compression performance in most cases. Though the HLNDM between the proposed method and Baseline 2 is \( 1.51 (\approx 1.96) \) when \( \eta_0 = 17 \), the probability that these two methods have the same level of accuracy is only 6.55%.

We count the per-time-slot CPU time to show the computational efficiency of the proposed method. For the off-line

\(^{10}\)The last minibatch of readings has the minibatch size \( M_{\text{test}} = 3 \).

\(^{11}\)Implementation details can be found at http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html

\(^{12}\)Implementation details can be found at https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.OrthogonalMatchingPursuit.html

\(^{13}\)The last minibatch of readings has the minibatch size \( M_{\text{test}} = 5 \).
methods, we count the per-time-slot CPU time on the test reading compression stage, i.e., we ignore the off-line training cost. In this case, the proposed online ODL method still costs fewer computational resources. This is because it only executes one iteration at each time index with simple thresholding of a few components.

Fig. 8 shows the compression results of the proposed online ODL method for 1000 temperature readings of the 45th wireless sensor from 21:00, October 27, 2017, to 12:00, December 08, 2017. Top to bottom rows: \( \eta_0 = 8, 17 \) and compression ratio = 3, 7.

D. Generalization Study

To demonstrate the generalization of the proposed Algorithm 1 based on the generic problem (9), we evaluate the performance of Algorithm 1 on the sparse principal component analysis (SPCA) problem [27], [40], which is a powerful framework to find a sparse principal component for seeking a reasonable tradeoff between the statistical fidelity and interpretability of data. Specifically, for the single-unit SPCA problem, one can have the following formulation:

\[
\min_{\mathbf{z} \in \mathbb{B}^N} \mathbb{E}_{\mathbf{y} \sim P} \left[-\mathbf{y}^T \mathbf{y} \mathbf{z} + \lambda H_\mu(\mathbf{z}) \right] 
\]

(36)

where \( \mathbb{B}^N \) is the \( N \) dimensional closed unit ball, which is a convex set, \( H_\mu(\cdot) \) is the Huber loss with parameter \( \mu \) to promote the sparsity of the principal component, and \( \lambda \) is a regularization parameter. The objective of Problem (36) is nonconvex. Problem (36) can be solved by the proposed generic Algorithm 1, with the LMO over the unit ball as

\[
s_i = \arg \min_{\mathbf{g}_i \in \mathbb{B}^q} \langle \mathbf{g}_i, \mathbf{s} \rangle = (-\mathbf{g}_i / \| \mathbf{g}_i \|_2)
\]

We test the convergence property of the proposed Algorithm 1 for Problem (36) in a number of experimental settings with synthetic data. In the experiments, we follow the procedures in [27], [40] to generate random data with a covariance matrix \( \Sigma = \mathbf{V} \mathbf{Y} \mathbf{V}^T \in \mathbb{R}^{N \times N} \) containing a sparse leading eigenvector. To do so, \( \Sigma \) is synthesized by constructing \( \mathbf{Y} \) and \( \mathbf{V} \). Specifically, we have \( \mathbf{Y} = \text{diag}(\mathbf{y}) \), where \( \mathbf{y} = [100, 1, 1, \ldots, 1] \). To synthesize \( \mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_N] \), we first construct the sparse leading eigenvector \( \mathbf{v}_1 \). Specifically, the \( i \)-th component of \( \mathbf{v}_1 \) is constructed by the following expression:

\[
u_{1,i} = \begin{cases} 
\frac{1}{\sqrt{q}}, & i = 1, \ldots, q \\
0, & \text{otherwise}
\end{cases}
\]

(37)

Next, vectors \( \mathbf{v}_2, \ldots, \mathbf{v}_N \) are randomly drawn from \( \mathbb{S}^{N-1} \) until a full-rank \( \mathbf{V} \) is obtained. Then, we apply the Gram–Schmidt orthogonalization method to the full-rank \( \mathbf{V} \) to obtain an orthogonal matrix \( \mathbf{V} \). Finally, \( T \) minibatches of data with minibatch size \( B \) are generated by drawing independent samples from a zero-mean normal distribution with covariance matrix \( \Sigma = \mathbf{V} \mathbf{Y} \mathbf{V}^T \). The resultant data samples are denoted by \( \mathbf{y}_j^i (j \in [B], 1 \leq i \leq T) \), where \( i \) is the index for the minibatch and \( j \) is the index for a sample within one minibatch.

---

### Table I

**Performance of Different Methods in Compressing Sensor Readings of Temperature in 2017 in Krakow, Poland [38]**

| Compression ratio | 28 (\( \eta_0 = 2 \)) | 7 (\( \eta_0 = 8 \)) | 5 (\( \eta_0 = 10 \)) |
|-------------------|-----------------------|---------------------|---------------------|
| Methods           | Time (ms)             | RMSE                | HLNDM               | Time (ms)             | RMSE                | HLNDM               | Time (ms)             | RMSE                | HLNDM               |
| Proposed          | 1.018                 | 4.82%               | Ref                 | 1.021                 | 2.74%               | Ref                 | 1.034                 | 2.53%               | Ref                 |
| Baseline 1 (SPW)  | 1.022                 | 22.53%              | 7.56                | 1.030                 | 14.78%              | 3.57                | 1.034                 | 13.90%              | 3.48                |
| Baseline 2 (\( \ell_0 \)-NonevxxSPW) | 1.076 | 5.22% | 6.212 | 1.083 | 3.16% | 2.65 | 1.088 | 3.04% | 2.70 |
| Baseline 3 (Online AODL) | 8.932 | 47.45% | 4.38 | 19.58 | 25.06% | 3.29 | 39.65 | 24.71% | 2.11 |
| Baseline 4 (Offline-DeepSTGDL) | 4.606 | 24.8% | 2.42 | 10.14 | 16.6% | 5.47 | 18.04 | 7.27% | 3.41 |
| Baseline 5 (Offline-NCBDL) | 2.717 | 34.72% | 5.94 | 4.028 | 24.87% | 6.17 | 4.347 | 24.72% | 6.51 |

* For \( \eta_0 = 2, 8, 10, 17, 25, 35 \), the hyper-parameter \( \lambda \) for Baseline 3 is set to \( 18, 5, 1, 0.3, 0.2, 0.1 \), and \( \lambda \) for Baseline 4 is set to \( 20, 8, 4, 0.4, 0.4, 0.2 \), for better performance.
The target for the algorithm is to find the sparse leading principal component $v_l$ with the streaming input $y_t^j (j \in [B], 1 \leq t \leq T)$ by solving Problem (36). We set $\mu = 0.2$ and $\lambda = 1$ for the problem and generate $T = 3 \times 10^3$ minibatches of data from the above procedures. In the experiments, we test the recovery error by varying the dimensions $N$, the minibatch size $B$, and the number of nonzero elements in the principal component $q$. The recovery error is defined by

$$\text{Error}_t = |1 - |z_T^Tv_l||$$

since the maximum of $|z_T^Tv_l|$ is 1, and it is achieved when $z_t = v_l$. The results with Error, against the number of iterations are shown in Fig. 9, where Error, at each $t$ is averaged over 100 independent Monte Carlo trials. As indicated by the results under various experimental settings, the proposed method for a generic problem (9) generalizes well to this SPCA problem.

VI. CONCLUSION

We have proposed in this article a novel online ODL scheme with a new relaxed problem formulation, a Frank–Wolfe-based online algorithm, and convergence analysis. Experiments on synthetic data and a real-world dataset show the superior performance of our proposed scheme compared to the baselines and verify the correctness of our theoretical results. Future focus on the application will include a distributed or federated version of the proposed scheme to further leverage the spatially distributed data. As for the theory, the sample complexity analysis of the proposed scheme will be an important direction since it characterizes how close the result returned by the proposed scheme at a given iteration is to the ground-truth dictionary.

APPENDIX A

PROOF OF THEOREM 1

Let $W = D^TD$ true $\in \mathbb{R}^{N \times N}$. We have

$$\mathbb{E}_{y \sim \mathcal{BG}(\theta)}[-\|D^Ty\|_3^3] = -\sum_{n=1}^{n=N} \mathbb{E}[|W_n \cdot x|^3] = -\sum_{n=1}^{n=N} \mathbb{E}[|(W_n \cdot b^T)g|^3]$$

where we denote $b \sim_{i.i.d} \mathbb{E}[\theta]$ and $g \sim_{i.i.d} \mathcal{N}(0, 1)$. Using the rotation-invariant property of Gaussian random variables, we have $-\mathbb{E}[|W_n \cdot b^T g|^3] = -\gamma_1 \mathbb{E}[|W_n \cdot b^T g|^3]$, with $\gamma_1 = (2^{3/2}/(\pi)^{1/2})$ calculated by the third-order noncentral moment of the Gaussian distribution.

For Problem (4), we have $D \in \mathbb{R}(N, \mathbb{R})$ and $\|W_n\|_2 = \|D^TD\text{ true}\|_2 \leq 1$. Hence, we have $0 \leq \mathbb{E}[|W_n \cdot b^T|^3] \leq \mathbb{E}[|W_n \cdot b^T|^3] = \theta$. The equality holds if and only if $|W_n \cdot b^T| = 0$ for all $b$, which is only satisfied at $W_n : = \{\pm e_i^T : i \in [N]\}$ [7]. Therefore, we have $\mathbb{E}_{y \sim \mathcal{BG}(\theta)}[-\|D^Ty\|_3^3] = -\sum_{n=1}^{n=N} \mathbb{E}[|W_n \cdot b^T|^3] \geq -N\gamma_1 \theta$, and the minimum is achieved when $W_n : = \{\pm e_i^T : i \in [N]\}$. Furthermore, if $W_{n_1} = \pm e_i^T$ and $W_{n_2} = \pm e_i^T$, then $\text{Tr}(W_{n_1}W_{n_2}^T) = \pm 1$. However, we have $\text{Tr}(W_{n_1}W_{n_2}^T) = \text{Tr}(D^TD\text{ true}) = \text{Tr}(D^TD\text{ true}) = \text{Tr}(D^TD) = 0$. This indicates that two different columns of $W$ cannot simultaneously equal the same standard basis vector (up to a sign difference). Hence, $\mathbb{E}_{y \sim \mathcal{BG}(\theta)}[-\|D^Ty\|_3^3]$ achieves the minimum $-N\gamma_1 \theta$ when $D = D^{\text{opt}}$ with $D^{\text{opt}} = \mathbb{E}[\theta] = \mathbb{E}[\mathcal{N}(0, 1)]$. In other words, the optimal solution of Problem (4) is $D^{\text{opt}} = D\text{ true}$.

For Problem $\mathcal{P}$, we have $D \in \mathbb{R}_p(N, \mathbb{R})$, but $\|W_n\|_2 = \|D^TD\text{ true}\|_2 \leq 1$ still holds. Hence, we also have $\mathbb{E}_{y \sim \mathcal{BG}(\theta)}[-\|D^Ty\|_3^3] = -\sum_{n=1}^{n=N} \mathbb{E}[|W_n \cdot b^T|^3] \geq -N\gamma_1 \theta$, and the minimum is achieved when $W_n : = \{\pm e_i^T : i \in [N]\}$. Since $D^{\text{opt}} = D\text{ true}$ for Problem $\mathcal{P}$ is feasible for Problem $\mathcal{P}$, $D^{\text{opt}}$ is also an optimal solution for Problem $\mathcal{P}$. This finishes the proof of Theorem 1.

Now, we will demonstrate that, if the minimizers of Problem $\mathcal{P}$ are restricted to being full-rank, then they are also the minimizers of Problem (4). The minimum of Problem $\mathcal{P}$ is achieved when $W_{n_1} = \pm e_i^T$ and $W_{n_2} = \pm e_i^T$, then $\text{Tr}(W_{n_1}W_{n_2}^T) = 1$. However, if we have $D \in \mathbb{R}_p(N, \mathbb{R})$ and $D$ being full-rank, then $\text{Tr}(W_{n_1}W_{n_2}^T) = \text{Tr}(D^TD\text{ true}) \leq 1$. This indicates that two different columns of $W$ cannot simultaneously equal the same standard basis vector (up to a sign difference). Hence, the minimizers of Problem $\mathcal{P}$ (with additional full-rank requirement) are also the minimizers of Problem (4).

APPENDIX B

PROOF OF LEMMA 2

The definition of the stationary points for a constraint problem is given as follows.

**Definition 3 (Definition of Stationary Points):** We will say that $X^* \in \mathcal{C}$ is a stationary point if

$$(\nabla \Phi_{\text{gen}}(X^*)), S - X^* \geq 0 \quad \forall S \in \mathcal{C}.$$ 

Then, we first show the sufficient condition. If $g_{t}^{\text{gen}} = 0$, we have $\min_{S \in \mathcal{C}} \langle \nabla \Phi_{\text{gen}}(X_{t-1}), S - X_{t-1} \rangle = 0$. According to Definition 3, it is obvious that $X_{t-1}$ is a stationary point.

Next, we show the necessary condition. If $\langle \nabla \Phi_{\text{gen}}(X_{t-1}), S - X_{t-1} \rangle \geq 0, \forall S \in \mathcal{C}$, then we have $\langle -\nabla \Phi_{\text{gen}}(X_{t-1}), S - X_{t-1} \rangle \leq 0, \forall S \in \mathcal{C}$, which indicates

$$\max_{S \in \mathcal{C}} \langle -\nabla \Phi_{\text{gen}}(X_{t-1}), S - X_{t-1} \rangle = g_{t}^{\text{gen}} \leq 0.$$ (40)
Let $S^* = \arg \max_{S \in \mathcal{C}} (-\nabla F_{\text{gen}}(X_{t-1}, S))$. Then, we have $g^\text{gen}_t = (-\nabla F_{\text{gen}}(X_{t-1}, S^* - X_{t-1}) \geq 0$. Combining this with the result in (40), we have that if $\langle \nabla F_{\text{gen}}(X_{t-1}, S) - X_{t-1} \geq 0, \forall S \in \mathcal{C}$, then $g^\text{gen}_t = 0$.

The sufficient and necessary conditions complete the proof.

**APPENDIX C**

**PROOF OF THEOREM 2**

For simplicity, we let $F(\cdot)$ represent $F_{\text{gen}}(\cdot)$. To prove Theorem 2, we first prove the *Iterates Contraction* by the following Lemma.

**Lemma 6 (Iterates Contraction):** Under Assumptions 1, the Frank–Wolfe gap for Problem (9) using Algorithm (1) satisfies

$$
\gamma_t g^\text{gen}_t \leq F(X_t) - F(X_{t+1}) + \gamma_t \sqrt{e} \text{diam}(C) + \frac{L}{2} \gamma_t^2 \text{diam}(C)^2
$$

where $N$ is the dimensions of the subspace that $\mathcal{C}$ belongs to and

$$
e_t := \|G_t - \nabla F(X_{t-1})\|_F^2
$$

is the gradient estimation error.

**Proof:** We introduce the auxiliary variable

$$S^*_t := \arg \max_{S \in \mathcal{C}} (-\nabla F(X_{t-1}), S)
$$

for the proof. At the $t$th iteration, we have

$$F(X_t) \leq F(X_{t-1} + \gamma_t (S_t - X_{t-1}))
$$

$$\leq \left[ (a) \right] F(X_{t-1}) + \gamma_t \|\nabla F(X_{t-1}) - G_t\|_F \|S_t - S^*_t\|_F
$$

$$- \gamma_t g^\text{gen}_t + \frac{L}{2} \gamma_t^2 \|S_t - X_{t-1}\|_F^2
$$

$$= F(X_{t-1}) + \gamma_t \sqrt{e} \text{diam}(C) - \gamma_t g_t + \frac{L}{2} \gamma_t^2 \text{diam}(C)^2
$$

$$\Rightarrow \gamma_t g_t \leq F(X_{t-1}) - F(X_t) + \gamma_t \sqrt{e} \text{diam}(C)
$$

$$+ \frac{L}{2} \gamma_t^2 \text{diam}(C)^2
$$

where (a) is from Assumption 1, Algorithm (1), and the Cauchy–Schwarz inequality. □

The next key ingredient for the proof is the diminishing gradient estimation error, as formally shown in the following lemma.

**Lemma 7 (Diminishing Gradient Estimation Error):** Let the gradient estimation error at the $t$th iteration in Algorithm (1) be defined as

$$e_t := \|G_t - \nabla F(X_{t-1})\|_F^2.
$$

Using the updating rule of Algorithm (1), we have

$$E[\epsilon_t] \leq \frac{C_0 + C_1}{(t + 2)^{1/2}}
$$

where $C_0 = \|\nabla F(X_0)\|_F^2$ and $C_1 = (16V/M_t) + 2L^2 \text{diam}(C)^2$.

**Proof:** To prove Lemma 7, we have the following bound with $E_t$ the conditional expectation w.r.t. the randomness sampled at the $t$th iteration conditioned on all randomness up to the $t$th iteration

$$E_t \left[ \|G_t - \nabla F(X_{t-1})\|_F^2 \right]
$$

$$= E_t \left[ \|G_t - \nabla F(X_{t-1}) - \frac{1}{|M_t|} \sum_{i \in M_t} \nabla f_i(X_{t-1}, y_i) \right]
$$

$$+ (1 - \rho_t) (\nabla F(X_{t-1}) - G_{t-1})^2_F
$$

$$+ \frac{L^2}{|M_t|} + \frac{2}{\rho_t} \gamma_t^2 L^2 \text{diam}(C)^2
$$

$$+ (1 - \rho_t)(1 + \frac{L}{\rho_t}) \leq (1 - \rho_t)(1 + \frac{L}{\rho_t}) \leq \frac{2}{\rho_t}.
$$

Therefore, we have

$$E[\epsilon_t] = E_{t_0, \ldots, t}[\epsilon_t]
$$

$$\leq \left( 1 - \frac{4(t + 1)^{-1/2}}{2} \right) E_{t_0, \ldots, t-1}[\epsilon_{t-1}]
$$

$$+ \left( 2L^2 \text{diam}(C)^2 + 16 \frac{V}{M_t} \right) (\eta_0 + t)^{-1}.
$$

Then, using the result in [23, Lemma 17], we have finished the proof. □

We are now able to prove Theorem 2. Based on Lemma 6, Lemma 7, and Jensen’s inequality, we have

$$E[\gamma_{t+1} g^\text{gen}_t] \leq E\left[ F(X_{t-1}) - F(X_t) \right] + \gamma_t \sqrt{e} E[\epsilon_t] \text{diam}(C) + \frac{L}{2} \gamma_t^2 \text{diam}(C)^2.
$$

Define $C_v = \max F(X) - \min F(X)$. We have

$$\inf_{1 \leq t \leq T} \frac{1}{e_t} \sum_{s=1}^{t} \gamma_s \leq C_v + \sum_{s=1}^{t} \gamma_s \sqrt{\max \left\{ C_0, C_1 \right\}} \text{diam}(C)
$$

$$+ \frac{L}{2} \gamma_t^2 \text{diam}(C)^2.
$$
Hence, we have
\[
\inf_{1 \leq s \leq t} \mathbb{E}[g_s] \leq \left( C_s + \sum_{i=1}^{t} \left( (s+2)^{-1} \left( 2 \sqrt{\max\{C_0, C_1\}} \text{diam}(C) + 2L\text{diam}(C)^2 \right) \right) \right) \left( \sum_{i=1}^{2} (s+2)^{-3/4} \right).
\]
(52)
Since both \((s+2)^{-1}\) and \((s+2)^{-3/4}\) are decreasing in terms of \(s\), we have
\[
\inf_{1 \leq s \leq t} \mathbb{E}[g_s] \leq \left( 5C_s + 10 \ln(t+2) \left( \sqrt{\max\{C_0, C_1\}} \text{diam}(C) + L\text{diam}(C)^2 \right) / (4(t+3)^{1/4}) \right).
\]
(53)
This finishes the proof.\[^{14}\]

**Appendix D**

**Proof of Lemma 4**

Define \(\text{Polar}(D)\) as \(D^O\). Let \(W = D^T D^\text{true} \) and \(W^O = (D^O)^T D^\text{true} \). It is easy to determine that \(W^O\) has orthonormal rows, while rows of \(W^O\) lie in the unit ball. Using similar calculations to those in Appendix A, we have
\[
-\sum_{n=1}^{N} \mathbb{E}[||W^O_n \circ b^T||_2^2] \leq -\sum_{n=1}^{N} \mathbb{E}[||W_n \circ b^T||_2^2].
\]
(54)
Hence, we have \(F(\text{Polar}(D)) \leq F(D)\). This completes the proof.

**Appendix E**

**Proof of Lemma 5**

We first show that condition (1) is satisfied \(\forall D_1, D_2 \in \mathbb{B}_p(N, \mathbb{R})\), and we have
\[
\|D_1 - D_2\|_F = \sqrt{\text{Tr}((D_1 - D_2)^T (D_1 - D_2))} \leq \sqrt{\text{Tr}(D_1^T D_1 + D_2^T D_2)} \leq \sqrt{N + N} = \sqrt{2N}.
\]
(55)
For condition (2), we define \(d = \text{vec}(D)\) and \(f_o(d) = f(\text{vec}(D)) = \nabla F(D) = \nabla \mathbb{E}_{y \sim p[-\|D^T y\|_2^2]} \). Since we have \(\|D_1 - D_2\|_2 = \|d_1 - d_2\|_2\), the following holds:
\[
\|\nabla f_o(d_1) - \nabla f_o(d_2)\|_2 \leq \|D_d[d_o(d_1)]\|_2 \leq \sqrt{\sum_{i=1}^{N^2} \sum_{j=1}^{N^2} (D_d[d_o(d_1)]_{i,j})^2} \]
(56)
where \(D_d[d_o(d)]\) is the differential of \(f_o(d)\) in terms of \(d\).
To prove condition (2), we only need to show that
\[
\sqrt{\sum_{i=1}^{N^2} \sum_{j=1}^{N^2} (D_d[d_o(d_1)]_{i,j})^2} \leq \sqrt{2} N^{3/2}(N+1) \theta.
\]
(57)
Letting \(W = D^T D^\text{true} \) and using the strategy in [7, B.1], we have the \(j\)-th element of \(f_o(d)\) being
\[
f_o(d_1)_{j} = \mathbb{E}_Q \left[ D^\text{true}_{\cup N^2} \left( \left\| \frac{W^Q}{|x|^{\gamma}} \right\|_{\ell_{2,2}}, \left\| \frac{W^Q}{|x|^{\gamma}} \right\|_{\ell_{2,2}} \right)^T \right]
\]
(58)
where \(\Omega\) is used to denote the generic support set of the Bernoulli random variable contained in \(x\) with \(y = D^\text{true} x\). Then, we have the \(i\)-th element in \(D_d[f_o(d_1)]\) as
\[
(D_d[f_o(d_1)]_{i,j})^2 \leq \left( \mathbb{E}_Q \left[ \left\| \frac{W^Q_{\ell_{2,2}}}{|x|^{\gamma}} \right\|_{\ell_{2,2}} + \left\| \frac{W^Q_{\ell_{2,2}}}{|x|^{\gamma}} \right\|_{\ell_{2,2}} \right] \right)^2
\]
if \(N\left( \frac{1}{\theta} - 1 \right) + 1 \leq i \leq N\left( \frac{1}{\theta} \right)
\]
(59)
Combining (60) with (59), we have (57). This finishes the proof.
Condition (3) holds obviously due to the i.i.d assumption of \(x_t\).
To prove that condition (4) is satisfied, we have
\[
\mathbb{E} \left[ \left\| \frac{1}{M_t} \sum_{j \in [M_t]} -\nabla \|D^T y_j\|_3^2 - \nabla F(D) \right\|_F^2 \right]
\]
\[
\leq \mathbb{E} \left[ \left\| \frac{1}{M_t} \sum_{j \in [M_t]} -\nabla \|D^T y_j\|_3^2 \right\|_F^2 \right]
\]
\[
= \frac{1}{M_t^2} \sum_{j \in [M_t]} \mathbb{E} \left[ \left( \nabla \|D^T y_j\|_3^2 \right)^T \left( \nabla \|D^T y_j\|_3^2 \right) \right]
\]
(61)
where (a) is due to the fact that matrix \(\nabla \|D^T y_j\|_3^2\) is rank one and the Cauchy–Schwarz inequality. Let \((D^\text{true})^T D = W\) and \(W_{i,:}\) represent the \(i\)-th column vector in \(W\). We have
1) \[
\mathbb{E} \left[ \left\| x^T (D^\text{true})^T D \right\|_{\ell_{2,2}}^4 \right] = \mathbb{E} \left[ \left( \sum_{i=1}^{N} \|x^T W_{i,:}\|_4^2 \right)^2 \right]
\]
\[
= \sum_{i=1}^{N} \|x^T W_{i,:}\|_4^4 + \mathbb{E} \left[ \sum_{i \neq i'} \|x^T W_{i,:}\|_4^2 \|x^T W_{i',:}\|_4^2 \right]
\]
\[
= \sum_{i=1}^{N} 3\theta + \sum_{i \neq i'} 3\theta = N^2 \theta
\]
(62)
where the last inequality is from [19, Lemma B.1].
2) Let \(D_{\text{true}(i,:)}\) represent the \(i\)-th row vector in \(D^\text{true}\). We have
\[
\mathbb{E} \left[ \left\| D^\text{true} x \right\|_1^4 \right] = \mathbb{E} \left[ \left( \sum_{i=1}^{N} \|D_{\text{true}(i,:)} x\|_1^2 \right)^2 \right]
\]
\[
\leq \sum_{i=1}^{N} 3\theta + \sum_{i \neq i'} 3\theta = N^2 \theta.
\]
(63)
Substituting (62) and (63) into (61), we finish the proof.

\[^{14}\text{Full-version proof can be found in https://arxiv.org/abs/2103.01484}\]
