University of California

Los Angeles

Online Nonnegative CP-Dictionary Learning with Applications to Multimodal Data

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Mathematics

by

Christopher William Strohmeier

2023

Abstract of the Dissertation

Online Nonnegative CP-Dictionary Learning with Applications to Multimodal Data

by

Christopher William Strohmeier
Doctor of Philosophy in Mathematics
University of California, Los Angeles, 2023
Professor Deanna Needell, Chair

In this thesis, we present a myriad of applications of online matrix and tensor dictionary learning algorithms to the analysis of time series and image data, as well as a theoretical analysis of our algorithm, Online CP-Dictionary Learning (OCPDL). First, we present a method which applies online nonnegative matrix factorization (ONMF), an algorithm which learns a sparse, nonnegative representation of streaming data, to perform joint dictionary learning on multivariate COVID-19 time-series data, followed by a certain "restrict and predict" algorithm to tackle the future time regression problem. Next, we apply ONMF to meteorological time series data, as well as to video data, and demonstrate the particular utility of online as opposed to offline algorithms in dealing with said data. In the following section, we present our extension of ONMF, our OCPDL algorithm, as well as a proof of some convergence guarantees.

The dissertation of Christopher William Strohmeier is approved.

Stanley Osher

Guido Montúfar

Chris Anderson

Deanna Needell, Committee Chair

University of California, Los Angeles

2023

To my parents,

without whose love and support this thesis

would not have come to fruition.

Table of Contents

List of Figures

Acknowledgments

I would like to take this opportunity to extend my heartfelt thanks to all of the faculty, friends, and family who have helped me along the way to completing my PhD.

First, I would like to thank my advisor, Deanna Needell. Deanna has always gone above and beyond to support her students. At times when I was really struggling during the program, she provided the guidance necessary to overcome the many obstacles which come with research and the struggles with the uncertainty that come with them. Moreover, at what was a difficult time for everyone during the COVID19 pandemic, Deanna went out of her way to ensure the well-being of her students, fostering what is surely one of the most inclusive and supportive research groups anywhere.

I would also like to thank the other members of my ATC committee, Chris Anderson, Guido Montúfar, and Wotao Yin, for there helpful comments and suggestions during my Advancement to Candidacy. Additionally, I would like to thank Stanley Osher for joining my thesis defense committee.

On a more technical note, it is no exaggeration to say that the works which constitute this thesis could not have come to fruition without the joint effort of my primary collaborators, my advisor Deanna Needell and her former postdoc Hanbaek Lyu, as well as professor Georg Menz.

Furthermore, I would like to thank all of my friends who dragged me out of my sad, depressing basement office over the years. Stan Palasek, I've enjoyed our conversations about partial differential equations as much as I have eating nachos and playing Evans tennis in the hallway. Abigail Hickock, words cannot express how appreciative I am to have a friend almost as clumsy and gullible as I am. Big thanks to my friend Yonatan Duckler; may our continued dialogue on everything from AI to consciousness extend well into the future. I'd also like to thank my friend Alex Frederick, whom I have known since we were both snotty undergraduates at Cornell. Last but not least, a big shout-out to my "stats friends" Sam Baugh, Sidney Kahman, and Stephanie Stacy. Ya'll were my first non-math friends at

UCLA; our game-nights and cooking misadvenures were a highlight of my time in graduate school.

Finally, I would like to thank my mom and dad. You've been there for me every step of the way, always encouraging me to never give up. At times when I've been really struggling, you were always there to pick up the phone. Love you guys!

Vita

2014-2017	Mathematics Tutor, Mathematics Support Center, Cornell University, Ithaca, New York.
2017	B.A. (Mathematics), Cornell University.
2020	M.A. (Mathematics), UCLA, Los Angeles, California.
2017-2022	Teaching Assistant, Mathematics Department, UCLA.
2019-2023	Research Assistant, Mathematics Department, UCLA.

Publications

Hanbaek Lyu, Georg Menz, Deanna Needell, Christopher Strohmeier, "Applications of Online Nonnegative Matrix Factorization to Image and Time-Series Data." 2020 Information Theory and Applications Workshop, 2020.

Hanbaek Lyu, Christopher Strohmeier, Georg Menz, Deanna Needell, "COVID-19 Time-Series Prediction by Joint Dictionary Learning and Online NMF" https://arxiv.org/abs/2004.09112

Hanbaek Lyu, Christopher Strohmeier, Deanna Needell, "Online Nonnegative CP-dictionary Learning for Markovian Data." JMLR 23(148):1-50, 2022

CHAPTER 1

Introduction

Dictionary learning is a relatively classical field of applied mathematics and signal processing which consists of uncovering sparse, linear representations of data by learning a set of vectors which constitute a so-called "dictionary", to efficiently and accurately represent data. A close relative of matrix factorizaton, dictionary learning differentiates itself in that the learned dictionary can be used to represent data distinct from its input data.

The two primary components of a dictionary learning algorithm are the coding step, in which one represents a given minibatch of data as a linear combination of dictionary elements, often referred to as "atoms," and the dictionary update step, in which one seeks to modify the atoms in such a way as to optimize their capacity to represent the data. Just what this means depends heavily on context. At a minimum, the learned dictionary should be such that the average reconstruction error over the data set is small. To be more concrete, if we denote our dictionary by $D \in \mathbb{R}^{m \times r}$ and define for a minibatch of data $X \in \mathbb{R}^{m \times b}$ the optimal coding matrix with sparsity regularization parameter $\lambda > 0$

$$C^{opt}(D, X) = \arg\min_{C \in \mathcal{C}} \frac{1}{2}||X - DC||_2^2 + \lambda ||C||_1$$

, then a typical dictionary learning problem would be to obtain

$$D = \arg\min_{D \in \mathcal{D}} \mathbb{E}_{X \sim p} \frac{1}{2}||X - DC^{opt}(D, X)||_2^2$$

where $\mathcal{C}, \mathcal{D}$ are the constraint sets for the codes and dictionary, respectively, and p is the (virtually always unknown) underlying data distribution. Although the choice of loss function can differ from the standard l2-loss, this is by far the most common.

There are, however, criteria which one often want to use to evaluate the utility of a learned dictionary beyond its capacity to minimize reconstruction error. The interpretability of the representation, which for sufficiently sparse representations reduces the the interpretability of the learned atoms, is also a chief concern. Indeed, in an age where deep learning has seen an explosion of interest throughout the computational sciences, an almost universal complaint among researchers hoping to gain insight into the form of "reasoning" of deep learning algorithms is the lack of such interpretability of the learned representations. It is here that structured dictionary learning, i.e. dictionary learning in which one chooses nontrivial constraints on the codes or dictionaries, is remarkably effective. Among the most common choice of constraints is that of nonnegativity, in which both the entries of the codes and dictionary atoms are constraied to be nonnegative. The benefit of this constraint is that the absence of cancellation implies that once an atom has contributed to reconstructing a data point, the whole atom must contribute. For many applications, and image and time series applications in particular, this forces the learned atoms to be well-localized.

Beyond the choice of constraint set, one can glean new insights from data by considering tensor dictionaries. In the present context, a tensor dictionary is perfectly analogous to an ordinary dictionary, the chief difference being that the learned atoms are allowed to be a tensor of arbitrary modality, as opposed to merely vectors (1-mode tensors). This is an important generalization, as many datasets consist of data which have truly distinct modalities, e.g. location attributes, pixel intensities, time. In general, one wants to preserve the tensorial structure, and so the typical practice of vectorizing the data and learning an ordinary dictionary is often not ideal. It is also computatonally very expensive whenever the product of the dimensions of the modes is much larger than their sum.

There is yet another consideration when evaluating the use-case of a dictionary learning algorithm, and that is whether it is "offline" vs "online." To first approximation, and speaking pragmatically, offline learning algorithms are those for which the entire dataset which they make use of fit in memory, while online algorithms are those for which the data is too large to fit in memory, whether that be by virtue of its high-dimensionality or cardinality, so that the full dataset can only be used by streaming in data. Online learning is thus a form of minibatch learning, however there is one key refinement: in addition to the coding and dictionary update components, there is an “aggregation" component which modifies the objective used to update the dictionary without having to store the incoming data.

To address some of the concerns raised above, in the seminal work [MBPS10], the authors introduce Online Nonnegative Matrix Factorization (ONMF) to address the quintessential problems associated with the “Big Data regime." By streaming data in batches, one produces a sequence of so-called “dictionary matrices" which are intended to form successively better models of the data. This, one can train the model on all of a data set which does not fit in memory, by streaming said data sets in minibatches which do fit in memory.

In the present work, we present our algorithm, Online CP-Dictionary learning (OCPDL). We are chiefly concerned with generalizing the original ONMF paper [MBPS10] in two ways. First, while the original paper only treated data under the i.i.d. assumption, our theoretical guarantees extend to the more general markovian setting. Second, our algorithm extends the matrix setting to the setting of tensors with an arbitrary number of modes. We test our algorithm on a variety of different data sets, and highlight the qualitative distinction between dictionary atoms (elements) obtained from OCPDL with those obtained by “flattening" tensor data, applying ONMF, and reshaping as well as the relative computational costs.

The outline of this thesis is as follows. In chapter 2, we apply ONMF to COVID-19 time series data and use the dictionary thus obtained to perform multivariate time-series regression. In chapter 3, we apply ONMF to meteorological time series, static images, as well as video data; the emphasis in this section is on the qualitative distinction between the online vs. offline learned dictionaries. Next in chapter 4, we come to the heart of the matter, namely the theory and applications of our extension of ONMF to our Online CP-Dictionary Learning Algorithm (OCPDL). Finally, in chapter 5 we present a hierarchical generalization of OCPDL, called Online Molecular Dictionary Learning (OMDL), as well as some applications to multivariate time series data.

CHAPTER 2

COVID-19 Time-Series Prediction by Joint Dictionary Learning and Online NMF

2.1 Attribution

This section is derived from the joint work [LSMN20]. Our application of ONMF to joint dictionary learning for time series forecasting grew out of another of our joint works, [LMNS20], discussed in the next chapter. I contributed the idea to learn a joint dictionary not only on cases corresponding to different cities, but also on the case outcomes (confirmed case, death, recovery), pre-processing the data, hyper-parameter tuning, and early implementations of the algorithm. Deanna Needell oversaw the project and discussions regarding the data representation used and interpretations of our predictions, in addition to contributing much of the introduction which discussed the relative advantages of our model compared to traditional epidemiological and deep learning models as well as possible extensions of our work. Hanbaek Lyu and Georg Menz contributed the application of ONMF to time series forecasting and recursive extrapolation technique. Hanbaek Lyu also created Python code for the final implementation of the algorithm.

2.2 Introduction

The rapid spread of coronavirus disease (COVID-19) has had devastating effects globally. The virus first started to grow significantly in China and then in South Korea around January of 2020, and then had a major outbreak in European countries within the next month, and as of April the US alone has over 400,000 cases with over 12,000 deaths. Predicting the rapid spread of COVID-19 is a challenge of utmost importance that the broader scientific community is currently facing.

A conventional approach to this problem is to use *compartmental models* (see, e.g. [KE05, Bra08]), which are mathematical models used to simulate the spread of infectious diseases governed by stochastic differential equations describing interactions between different compartments of the population (e.g. susceptible, infectious, and recovered). Namely, one may postulate a compartmental model tailored to COVID-19 and find optimal parameters for the model by fitting it them the available data. An alternative approach is to use data-driven machine learning techniques, especially deep learning algorithms [DLH+13, Ben13, Den14], which have had great success in various problems including image classification, computer vision, and voice recognition [KSH12, BPL10, HCC+14, AAB+16].

In this chapter, we propose an entirely different approach to predicting the spread of COVID-19 based on *dictionary learning* (or *topic modeling*), which is a machine learning technique that is typically applied to text or image data in order to extract important features of a complex dataset so that one can represent said dataset in terms of a reduced number of extracted features, or dictionary atoms [SG07, BCD10]. Although dictionary learning has seen a wide array of applications in data science, to our best knowledge this work is the first to apply such an approach to time series data and time series prediction.

Our proposed method has four components:

1. (*Minibatch learning*) Use online nonnegative matrix factorization (online NMF) to learn "elemental" dictionary atoms which may be combined to approximate short time evolution patterns of correlated time series data.

2. (*Online learning*) Further adapt the minibatch-learned dictionary atoms by traversing the time series data using online NMF.

3. (*Partial fitting and one-step prediction*) Progressively improve our learned dictionary atoms by online learning while concurrently making one-step predictions by partial fitting.

4. (*Recursive extrapolation*) By recursively using the one-step predictions above, extrapolate into the future to predict future values of the time series.

Our method enables us to learn dictionary atoms from a diverse collection of correlated time series data (e.g. new daily cases of COVID-19, number of fatal and recovered cases, and degree of observance of social distancing measures). The learned dictionary atoms describe “elemental" short-time evolution patterns from the correlated data which may be superimposed to recover and even predict the original time series data. *Online Nonnegative Matrix Factorization* is at the core of our learning algorithm, which continuously adapts and improves the learned dictionary atoms to newly arrived time series data sets.

There are a number of advantages of our proposed approach that may complement some of the shortcomings of the more traditional model-based approach or large-data-based machine learning approach. First, Our method is completely model-free and has the universality of data types, as the dictionary atoms directly learned from the data serve as the ‘model' for prediction. Hence a similar method could be applied to predict not only the spread of the virus but also other related parameters. These include the spread of COVID-19 media information, medical and food supply shortages and demands, patient subgroup infections, immunity and many more. Second, our method does not lose interpretability as some of the deep-learning-based approaches do, which is particularly important in making predictions for health-related areas. Third, our method is computationally efficient and can be executed on a standard personal computer in a short time. This enables our method to be applied in real-time in online-setting to generate continuously improving prediction. Lastly, our method has a strong theoretical foundation based on the recent work [LNB19].

In this chapter, we demonstrate our general online NMF-based time series prediction method on COVID-19 time series data by learning a small number of fundamental time evolution patterns in joint time series among the six countries in three different cases (confirmed/death/recovered) concurrently. Our analysis shows that we can indeed extract interpretable dictionary atoms for short-time evolution of such correlated time series and use them to get accurate short-time predictions with a small variation. This approach could further be extended by augmenting various other types of correlated time series data set that may contain nontrivial information on the spread of COVID-19 (e.g. time series quantifying commodity, movement, and media data).

This chapter is organized as follows. In Section 2.3, we give a brief overview of dictionary learning by nonnegative matrix factorization, and provide the full statement of our learning and prediction algorithms. In Subsection 2.4.1, we give a description of the time series data set of new COVID-19 cases and discuss a number of pre-processing methods for regularizing high fluctuations in the data set. Then we discuss our data analysis scheme and simulation setup in Subsection 2.4.2. In the following subsections 2.4.3-2.4.5, we present our main simulation results. Finally, we conclude and suggest further directions in Section 2.5.

2.3 Time Series Prediction by Online NMF

2.3.1 Dictionary learning by nonnegative matrix factorization

Matrix factorization provides a powerful mathematical setting for dictionary learning problems. We first organize n observations of d-dimensional samples a data matrix $X \in \mathbb{R}^{d\times n}$, and then seek a factorization of X into the product WH for some $W \in \mathbb{R}^{d\times r}$ and $H \in \mathbb{R}^{r\times n}$. This means that each column of the data matrix is approximated by the linear combination of the columns of the *dictionary matrix* W with coefficients given by the corresponding column of the *code matrix* H (see Figure 2.1). This problem has been extensively studied under many names, each with different constraints: dictionary learning, factor analysis, topic modeling, component analysis. It has also found applications in text analysis, image reconstruction, medical imaging, bioinformatics, and many other scientific fields [SGH02, BB05, BBL+07, CWS+11, TN12, BMB+15, RPZ+18].

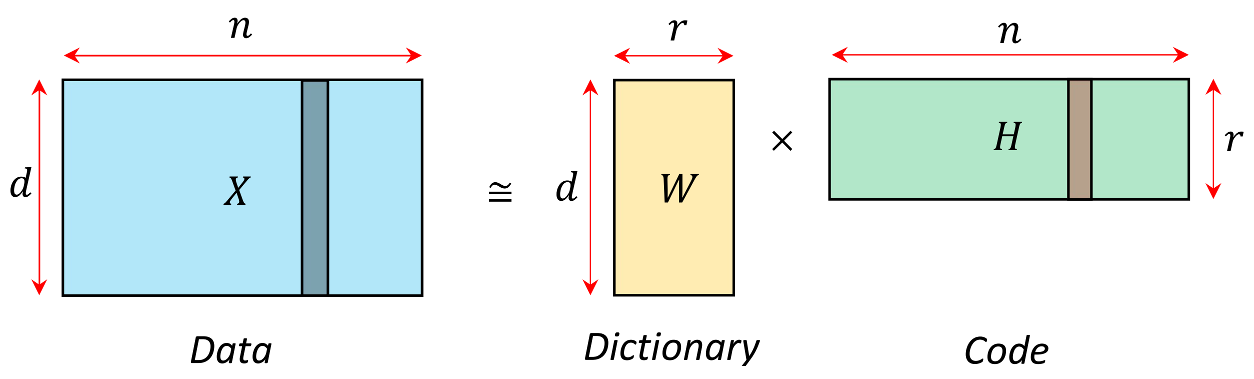

Figure 2.1: Illustration of matrix factorization. Each column of the data matrix is approximated by the linear combination of the columns of the dictionary matrix with coefficients given by the corresponding column of the code matrix.

Nonnegative matrix factorization (NMF) is an instance of matrix factorization where one seeks two nonnegative matrices whose product approximates a given nonnegative data matrix. Below we give an extension of NMF with an extra sparsity constraint on the code matrix which is particularly suited for dictionary learning problems [Hoy04]. Given a data matrix $X \in \mathbb{R}^{d\times n}_{\geq 0}$, the goal is to find a nonnegative dictionary $W \in \mathbb{R}^{d\times r}$ and nonnegative code matrix $H \in \mathbb{R}^{r\times n}$ by solving the following optimization problem:

$$\inf_{W\in\mathbb{R}^{d\times r}_{\geq 0},\, H\in\mathbb{R}^{r\times n}_{\geq 0}} \|X - WH\|_F^2 + \lambda\|H\|_1, \tag{2.1}$$

where $\|A\|_F^2 = \sum_{i,j} A_{ij}^2$ denotes the matrix Frobenius norm and $\lambda \geq 0$ is the L_1-regularization parameter for the code matrix H.

A consequence of the nonnegativity constraints is that one must represent the data using the dictionary W without exploiting cancellation. This is a critical mechanism that gives a parts-based representation of the data (see [LS99]). Many efficient iterative algorithms for NMF are based on block optimization schemes that have been proposed and studied following the introduction of the first, and most well-known, multiplicative update method by Lee and Seung [LS01] (see [Gil14] for a survey).

2.3.2 Our algorithms

In this section, we provide algorithms for online dictionary learning and prediction for ensembles of correlated time series. At the core of our online dictionary learning algorithm is the well-known online nonnegative matrix factorization (ONMF) algorithm [MBPS10, LNB19], which is an online extension of NMF that learns a sequence of dictionary matrices from a sequence of data matrices.

We first illustrate the key idea in the simplest setting of time series data for a single entity. Suppose we observe a single numerical value $x_s \in \mathbb{R}$ at each discrete time s. By adding a suitable constant to all observed values, we may assume without loss of generality that $x_s \geq 0$ for all $s \geq 0$. Fix integer parameters $k, N, r \geq 0$. Suppose we only store N past data points at any given time, due to memory constraints. So at time t, we hold the vector $\mathcal{D}_t = [x_{t-N+1}, x_{t-N+2}, \cdots, x_t]$ in our memory. The goal is to learn a dictionary of k-step evolution patterns from the observed history $(x_s)_{0\leq s\leq t}$ up to time t. A possible approach is to form a k by $N-k+1$ *Hankel matrix* X_t (see, e.g., [Ris73]), whose ith column consists of the k consecutive values of $\mathcal{D}_t$ starting from its ith coordinate. We can then factorize this into k by r dictionary matrix W and r by $t-k$ code matrices using an NMF algorithm:

$$X_t = \begin{bmatrix} x_{t-N+1} & x_{t-N+2} & \cdots & x_{t-k+1} \\ x_{t-N+2} & x_{t-N+3} & \cdots & x_{t-k+2} \\ \vdots & \vdots & \vdots & \vdots \\ x_{t-N+k} & x_{t-N+k+1} & \cdots & x_t \end{bmatrix} \approx WH. \tag{2.2}$$

This approximate factorization tells us that we can approximately represent any k-step evolution pattern from our past data $(x_s)_{t-N<s\leq t}$ by a nonnegative linear combination of the r columns of W. Hence the columns of W can be regarded as dictionary patterns for all k-step time evolution patterns in our current data set $\mathcal{D}_t$ for each time t.

Below we provide an online mini-batch implementation of the above sketch of dictionary learning for time series data in an online setting, as well as an online prediction algorithm.

=0

Algorithm 1 Online dictionary learning for time-series data

```
Input: Time series $(\mathbf{x}_t)_{1\le t\le T}$, $\mathbf{x}_t \in \mathbb{R}^{d\times 1}_{\ge 0}$
Variables: $N \ge k \in \mathbb{N}$, $\lambda > 0$, $\beta > 0$,
        $\mathbf{W}_0 \in \mathbb{R}^{d\times k\times r}_{\ge 0}$, $A_0 \in \mathbb{R}^{r\times r}_{\ge 0}$, $B_0 \in \mathbb{R}^{r\times dk}_{\ge 0}$
for $t = k, \cdots, T$ do
    Update sparse code:
        $H_t = \operatorname*{arg\,min}_{H\in\mathbb{R}^{r\times(N-k+1)}_{\ge 0}} ||\mathbf{X}^{(3)}_t - \mathbf{W}^{(3)}_{t-1}H||^2_F + \lambda||H||_1,$    (2.3)
    where $\mathbf{X}^{(3)}_t$ is the mode-3 unfolding of the $d \times k \times (N-k+1)$ tensor $\mathbf{X}_t$ (similarly for
    $\mathbf{W}^{(3)}_{t-1}$), whose mode-1 slices $\mathbf{X}_t(1), \cdots, \mathbf{X}_t(d)$ are given by the following Hankel matrix
        $\mathbf{X}_t(i) = \begin{bmatrix} \mathbf{x}_{t-N+1}(i) & \mathbf{x}_{t-N+2}(i) & \cdots & \mathbf{x}_{t-k+1}(i) \\ \mathbf{x}_{t-N+2}(i) & \mathbf{x}_{t-N+3}(i) & \cdots & \mathbf{x}_{t-k+2}(i) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{x}_{t-N+k}(i) & \mathbf{x}_{t-N+k+1}(i) & \cdots & \mathbf{x}_t(i) \end{bmatrix}.$    (2.4)
    Aggregate data:
        $A_t = (1 - t^{-\beta})A_{t-1} + t^{-\beta}H_tH_t^T$    (2.5)
        $B_t = (1 - t^{-\beta})B_{t-1} + t^{-\beta}H_tX_t^T$    (2.6)
    Update dictionary:
        $\mathbf{W}_t = \operatorname*{arg\,min}_{\mathbf{W}\in\mathbb{R}^{d\times k\times r}_{\ge 0}} \operatorname{tr}(\mathbf{W}^{(3)}A_t(\mathbf{W}^{(3)})^T) - 2\operatorname{tr}(B_t\mathbf{W}^{(3)})$    (2.7)
end for
```

Algorithm 2 Partial fitting and Prediction

Input: Time-series $(\mathbf{x}_t)_{t-k+2\le s\le t}$, $\mathbf{x}_t \in \mathbb{R}^{d\times 1}_{\ge 0}$

Dictionary tensor $\mathbf{W}_t \in \mathbb{R}^{d\times k\times r}$

Output: Prediction $\hat{\mathbf{x}}_{t+1}$ for $\mathbf{x}_{t+1}$

Variables: $\lambda' > 0$

Do:

Partial fitting:

$$H_* = \operatorname*{arg\,min}_{H\in\mathbb{R}^{r\times 1}_{\ge 0}} ||\widetilde{\mathbf{x}}_t - \widetilde{W}_t H||_F^2 + \lambda' ||H||_1, \tag{2.8}$$

where $\widetilde{\mathbf{x}}_t \in \mathbb{R}^{d(k-1)\times 1}_{\ge 0}$ is obtained by concatenating the columns of the following matrix

$$\begin{bmatrix} \mathbf{x}_{t-k+2}(1) & \cdots & \mathbf{x}_{t-k+2}(d) \\ \vdots & \ddots & \vdots \\ \mathbf{x}_{t-1}(1) & \cdots & \mathbf{x}_{t-1}(d) \\ \mathbf{x}_t(1) & \cdots & \mathbf{x}_t(d) \end{bmatrix}, \tag{2.9}$$

and $\widetilde{W}_t \in \mathbb{R}^{d(k-1)\times r}_{\ge 0}$ is the mode-3 unfolding of the $d\times(k-1)\times r$ tensor $\mathbf{W}_t[:, : (k-1), :]$ that consists of the first $(k-1)$ entries of $\mathbf{W}$ in mode 2.

Prediction:

$\hat{\mathbf{x}}_{t+1}$ = the last row of the $k\times d$ matrix obtained by reshaping $\mathbf{W}_t^{(3)} H^*$, where $\mathbf{W}_t^{(3)}$ is the mode-3 unfolding of $\mathbf{W}_t$

=0

=0

The dictionary update rule in both Algorithms 1 and 3 are based on the well-known online NMF algorithm in [MBPS10, LNB19]. Furthermore, in Algorithms 1 and 2, we also outline how to make predictions using the online-learned dictionary atoms via partial fitting (Algorithm 2) and extrapolation. Algorithm 3 is useful for initializing the dictionary tensor $\mathbf{W}_0$ in Algorithm 1, especially when the time-series data is limited (small T) so that one

Algorithm 3 Minibatch dictionary learning for time-series data

```
Input: Time-series $(\mathbf{x}_t)_{1\le t\le T}$, $\mathbf{x}_t \in \mathbb{R}^{d\times 1}_{\ge 0}$
Output: Dictionary tensor $\mathbf{W}_M \in \mathbb{R}^{d\times k\times r}_{\ge 0}$ and aggregate matrices $A_M \in \mathbb{R}^{r\times r}_{\ge 0}$, $B_M \in \mathbb{R}^{r\times dk}_{\ge 0}$

Variables: $N \ge k \in \mathbb{N}$, $\lambda > 0$, $\beta > 0$, $M \in \mathbb{N}$
Random initialization:
    Randomly initialize the tensors $\mathbf{W}_0 \in \mathbb{R}^{d\times k\times r}$, $A_0 \in \mathbb{R}^{r\times r}_{\ge 0}$, $B_0 \in \mathbb{R}^{r\times dk}_{\ge 0}$ so that the entries are i.i.d. Uniform([0, 1]) variables.
for $j = 1, 2, \cdots, M$ do
  Choose $t$ uniformly at random from $\{T-N+1, T-N+2, \cdots, T\}$
  Update $\mathbf{W}_j \leftarrow \mathbf{W}_{j-1}$ by using (2.3) - (2.7) in Algorithm 1.
end for
```

may not expect the online dictionary learning in Algorithm 1 in T steps. Also, we may use a different time series $(\mathbf{y}_t)_{1\le t\le T}$ to find a dictionary tensor $\mathbf{W}_M$ and use it as the initial dictionary for the given time series $(\mathbf{x}_t)_{1\le t\le T}$ in Algorithm 1. This "transfer learning" would be effective when the two time-series share a similar structure.

The sparse coding problems in (2.3) and (2.8) can be solved by LASSO, whereas the constrained quadratic problem (2.7) can be solved by projected gradient descent algorithms. See [LNB19] for more details and background. For practical use, we provide a python implementation of our algorithms in our GihHub repository (see the footnote of the front page).

We also remark that, using the recent contribution of Lyu, Needell, and Balzano [LNB19] on online matrix factorization algorithms on dependent data streams, we can give a theoretical guarantee of convergence of the sequence of learned dictionary atoms by Algorithm 1 under suitable assumptions. The essential requirement is that the time series data satisfies a weak "stochastic periodicity" condition. A complete statement of this result and proof will be provided in our follow-up paper.

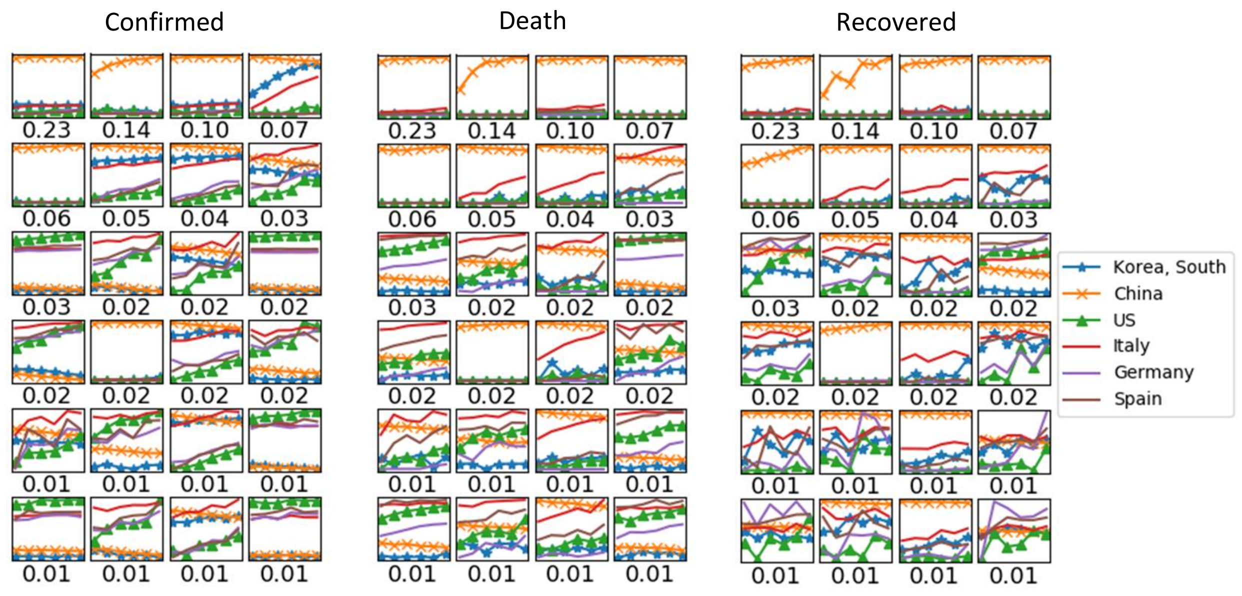

Figure 2.2: 24 Joint dictionary atoms of 6-day evolution patterns of new daily cases (confirmed/death/recovered) in six countries (S. Korea, China, US, Italy, Germany, and France). Each dictionary atom is a $6*6*3 = 108$ dimensional vector corresponding to `time` $*$ `country` $*$ `case type`. The corresponding importance metric is shown below each atom. 50 atoms are learned and the figure shows top 24 with the highest importance metric.

While this is a substantial extension of the usual independence assumption in the streaming data set, unfortunately, most COVID-19 time series do not verify any type of weak periodicity condition directly, which is one of the biggest challenges in predicting the spread of COVID-19. In the next section, we describe our experiment setting and how we may overcome this issue of “lack of periodicity" by combining the minibatch and online learning algorithms. It is important to note that minibatch learning algorithm (Algorithm 3) converges for fixed T as $M \to \infty$, as it is a version of online NMF on a i.i.d. sequence of data, which satisfies our stochastic periodicity condition.

2.4 Application to COVID-19 time-series data sets

2.4.1 Data set and pre-processing

To illustrate our dictionary learning and prediction algorithms for time series data, we analyze the historical daily reports of confirmed/death/recovered COVID-19 cases in six countries – South Korea, China, US, Italy, Spain, and Germany – from Jan 19, 2020 to Apr. 12, 2020. The input data can be represented as a tensor of shape $6 \times 80 \times 3$ corresponding to countries, days, and types of cases, respectively[1]. In order to apply our dictionary learning algorithms, we first unfold this data tensor into a matrix X of shape $(d \times T) = (6 \times 3) \times 80$, whose tth column, which we denote by $\mathbf{x}_t$, gives the full 18 dimensional observation in day $t, 0 \leq t \leq 80$. Also, we find that the fluctuation in the original data set is too large to yield stable representations, let alone predictions. In order to remedy this, we pre-process the time series data by taking a 5-day moving average and then entry-wise log-transform $x \mapsto \log(x+1)$. After applying dictionary learning and prediction algorithms, we take the inverse transform $x \mapsto \exp(x) - 1$ and plot the result.

2.4.2 Analysis scheme and experiment setup

As the COVID-19 time series data set we analyze here only consists of $T = 80$ highly non-repetitive observations, applying the online dictionary learning algorithm (Algorithm 1) with random initialization is not sufficient for proper learning and accurate prediction. We overcome this by using the minibatch learning algorithm (Algorithm 3) for the initialization for the online learning and prediction. Namely, we use the following scheme:

1. (*Minibatch learning*) Use minibatch Algorithm 3 for the time series $(\mathbf{x}_t)_{0 \leq t \leq T}$ to obtain dictionary tensor $\mathbf{W}_M \in \mathbb{R}_{\geq 0}^{d \times k \times r}$ and aggregate matrices $A_M \in \mathbb{R}_{\geq 0}^{r \times r}$, $B_M \in \mathbb{R}_{\geq 0}^{r \times dk}$.

2. (*Online learning and one-step prediction*) Use the output in step 1 as the initialization

[1]Raw data obtain from https://github.com/CSSEGISandData/COVID-19/tree/master/csse_covid_19_data

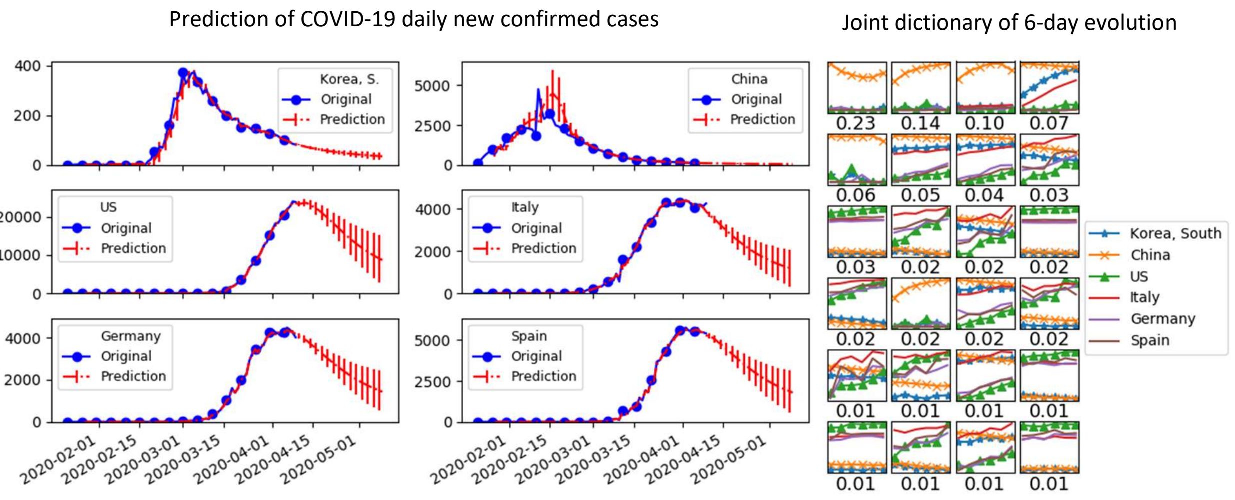

Figure 2.3: Joint dictionary learning and prediction for the time series of new daily cases (confirmed/death/recovered) in six countries (S. Korea, China, US, Italy, Germany, and France). After joint dictionary atoms are learned by minibatch learning, they are further adapted to the time series data by concurrent online learning and predictions. (Right) Joint dictionary atoms of 6-day evolution patterns of new confirmed cases. The corresponding importance metric is shown below each atom. (Left) Plot of the original and predicted daily new confirmed cases of the six countries. The errorbar in the red plot shows standard deviation of 1000 trials.

for Algorithm 1. For each $t = 1, \cdots, T$, iterates the steps in Algorithm 1 as well as Algorithm 2. This outputs a dictionary tensor $\mathbf{W}_T$ and a prediction $(\hat{x}_t)_{k \le t \le T+1}$.

3. (*Recursive extrapolation*) For $T < t \le T + L$, recursively use Algorithm 2

The hyperparameters we used in each steps above are given below:

1. (*Minibatch learning*)

$$N = 100 \qquad \text{(Memory size)}$$

$$M = 20 \qquad \text{(\# of minibatch iterations)}$$

$$k = 6 \qquad \text{(segment length)}$$

$$\lambda = 3 \qquad \text{(L_1-regularizer of sparse coding in (2.3))}$$

$$r = 50 \qquad \text{(Number of dictionary atoms)}$$

$$\beta = 1 \qquad \begin{pmatrix}\text{learning rate exponent for} \\ \text{minibatch learning}\end{pmatrix}$$

2. (*Online learning and one-step prediction*)

$$N = 100 \qquad \text{(Memory size)}$$

$$k = 6 \qquad \text{(segment length)}$$

$$\lambda' = 1 \qquad (L_1\text{-regularizer of sparse coding in (2.8))}$$

$$r = 50 \qquad \text{(Number of dictionary atoms)}$$

$$\beta = 4 \qquad \begin{pmatrix}\text{learning rate exponent for} \\ \text{online learning}\end{pmatrix}$$

$$\lambda' = 0 \qquad (L_1\text{-regularizer of sparse coding in (2.8))}$$

3. (*Recursive extrapolation*)

$$\lambda' = 0 \qquad (L_1\text{-regularizer of sparse coding in (2.8))}$$

$$L = 30 \qquad \text{(Future extrapolation length)}$$

The role of the parameter β becomes clear when examining the equations (2.5) and (2.6). It weights how much of the past data is used when updating the new dictionary matrix W_t. For example, in (2.7), in the extreme case $\beta = 0$ the present observation at time t is not used, wherease in the other extreme case $\beta = \infty$ only the present observation is used.

2.4.3 Simulation results - Minibatch learning

An example of dictionary atoms obtained from the minibatch learning algorithm (Algorithm 3) from the COVID-19 daily new cases time series data set is presented in Figure 2.2. We note that the time evolution structure in the data set is not used in this minibatch learning process, as slices of length k evolution are sampled independently and uniformly at random from the entire history. Each dictionary atom is a $6*6*3 = 108$ dimensional vector corresponding to `time` $*$ `country` $*$ `case type`.

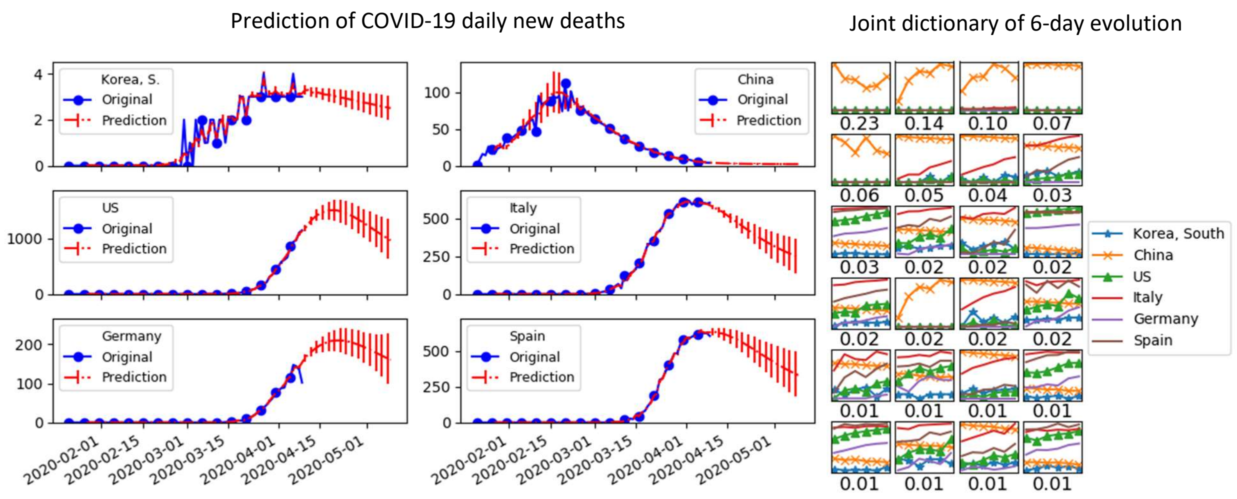

Figure 2.4: Joint dictionary learning and prediction for the time series of new daily cases (confirmed/death/recovered) in six countries (S. Korea, China, US, Italy, Germany, and France). After dictionary atoms representing fundamental joint time series patterns are obtained by minibatch learning, they are further adapted to the time series data by online learning while making predictions. (Right) Joint dictionary atoms of 6-day evolution patterns of new death cases. The corresponding importance metric is shown below each atom. (Left) The plot of the original and predicted daily new death cases of the six countries. The error bar in the red plot shows the standard deviation of 1000 trials.

To each atom, we associate an “importance metric" first introduced in [LNB19] as a measure of the total contribution of the atom in representing the original data set. Namely, the importance metric of each atom is the total sum of its linear coefficients in the sparse coding problem (2.3) during the entire learning process. This is computed as the row sums of the sum of all code matrices H_t in (2.3) obtained from the learning process.

For example, the $(1, 1)$ atom (in matrix coordinates) with importance 0.23 in Figure 2.2 indicates that the number of daily new confirmed cases in all six countries are almost constant and that China has significantly higher values than other countries. Also, the $(1, 4)$ atom (in matrix coordinates) with importance 0.07 in Figure 2.2 indicates that the number of daily new confirmed cases are growing rapidly in Korea and Italy, while for the other four countries the values are almost constant. It is important to note that these dictionary

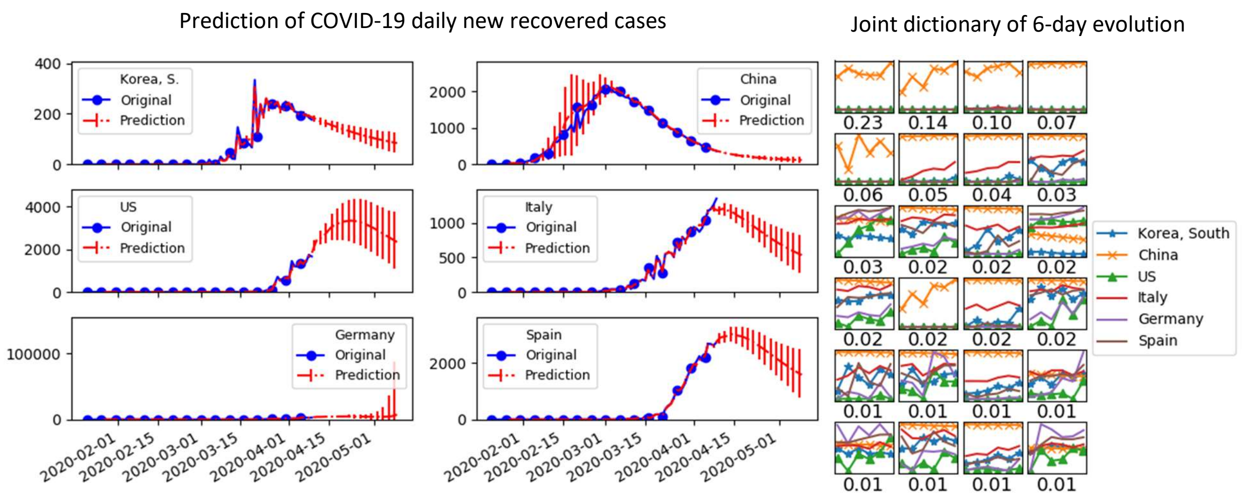

Figure 2.5: Joint dictionary learning and prediction for the time series of new daily cases (confirmed/death/recovered) in six countries (S. Korea, China, US, Italy, Germany, and France). After joint dictionary atoms are learned by minibatch learning, they are further adapted to the time series by online learning while making predictions. (Right) Joint dictionary atoms of 6-day evolution patterns of new recovered cases. The corresponding importance metric is shown below each atom. (Left) The plot of the original and predicted daily new recovered cases of the six countries. The errorbar in the red plot shows the standard deviation of 1000 trials.

atoms are learned by a nonnegative matrix factorization algorithm, so they maintain their individual interpretation we described before in representing the entire data set. Indeed, such patterns were dominant in the COVID-19 time-series data during the period of Jan. 21 - Mar. 1, 2020 (see e.g. the blue plot in Figure 2.3). Similarly, a direct interpretation of other dictionary atoms can be associated with features in the original data set.

2.4.4 Simulation results - Online learning and one-step prediction

The learned dictionary atoms in Figure 2.2 are not only directly interpretable but also provide a compressed representation of the original data set. Namely, we will be able to approximate any 6-day evolution pattern in the data set by only using the 24 atoms in Figure 2.2 with suitable nonnegative linear coefficients found from the sparse coding problem (2.3). Obtaining a global approximation of the entire data set based on such local approximations by dictionary atoms is called *reconstruction* (see for instance the image reconstruction example in [LNB19]). Our partial fitting and prediction algorithm (Algorithm 2) extends this reconstruction procedure by using the learned patterns to make one step predictions adapted to the time series data.

The main point of our application is to illustrate that we may obtain reasonable predictions on our very limited $T = 80$ COVID-19 time series data set by learning a small number of fundamental patterns in joint time evolution among the six countries in three different cases (confirmed/death/recovered) concurrently. This approach could further be extended by augmenting various other types of correlated time series which contain nontrivial information on the spread of COVID-19 (e.g. time series quantifying commodity, movement, and media data trends). One can think of learning the highly correlated temporal evolution patterns across different countries and cases as a model construction process for the joint time evolution of the data set. There are relatively few hyper-parameters to train in our algorithm compared to deep neural network-based models, and parameters in our method are in some sense built-in to the temporal dictionary atoms so that one does not need to begin by choosing the model to use.

However, recall that the dictionary atoms learned by the minibatch algorithm are not adapted to the temporal structure of the data set. We find that further adapting them in the direction of time evolution by using our online learning algorithm (Algorithm 1) according to the scheme in subsection 2.4.2 significantly improves the prediction accuracy by reducing the standard deviation of the prediction curves over a number of trials. An example of the result of this further adaptation of the minibatch-learned dictionary atoms is shown in Figures 2.3, 2.4, and 2.5. In each figure, the online-improved dictionary atoms are shown in the right, and the original time series data and its prediction computed by Algorithms 1-2 are shown in blue and red, respectively. Prediction curves also show error bars of one standard deviation from 1000 trials of the entire scheme (minibatch + online + extrapolation) under the same hyperparameters in Subsection 2.4.2.

By comparing the corresponding dictionary atoms in Figures 2.2 and 2.3 for example, we find that the online learning process does not change the importance metric on the top 24 atoms, and only a few atoms change in their shape significantly (especially the curves for China in atoms at $(1,1), (1,2), (1,3), (2,1)$, and $(4,2)$). Such new patterns were not able to be learned by the minibatch learning, but they were picked up by traversing the time series data from the past to the present with our online learning algorithm. The "correctness" of the learned dictionary atoms can be verified by the accuracy of the 1-step prediction up to time $T = 80$ (the end of blue curve in Figure 2.3) in Figures 2.3, 2.4, and 2.5.

We remark that our one-step predictions up to time $T = 80$ are not exact predictions, as our initial dictionary tensor for this step, learned by the minibatch algorithm, uses all information in the entire time-series data. More precisely, one can think of this as a reconstruction procedure without seeing the last coordinate. This would have been a proper prediction if our initialization were independent of the data, but we find that online learning alone without the minibatch initialization gives inferior reconstruction results. We believe this is due to the fact that the COVID-19 time series data set is too short (and far from periodic) for the online dictionary learning algorithm to converge in a single run. Nevertheless, in some sense the mini-batch method is implemented to compensate for the lack of data; given a sufficient amount of data, we could omit this step.

2.4.5 Simulation results - Recursive extrapolation

Next, we discuss the recursive extrapolation step, which gives the 30-day prediction of the new daily cases of all three types and all six countries simultaneously, shown as in Figures 2.3, 2.4, and 2.5. For instance, by partially fitting the first five coordinates of the length-6 dictionary atoms to the last five days of the data, we can use Algorithm 2 to obtain a prediction of the future values at time $T+1$. We can then recursively continue this extrapolation step using the predicted data into the future ad infinitum. Our 30-day prediction results show reasonable variation among trials. However, the variation in prediction grows in the prediction length, so only a moderate range of predictions would have meaningful implications.

We remark that we did not enforce any additional assumption on the prediction curves (e.g. finite carrying capacity, logistic-like growth for the total, SIR-type structure) that are standard in many epidemiological models. Instead, we chose our hyperparameters in subsection 2.4.2 so that the future prediction curves approximately satisfy such assumptions, highlighting the model-free nature of our approach. This high-level fitting is not without compromise in the uncertainty of the prediction. For example, choosing large values of the L_1-regularization parameter λ' used in the recursive extrapolation step reduces the variability of prediction significantly, but the prediction curve drops to zero very rapidly right after the end of the current data, which is absurd.

Lastly, we also mention that our recursive extrapolation defines a deterministic dynamical system in multidimensional space (dimension 18 in this case). The evolution is determined by the hyperparameters and the set of dictionary atoms at the end of the given time series data ($T = 80$ in this case). Even though our dictionary learning algorithms are randomzied, we find our 30-day predictions over many trials are within a modest standard error. However, we find that the mean trajectory of the prediction could vary significantly with respect to changes in the hyperparameters. Developing a more systematic method of choosing these hyperparameters is a future direction of inquiry.

2.5 Conclusion

With the rapidly changing situation involving COVID-19, it is critical to have accurate and effective methods for predicting short-term and long-term behavior of many parameters relating to the virus. In this paper, we proposed a novel approach that uses dictionary learning to predict time-series data. We then applied this approach to analyze and predict the new daily reports of COVID-19 cases in multiple countries. Usually, dictionary learning is used for text and image data; often with impressive results. To our best knowledge, our work is the first to implement dictionary learning to time-series data.

There are a number of advantages of our approach that may complement some of the shortcomings of the more traditional model-based approach or the large-data-based machine learning approach. First, our method is completely model-free as the dictionary atoms directly learned from the data serve as the 'model' for prediction. Our approach also works with universal data types. For example, it could be applied to predict not only the spread of the virus but also other related parameters. These include the spread of COVID-19 media information, medical and food supply shortages and demands, subgroup infections, immunity and many more. Second, the method works with small data sets. Most machine-learning methods need either model-specific input or large data sets. Because COVID19 only appeared recently, there are no large data sets yet available. Third, the method does not lose interpretability as some of the deep-learning-based approaches do. This is particularly important when making predictions for health-related areas. The learned dictionary atoms are not only interpretable but also identify hidden correlations between multiple entities. Fourth, our approach uses only a few hyperparameters that are model-free and independent of the data set. Therefore our approach avoids the issue of over-fitting. Furthermore, the method is computationally efficient and can be executed on a standard personal computer in a short time. Hence, it could be applied in real-time or online-settings to generate continuously improving predictions. Lastly, the method has a strong theoretical foundation: Convergence of the minibatch algorithm is always guaranteed and the convergence of the online learning algorithm is guaranteed under the assumption of quasi-periodicity. Therefore we expect our method to be robust i.e. small changes to the data-set or to the parameters should not destroy the learning outcome.

There are a number of future directions that we are envisioning. First, one could apply our method to county-level time-series data. One would obtain dictionary atoms describing county-wise correlation and prediction. This analysis could be critically used in distributing medical supplies and also in measuring the effect of re-opening the economy successively by a few counties at a time. Second, as not every individual can be tested, it is valuable to be able to transfer the knowledge on tested subjects to the ones yet to be tested. This 'transfer learning' could naturally be done with our method, by learning a dictionary from one subject group and apply that to make predictions for the unknown group. Lastly, one may extend the method to a fully tensor-based setting, where a large number of related variables of different types could be encoded in a single tensor. Then one could use various direct tensor-factorization methods to learn higher-order dictionary atoms. For example, this might be useful in identifying a critical subgroup of variables for clinical data and therapeutics.

CHAPTER 3

Applications of Online Nonnegative Matrix Factorization to Image and Time-Series Data

3.1 Attribution

This section is based on the joint work [LMNS20]. My primary contributions were to the image and video applications, in particular the hybrid deep learning-dictionary learning approach to color restoration, and to the experiment which illustrates ONMF's capacity to learn qualitatively distinct dictionary atoms for video data from those learned from offline NMF. Hanbaek Lyu and Georg Menz contributed the time series forecasting application, and in particular the "restrict and predict" method below. Deanna Needell suggested the use of the meterological and video data, in addition to clarifying the results in the paper.

3.2 Introduction

In the last few decades, the quantity of data available and the need to effectively exploit this data have grown exponentially. At the same time, modern data also presents new challenges in its analysis for which new techniques and ideas have become necessary. Many of these techniques may be classified as *topic modeling* (or *dictionary learning*), which aim to extract important features of a complex dataset so that one can represent the dataset in terms of a reduced number of extracted features, or topics. One of the advantages of topic modeling-based approaches is that the extracted topics are often directly interpretable, as opposed to the arbitrary abstractions of deep neural networks.

Matrix factorization provides a powerful setting for dimensionality reduction and dictionary learning problems. In this setting, we have a data matrix $X \in \mathbb{R}^{d \times n}$, and we seek a factorization of X into the product WH for $W \in \mathbb{R}^{d \times r}$ and $H \in \mathbb{R}^{r \times n}$ (see Figure 2.1). Hence each column of the data matrix is approximated by the linear combination of the columns of the *dictionary matrix* W with coefficients given by the corresponding column of the *code matrix* H. This problem has been extensively studied under many names each with different constraints: dictionary learning, factor analysis, topic modeling, component analysis. It has also found applications in text analysis, image reconstruction, medical imaging, bioinformatics, and many other scientific fields more generally [SGH02, BB05, BBL+07, CWS+11, TN12, BMB+15, RPZ+18].

In today's data world, large companies, scientific instruments, and healthcare systems are collecting massive amounts of data every day so that an entire data matrix is hardly available at any single time. *Online matrix factorization* is a matrix factorization problem in the online setting where data are accessed in a streaming fashion and the matrix factors are updated each time. This enables factor analysis to be performed concurrently with the arrival of new data samples.

In this article, we demonstrate three application contexts of matrix factorization algorithms, namely, historical temperature data, color images, and video frames. In doing so, we demonstrate how one can use matrix factorization techniques to learn joint dictionary atoms from an ensemble of correlated data sets. In the case of historic temperature data, we show that the online-learned joint dictionaries reveal key features of dependence between temperatures of four cities (LA, SD, SF, and NYC), which we use to "in-paint" missing data in one city by inferring from observed data in other cities. By a suitable matricization, we learn a small number of dictionary patches from a color image and reconstruct the original image from this low-rank image basis. The same procedure, used in conjunction with a convolutional neural network for classifying image patches, enables us to restore color to a grayscale image. Lastly, for video frames, we compare dictionary frames learned from "offline" and online matrix factorization algorithms and show that they capture different features of the video frames. By factorizing along the time dimension, we also demonstrate that one can detect a significant temporal change in the dataset using matrix factorization algorithms.

3.3 Dictionary learning by online Nonnegative Matrix Factorization

In this section, we describe a dictionary learning algorithm based on online nonnegative matrix factorization.

3.3.1 Nonnegative Matrix Factorization

Nonnegative matrix factorization (NMF) is a variant of matrix factorization where one seeks find two smaller matrices whose product approximates a given nonnegative data matrix. Below we give an extension of NMF with extra sparsity constraint on the code matrix, which is often used for dictionary learning problems. Given a data matrix $X \in \mathbb{R}^{d\times n}_{\geq 0}$, the goal is to find nonnegative dictionary $W \in \mathbb{R}^{d\times r}$ and code matrices $H \in \mathbb{R}^{r\times n}$ by solving the following optimization problem:

$$\inf_{W\in\mathbb{R}^{d\times r}_{\geq 0},\, H\in\mathbb{R}^{r\times n}_{\geq 0}} \|X - WH\|_F^2 + \lambda \|H\|_1, \tag{3.1}$$

where $\|A\|_F^2 = \sum_{i,j} A_{ij}^2$ denotes the matrix Frobenius norm and $\lambda \geq 0$ is the L_1-regularization parameter for the code matrix H.

A consequence of the nonnegativity constraints is that one must represent the data using the dictionary W without exploiting cancellation. In practice, this forces the learned atoms to be “atomic," or localized. A famous comparison was presented in [LS99]. There, the authors apply PCA and NMF to (vectorized) images of human faces. The (matricized) learned dictionary atoms from PCA resembled entire faces, while the matricized dictionary atoms from NMF resembled parts of faces, edges, etc. Furthermore, by enforcing a sparsity constraint on the code matrix, the learned dictionary elements have to appear sparsely in approximating each column of the data matrix. To make the dictionary components more localized and reduce the overlaps between them, one can also enforce sparseness on the dictionary matrix [Hoy04] in the optimization problem for NMF as in (3.1).

Many efficient iterative algorithms for NMF are based on block optimization schemes that have been proposed and studied, including the well-known multiplicative update method by Lee and Seung [LS01] (see [Gil14] for a survey).

Algorithm 4 NMF by Multiplicative Updates

```
Input: $W_0, H_0, K$
for $k = 1, ..., K$ do
    Update code:
        $H_{ij}^{k+1} \leftarrow H_{ij}^{k} \frac{((W^k)^T X)_{ij}}{((W^k)^T W^k H^k)_{ij}}$
    Update dictionary:
        $W_{ij}^{k+1} \leftarrow W_{ij}^{k} \frac{(X(H^{k+1})^T)_{ij}}{(W^k H^{k+1}((H^{k+1})^T))_{ij}}$
end for
```

=0

3.3.2 Online Nonnegative Matrix Factorization

Online Matrix Factorization (OMF) is an optimization problem where given a sequence $(X_t)_{t\geq 0}$ of data matrices, one seeks to find sequence of dictionary and code matrices $(W_t, H_t)_{t\geq 0}$ which asymptotically minimizes a loss function of choice. If we have the additional nonnegativity constraint on the factors, the resulting problem is called the *Online Nonnegative Matrix Factorization* (ONMF).

One of the most fundamental ideas in many algorithms for OMF is empirical loss minimization. Roughly speaking, the idea is to choose W_{t-1} be to the minimizer of an empirical loss function up to time $t-1$, and when the new data matrix X_t arrives, we find an improved dictionary W_t in response to the empirical loss up to time t. However, as the empirical loss function is usually nonconvex and difficult to optimize, many successful algorithms for OMF exploit techniques such as block optimization, convex relaxation, and majorization-minimization. Below we present one of the most well-known OMF algorithms proposed in [MBPS10]. The superscript of matrices in the algorithm below denotes iterates.

Algorithm 5 Online Nonnegative Matrix Factorization

```
Input: $W_0, \lambda$
for $t = 1, ..., T$ do
    Update sparse code:
        $H_t = \arg\min_{H \geq 0} ||X_t - W_{t-1}H||_F^2 + \lambda||H||_1$
    Aggregate data:
        $A_t = \frac{1}{t}((t-1)A_{t-1} + H_t H_t^T)$
        $B_t = \frac{1}{t}((t-1)B_{t-1} + H_t X_t^T)$
    Update dictionary:
        $W_t = \arg\min_{W \geq 0} \frac{1}{2}tr(W A_t W^T) - tr(B_t W)$
end for
```

=0

Rigorous convergence guarantees for online NMF algorithms have been obtained in [MBPS10] for independent and identically distributed input data. Recently, convergence guarantees of online NMF algorithms have been established when the data matrices have hidden Markov dependence [LNB19], ensuring further versatility of NMF based topic modeling from input sequences generated by Markov Chain Monte Carlo algorithms.

A simple computation reveals that the minimization problem in the update of the dictionary W equivalant to the more intuitive problem,

$$W_t = \arg\min_{W \geq 0} \frac{1}{t} \sum_{s=1}^{t} ||X_s - WH_s||_F^2 + \lambda||H_s||_1$$

where the objective function is a convex upper bounding surrogate of the corresponding empirical loss function. In practice, there is critical difference: the latter optimization problem requires one to have access to all of the data up to time t in order to obtain W_t, while the former only needs the "aggregation matrix" of the data and the data sample X_t. This property of the formulation of the algorithm is what enables ONTF to deal with the data size, acquisition, and evolution problems mentioned in the introduction.

We comment on a related important contrast between NMF and ONMF. While the NMF algorithm in the last subsection was essentially symmetric in the update of the dictionary and code for $\lambda = 0$, this is far from true in the ONMF algorithm. This is intuitive: in ONMF, one attempts to learn a fixed dictionary that can well-model all data in expectation. However one particular data sample generally has little to do with another, and so there is no reason for their respective codes to be similar.

3.4 Time-series application

In this section, we apply ONMF to time-series data for online dictionary learning and online reconstruction. We will be using the online nature of the dictionary learning algorithm there.

Suppose we observe a single numerical value $x_s \in \mathbb{R}$ at each discrete time s. By adding a suitable constant to all observed values, we may assume that $x_s \geq 0$ for all $s \geq 0$. Fix integer parameters $k, N, r \geq 0$. Suppose we only store N past data at any given time, due to a memory constraint. So at time t, we hold the vector $\mathcal{D}_t = [x_{t-N+1}, x_{t-N+2}, \cdots, x_t]$ in our memory. The goal is to learn dictionary patterns of k-step evolution from the observed history $(x_s)_{0\leq s\leq t}$ up to time t. A possible approach is to form a k by $N-k+1$ data matrix X_t, whose ith column consists of the k consecutive values of $\mathcal{D}_t$ starting from its ith coordinate. We can then factorize this into k by r dictionary matrix W and r by $t-k$ code matrices using an NMF algorithm:

$$X_t = \begin{bmatrix} x_{t-N+1} & x_{t-N+2} & \cdots & x_{t-k+1} \\ x_{t-N+2} & x_{t-N+3} & \cdots & x_{t-k+2} \\ \vdots & \vdots & \vdots & \vdots \\ x_{t-N+k} & x_{t-N+k+1} & \cdots & x_t \end{bmatrix} \approx WH. \tag{3.2}$$

This approximate factorization tells us that we can represent any k-step evolution from our past data $(x_s)_{t-N<s\leq t}$ approximately by a nonnegative linear combination of the r columns

of W. Hence the columns of W can be regarded as dictionary patterns for all k-step time evolution patterns in our current dataset $\mathcal{D}_t$ at each time t.

In order to extend the above 'temporal dictionary learning' scheme into an online setting, we apply the ONMF algorithm given in Subsection 3.3.2. This only requires us to store additional aggregate matrices $A_t \in \mathbb{R}^{r\times r}$ and $B_t \in \mathbb{R}^{r\times d}$. We denote the resulting dictionary matrix at time t as W_t. Then the r columns of W_t are temporal dictionary for all k-step evolution in the time series $(x_s)_{0\le s\le t}$ up to time t.

Lastly, we can further extend the above online temporal dictionary learning scheme for a collection of time-series data. We explain this in a concrete context. We analyze the historical monthly temperature (measured in Fahrenheit) data of four cities – Los Angeles (LA), San Diego (SD), San Francisco (SF), and New York City (NYC), which is published as part of the *NOAA online weather data (NOWData)* [Sta06]. The time period of our data set spans the years 1944-2020, a total of 869 months. At each time step $0 \le t \le 868$, we obtain four numerical values for the average monthly temperatures from the four cities, which we denote as a four-dimensional column vector $[LA(t), SD(t), SF(t), NYC(t)]^T$. Instead of applying our online temporal dictionary learning scheme separately for each city, we stack the resulting data matrices from the time series from each city vertically and form a joint data matrix at each time t. We use the parameters $k = 6$, $N = 50$, and $r = 16$. Then we can learn a sequence $(W_t)_{t\ge 0}$ of $4k$ by r dictionary matrices using the ONMF algorithm. We plot the last dictionary W_{868} after a suitable reshaping, which gives us 16 joint temporal dictionary elements as shown in Figure 3.1.

The online-learned joint temporal dictionary elements shown in Figure 3.1 reveal an interesting dependency structure between the temperature evolution of the four cities. For instance, LA and SD (blue and red) evolve almost in synchrony with a small amplitude. On the other hand, NYC (black) has a larger amplitude and has a mostly positive correlation with LA and SD. Lastly, SF (yellow) has an intermediate amplitude with a negative correlation between LA and SD.

One way to evaluate the accuracy of learned dictionary patterns is to use them to recon-

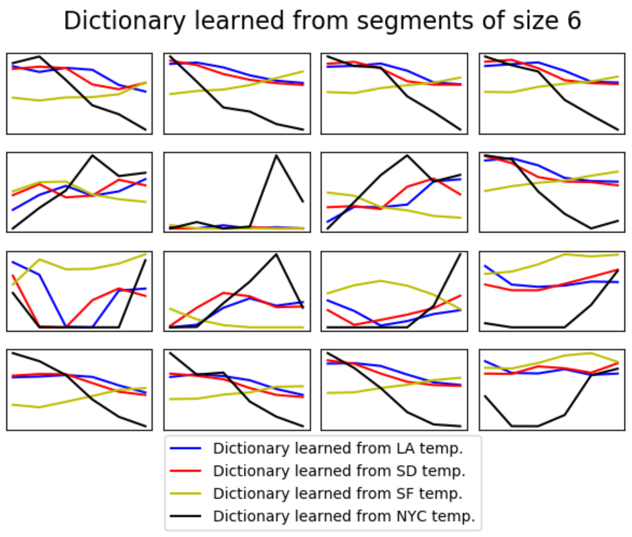

Figure 3.1: 16 joint temporal dictionary elements learned from the average monthly temperature of LA, SD, SF, and NYC during years 1944-2020. Each element indicates a fundamental pattern in the 6-month joint evolution of the temperatures in the four cities.

struct the original dataset, as shown in Figure 3.2. Here for each time t, we reconstruct the 6-step (6-month) joint time evolution vector $\mathbf{v}_t$ during the period $[t-5, t]$ using the online-learned joint dictionary W_t from the observe data up to time t. This can be easily done by obtaining the best coefficient matrix H such that $\|\mathbf{v}_t - W_t H\| + \lambda \|H\|_1$ is minimized, which is a convex quadratic problem. Then the product $W_t H$ gives the reconstruction for $\mathbf{v}_t$. We take the time-t-coordinates from this reconstruction to get the four red curves in Figure 3.2, plotted on top of the original temperature data in blue.

We emphasize that there is no a priori training. Initial dictionary matrix W_0 is completely random, but we progressively learn better dictionary matrix W_t over time. Also, we remark that for the three cities SD, SF, and NYC, there are several missing entries, which were coded as -100 °F. For a plotting purpose, we have modified these missing entries to 0 °F, which are shown as horizontal bars in the blue curve at height 0 in Figure 3.2. Our reconstruction (red) matches the original data (blue) accurately for the observed entires. Furthermore, we used a modified reconstruction algorithm so that we only approximate observed entries, and use the resulting linear coefficients to “fill-in" the missing entries using our joint dictionary.

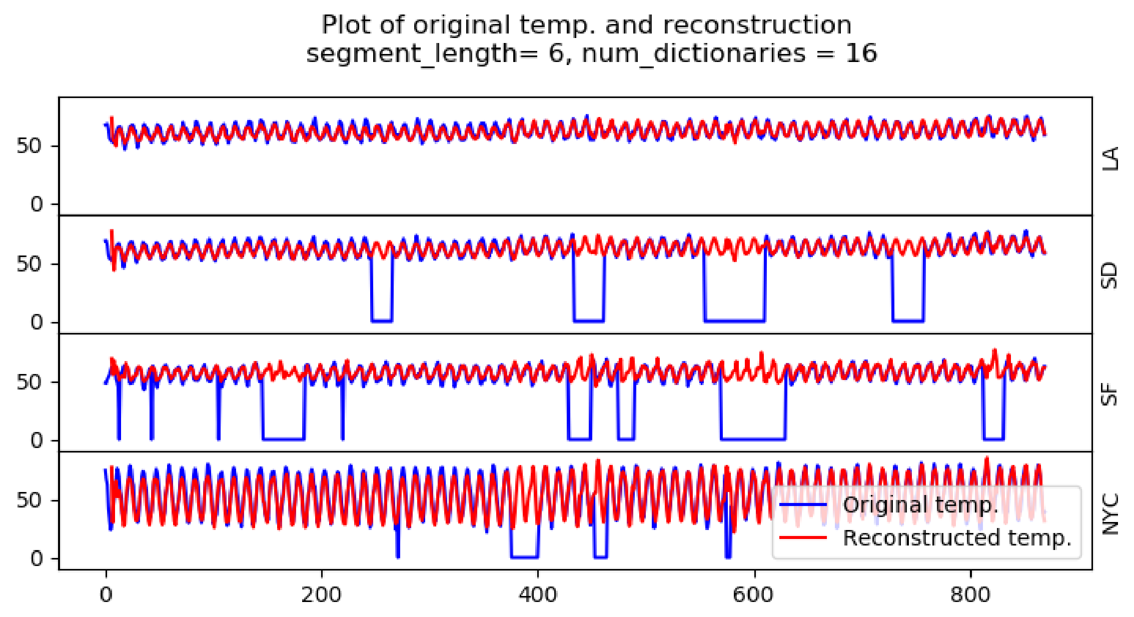

Figure 3.2: (Blue) Plot of historical average monthly temperature (measured in Fahrenheit) data of four cities Los Angeles (LA), San Diego (SD), San Francisco (SF), and New York City (NYC). Missing entries are coded as 0 °F. (Red) Online-reconstructed data using online-learned joint temporal dictionary.

The result indicates that we can use our online joint dictionary learning to inferring missing entries in our dataset, purely based on learning the dependency structure.

3.5 Video Application

In the previous section, we applied ONMF to two important problems in image processing which did not use any particular advantage of the algorithm, which serves to demonstrate its utility. In this section, we turn to applications of ONMF to video processing. It is here that the sequential nature of the ONMF algorithm enforces a qualitatively different structure on the learned dictionary atoms when compared to standard offline NMF.

We now describe our experiment. Our data consisted of a grayscale video of a candle, which consisted of 75 time frames. Each time frame was represented by an 80 x 30 matrix of pixel intensities.

We reshaped our data by vectorizing each time frame and then concatenating these

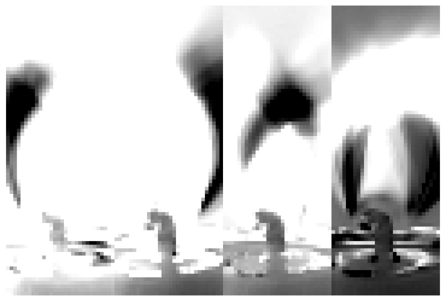
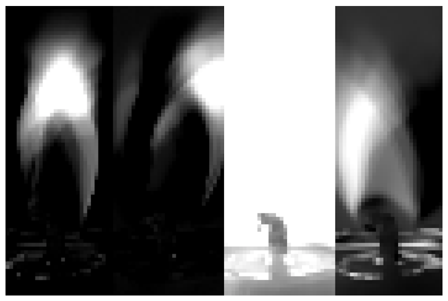
Figure 3.3: Candle Video Dictionaries. The first dictionary consists of four elements and was trained by an alternating least squares-based, offline NMF, the second dictionary below was trained using ONMF, where each time frame of the video represented a new data point.

vectors to form a single 2400 x 75 data matrix of pixel intensities. We applied an alternating least squares-based offline NMF to learn four dictionary elements. Next, we applied ONMF in natural way: each vectorized time frame corresponded to a new data sample. Again we learned four dictionary elements, which were then matricized and displayed in 3.3.

Additionally, we plotted the dictionaries learned by ONMF up to various time frames and displayed the results. This reveals an interesting, and intuitive, illustration of how ONMF uncovers temporal structure in the video. In particular, like the candle in the video moves, initially trivial atoms one by one become nontrivial, and adopt the shape of a particular configuration of the candle, e.g. left, center, right. When one plots the full evolution of the learned dictionaries, once the basic patterns have been established we empirically observe that when the candle is in a certain configuration at a particular time frame, only the atom with the corresponding configuration exhibits any significant update. By the end of the learning process, the final atoms look like superpositions of small perturbations of a single configuration (e.g. the second atom resembles a superposition of right configurations, the fourth atom resembles a superposition of left configurations). The results are displayed in 3.4.

Lastly, we apply NMF to the video frame along the temporal dimension to learn the dictionary for time evolution. This will enable us to detect a temporally significant change

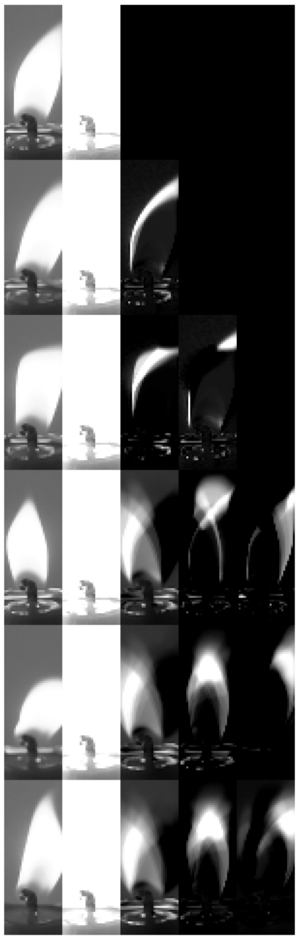
Figure 3.4: Candle video and learned dictionary at various time frames (time goes from top to bottom). The left column corresponds to the actual video frame. The remaining four columns each correspond to a particular dictionary element. The six rows correspond to different time frames, 1, 5, 7, 15, 35, and 75

in the video frame through the learned dictionary elements. To explain our simulation setup, first recall that the candle video frame is stored as an 80 x 30 x 75 tensor, where the last dimension corresponds to time. By flattening the first two spatial dimensions, this video frame can be represented as a 2400 x 75 matrix. In the previous experiments, we have factorized this data matrix using NMF and ONMF algorithms. Here, we take the transpose of this data matrix so that the dimension of 75 x 2400 correspond to [time x space]. We generate another video frame with the same size and length using white noise and concatenate

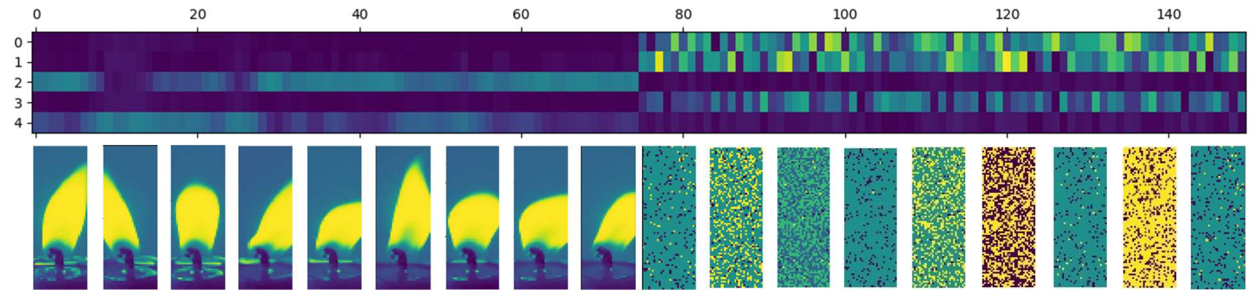

Figure 3.5: Learning time evolution dictionary from a video frame using NMF. The first and last 75 frames of the video are from a candle video and white noise, respectively, as shown below. By an approximate factorization of shape [time x space] = [time x 5] [5 x space], we learn 150 x 5 dictionary matrix, whose columns give an approximate basis for the time evolution of each pixel of the video frame. The learned time evolution dictionaries detect the planted 'phase transition' between frames 75 and 76.

with the candle video. Altogether, we form a data matrix X of shape 150 x 2400, where the first 75 rows are coming from the candle video, and the last 75 are from white noise.

Next, we approximately factorize the data matrix X into WH using an NMF algorithm, where W is of shape 150 x 5. This factorization tells us that each column of X, which corresponds to the time evolution of a single pixel for 150 frames, can be approximated by a nonnegative linear combination of the 5 columns of W. Hence the columns of W give a dictionary for the time evolution of each of the 2400 pixels in the video frame. Since we have planted a significant change between times 75 and 76, these five time-evolution dictionary elements should be able to detect this "phase transition" in our video. Indeed, as in Figure 3.5, all five columns of W (shown as rows) exhibit significant change between frames 75 and 76, as expected. Furthermore, the first halves of the third and last dictionary elements in Figure 3.5 contains information on the time evolution for the candle video, which disappears after the phase transition at frame 76. This is also expected since a basis for time evolution in the candle video should not contain meaningful information for the frames generated by white noise.

3.6 ONMF for Color Image Processing

In this section, we describe how ONMF can be applied to the two important image processing problems.

First, a quick introduction to patch-based image processing. A typical e.g. jpeg grayscale image may be represented by a matrix of unsigned integers with values between 0 and 255, representing the intensity. Lower values correspond to black, higher values to white. Color images may be stored similarly: to each of the three color channels red, blue and green, there corresponds an analogous matrix of color intensities.

In general, images may contain thousands of pixels, so working with the full image directly can be computationally infeasible, even when using online algorithms. Thankfully, one can achieve success on various image processing applications, such as compression, denoising, and impainting, by taking a "patch-based" approach. The idea is to extract (typically, overlapping) small patches, usually of size in the range 8×8 to 30×30 pixels. One may apply some procedures to each patch and recover the full image by patch averaging.

3.6.1 Color Image Compression

In this subsection, we describe how one can utilize ONMF to implement a patch-based approach to image compression. These applications do not leverage any particular advantage of ONMF compared to other nonnegative matrix factorization algorithms, but serve to illustrate its versatility. However, in the video processing application we will have seen that ONMF can produce qualitatively distinct dictionaries compared to those obtained from standard offline nonnegative matrix factorization algorithms.

First, we consider the patch-based compression of images. The motivation for the approach presented here is as follows. A typical image, in its uncompressed form, is very high dimensional. Indeed, images of moderate size may contain thousands to millions of pixels. Thankfully, most real-world images are sufficiently regular to enable a substantial reduction in the effective dimension. For instance, while it is true that an image of a clear sky may contain a large number of pixels, there is little variation in this image, and so intuitively one may expect a representation by an object of dimension far lower than the number of pixels to be possible (indeed, a single blue image patch ought to suffice).

One of the most popular ways of exploiting the regularity of natural images is through the use of small image patches. The general scheme is to first learn a collection of relatively few "atomic image patches" through some dictionary learning algorithms such as KSVD or NMF. One then may then implement one's application of choice, e.g. compression, denoising, inpainting, to some collection of overlapping small image patches that cover the full image. Finally, one can recover a full-size image through patch averaging.

We now describe in more detail how we used ONMF to compress color images. ONMF sequentially receives data and updates the learned dictionary to effectively model new data samples. In our implementation, a single data sample will consist of a matrix whose columns consist of vectorized color image patches. For instance, we may select 1000 20 x 20 color image patches so that our data sample is a 1200 x 1000 matrix. The first 400 entries of the jth column are obtained by extracting the red channel of the jth image patch and vectorizing along the first axis. The next 400 entries are analogously obtained from the blue channel and the final 400 from the green channel. We may then apply ONMF as described in section II with some number of data samples. Figure 3.6 displays the results of ONMF applied to a famous painting.

We comment that for quality compression, it is often necessary to use the significant overlap of patches. Too small overlap can lead to a certain "blockiness" of the recovered image. For more information, consult the references [Juv17], [AES17], [Ela10].

3.6.2 Color Restoration

Next, we describe how ONMF can be used in conjunction with convolutional neural networks to approach the problem of restoring color to grayscale images.

Color restoration without additional qualification of the term "restoration" is an ill-posed

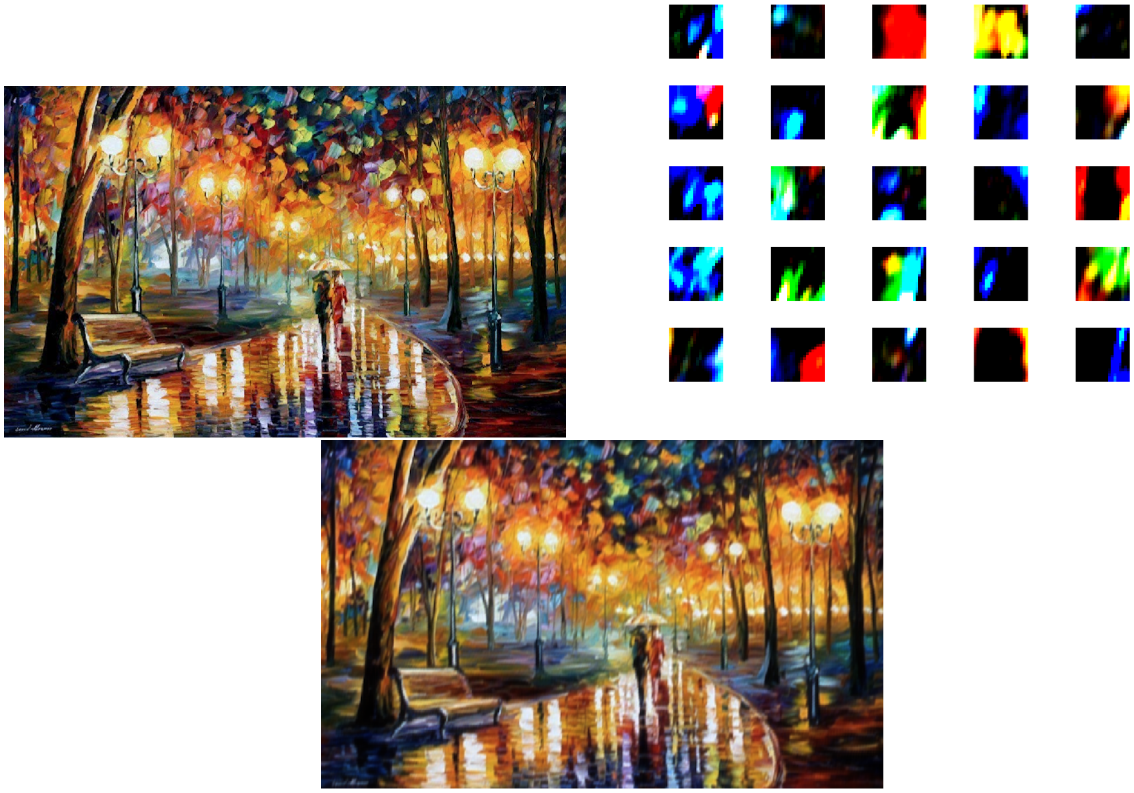

Figure 3.6: Image Compression Via ONMF. (Top) uncompressed image of Leonid Afremov's famous painting "Rain's Rustle." (Middle) 25 of the 100 learned dictionary elements, reshaped from their vectorized form to color image patch form. (Bottom): Painting compressed using a dictionary of 100 vectorized 20×20 color image patches obtained from 30 data samples of ONMF, each consisting of 1000 randomly selected sample patches. We used an overlap length of 15 in the patch averaging for the construction of the compressed image.

problem. Indeed, any sensible conversion from color to grayscale images is necessarily lossy. That said, of primary interest, are real-world images. For the sake of concreteness, one may have an old black and white photograph of a landscape, and wish to restore color to this photo. It is reasonable to assume a great degree of similarity between elements of old and recent photographs. Naturally occurring objects like grass, water, mountains, etc. look similar regardless of period. Thus the following idea suggests itself: learn a "landscape dictionary," trained on color images of modern landscapes, and use this dictionary to restore color to the grayscale landscape image.

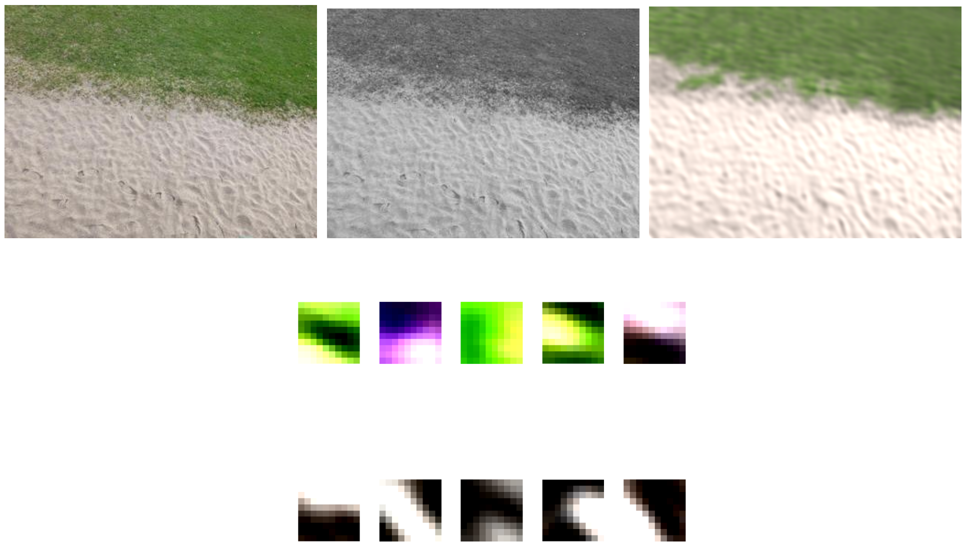

Figure 3.7: Color Restoration. (First) Original image of grass and sand. (Second) Conversion of first image using Matlab's default (linear) color to grayscale conversion function, rgb2gray. (Third) Restored color image, obtained by applying our method to the second (grayscale) image using the dictionary below. (Fourth) The first row displays the grass dictionary, the second row displays the sand dictionary. Each dictionary contains five 10 x 10 color image patches. These were trained on our ONMF with 20 batches of 1000 randomly selected sample patches each. We used maximal patch overlap in the restoration process.

One way to attempt the color restoration process is to take the landscape dictionary of color image patches and convert said patches to grayscale. There are many ways of doing so, but it is important to choose a linear conversion. With this dictionary of grayscale patches, one can use nonnegative least squares to approximate the image patches of the black and white landscape photo we wish to restore. The key idea is to use the coefficients in the representation by grayscale patches to form a color image patch using the corresponding color image dictionary elements. The linearity of the conversion from color to grayscale which produced the grayscale dictionary patches ensures that this procedure is consistent.

For this color restoration algorithm to be successful, it is necessary that in the above representation of an arbitrary grayscale image patch of the black and white photo, only (grayscale conversions of) the color landscape dictionary elements associated with this patch make a significant contribution. For example, if a grayscale image patch to be restored is extracted from a grass portion of the image, one would hope that ocean, mountain, cloud, etc. patches do not contribute. This may seem plausible at first, since the textures of all of these objects vary significantly, and so one might hope that in general only grayscale conversions of color grass image patches are used to represent black and white grass image patches. Unfortunately, this is not the case.

There are many difficulties in using a pure dictionary learning-based approach to color restoration. On one hand, images may abruptly change from pixel to pixel between different components of an image, e.g. the transition from mountain to sky. This suggests that one needs to use relatively small image patches, perhaps of size 10 x 10 or smaller, to avoid “mountain patches" spilling into the sky portion of the image or vise-versa. On the other hand, if there is any hope for the necessary condition for the success of color restoration that different components of the black and white image may be identified despite their absence of color, one needs e.g. spatial features such as texture to distinguish grass from other components. For small very image patches, these features are difficult to detect. One can see from the dictionary atoms in both of our color image patch dictionaries that there is little textural information. Even larger image patches exhibit this problem: the low-rank approximation of arbitrary image patches enforces a degree of “generality" in the few dictionary atoms learned to represent said patches and thus finer details like texture are obscured.

On the other hand, dictionary learning has its advantages. It takes relatively little data to learn an effective dictionary for tasks such as compression, and even color restoration of monochromatic objects. Motivated by this, we approached the color restoration problem in two stages. The first stage is to classify grayscale image patches of the black and white image to which we wish to restore color. Once the patch has been classified, we can use an appropriate dictionary of color image patches to restore color. For example, to restore color to black and white photos of landscapes, one may train a convolutional neural network to classify grayscale image patches as derived from grass, cloud, sky, mountain, water, etc. One can then train separate color image patch dictionaries corresponding to each of these elements.

We implemented this approach in a simple example of an image of grass juxtaposed with sand. The results of our algorithm are displayed in 3.7. We comment that extensions of this idea like those described above are possible, although they require more effort.

3.7 Conclusion

In this paper, we have presented various applications of ONMF to image and video processing as well as to the analysis of time-series. Our experiments demonstrate both the utility of ONMF on applications of general interest as well as the special insights its online nature can reveal in sequential data.

CHAPTER 4

Online Nonnegative CP-Dictionary Learning for Markovian Data

4.1 Attribution

This chapter is based off of the joint work "Online nonnegative CP-dictionary learning for Markovian data" by Hanbaek Lyu, Christopher Strohmeier, and Deanna Needell [LSN22] On the theoretical end, my contributions consisted of developing the tensor extension of ONMF which made use of "intermediate aggregation" steps, which were a key feature of the algorithm, ultimately enabling the theoretical work in the related paper [LNB20] , as well as in clarifying some of the technical details and developing appropriate extensions of key lemmas in the matrix setting to the tensor setting in the proof of our main theorem. On the applications side, I contributed the image and time series applications below 3.6 and 4.5, where the emphasis was on the contrast in structure between dictionary atoms learned from different unfoldings of the relevant data tensors. Hanbaek Lyu contributed both the general strategy of the proof, adapted from [LNB20], as well as many of the technical details. On the applications side, he contributed 4.4, and the idea to illustrate temporal activation patterns. Deanna Needell oversaw the project, highlighting weaknesses and errors in the theoretical component, suggesting interesting applications, and greatly improving the clarity of the work.

4.2 Introduction

In modern signal processing applications, there is often a critical need to analyze and understand data that is high-dimensional (many variables), large-scale (many samples), and multi-modal (many attributes). For unimodal (vector-valued) data, *matrix factorization* provides a powerful tool for one to describe data in terms of a linear combination of factors or atoms. In this setting, we have a data matrix $X \in \mathbb{R}^{d \times n}$, and we seek a factorization of X into the product WH for $W \in \mathbb{R}^{d \times R}$ and $H \in \mathbb{R}^{R \times n}$. Including two classical matrix factorization algorithms of Principal Component Analysis (PCA) [WEG87] and Nonnegative Matrix Factorization (NMF) [LS99], this problem has gone by many names over the decades, each with different constraints: dictionary learning, factor analysis, topic modeling, component analysis. It has applications in text analysis, image reconstruction, medical imaging, bioinformatics, and many other scientific fields more generally [SGH02, BB05, BBL+07, CWS+11, TN12, BMB+15, RPZ+18].

A *tensor* is a multi-way array that is a natural generalization of a matrix (which is itself a 2-mode tensor) and is suitable for representing multi-modal data. As matrix factorization is for unimodal data, *tensor factorization* (TF) provides a powerful and versatile tool that can extract useful latent information out of multi-modal data tensors. As a result, tensor factorization methods have witnessed increasing popularity and adoption in modern data science. One of the standard tensor factorization paradigms is CANDECOMP/PARAFAC (CP) decomposition [Tuc66, H+70, CC70]. In this setting, given a n-mode data tensor $\mathcal{X}$,

[H]

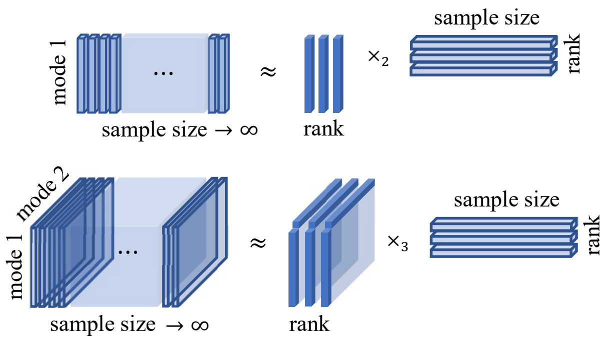

Figure 4.1: Illustration of online MF (top) and online CP-decomposition (bottom). n-mode tensors arrive sequentially and past data are not stored. One seeks n loading matrices that give approximate decomposition of all past data.

one seeks n *loading matrices* $U^{(1)}, \dots, U^{(n)}$, each with R columns, such that $\mathcal{X}$ is approximated by the sum of the outer products of the respective columns of U_i's. In other words, regarding the n-mode tensor $\mathcal{X}$ as the joint probability distribution of n random variables, the CP-decomposition approximates such a joint distribution as the sum of R product distributions, where the columns of the loading matrices give one-dimensional marginal distributions used to form the product distributions. A particular instance of CP-decomposition is when the data tensor and all of its loading matrices are required to have nonnegative entries. As pointed out in the seminal work of Lee and Seung [LS99] (in the matrix case), imposing a nonnegativity constraint in the decomposition problem helps one to learn interpretable features from multi-modal data.

Besides being multi-modal, another unavoidable characteristic of modern data is its enormous volume and the rate at which new data are generated. *Online learning* algorithms permit incremental processing that overcomes the sample complexity bottleneck inherent to batch processing, which is especially important when storing the entire data set is cumbersome. Not only do online algorithms address capacity and accessibility, but they also have the ability to learn qualitatively different information than offline algorithms for data that admit such a “sequential" structure (see e.g. [LMNS20]). In the literature, many “online" variants of more classical “offline" algorithms have been extensively studied — NMF [MBPS10, GTLY12, LNB20], TF [ZVB$^+$16, HNHA15, ZEB18, DZLZ18, SHSK18], and dictionary learning [RLH19, AGMM15, AGM14, KWSR17]. Online Tensor Factorization (OTF) algorithms with suitable constraints (e.g., nonnegativity) can serve as valuable tools that can extract interpretable features from multi-modal data.

4.2.1 Contribution

In this work, we develop a novel algorithm and theory for the problem of *online CP-dictionary learning*, where the goal is to progressively learn a dictionary of rank-1 tensors (CP-dictionary) from a stream of tensor data. Namely, given n-mode nonnegative tensors $(\mathcal{X}_t)_{t\geq 0}$, we seek to find an adaptively changing sequence of nonnegative CP-dictionaries such that the current CP-dictionary can approximate all tensor-valued signals in the past as a suitable nonnegative linear combination of its CP-dictionary atoms (see Figure 4.1 in Section 4.1). Our framework is flexible enough to handle general situations of an arbitrary number of modes in the tensor data, arbitrary convex constraints in place of the nonnegativity constraint, and a sparse representation of the data using the learned rank-1 tensors. In particular, our problem setting includes online nonnegative CP-decomposition.

Furthermore, we rigorously establish that under mild conditions, our online algorithm produces a sequence of loading matrices that converge almost surely to the set of stationary points of the objective function. In particular, our convergence results hold not only when the sequence of input tensors $(\mathcal{X}_t)_{t\geq 0}$ are independent and identically distributed (i.i.d.), but also when they form a Markov chain or functions of some underlying Markov chain. Such a theoretical convergence guarantee for online NTF algorithms has not been available even under the i.i.d. assumption on the data sequences. The relaxation to the Markovian setting is particularly useful in practice since often the signals have to be sampled from some complicated or unknown distribution, and obtaining even approximately independent samples is difficult. In this case, the Markov Chain Monte Carlo (MCMC) approach provides a powerful sampling technique (e.g., sampling from the posterior in Bayesian methods [VRCB18] or from the Gibbs measure for Cellular Potts models [VB12], or motif sampling from sparse graphs [LMS19]), where consecutive signals can be highly correlated.

4.2.2 Approach

Our algorithm combines the Stochastic Majorization-Minimization (SMM) framework [Mai13b], which has been used for online NMF algorithms [MBPS10, GTLY12, ZTX17, LNB20], and a recent work on block coordinate descent with diminishing radius (BCD-DR) [Lyu20]. In SMM, one iteratively minimizes a recursively defined surrogate loss function $\hat{f}_t$ that majorizes the empirical loss function f_t. A premise of SMM is that $\hat{f}_t$ is convex so that it is easy to minimize, which is the case for online matrix factorization problems in the aforementioned references. However, in the setting of factorizing n-mode tensors, $\hat{f}_t$ is only convex in each of the n loading matrices and nonconvex jointly in all loading matrices. Our main algorithm (Algorithm 6) only approximately minimizes $\hat{f}_t$ by a single round of cyclic block coordinate descent (BCD) in the n loading matrices. This additional layer of relaxation causes a number of technical difficulties in convergence analysis. One of our crucial innovations to handle them is to use a search radius restriction during this process [Lyu20], which is reminiscent of restricting step sizes in stochastic gradient descent algorithms and is in some sense 'dual' to proximal modifications of BCD [GS00, XY13].

Our convergence analysis on dependent data sequences uses the technique of "conditioning on a distant past", which leverages the fact that while the one-step conditional distribution of a Markov chain may be a constant distance away from the stationary distribution π, the N-step conditional distribution is exponentially close to π in N. This technique has been developed in [LNB20] recently to handle dependence in data streams for online NMF algorithms.

4.2.3 Related work

We roughly divide the literature on TF into two classes depending on *structured* or *unstructured* TF problems. The *structured TF problem* concerns recovering exact loading matrices of a tensor, where a structured tensor decomposition with loading matrices satisfying some incoherence or sparsity conditions is assumed. A number of works address this problem in the offline setting [TS15, AGJ15, SV17, SLLC17, BKS15, MSS16, SS17]. Recently, [RLH20] addresses an online structured TF problem by reducing it to an online MF problem using sparsity constraints on all but one loading matrices.

On the other hand, in the *unstructured TF problem*, one does not make any modeling assumption on the tensor subject to a decomposition so there are no true factors to be discovered. Instead, given an arbitrary tensor, one tries to find a set of factors (matrices or tensors) that gives the best fit of a chosen tensor decomposition model. In this case, convergence to a globally optimal solution cannot be expected, and global convergence to stationary points of the objective function is desired. For offline problems, global convergence to stationary points of the block coordinate descent method is known to hold under some regularity assumptions on the objective function [GS00, GS99, Ber99]. The recent works [ZVB$^+$16, HNHA15, ZEB18, DZLZ18, SHSK18] on online TF focus on computational considerations and do not provide a convergence guarantee. For online NMF, almost sure convergence to stationary points of a stochastic majorization-minimization (SMM) algorithm under i.i.d. data assumption is well-known [MBPS10], which has been recently extended to the Markovian case in [LNB20]. Similar global convergence for online TF is not known even under the i.i.d. assumption. The main difficulty of extending a similar approach to online TF is that the recursively constructed surrogate loss functions are nonconvex and cannot be jointly minimized in all n loading matrices when $n \geq 2$.

There are several recent works improving standard CP-decomposition algorithms such as the alternating least squares (ALS) (see, e.g., [KB09]). [BBK18] proposes a randomized ALS algorithm, that subsamples rows from each factor matrix update, which is an overdetermined least squares problem. A similar technique of row subsampling was used in the context of high-dimensional online matrix factorization [MMTV17]. [MYW18] proposed a randomized algorithm for online CP-decomposition but no theoretical analysis was provided. Also, CP-decomposition with structured factor matrices has been investigated in [GBB$^+$15]. On the other hand, [VMV15] considers a more efficient version of gradient descent type algorithms for CP-decomposition.

In the context of dictionary learning, there is an interesting body of work considering tensor-dictionary learning based on the Tucker-decomposition [SBS16, GSSB17, SSB18, GSSB19, SSB$^+$19]. When learning a reconstructive tensor dictionary for tensor-valued data, one can impose additional structural assumptions on the tensor dictionary in order to better exploit the tensor structure of the data and to gain reduced computational complexity. While in this work we consider the CP-decomposition model for the tensor-dictionary (also in an online setting), the aforementioned works consider the Tucker-decomposition instead and obtain various results on sample complexity, identifiability of a Tucker dictionary, and local convergence.

While our approach largely belongs to the SMM framework, there are related works using stochastic gradient descent (SGD). In [ZTX17], an online NMF algorithm based on projected SGD with general divergence in place of the squared ℓ_2-loss is proposed, and convergence to stationary points to the expected loss function for i.i.d. data samples is shown. In [SSY18], a similar convergence result for stochastic gradient descent algorithms for *unconstrained* nonconvex optimization problems with Markovian data samples is shown. While none of these results can be directly applied to our setting of online NTF for Markovian data, it may be possible to develop an SGD based approach for our setting, and it will be interesting to compare the performance of the algorithms based on SMM and SGD.

4.2.4 Organization

In Section 4.3 we first give a background discussion on NTF and CP-decomposition and then state the main optimization problem we address in this paper (see (4.13)). In Section 4.4, we provide the main algorithm (Algorithm 6) and give an overview of the main idea. Section 4.5 states the main convergence result in this paper, Theorem 1, together with a discussion on necessary assumptions and key lemmas used for the proof. In Section 4.6 we give the proof of the main result, Theorem 1. In Section 4.7, we compare the performance of our main algorithm on the offline nonnegative CP-decomposition problem against other baseline algorithms – Alternating Least Squares and Multiplicative Update. We then illustrate our approach on a diverse set of applications in Section 4.8; these applications are chosen to showcase the advantage of being able to flexibly reshape multi-modal tensor data and learn CP-dictionary atoms for any desired group of modes jointly.

In Appendix 4.9, we provide some additional background on Markov chains and Markov chain Monter Carlo sampling. Appendix 4.10 contains some auxiliary lemmas. In Appendix 4.11, we provide a memory-efficient implementation (Algorithm 7) of Algorithm 6 that uses bounded memory regardless of the length of the data stream.

4.2.5 Notation

For each integer $k \geq 1$, denote $[k] = \{1, 2, \dots, k\}$. Fix integers $n, I_1, \dots, I_n \geq 1$. An n-mode tensor $\mathbf{X}$ of shape $I_1 \times \cdots \times I_n$ is a map $(i_1, \dots, i_n) \mapsto \mathbf{X}(i_1, \dots, i_n) \in \mathbb{R}$ from the multi-index set $[I_1] \times \cdots \times [I_n]$ into the real line $\mathbb{R}$. We identify 2-mode tensors with matrices and 1-mode tensors with vectors, respectively. We do not distinguish between vectors and columns of matrices. For two real matrices A and B, we denote their Frobenius inner product as $\langle A, B \rangle := \mathrm{tr}(B^T A)$ whenever the sizes match. If we have N n-mode tensors $\mathbf{X}_1, \dots, \mathbf{X}_N$ of the same shape $I_1 \times \cdots \times I_n$, we identify the tuple $[\mathbf{X}_1, \dots, \mathbf{X}_N]$ as the $(n+1)$-mode tensor $\mathcal{X}$ of shape $I_1 \times \cdots \times I_n \times N$, whose the i^{th} slice along the $(n+1)^{\text{st}}$ mode equals $\mathbf{X}_i$. For given n-mode tensors $\mathbf{A}$ and $\mathbf{B}$, denote by $\mathbf{A} \odot \mathbf{B}$ and $\mathbf{A} \otimes_{kr} \mathbf{B}$ their Hadamard (pointwise) product and Katri-Rao product, respectively. When $\mathbf{B}$ is a matrix, for each $1 \leq j \leq n$, we also denote their mode-j product by $\mathbf{A} \times_j \mathbf{B}$. (See [KB09] for an excellent survey of tensor algorithms, albeit with notation that differs from our own).

4.3 Background and problem formulation

4.3.1 CP-dictionary learning and nonnegative tensor factorization

Assume that we are given N observed vector-valued signals $x_1, \dots, x_N \in \mathbb{R}^d_{\geq 0}$. Fix an integer $R \geq 1$ and consider the following approximate factorization problem (see (4.2) for a precise statement)

$$[x_1, \dots, x_N] \approx [u_1, \dots, u_R] \times_2 H \qquad \Longleftrightarrow \qquad X \approx UH, \tag{4.1}$$

where $\times_2$ denotes the mode-2 product, $X = [x_1, \dots, x_N] \in \mathbb{R}^{d \times N}_{\geq 0}$, $U = [u_1, \dots, u_R] \in \mathbb{R}^{d \times R}_{\geq 0}$, and $H \in \mathbb{R}^{R \times N}_{\geq 0}$. The right hand side (4.1) is the well-known *nonnegative matrix factorization* (NMF) problem, where the use of nonnegativity constraint is crucial in obtaining a "parts-based" representation of the input signals [LS99]. Such an approximate factorization learns R *dictionary atoms* $u_1, \dots, u_R$ that together can approximate each observed signal x_j by using the nonnegative linear coefficients in the j^{th} column of H. The factors U and H in

(4.1) above are called the *dictionary* and *code* of the data matrix X, respectively. They can be learned by solving the following optimization problem

$$\mathop{\arg\min}_{U'\in\mathbb{R}_{\geq 0}^{d\times R},\,H'\in\mathbb{R}_{\geq 0}^{R\times N}} \left(\|X-U'H'\|_F^2+\lambda\|H'\|_1\right), \tag{4.2}$$

where $\lambda \geq 0$ is a regularization parameter that encourages a sparse representation of the columns of X over the columns of U. Note that (4.2) is also known as a *dictionary learning problem* [OF97, EAH99, LS00, EA06, LHK05], especially when $R \geq d$.

Next, suppose we have N observed n-mode tensor-valued signals $\mathbf{X}_1, \dots, \mathbf{X}_N \in \mathbb{R}_{\geq 0}^{I_1\times\cdots\times I_n}$. A direct tensor analogue of the NMF problem (4.1) would be the following:

$$[\mathbf{X}_1,\dots,\mathbf{X}_N] \approx [\mathbf{D}_1,\dots,\mathbf{D}_R]\times_{n+1} H \qquad \Longleftrightarrow \qquad \mathcal{X}\approx\mathcal{D}\times_{n+1} H, \tag{4.3}$$

where $\mathcal{X}=[\mathbf{X}_1,\dots,\mathbf{X}_N]\in\mathbb{R}_{\geq 0}^{I_1\times\cdots\times I_n\times N}$, $\mathcal{D}=[\mathbf{D}_1,\dots,\mathbf{D}_R]\in\mathbb{R}_{\geq 0}^{I_1\times\cdots\times I_n\times R}$, and $H\in\mathbb{R}_{\geq 0}^{R\times N}$. As before, we call $\mathcal{D}$ and H above the dictionary and code of the data tensor $\mathcal{X}$, respectively. Note that this problem is equivalent to (4.1) since

$$\|\mathcal{X}-\mathcal{D}\times_{n+1} H\|_F^2 = \|\mathrm{MAT}(\mathcal{X})-\mathrm{MAT}(\mathcal{D})\times_2 H\|_F^2, \tag{4.4}$$

where $\mathrm{MAT}(\cdot)$ is the matricization operator that vectorizes (using lexicographic ordering of entries) each slice with respect to the last mode. For instance, $\mathrm{MAT}([\mathbf{X}_1,\dots,\mathbf{X}_N])$ is a $(I_1\cdots I_n)\times N$ matrix whose i^{th} column is the vectorization of $\mathbf{X}_i$.

Now, consider imposing an additional structural constraint on the dictionary atoms $\mathbf{D}_1,\dots,\mathbf{D}_N$ in (4.3). Specifically, suppose we want each $\mathbf{D}_i$ to be the sum of R rank 1 tensors. Equivalently, we assume that there exist *loading matrices* $[U^{(1)},\dots,U^{(n)}]\in\mathbb{R}_{\geq 0}^{I_1\times R}\times\cdots\times\mathbb{R}_{\geq 0}^{I_n\times R}$ such that

$$[\mathbf{D}_1,\dots,\mathbf{D}_R] = \mathtt{Out}(U^{(1)},\dots,U^{(n)}) \tag{4.5}$$

$$:= \left[\bigotimes_{k=1}^{n} U^{(k)}(:,1),\ \bigotimes_{k=1}^{n} U^{(k)}(:,2),\ \dots,\ \bigotimes_{k=1}^{n} U^{(k)}(:,R)\right]\in\mathbb{R}_{\geq 0}^{I_1\times\cdots\times I_n\times R}, \tag{4.6}$$

where $U^{(k)}(:,j)$ denotes the i^{th} column of the $I_k\times R$ matrix $U^{(k)}$ and $\otimes$ denotes the outer product. Note that we are also defining the operator $\mathtt{Out}(\cdot)$ here, which will be used throughout this paper. In this case, the tensor factorization problem in (4.3) becomes (a more precise statement is given in (4.13))

$$[\mathbf{X}_1, \dots, \mathbf{X}_N] \approx \mathtt{Out}(U^{(1)}, \dots, U^{(n)}) \times_{n+1} H. \tag{4.7}$$

When $N = 1$ and $\lambda = 0$, then $H \in \mathbb{R}_{\geq 0}^{R\times 1}$, so by absorbing the i^{th} entry of H into the $\mathbf{D}_i$, we see that the above problem (4.7) reduces to

$$\mathbf{X} \approx \sum \mathtt{Out}(U^{(1)}, \dots, U^{(n)}) := \sum_{i=1}^{R} \bigotimes_{k=1}^{n} U^{(k)}(:, i), \tag{4.8}$$

which is the nonnegative CANDECOMP/PARAFAC (CP) decomposition problem [Tuc66, H$^+$70, CC70]. On the other hand, if $n = 1$ so that $\mathbf{X}_i$ are vector-valued signals, then (4.7) reduces to the classical dictionary learning problem (4.2). For these reasons, we refer to (4.7) as the *CP-dictionary learning* (CPDL) problem. We call the $(n+1)$-mode tensor $\mathtt{Out}(U^{(1)}, \dots, U^{(n)}) = [\mathbf{D}_1, \dots, \mathbf{D}_R]$ in $\mathbb{R}^{(I_1 \times \cdots \times I_n \times R)}$ a *CP-dictionary* and the matrix $H \in \mathbb{R}_{\geq 0}^{R\times N}$ the *code* of the dataset $\mathcal{X} = [\mathbf{X}_1, \dots, \mathbf{X}_N]$, respectively. Here we call the rank-1 tensors $\mathbf{D}_i$ the *atoms* of the CP-dictionary.

4.3.2 Online CP-dictionary learning

Next, we consider an *online* version of the CPDL problem we considered in (4.7). Given a continuously arriving sequence of data tensors $(\mathbf{X}_t)_{t\geq 0}$, can we find an adaptively changing sequence of CP-dictionaries such that the current CP-dictionary can approximate all tensor-valued signals in the past as a suitable nonnegative linear combination of its CP-dictionary atoms (see Figure 4.1 in Section 4.1)? This online problem can be explicitly formulated as an empirical loss minimization framework, and we will also state an equivalent stochastic program (under some modeling assumption) that we rigorously address.

Fix constraint sets for code and loading matrices $\mathcal{C}^{\text{code}} \subseteq \mathbb{R}^{R\times b}$ and $\mathcal{C}^{(i)} \subseteq \mathbb{R}^{I_i \times R}$, $i = 1, \dots, n$, respectively (generalizing the nonnegativity constraints in Subsection 4.3.1). Write $\mathcal{C}^{\text{dict}} := \mathcal{C}^{(1)} \times \cdots \times \mathcal{C}^{(n)}$. For each $\mathcal{X} \in \mathbb{R}_{\geq 0}^{I_1 \times \cdots \times I_n \times b}$, $\mathcal{D} := [U^{(1)}, \dots, U^{(n)}] \in \mathbb{R}^{I_1 \times R} \times \cdots \times \mathbb{R}^{I_n \times R}$, $H \in \mathbb{R}^{R\times b}$, define

$$\ell(\mathcal{X}, \mathcal{D}, H) := \|\mathcal{X} - \mathtt{Out}(\mathcal{D}) \times_{n+1} H\|_F^2 + \lambda \|H\|_1, \tag{4.9}$$

$$\ell(\mathcal{X},\mathcal{D}) := \inf_{H\in\mathcal{C}^{\mathrm{code}}} \ell(\mathcal{X},\mathcal{D},H), \tag{4.10}$$

where $\lambda \geq 0$ is a regularization parameter. Fix a sequence of non-increasing weights $(w_t)_{t\geq 0}$ in $(0,1]$. Here $\mathcal{X}$ denotes a minibatch of b tensors in $\mathbb{R}^{I_1\times\cdots\times I_n}$, so minimizing $\ell(\mathcal{X},\mathcal{D})$ with respect to $\mathcal{D}$ amounts to fitting the CP-dictionary $\mathcal{D}$ to the minibatch of b tensors in $\mathcal{X}$.

The *online CP-dictionary learning* (online CPDL) problem is the following empirical loss minimization problem:

$$\text{Upon arrival of } \mathcal{X}_t\text{:}\quad \mathcal{D}_t \in \operatorname*{arg\,min}_{\mathcal{D}\in\mathcal{C}^{\mathrm{dict}}} \left(f_t(\mathcal{D}) := (1-w_t)f_{t-1}(\mathcal{D}) + w_t\,\ell(\mathcal{X}_t,\mathcal{D})\right), \tag{4.11}$$

where f_t is the *empirical loss function* recursively defined by the weighted average in (4.11) with $f_0 \equiv 0$. One can solve the recursion in (4.11) and obtain the more explicit formula for the empirical loss:

$$f_t(\mathcal{D}) = \sum_{k=1}^{t} \ell(\mathcal{X}_k,\mathcal{D})\,w_k^t, \qquad w_k^t := w_k \prod_{i=k+1}^{t}(1-w_i). \tag{4.12}$$

The weight w_t in (4.11) controls how much we want our new loading matrices in $\mathcal{D}_t$ to deviate from minimizing the previous empirical loss f_{t-1} to adapting to the newly observed tensor data $\mathcal{X}_t$. In one extreme case of $w_t \equiv 1$, $\mathcal{D}_t$ is a minimizer of the time-t loss $\ell(\mathcal{X}_t,\cdot)$ and ignores the past f_{t-1}. If $w_t \equiv \alpha \in (0,1)$ then the history is forgotten exponentially fast, that is, $f_t(\cdot) = \sum_{s=1}^{t}\alpha(1-\alpha)^{t-s}\,\ell(\mathcal{X}_s,\cdot)$. On the other hand, the 'balanced weight' $w_t = 1/t$ makes the empirical loss to be the arithmetic mean: $f_t(\cdot) = \frac{1}{t}\sum_{s=1}^{t}\ell(\mathcal{X}_s,\cdot)$, which is the choice made for the online NMF problem in [MBPS10]. Therefore, one can choose the sequence of weights $(w_t)_{t\geq 1}$ in Algorithm 6 in a desired way to control the sensitivity of the algorithm to the newly observed data. That is, make the weights decay fast for learning average features and decay slow (or keep it constant) for learning trending features. We mention that our theoretical convergence analysis covers only the former case.

We note that the online CPDL problem (4.11) involves solving a constrained optimization problem for each t, which is practically infeasible. Hence we may compute a sub-optimal sequence $(\mathcal{D}_t)_{t\geq 0}$ of tuples of loading matrices (see Algorithm 6) and assess its asymptotic fitness to the original problem (4.11). We seek to perform some rigorous theoretical analysis at the expense of some suitable but non-restrictive assumption on the data sequence $\mathcal{X}_t$ as well as the weight sequence $(w_t)_{t\geq 1}$. A standard modeling assumption in the literature is to assume the data sequence $\mathcal{X}_t$ are independent and identically distributed (i.i.d.) according to some distribution π [MBPS10, Mai13b, MMTV17, ZTX17]. We consider a more relaxed setting where $\mathcal{X}_t$ is given as a function of some underlying Markov chain (see (A1)) and π is the stationary distribution of $(\mathcal{X}_t)_{t\geq 1}$ viewed as a stochastic process. Under this assumption, consider the following stochastic program

$$\operatorname*{arg\,min}_{\mathcal{D}\in\mathcal{C}^{\text{dict}}} \big(f(\mathcal{D}) := \mathbb{E}_{\mathcal{X}\sim\pi}\left[\ell(\mathcal{X},\mathcal{D})\right] \big), \tag{4.13}$$

where the *random* tensor $\mathcal{X}$ is sampled from the distribution π, and we call the function f defined in (4.13) the *expected loss function*. The connection between the online (4.11) and the stochastic (4.13) formulation of the CPDL problem is that, if the parameter space $\mathcal{C}^{\text{dict}}$ is compact and the weights w_t satisfy some condition, then $\sup_{\mathcal{D}} |f_t(\mathcal{D}) - f(\mathcal{D})| \to 0$ almost surely as $t\to\infty$. (see Lemma 9 in Appendix 4.10). Hence, under this setting, we seek to find a sequence $(\mathcal{D}_t)_{t\geq 1}$ that converges to a solution to (4.13). In other words, fitness to the *single* expected loss function f is enough to deduce the asymptotic fitness to *all* empirical loss functions f_t.

Once we find an optimal CP-dictionary $\mathcal{D}^* = \texttt{Out}(U^{(1)},\dots,U^{(n)})$ for (4.13), then we can obtain an optimal code matrix $H = H(\mathcal{X}) \in \mathbb{R}^{R\times b}$ for each realization of the random tensor $\mathcal{X}$ by solving the convex problem in (4.10). Demanding optimality of the CP-dictionary $\mathcal{D}^*$ and leaving the code matrix $H = H(\mathcal{X})$ adjustable in this way is a more flexible framework for CP-decomposition of a random (as well as online) tensor than seeking a pair of jointly optimal CP-dictionary $\mathcal{D}^*$ and code matrix H^*, especially when the variation of the random tensor is large. However, if we have a single deterministic tensor $\mathcal{X}$ to be factorized, then these two formulations are equivalent since $\min_{\mathcal{D},\mathcal{H}} \ell(\mathcal{X},\mathcal{D},H) = \min_{\mathcal{D}}\min_{H} \ell(\mathcal{X},\mathcal{D},H)$.

4.4 The Online CP-Dictionary Learning algorithm

In this section, we state our main algorithm (Algorithm 6). For simplicity, we first give a preliminary version for the case of $n = 2$ modes, minibatch size $b = 1$, with nonnegativity constraints. Suppose we have learned $n = 2$ loading matrices $\mathcal{D}_{t-1} := [U_{t-1}^{(1)}, U_{t-1}^{(2)}]$ from the sequence $\mathcal{X}_1, \dots, \mathcal{X}_{t-1}$ of data tensors, $\mathcal{X}_i \in \mathbb{R}^{I_1 \times I_2 \times 1}$. Then we compute the updated loading matrices $\mathcal{D}_t = [U_t^{(1)}, U_t^{(2)}]$ by

$$\begin{cases} H_t & \leftarrow \operatorname*{arg\,min}_{H \in \mathbb{R}^{R\times 1}_{\geq 0}} \ell(\mathcal{X}_t, \mathcal{D}_{t-1}, H) \\ \hat{f}_t(\mathcal{D}) & \leftarrow (1-w_t)\hat{f}_{t-1}(\mathcal{D}) + w_t \ell(\mathcal{X}_t, \mathcal{D}, H_t) \\ U_t^{(1)} & \leftarrow \operatorname*{arg\,min}_{U \in \mathbb{R}^{I_i \times R}_{\geq 0},\, \|U - U_{t-1}^{(1)}\|_F \leq c' w_t} \hat{f}_t(U, U_{t-1}^{(2)}) \\ U_t^{(2)} & \leftarrow \operatorname*{arg\,min}_{U \in \mathbb{R}^{I_i \times R}_{\geq 0},\, \|U - U_{t-1}^{(2)}\|_F \leq c' w_t} \hat{f}_t(U_t^{(1)}, U), \end{cases} \tag{4.14}$$

where $\lambda \geq 0$ is an absolute constant and $(w_t)_{t\geq 1}$ is a non-increasing sequence of weights in $(0,1]$. The recursively defined function $\hat{f}_t : \mathcal{D} = [U^{(1)}, \dots, U^{(n)}] \mapsto [0, \infty)$ is called the *surrogate loss function*, which is quadratic in each factor $U^{(i)}$ but is not jointly convex. Namely, when the new tensor data $\mathcal{X}_t$ arrives, one computes the code $H_t \in \mathbb{R}^{R\times 1}_{\geq 0}$ for $\mathcal{X}_t$ with respect to the previous loading matrices in $\mathcal{D}_{t-1}$, updates the surrogate loss function $\hat{f}_t$, and then *sequentially* minimizes it to find updated loading matrices within diminishing search radius $c' w_t$.

Note that the surrogate loss function $\hat{f}_t$ in (4.16) is defined by the same recursion that defines the empirical loss function in (4.11). However, notice that the loss term $\ell(\mathcal{X}_t, \mathcal{D})$ in the definition of the empirical loss function f_t in (4.13) involves optimizing over the code matrices H in (4.10), which should be done for every $\mathcal{X}_s$, $1 \leq s \leq t$, in order to evaluate f_t. On the contrary, in the definition of the surrogate loss function $\hat{f}_t$ in (4.16), this term is replaced with the sub-optimal loss $\ell(\mathcal{X}_t, \mathcal{D}, H_t)$, which is sub-optimal since H_t was found by decomposing $\mathcal{X}_t$ using the previous CP-dictionary $\texttt{Out}(\mathcal{D}_{t-1})$. From this, it is clear that $\hat{f}_t \geq f_t$ for all $t \geq 0$. In other words, $\hat{f}_t$ is a majorizing surrogate of f_t.

Now we state our algorithm in the general mode case in Algorithm 6.

Algorithm 6 Online CP-Dictionary Learning (online CPDL)

```
0: Input: $(\mathcal{X}_t)_{1\le t\le T}$ (minibatches of data tensors in $\mathbb{R}^{I_1\times\cdots\times I_n\times b}$); $[U_0^{(1)},\dots,U_0^{(n)}]\in\mathbb{R}^{I_1\times R}\times\cdots\times\mathbb{R}^{I_n\times R}$ (initial loading matrices); $c'>0$ (search radius constant);
0: Constraints: $\mathcal{C}^{(i)}\subseteq\mathbb{R}^{I_i\times R}$, $1\le i\le n$, $\mathcal{C}^{\text{code}}\subseteq\mathbb{R}^{R\times b}$ (e.g., nonnegativity constraints)
0: Parameters: $R\in\mathbb{N}$ (# of dictionary atoms); $\lambda\ge 0$ ($\ell_1$-regularization parameter); $(w_t)_{t\ge 1}$ (weights in $(0,1]$);
0:   Initialize surrogate loss $\hat{f}_0\equiv 0$;
0:   For $t=1,\dots,T$ do:
0:     Coding: Compute the optimal code matrix
         $H_t \leftarrow \underset{H\in\mathcal{C}^{\text{code}}\subseteq\mathbb{R}^{R\times b}}{\arg\min}\ \ell(\mathcal{X}_t,U_{t-1}^{(1)},\dots,U_{t-1}^{(n)},H);$   (using Algorithm 9)   (4.15)
0:     Update surrogate loss function:
         $\hat{f}_t(U^{(1)},\dots,U^{(n)})\leftarrow(1-w_t)\hat{f}_{t-1}(U^{(1)},\dots,U^{(n)})+w_t\,\ell(\mathcal{X}_t,U^{(1)},\dots,U^{(n)},H_t)$   (4.16)
0:     Update loading matrices by restricted cyclic block coordinate descent:
0:       For $i=1,\dots,n$ do:
0:
         $\mathcal{C}_t^{(i)}\leftarrow\left\{U\in\mathcal{C}^{(i)}\,\middle|\,\|U-U_{t-1}^{(i)}\|_F\le c'w_t\right\}$; (▷ Restrict the search radius by $c'w_t$)   (4.17)
         $U_t^{(i)}\leftarrow\underset{U\in\mathcal{C}_t^{(i)}}{\arg\min}\ \hat{f}_t\left(U_t^{(1)},\dots,U_t^{(i-1)},U,U_{t-1}^{(i+1)},\dots,U_{t-1}^{(n)}\right)$;   (4.18)
             (▷Update the $i^{\text{th}}$ loading matrix)
0:       End for
0:   End for
0: Return: $[U_T^{(1)},\dots,U_T^{(n)}]\in\mathcal{C}^{(1)}\times\cdots\times\mathcal{C}^{(n)}$; =0
```

Algorithm 6 combines two key elements: stochastic majorization-minimization (SMM) [Mai13b] and block coordinate descent with diminishing radius (BCD-DR) [Lyu20]. SMM amounts to iterating the following steps: sampling new data points, constructing a strongly convex surrogate loss, and then minimizing the surrogate loss to update the current estimate. This framework has been successfully applied to online matrix factorization problems [MBPS10, MMTV17]. However, the biggest bottleneck in using a similar approach in the tensor case is that the surrogate loss function $\hat{f}_t$ in (4.16) is only block multi-convex, meaning that it is convex in each block of coordinates but nonconvex jointly. Hence we cannot find an exact minimizer for $\hat{f}_t$ to update all n loading matrices at the same time.

In order to circumvent this issue, one could try to perform a few rounds of block coordinate descent (BCD) on the surrogate loss function $\hat{f}_t$, which can be easily done since $\hat{f}_t$ is convex in each loading matrix. However, this results in sub-optimal loading matrices in each iteration, causing a number of difficulties in convergence analysis. Moreover, global convergence of BCD to stationary points is not guaranteed in general even for the deterministic tensor CP-decomposition problems without constraints [KB09], and such a guarantee is known only with some additional regularity conditions [GS99, GS00, Ber99]. There are other popular strategies of using proximal [GS00] or prox-linear [XY13] modifications of BCD to improve convergence properties. While these methods only ensure square-summability $\sum_{t=1}^{\infty} \|\mathcal{D}_t - \mathcal{D}_{t-1}\|_F^2 < \infty$ of changes (see, e.g., [XY13, Lem 2.2]), we find it crucial for our stochastic analysis that we are able to control the individual changes $\|\mathcal{D}_t - \mathcal{D}_{t-1}\|_F$ of the loading matrices in each iteration. This motivates to use BCD with diminishing radius in [Lyu20]. More discussions on technical points in convergence analysis are given in Subsection 4.5.2.

The coding step in (4.15) is a convex problem and can be easily solved by a number of known algorithms (e.g., LARS [EHJ+04], LASSO [Tib96], and feature-sign search [LBRN07]). As we have noted before, the surrogate loss function $\hat{f}_t$ in (4.16) is quadratic in each block coordinate, so each of the subproblems in the factor matrix update step in (4.18) is a constrained quadratic problem and can be solved by projected gradient descent (see [MBPS10, LNB20]).

Notice that implementing Algorithm 6 may seem to require unbounded memory as one needs to store all past data $\mathcal{X}_1, \dots, \mathcal{X}_t$ to compute the surrogate loss function $\hat{f}_t$ in (4.16).

However, it turns out that there are certain bounded-sized statistics that aggregates the past information that are sufficient to parameterize $\hat{f}_t$ and also to update the loading matrices. This bounded memory implementation of Algorithm 6 is given in Algorithm 7, and a detailed discussion on the memory efficiency is relegated to Appendix 4.11.

4.5 Convergence results

In this section, we state our main convergence result of Algorithm 6. Note that all results that we state here also apply to Algorithm 7, which is a bounded-memory implementation of Algorithm 6.

4.5.1 Statement of main results

We first layout all technical assumptions required for our convergence results to hold.

(A1). *The observed minibatch of data tensors* $\mathcal{X}_t = [\mathbf{X}_{t;1}, \dots, \mathbf{X}_{t;b}]$ *are given by* $\mathcal{X}_t = \varphi(Y_t)$*, where* Y_t *is an irreducible and aperiodic Markov chain defined on a finite state space* Ω *and* $\varphi : \Omega \to \mathbb{R}^{I_1 \times \cdots \times I_n \times b}$ *is a bounded function. Denote the transition matrix and the unique stationary distribution of* Y_t *by* P *and* π*, respectively.*

(A2). *For each* $1 \le i \le n$*, the* i^{th} *loading matrix is constrained to a compact and convex subset* $\mathcal{C}^{(i)} \subseteq \mathbb{R}^{I_i \times R}$ *that contains at least two points. Furthermore, the code matrices* H_t *belong to a compact and convex subset* $\mathcal{C}^{\text{code}} \subseteq \mathbb{R}^{R \times b}$.

(A3). *The sequence of non-increasing weights* $w_t \in (0, 1]$ *in Algorithm 6* $\sum_{t=1}^{\infty} w_t = \infty$*, and* $\sum_{t=1}^{\infty} w_t^2 \sqrt{t} < \infty$*. Furthermore,* $w_{t+1}^{-1} - w_t^{-1} \le 1$ *for all sufficiently large* t*.*

(A4). *The expected loss function* f *defined in* (4.13) *is continuously differentiable and has Lipschitz gradient.*

It is standard to assume that the sequence of signals is drawn from a data distribution of compact support in an independent fashion [MBPS10, Mai13a], which enables the processing of large data by using i.i.d. subsampling of minibatches. However, when the signals have to be sampled from some complicated or unknown distribution, obtaining even approximately independent samples is difficult. In this case, Markov Chain Monte Carlo (MCMC) provides a powerful sampling technique (e.g., sampling from the posterior in Bayesian methods [VRCB18] or from the Gibbs measure for Cellular Potts models [VB12], or motif sampling from sparse graphs [LMS19]), where consecutive signals could be highly correlated. (See Appendix 4.9 for more background on Markov chains and MCMC.)

An important notion in MCMC sampling is "exponential mixing" of the Markov chain[1]. For a simplified discussion, suppose in (A1) that our data tensors $\mathcal{X}_t$ themselves form a Markov chain with unique stationary distribution π. Under the assumption of finite state space, irreducibility, and aperiodicity in (A1), the Markov chain $\mathcal{X}_t$ "mixes" to the stationary distribution π at an exponential rate. Namely, for any $\varepsilon > 0$, one can find a constant $\tau = \tau(\varepsilon) = O(\log \varepsilon^{-1})$, called the "mixing time" of $\mathcal{X}_t$, such that the conditional distribution of $\mathcal{X}_{t+\tau}$ given $\mathcal{X}_t$ is within total variation distance ε from π regardless of the distribution of $\mathcal{X}_t$ (see (4.87) for the definition of total variation distance). This mixing property of Markov chains is crucial both for practical applications of MCMC sampling as well as our theoretical analysis. For instance, a common practice of using MCMC sampling to obtain approximate i.i.d. samples is to first obtaining a long Markov chain trajectory $(\mathcal{X}_t)_{t\geq 1}$ and then thinning it to the subsequence $(\mathcal{X}_{k\tau})_{k\geq 1}$ [BGJM11, Sec. 1.11]. Due to the choice of mixing time τ, this forms an ε-approximate i.i.d. samples from π.

However, thinning a Markov chain trajectory does not necessarily produce truly independent samples, so classical stochastic analysis that relies crucially on independence between data samples is not directly applicable. For instance, if $\mathcal{X}_t$ is a reversible Markov chain then the correlation within the subsequence is nonzero and at least of order ε (see Appendix 4.9.2 for the definition of reversibility and detailed discussion). In order to obtain truly independent samples, one may independently re-initialize a Markov chain trajectory, run it

[1]For our analysis, it is in fact sufficient to have a sufficiently fast polynomial mixing of the Markov chain. See (A1)' in Appendix 4.9 for a relaxed assumption using countable state-space.

for τ iterations, and keep the last samples in each run (e.g., see the discussion in [SSY18]). However, in both approaches, only one out of τ Markov chain samples are used for optimization, which could be extremely wasteful if the Markov chain converges to the stationary distribution slowly so that the implied constant in $\tau(\varepsilon) = O(\log \varepsilon^{-1})$ is huge.

Instead, our assumption on input signals in (A1) allows us to use every single sample in the same MCMC trajectory without having to "burn" lots of samples. Such Markovian extension of the classical OMF algorithm in [MBPS10] has recently been achieved in [LNB20], which has applications in dictionary learning, denoising, and edge inference problems for network data [LKVP21].

Assumption (A2) is also standard in the literature of dictionary learning (see [MBPS10, Mai13b]). A particular instance of interest is when they are confined to having nonnegative entries, in which case the learned dictionary components give a “parts-based” representation of the input signals [LS99].

Assumption (A3) states that the sequence of weights $w_t \in (0, 1]$ we use to recursively define the empirical loss (4.11) and surrogate loss (4.16) does not decay too fast so that $\sum_{t=1}^{\infty} w_t = \infty$ but decay fast enough so that $\sum_{t=1}^{\infty} w_t^2 \sqrt{t} < \infty$. This is analogous to requirements for stepsizes in stochastic gradient descent algorithms, where the stepsizes are usually required to be non-summable but square-summable (see, e.g., [SSY18]). The additioanl factor $\sqrt{t}$ is used in our analysis to deduce the uniform convergence of the empirical loss f_t to the expected loss f (see Lemma 9 in Appendix 4.10), which was also the case in the literature [MBPS10, Mai13b, MMTV17, LNB20]. Also, the condition $w_t^{-1} - w_{t-1}^{-1} \leq 1$ for all suficiently large t is equivalent to saying the recursively defined weights w_k^t in (4.12) are non-decreasing in k for all sufficiently large k, which is required to use Lemma 9 in Appendix 4.10. We also remark that (A3) is implied by the following simpler condition:

(A3'). *The sequence of non-increasing weights $w_t \in (0, 1]$ in Algorithm 6 satisfy either $w_t = t^{-1}$ for $t \geq 1$ or $w_t = \Theta(t^{-\beta}(\log t)^{-\delta})$ for some $\delta \geq 1$ and $\beta \in [3/4, 1)$.*

For Assumption (A4), we remark that it follows from the uniqueness of the solution of (4.15) (see [MBPS10, Prop. 1]). This can be enforced for example by the elastic net penalization [ZH05]. Namely, we may add a quadratic regularizer $\lambda' \|H\|_F^2$ to the loss function ℓ in (4.9) for some $\lambda' > 0$. Then the resulting quadratic function is strictly convex and hence it has a unique minimizer in the convex constraint set $\mathcal{C}^{\text{code}}$. (See [MBPS10, Sec. 4.1] and [LNB20, Sec. 4.1] for more detailed discussion on this assumption).

The main result in this paper, which is stated below in Theorem 1, states that the sequence $\mathcal{D}_t$ of CP-dictionaries produced by Algorithm 6 converges to the set of stationary points of the expected loss function f defined in (4.13). To the best of our knowledge, Theorem 1 is the first convergence guarantee for any online *constrained* dictionary learning algorithm for tensor-valued signals or as an online *unconstrained* CP-factorization algorithm, which have not been available even under the classical i.i.d. assumption on input signals. Recall that f_t and $\hat{f}_t$ denote the empirical and surrogate loss function defined in (4.11) and (4.16), respectively.

Theorem 1. *Suppose (A1)-(A3) hold. Let $(\mathcal{D}_t)_{t\geq 1}$ be an output of Algorithm 6. Then the following hold.*

(i) $\lim_{t\to\infty} \mathbb{E}[f_t(\mathcal{D}_t)] = \lim_{t\to\infty} \mathbb{E}[\hat{f}_t(\mathcal{D}_t)] < \infty$.

(ii) $f_t(\mathcal{D}_t) - \hat{f}_t(\mathcal{D}_t) \to 0$ *and* $f(\mathcal{D}_t) - \hat{f}_t(\mathcal{D}_t) \to 0$ *as* $t \to \infty$ *almost surely.*

(iii) *Further assume (A4). Then the distance (measured by block-wise Frobenius distance) between $\mathcal{D}_t$ and the set of all stationary points of f in $\mathcal{C}^{\text{dict}}$ converges to zero almost surely.*

We acknowledge that a similar asymptotic convergence result under Markovian dependency in data samples has been recently obtained in [SSY18, Thm. 2] in the context of stochastic gradient descent for nonconvex optimization problems. The results are not directly comparable since [SSY18, Thm. 2] only handles unconstrained problems.

4.5.2 Key lemmas and overview of the proof of Theorem 1.

In this subsection, we state the key lemmas we use to prove Theorem 1 and illustrate our contribution to techniques for convergence analysis.

As we mentioned in Section 4.4, there is a major difficulty in convergence analysis in the multi-modal case $n \geq 2$ as the surrogate loss function $\hat{f}_t$ (see (4.16)) to be minimized for updating the loading matrices is only multi-convex in n blocks. Note that we can view our algorithm (Algorithm 6) as a multi-modal extension of stochastic majorization-minimization (SMM) in the sense that it reduces to standard SMM in the case of vector-valued signals ($n = 1$). We first list the properties of SMM that have been critically used in convergence analysis in related works [MBPS10, Mai13b, MMTV17, LNB20]:

libel=**0**, leftmirgin=0.6cm (Surrogate Optimality) $\mathcal{D}_t$ is a minimizer of $\hat{f}_t$ over $\mathcal{C}^{\text{dict}}$.

liibel=**0**, leftmiirgiin=0.6cm (Forward Monotonicity) $\hat{f}_t(\mathcal{D}_{t-1}) \geq \hat{f}_t(\mathcal{D}_t)$.

liiibel=**0**, leftmiiirgiiin=0.6cm (Backward Monotonicity) $\hat{f}_{t-1}(\mathcal{D}_{t-1}) \leq \hat{f}_{t-1}(\mathcal{D}_t)$.

livbel=**0**, leftmivrgivn=0.6cm (Second-Order Growth Property) $\hat{f}_t(\mathcal{D}_{t-1}) - \hat{f}_t(\mathcal{D}_t) \geq c\|\mathcal{D}_t - \mathcal{D}_{t-1}\|_F^2$ for some constant $c > 0$.

lvbel=**0**, leftmvrgvn=0.6cm (Stability of Estimates) $\|\mathcal{D}_t - \mathcal{D}_{t-1}\|_F = O(w_t)$.

lvibel=**0**, leftmvirgvin=0.6cm (Stability of Errors) For $h_t := \hat{f}_t - f_t \geq 0$, $|h_t(\mathcal{D}_t) - h_{t-1}(\mathcal{D}_{t-1})| = O(w_t)$.

For $n = 1$, it is crucial that $\hat{f}_t$ is convex so that $\mathcal{D}_t$ is a minimizer of $\hat{f}_t$ in the convex constraint set $\mathcal{C}^{\text{dict}}$, as stated in i. From this the monotonicity properties ii and iii follow immediately. The second-order growth property iv requires additional assumption that the surrogates $\hat{f}_t$ are strongly convex uniformly in t. Then iii and iv imply v, which then implies vi. Lastly, i is also crucially used to conclude that every limit point of $(\mathcal{D}_t)_{t\geq 1}$ is a stationary point of

f over $\mathcal{C}^{\text{dict}}$. Now in the multi-modal case $n \geq 2$, we do not have i so all of the implications mentioned above are not guaranteed. Hence the analysis in the multi-modal case requires a significant amount of technical innovation.

Now we state our key lemma that handles the nonconvexity of the surrogate loss $\hat{f}_t$ in the general multi-modal case $n \geq 1$.

Lemma 1 (Key Lemma). *Assume (A1)-(A3). Let $(\mathcal{D}_t)_{t\geq 1}$ be an output of Algorithm 6. For all $t \geq 1$, the following hold:*

(i) *(Forward Monotonicity)* $\hat{f}_t(\mathcal{D}_{t-1}) \geq \hat{f}_t(\mathcal{D}_t)$;

(ii) *(Stability of Estimates)* $\|\mathcal{D}_t - \mathcal{D}_{t-1}\|_F = O(w_t)$;

(iii) *(Stability of Errors)* $|h_t(\mathcal{D}_t) - h_{t-1}(\mathcal{D}_{t-1})| = O(w_t)$, *where* $h_t := \hat{f}_t - f_t$.

(iv) *(Asymptotic Surrogate Stationarity) Let $(t_k)_{k\geq 1}$ be any sequence such that $\mathcal{D}_{t_k}$ and $\hat{f}_{t_k}$ converges almost surely. Then $\mathcal{D}_\infty := \lim_{k\to\infty} \mathcal{D}_{t_k}$ is almost surely a stationary point of $\hat{f}_\infty := \lim_{k\to\infty} \hat{f}_{t_k}$ over $\mathcal{C}^{\text{dict}}$.*

We show Lemma 1 **(i)** using a monotonicity property of block coordinate descent. One of our key observations is that we can directly ensure the stability properties v and vi (Lemma 1 **(ii)** and **(iii)**) by using a search radius restriction (see (4.17) in Algorithm 6). In turn, we do not need the properties iii and iv. In particular, our analysis does not require strong convexity of the surrogate loss $\hat{f}_t$ in each loading matrices as opposed to the existing analysis (see, e.g., [MBPS10, Assumption **B**] and [Mai13b, Def. 2.1]). Lastly, our analysis requires that estimates $\mathcal{D}_t$ are only asymptotically stationary to the limiting surrogate loss function along convergent subsequences, as stated in Lemma 1 **(iv)**. The proof of this statement is nontrivial and requires a substantial work. On a high level, the argument consists of demonstrating that the effect of search radius restriction by $O(w_t)$ vanishes in the limit, and the negative gradient $-\nabla \hat{f}_\infty(\mathcal{D}_\infty)$ is in the normal cone of $\mathcal{C}^{\text{dict}}$ at $\mathcal{D}_\infty$.

The second technical challenge is to handle dependence on input signals, as stated in (A1). The theory of quasi-martingales [Fis65, Rao69] is a key ingredient in convergence analysis under i.i.d input in [MBPS10, Mai13b, APM19]. However, dependent signals do not induce quasi-martingale since conditional on the information $\mathcal{F}_t$ at time t, the following signal $\mathcal{X}_{t+1}$ could be heavily biased. We use the recently developed technique in [LNB20] to overcome this issue of data dependence. The essential fact is that for irreducible and aperiodic Markov chains on finite state space, the N-step conditional distribution converges exponentially in N to the unique stationary distribution regardless of the initial distribution (exponential mixing). The key insight in [LNB20] was that in the analysis, one can condition on "distant past" $\mathcal{F}_{t-\sqrt{t}}$, not on the present $\mathcal{F}_t$, in order to allow the underlying Markov chain to mix close enough to the stationary distribution π for $\sqrt{t}$ iterations. This is opposed to a common practice of thinning Markov chain samples in order to reduce the dependence between sample points we mentioned earlier in Subsection 4.5.1. We provide the estimate based on this technique in Lemma 2.

For the statement of Lemma 2, recall that under (A1), the data tensor at time t is given by $\mathcal{X}_t = \varphi(Y_t)$, where Y_t is an irreducible and aperiodic Markov chain on a finite state space Ω with transition matrix P. For $y, y' \in \Omega$ and $k \in \mathbb{N}$, $P^k(y, y')$ equals the k-step transition probability of Y_t from y to y', and $P^k(y, \cdot)$ equals the distribution of Y_k conditional on $Y_0 = y$. We also use the notation $a^+ = \max(0, a)$ for $a \in \mathbb{R}$.

Lemma 2 (Convergence of Positive Variation)**.** *Let $(\mathcal{D}_t)_{t\geq 1}$ be an output of Algorithm 6. Suppose (A1)', (A2), and (A3) hold.*

(i) *Let $(a_t)_{t\geq 0}$ be a sequence of non-decreasing non-negative integers such that $a_t = o(t)$. Then there exists absolute constants $C_1, C_2, C_3 > 0$ such that for all sufficiently large $t \geq 0$,*

$$\mathbb{E}\left[\mathbb{E}\left[w_{t+1}\big(\ell(\mathcal{X}_{t+1}, \mathcal{D}_t) - f_t(\mathcal{D}_t)\big)\,\middle|\,\mathcal{F}_{t-a_t}\right]^+\right] \tag{4.19}$$

$$\leq C_1 w_{t-a_t}^2 \sqrt{t} + C_2 w_t^2 a_t + C_3 w_t \sup_{\mathbf{y}\in\Omega} \|P^{a_t+1}(\mathbf{y}, \cdot) - \pi\|_{TV}. \tag{4.20}$$

(ii) $\displaystyle\sum_{t=0}^{\infty}\left(\mathbb{E}\left[\hat{f}_{t+1}(\mathcal{D}_{t+1}) - \hat{f}_t(\mathcal{D}_t)\right]\right)^+ \leq \sum_{t=0}^{\infty} w_{t+1}\left(\mathbb{E}\left[(\ell(\mathcal{X}_{t+1}, \mathcal{D}_t) - f_t(\mathcal{D}_t))\right]\right)^+ < \infty.$

We give some remarks on the statement of Lemma 2. According to Proposition 3 in Section 4.6, one of the main quantities we would like to bound is $\mathbb{E}[\ell(\mathcal{X}_{t+1}, \mathcal{D}_t) - \hat{f}_t(\mathcal{D})]^+$, which is the expected positive variation of the one-step difference between the one-point error $\ell(\mathcal{X}_{t+1}, \mathcal{D}_t)$ of factorizing the new data $\mathcal{X}_{t+1}$ using the current dictionary $\mathcal{D}_t$ and the empirical error $\hat{f}_t(\mathcal{D}_t)$. According to the recursive update of the empirical and surrogate losses in (4.11) and (4.16), we want the weighted sum of such expected positive variations in Lemma 2 **(ii)** is finite. This follows from the bound in Lemma 2 **(i)**, as long as $\sum_{t=1}^{\infty} w_t^2 \sqrt{t} < \infty$, $a_t = O(\sqrt{t})$, and the Markov chain Y_t modulating the data tensor $\mathcal{X}_t$ mixes fast enough (see (A1)). Such conditions are satisfied from the assumptions (A1) and (A3).

4.6 Proof of the main result

In this section, we prove our main convergence result, Theorem 1. Throughout this section, we assume the code matrices H_t and loading matrices $U_t^{(i)}$ belong to convex and compact constraint sets $H_t \in \mathcal{C}^{\text{code}} \subseteq \mathbb{R}^{R\times b}$, $U_t^{(i)} \in \mathcal{C}^{(i)} \subseteq \mathbb{R}^{I_i\times R}$ as in (A2) and denote $\mathcal{C}^{\text{dict}} = \mathcal{C}^{(1)} \times \cdots \times \mathcal{C}^{(n)} \subseteq \mathbb{R}^{I_1\times R} \times \cdots \times \mathbb{R}^{I_n\times R}$.

4.6.1 Deterministic analysis

In this subsection, we provide some deterministic analysis of our online algorithm (Algorithm 6), which will be used in the forthcoming stochastic analysis.

First, we derive a parameterized form of Algorithm 6, where the surrogate loss function $\hat{f}_t$ is replaced with $\hat{g}_t$, which is a block-wise quadradtic function with recursively updating parameters. This will be critical in our proof of Lemma 1 **(iv)** as well as deriving the bounded-memory implementation of Algorithm 6 stated in Algorithm 7 in Appendix 4.11.

Consider the following block optimization problem

$$\text{Upon arrival of } \mathcal{X}_t: \quad \begin{cases} H_t = \arg\min_{H \in \mathcal{C}^{\text{code}} \subseteq \mathbb{R}^{R\times b}} \ \ell(\mathcal{X}_t, \mathcal{D}_{t-1}, H) \\ A_t = (1-w_t)A_{t-1} + w_t H_t H_t^T \\ \mathbf{B}_t = (1-w_t)\mathbf{B}_{t-1} + w_t(\mathcal{X}_t \times_{n+1} H_t^T) \\ \mathcal{D}_t = \underset{\substack{\mathcal{D}=[U^{(1)},\dots,U^{(n)}]\in\mathcal{C}^{\text{dict}} \\ \|U^{(i)}-U^{(i)}_{t-1}\|_F \le c' w_t \ \forall i}}{\arg\min} \ \hat{g}_t(\mathcal{D}) \end{cases}, \tag{4.21}$$

where for each $\mathcal{D} = [U_1, \dots, U_n] \in \mathcal{C}^{\text{dict}}$ (here we use subscripts to denote modes for taking their transpose) and $\hat{g}_t$ in (4.21) is defined as

$$\hat{g}_t(\mathcal{D}) := \operatorname{tr}(A_t\,(U_n^T U_n \odot \dots \odot U_1^T U_1)) - 2\operatorname{tr}\left(\mathbf{B}_t^{(n+1)}(U_n \otimes_{kr} \dots \otimes_{kr} U_1)^T\right), \tag{4.22}$$

where $\mathbf{B}_t^{(n+1)} \in \mathbb{R}^{I_1\cdots I_n \times R}$ denotes the mode-$(n+1)$ unfolding of $\mathbf{B}_t \in \mathbb{R}^{I_1\times\dots\times I_n\times R}$, and $A_0 \in \mathbb{R}^{R\times R}$ and $\mathbf{B}_0 \in \mathbb{R}^{I_1\times\dots\times I_n\times R}$ are tensors of all zero entries. The following proposition shows that minimizing $\hat{f}_t$ in (4.16) is equivalent to minimizing $\hat{g}_t$ in (4.21). This shows that $\hat{f}_t$, which requires the storage of all past tensors $\mathcal{X}_1, \dots, \mathcal{X}_t$ for its definition, can in fact be parameterized by an aggregate matrix $A_t \in \mathbb{R}^{R\times R}$ and an aggregate tensor $\mathbf{B}_t \in \mathbb{R}^{I_1\times\dots\times I_n\times R}$, whose dimensions are independent of t. This implies that Algorithm 6 can be implemented using a bounded memory, without needing to store a growing number of full-dimensional data tensors $\mathcal{X}_1, \dots, \mathcal{X}_t$. See Appendix 4.11 for more details.

Proposition 2. *The following hold:*

(i) *Let $\hat{f}_t$ be as in (4.16) and $\hat{g}_t$ be in (4.22). Then*

$$\hat{f}_t(\mathcal{D}) = \hat{g}_t(\mathcal{D}) + \sum_{s=1}^{t} \operatorname{tr}\left(\operatorname{MAT}(\mathcal{X}_s)\operatorname{MAT}(\mathcal{X}_s)^T\right) + \lambda \sum_{s=1}^{t} \|H_s\|_1, \tag{4.23}$$

(ii) *For each $1 \le j \le n$ and $t \ge 1$, let $\overline{A}_{t;j} \in \mathbb{R}^{R\times R}$, $\overline{B}_{t;j} \in \mathbb{R}^{I_j\times R}$ be the output of Algorithm 8 with input $A_t, \mathbf{B}_t, U_1, \dots, U_n$, and j. Then can rewrite $\hat{g}_t(\mathcal{D}) = \hat{g}_t(U_1, \dots, U_n)$ in (4.22) as*

$$\hat{g}_t(U_1, \dots, U_n) = \operatorname{tr}\left(U_i \overline{A}_{t;j} U_i^T\right) - 2\operatorname{tr}\left(U_i \overline{B}_{t;j}^T\right). \tag{4.24}$$

Proof. Let $\mathrm{MAT}(\mathcal{X}_s) = [\mathrm{vec}(\mathcal{X}_{s;1}), \dots, \mathrm{vec}(\mathcal{X}_{s;b})] \in \mathbb{R}^{(I_1 \dots I_n) \times b}$ denote the matrix whose i^{th} column is the vectorization $\mathrm{vec}(\mathcal{X}_{s;j})$ of the tensor $\mathcal{X}_{s;j} \in \mathbb{R}^{I_1 \times \dots \times I_n}$. The first assertion follows easily from observing that, for each $[U_1, \dots, U_n] \in \mathcal{C}^{\text{dict}}$ and $H \in \mathbb{R}^{R \times b}$,

$$
\begin{aligned}
&\|\mathcal{X}_s - \mathtt{Out}(U_1, \dots, U_n) \times_{n+1} H\|_F^2 \\
&\quad = \|\mathrm{MAT}(\mathcal{X}_s) - (U_n \otimes_{kr} \dots \otimes_{kr} U_1) H\|_F^2 \\
&\quad = \mathrm{tr}\left((U_n \otimes_{kr} \dots \otimes_{kr} U_1) H H^T (U_n \otimes_{kr} \dots \otimes_{kr} U_1)^T\right) \\
&\qquad\quad - 2\,\mathrm{tr}\left(\mathrm{MAT}(\mathcal{X}_s) H^T (U_n \otimes_{kr} \dots \otimes_{kr} U_1)^T\right) + \mathrm{tr}\left(\mathrm{MAT}(\mathcal{X}_s)\,\mathrm{MAT}(\mathcal{X}_s)^T\right),
\end{aligned}
$$

and also note that

$$
\begin{aligned}
&\mathrm{tr}\left((U_n \otimes_{kr} \dots \otimes_{kr} U_1) H H^T (U_n \otimes_{kr} \dots \otimes_{kr} U_1)^T\right) \\
&\quad = \mathrm{tr}(H H^T (U_n \otimes_{kr} \dots \otimes_{kr} U_1)^T (U_n \otimes_{kr} \dots \otimes_{kr} U_1)) \\
&\quad = \mathrm{tr}(H H^T\, (U_n^T U_n \odot \dots \odot U_1^T U_1)).
\end{aligned}
$$

Then the linearity of trace show

$$
\begin{aligned}
\hat{f}_t(U_1, \dots, U_n) = {} & \mathrm{tr}(A_t\, (U_n^T U_n \odot \dots \odot U_1^T U_1)) - 2\,\mathrm{tr}\left(\widetilde{\mathbf{B}}_t (U_n \otimes_{kr} \dots \otimes_{kr} U_1)^T\right) \\
& + \sum_{s=1}^{t} \mathrm{tr}\left(\mathrm{MAT}(\mathcal{X}_s)\,\mathrm{MAT}(\mathcal{X}_s)^T\right) + \lambda \sum_{s=1}^{t} \|H_s\|_1,
\end{aligned}
\tag{4.25}
$$

where A_t is recursively defined in (4.21) and $\widetilde{\mathbf{B}}_t \in \mathbb{R}^{(I_1 \times \dots \times I_n) \times b}$ is defined recursively by

$$
\widetilde{\mathbf{B}}_s = (1 - w_t)\widetilde{\mathbf{B}}_{s-1} + w_t\,\mathrm{MAT}(\mathcal{X}_s) H_s^T.
$$

By a simple induction argument, one can show that $\widetilde{B}_t$ equals the mode-$(n+1)$ unfolding $\mathbf{B}_t^{(n+1)}$ of $\mathbf{B}_t$ defined recursively in (4.21), as desired.

For **(ii)**, first note that

$$
\begin{aligned}
&\mathrm{tr}(A\, (U_n^T U_n \odot \dots \odot U_1^T U_1)) \\
&\quad = \mathrm{tr}((A \odot U_1^T U_1 \odot \dots U_{i-1}^T U_{i-1} \odot U_{i+1}^T U_{i+1} \odot \dots \odot U_n^T U_n)\, U_j^T U_j) \\
&\quad = \mathrm{tr}(U_j\, (A \odot U_1^T U_1 \odot \dots U_{i-1}^T U_{i-1} \odot U_{i+1}^T U_{i+1} \odot \dots \odot U_n^T U_n)\, U_j^T)
\end{aligned}
$$

$$= \operatorname{tr}(U_j \,\overline{A}_{t;j}\, U_j^T).$$

Also, recall that $\mathbf{B}_t^{(n+1)}$ and $U_n \otimes_{kr} \dots \otimes_{kr} U_1$ are $(\prod_{i=1}^n I_j) \times R$ matrices. Let $\mathbf{B}_t(,r) \in \mathbb{R}^{I_1\times\cdots\times I_n}$ denote the r^{th} mode-$(n+1)$ slice of $\mathbf{B}_t$. We note that

$$\begin{aligned}
&\operatorname{tr}\left(\mathbf{B}_t^{(n+1)}(U_n \otimes_{kr} \dots \otimes_{kr} U_1)^T\right)\\
&= \sum_{r=1}^{R} \operatorname{tr}\left(\mathbf{B}_t(,r) \times_1 U_1(:,r) \times_2 \cdots \times_{i-1} U_{i-1}(:,r) \times_i U_i(:,r) \times_{i+1} U_{i+1}(:,r) \times_{i+2} \cdots \times_n U_n(:,r)\right)\\
&= \operatorname{tr}\left(\sum_{r=1}^{R} [\mathbf{B}_t(,r) \times_1 U_1(:,r) \times_2 \cdots \times_{i-1} U_{i-1}(:,r) \times_{i+1} U_{i+1}(:,r) \times_{i+2} \cdots \times_n U_n(:,r)]\, U_i(:,r)^T\right)\\
&= \operatorname{tr}\left(U_i \overline{B}_{t;j}^T\right),
\end{aligned}$$

where $\overline{B}_{t;j}^T$ is as in the assertion. Then the assertion follows. ☐

Proof. **of Lemma 1 (i)-(iii).** First, we show **(i)**. Write $\mathcal{D}_{t-1} = [U_1, \dots, U_n]$ and $\mathcal{D}_t = [U_1', \dots, U_n']$ (here we use subscripts to denote modes). Using Proposition 2 **(i)**, we write

$$\hat{f}_t(\mathcal{D}_{t-1}) - \hat{f}_t(\mathcal{D}_t) \tag{4.26}$$
$$= \hat{f}_t([U_1, \dots, U_n]) - \hat{f}_t([U_1', \dots, U_n']) \tag{4.27}$$
$$= \sum_{i=1}^{n} \hat{f}_t([U_1', \dots, U_{i-1}', U_i, U_{i+1}, \dots, U_n]) - \hat{f}_t([U_1', \dots, U_{i-1}', U_i', U_{i+1}, \dots, U_n]). \tag{4.28}$$

Recall that U_i' is a minimizer of the function $U \mapsto \hat{f}_t([U_1', \dots, U_{i-1}', U, U_{i+1}, \dots, U_n])$ (which is convex by Proposition 2) over the convex set $\mathcal{C}_i$ defined in Algorithm 6. Also, U_i' belongs to $\mathcal{C}_i$. Hence each summand in the last expression above is nonnegative. This shows $\hat{f}_t(\mathcal{D}_{t-1}) - \hat{f}_t(\mathcal{D}_t) \geq 0$, as desired. Also note that **(ii)** is trivial by the search radius restriction in Algorithm 6.

Lastly, we show **(iii)**. Both $\hat{f}_t$ and f_t are uniformly bounded and Lipschitz by Lemma 6 in Appendix 4.10. Hence $h_t = \hat{f}_t - f_t$ is also Lipschitz with some constant $C > 0$ independent of t. Then by the recursive definitions of $\hat{f}_t$ and f_t (see (4.16) and (4.11)) and noting that $\ell(\mathcal{X}_t, \mathcal{D}_{t-1}, H_t) = \ell(\mathcal{X}_t, \mathcal{D}_{t-1})$, we have

$$|h_t(\mathcal{D}_t) - h_{t-1}(\mathcal{D}_{t-1})| \tag{4.29}$$

$$
\begin{aligned}
&\leq |h_t(\mathcal{D}_t) - h_t(\mathcal{D}_{t-1})| + |h_t(\mathcal{D}_{t-1}) - h_{t-1}(\mathcal{D}_{t-1})| && (4.30)\\
&\leq C\|\mathcal{D}_t - \mathcal{D}_{t-1}\|_F + \left|\left(\hat{f}_t(\mathcal{D}_{t-1}) - \hat{f}_{t-1}(\mathcal{D}_{t-1})\right) - (f_t(\mathcal{D}_{t-1}) - f_{t-1}(\mathcal{D}_{t-1}))\right| && (4.31)\\
&= C\|\mathcal{D}_t - \mathcal{D}_{t-1}\|_F + w_t|\hat{f}_{t-1}(\mathcal{D}_{t-1}) - f_{t-1}(\mathcal{D}_{t-1})|. && (4.32)
\end{aligned}
$$

Hence this and **(ii)** show $|h_t(\mathcal{D}_t) - h_{t-1}(\mathcal{D}_{t-1})| = O(w_t)$, as desired. □

Next, we establish two elementary yet important inequalities connecting the empirical and surrogate loss functions. This is trivial in the case of vector-valued signals, in which case we can directly minimize $\hat{f}_t$ over a convex constraint set $\mathcal{C}^{\text{dict}}$ to find $\mathcal{D}_t$ so we have the 'forward monotonicity' $\hat{f}_t(\mathcal{D}_t) \leq \hat{f}_t(\mathcal{D}_{t-1})$ immediately from the algorithm design. In the tensor case, this still holds since we use block coordinate descent to progressively minimize $\hat{f}_t$ in each loading matrix.

Proposition 3. *Let $(\mathcal{D}_t)_{t\geq 1}$ be an output of Algorithm 6. Then for each $t \geq 0$, the following hold:*

(i) $\hat{f}_{t+1}(\mathcal{D}_{t+1}) - \hat{f}_t(\mathcal{D}_t) \leq w_{t+1}\left(\ell(\mathcal{X}_{t+1}, \mathcal{D}_t) - f_t(\mathcal{D}_t)\right)$.

(ii) $0 \leq w_{t+1}\left(\hat{f}_t(\mathcal{D}_t) - f_t(\mathcal{D}_t)\right) \leq w_{t+1}\left(\ell(\mathcal{X}_{t+1}, \mathcal{D}_t) - f_t(\mathcal{D}_t)\right) + \hat{f}_t(\mathcal{D}_t) - \hat{f}_{t+1}(\mathcal{D}_{t+1})$.

Proof. We begin by observing that

$$
\begin{aligned}
\hat{f}_{t+1}(\mathcal{D}_t) &= (1 - w_{t+1})\hat{f}_t(\mathcal{D}_t) + w_{t+1}\ell_{t+1}(\mathcal{X}_{t+1}, \mathcal{D}_t, H_{t+1}) && (4.33)\\
&= (1 - w_{t+1})\hat{f}_t(\mathcal{D}_t) + w_{t+1}\ell_{t+1}(\mathcal{X}_{t+1}, \mathcal{D}_t) && (4.34)
\end{aligned}
$$

for all $t \geq 0$. The first equality above uses the definition of $\hat{f}_t$ in (4.16) and the second equality uses the fact that H_{t+1} is a minimizer of $\ell(\mathcal{X}_{t+1}, \mathcal{D}_t, H)$ over $\mathcal{C}^{\text{code}}$. Hence

$$
\begin{aligned}
&\hat{f}_{t+1}(\mathcal{D}_{t+1}) - \hat{f}_t(\mathcal{D}_t) && (4.35)\\
&\quad= \hat{f}_{t+1}(\mathcal{D}_{t+1}) - \hat{f}_{t+1}(\mathcal{D}_t) + \hat{f}_{t+1}(\mathcal{D}_t) - \hat{f}_t(\mathcal{D}_t)\\
&\quad= \hat{f}_{t+1}(\mathcal{D}_{t+1}) - \hat{f}_{t+1}(\mathcal{D}_t) + (1 - w_{t+1})\hat{f}_t(\mathcal{D}_t) + w_{t+1}\ell(\mathcal{X}_{t+1}, \mathcal{D}_t) - \hat{f}_t(\mathcal{D}_t)\\
&\quad= \hat{f}_{t+1}(\mathcal{D}_{t+1}) - \hat{f}_{t+1}(\mathcal{D}_t) + w_{t+1}(\ell(\mathcal{X}_{t+1}, \mathcal{D}_t) - f_t(\mathcal{D}_t)) + w_{t+1}(f_t(\mathcal{D}_t) - \hat{f}_t(\mathcal{D}_t)).
\end{aligned}
$$

Now note that $f_t \leq \hat{f}_t$ by definition. Furthermore, $\hat{f}_{t+1}(\mathcal{D}_{t+1}) - \hat{f}_{t+1}(\mathcal{D}_t) \leq 0$ by Lemma 1 **(i)**, so the inequalities in both **(i)** and **(ii)** follow. $\square$

4.6.2 Stochastic analysis

In this subsection, we develop stochastic analysis on our online algorithm, a major portion of which is devoted to handling Markovian dependence in signals as stated in assumption (A1). The analysis here is verbatim as the one developed in [LNB20] for the vector-valued signal (or matrix factorization) case, which we present some of the important arguments in detail here for the sake of completeness. However, the results in this subsection crucially rely on the deterministic analysis in the previous section that was necessary to handle difficulties arising in the tensor-valued signal case.

Recall that under our assumption (A1), the signals $(\mathcal{X}_t)_{t\geq 0}$ are given as $\mathcal{X}_t = \varphi(Y_t)$ for a fixed function φ and a Markov chain $(Y_t)_{t\geq 0}$. Note that Proposition 3 gives a bound on the change in surrogate loss $\hat{f}_t(\mathcal{D}_t)$ in one iteration, which allows us to control its *positive variation* in terms of difference $\ell(\mathcal{X}_{t+1}, \mathcal{D}_t) - f_t(\mathcal{D}_t)$. The core of the stochastic analysis in this subsection is to show that $w_{t+1}\mathbb{E}[\ell(\mathcal{X}_{t+1}, \mathcal{D}_t) - f_t(\mathcal{D}_t)]^+$ is summable. In the classical setting when Y_t's are i.i.d., our signals $\mathcal{X}_t = \varphi(Y_t)$ are also i.i.d., so we can condition on the information $\mathcal{F}_t$ up to time t so that

$$\mathbb{E}\left[\ell(\mathcal{X}_{t+1}, \mathcal{D}_t) - f_t(\mathcal{D}_t) \,\middle|\, \mathcal{F}_t\right] = f(\mathcal{D}_t) - f_t(\mathcal{D}_t). \tag{4.36}$$

Note that for each fixed $\mathcal{D} \in \mathcal{C}^{\text{dict}}$, $f_t(\mathcal{D}) \to f(\mathcal{D})$ almost surely as $t \to \infty$ by the strong law of large numbers. To handle time dependence of the evolving dictionaries $\mathcal{D}_t$, one can instead look that the convergence of the supremum $\|f_t - f\|_\infty$ over the compact set $\mathcal{C}^{\text{dict}}$, which is provided by the classical Glivenko-Cantelli theorem. This is the approach taken in [MBPS10, Mai13b] for i.i.d. input.

However, the same approach is not applicable for dependent signals, for instance, when $(Y_t)_{t\geq 0}$ is a Markov chain. This is because, in this case, conditional on $\mathcal{F}_t$, the distribution of Y_{t+1} is not necessarily the stationary distribution π. In fact, when Y_t's form a Markov chain with transition matrix P, Y_t given Y_{t-1} has distribution $P(Y_{t-1}, \cdot)$, and this conditional distribution is a constant distance away from the stationary distribution π. (For instance, consider the case when Y_t takes two values and it differs from Y_{t-1} with probability $1-\varepsilon$. Then $\pi = [1/2, 1/2]$ and the distribution of Y_t converges exponentially fast to π, but $P(Y_{t-1}, \cdot)$ is either $[1-\varepsilon, \varepsilon]$ or $[\varepsilon, 1-\varepsilon]$ for all $t \geq 1$.)

To handle dependence in data samples, we adopt the strategy developed in [LNB20] in order to handle a similar issue for vector-valued signals (or matrix factorization). The key insight in [LNB20] is that, while the 1-step conditional distribution $P(X_{t-1}, \cdot)$ may be far from the stationary distribution π, the N-step conditional distribution $P^N(X_{t-N}, \cdot)$ is exponentially close to π under mild conditions. Hence we can condition much early on – at time $t-N$ for some suitable $N = N(t)$. Then the Markov chain runs $N+1$ steps up to time $t+1$, so if N is large enough for the chain to mix to its stationary distribution π, then the distribution of Y_{t+1} conditional on $\mathcal{F}_{t-N}$ is close to π. The error of approximating the stationary distribution by the $N+1$ step distribution can be controlled using total variation distance and Markov chain mixing bound. This is stated more precisely in the proposition below.

Proposition 4. *Suppose (A1) hold. Fix a CP-dictionary $\mathcal{D}$. Then for each $t \geq 0$ and $0 \leq N < t$, conditional on the information $\mathcal{F}_{t-N}$ up to time $t-N$,*

$$\left(\mathbb{E}\left[\ell(\mathcal{X}_{t+1}, \mathcal{D}) - f_t(\mathcal{D}) \,\middle|\, \mathcal{F}_{t-N}\right]\right)^+ \leq |f(\mathcal{D}) - f_{t-N}(\mathcal{D})| + N w_t f_{t-N}(\mathcal{D}) \tag{4.37}$$

$$+ 2\|\ell(\cdot, \mathcal{D})\|_\infty \sup_{\mathbf{y}\in\Omega} \|P^{N+1}(\mathbf{y}, \cdot) - \pi\|_{TV}. \tag{4.38}$$

Proof. The proof is identical to that of [LNB20, Prop. 7.5]. □

Proof. **of Lemma 2**. Part **(i)** can be derived from Proposition 4 and Lemma 9 in Appendix 4.10. See the proof of [LNB20, Prop. 7.8 **(i)**] for details. Next, part **(ii)** can be derived from part **(i)** with Proposition 3 **(i)**. See the proof of [LNB20, Prop. 7.8 **(ii)**] for details. □

Lemma 3. *Let $(\mathcal{D}_t)_{t\geq 1}$ be the output of Algorithm 6. Suppose (A1)-(A3) hold. Then the following hold.*

(i) $\displaystyle\sum_{t=0}^{\infty} \mathbb{E}\left[w_{t+1}\left(\ell(\mathcal{X}_{t+1},\mathcal{D}_t) - f_t(\mathcal{D}_t)\right)\right]^+ < \infty$*;*

(ii) $\mathbb{E}[\hat{f}_t(\mathcal{D}_t)]$ *converges as* $t \to \infty$*;*

(iii) $\displaystyle\mathbb{E}\left[\sum_{t=0}^{\infty} w_{t+1}\left(\hat{f}_t(\mathcal{D}_t) - f_t(\mathcal{D}_t)\right)\right] = \sum_{t=0}^{\infty} w_{t+1}\left(\mathbb{E}[\hat{f}_t(\mathcal{D}_t)] - \mathbb{E}[f_t(\mathcal{D}_t)]\right) < \infty$*;*

(iv) $\displaystyle\sum_{t=0}^{\infty} w_{t+1}\left(\hat{f}_t(\mathcal{D}_t) - f_t(\mathcal{D}_t)\right) < \infty$ *almost surely.*

Proof. Part **(i)** can be derived from Proposition 4 and Jensen's inequality. See the proof of [LNB20, Lem. 12 **(ii)**] for details. Parts **(ii)**-**(iv)** can be shown by using Propositions 3, 4, and part **(i)**. See the proof of [LNB20, Lem. 13] for details. □

4.6.3 Asymptotic surrogate stationarity

In this subsection, we prove Lemma 1 **(iv)**, which requires one of the most nontrivial arguments we give in this work. Throughout this subsection, we will denote by $(\mathcal{D}_t)_{t\ge1}$ the output of Algorithm 6 and $\Lambda := \{\mathcal{D}_t \,|\, t \ge 1\} \subseteq \mathcal{C}^{\text{dict}}$. Note that by Proposition 2, $\hat{f}_{t_k}$ converges almost surely if and only if $A_{t_k}, \mathbf{B}_{t_k}, \mathcal{X}_{t_k}, H_{t_k}$ converge a.s. as $k \to \infty$. In what follows, we say $\mathcal{D}_\infty \in \mathcal{C}^{\text{dict}}$ a *stationary point* of Λ if it is a limit point $\mathcal{D}_\infty$ of Λ and there exists a sequence $t_k \to \infty$ such that $\mathcal{D}_{t_k} \to \mathcal{D}_\infty$ and $\hat{f}_\infty := \lim_{k\to\infty} \hat{f}_{t_k}$ exists almost surely and $\mathcal{D}_\infty$ is a stationary point of $\hat{f}_\infty$ over $\mathcal{C}^{\text{dict}}$. Our goal is to show that every limit point of Λ is stationary.

The following observation is key to our argument.

Proposition 5. *Assume (A1)-(A3) hold. Let* $(\mathcal{D}_t)_{t\ge1}$ *be an output of Algorithm 6. Then almost surely,*

$$\sum_{t=1}^{\infty} \left| \left(\nabla \hat{f}_{t+1}(\mathcal{D}_{t+1})^T (\mathcal{D}_t - \mathcal{D}_{t+1})\right) \right| < \infty.$$

Proof. Since $\mathcal{C}^{\text{dict}}$ is compact by (A2) and the aggregate tensors $A_t, \mathbf{B}_t$ are uniformly bounded by Lemma 5 in Appendix 4.10, we can see from Proposition 2 that $\nabla \hat{f}_{t+1}$ over $\mathcal{C}^{\text{dict}}$ is Lipschitz with some uniform constant $L > 0$. Hence by Lemma 4 in Appendix 4.10, for all $t \geq 1$,

$$\left| \hat{f}_{t+1}(\mathcal{D}_t) - \hat{f}_{t+1}(\mathcal{D}_{t+1}) - \operatorname{tr}\left(\nabla \hat{f}_{t+1}(\mathcal{D}_{t+1})^T (\mathcal{D}_t - \mathcal{D}_{t+1}) \right) \right| \leq \frac{L}{2} \|\mathcal{D}_t - \mathcal{D}_{t+1}\|_F^2.$$

Also note that $\hat{f}_{t+1}(\mathcal{D}_t) \geq \hat{f}_{t+1}(\mathcal{D}_{t+1})$ by Lemma 1 **(i)**. Hence it follows that

$$\left| \operatorname{tr}\left(\nabla \hat{f}_{t+1}(\mathcal{D}_{t+1})^T (\mathcal{D}_t - \mathcal{D}_{t+1}) \right) \right| \leq \frac{L}{2} \|\mathcal{D}_t - \mathcal{D}_{t+1}\|_F^2 + \hat{f}_{t+1}(\mathcal{D}_t) - \hat{f}_{t+1}(\mathcal{D}_{t+1}) \tag{4.39}$$

On the other hand, (4.35) and $\hat{f}_t \geq f_t$ yields

$$0 \leq \hat{f}_{t+1}(\mathcal{D}_t) - \hat{f}_{t+1}(\mathcal{D}_{t+1}) \leq \hat{f}_t(\mathcal{D}_t) - \hat{f}_{t+1}(\mathcal{D}_{t+1}) + w_{t+1}(\ell(\mathcal{X}_{t+1}, \mathcal{D}_t) - f_t(\mathcal{D}_t)).$$

Hence using Lemma 3, we have

$$\sum_{t=1}^{\infty} \mathbb{E}\left[\hat{f}_{t+1}(\mathcal{D}_t) - \hat{f}_{t+1}(\mathcal{D}_{t+1}) \right] < \infty.$$

Then from (4.39) and noting that $\|\mathcal{D}_t - \mathcal{D}_{t+1}\|_F^2 = O(w_{t+1}^2)$ and $\sum_{t=1}^{\infty} w_t^2 < \infty$ (see (A3)), it follows that

$$\begin{aligned}\sum_{t=1}^{\infty} \mathbb{E}\left[\left| \operatorname{tr}\left(\nabla \hat{f}_{t+1}(\mathcal{D}_{t+1})^T (\mathcal{D}_t - \mathcal{D}_{t+1}) \right) \right| \right] &= \frac{L}{2} \sum_{t=1}^{\infty} \mathbb{E}\left[\|\mathcal{D}_t - \mathcal{D}_{t+1}\|_F^2 \right] \\ &\quad + \sum_{t=1}^{\infty} \mathbb{E}\left[\hat{f}_{t+1}(\mathcal{D}_t) - \hat{f}_{t+1}(\mathcal{D}_{t+1}) \right] < \infty.\end{aligned}$$

Then the assertion follows by Fubini's theorem and the fact that $\mathbb{E}[|X|] < \infty$ implies $|X| < \infty$ almost surely for any random variable X, where $|\cdot|$ denotes the largest absolute value among the entries of X. □

Next, we show that the block coordinate descent we use to obtain $\mathcal{D}_{t+1}$ should always give the optimal first-order descent up to a small additive error.

Proposition 6 (Asymptotic first-order optimality)**.** *Assume (A1)-(A3) and $w_t = o(1)$. Then there exists a constant $c_1 > 0$ such that for all $t \geq 1$,*

$$\operatorname{tr}\left(\nabla \hat{f}_{t+1}(\mathcal{D}_{t+1})^T \frac{(\mathcal{D}_{t+1} - \mathcal{D}_t)}{\|\mathcal{D}_{t+1} - \mathcal{D}_t\|_F} \right) \leq \inf_{\mathcal{D} \in \mathcal{C}^{\text{dict}}} \operatorname{tr}\left(\nabla \hat{f}_{t+1}(\mathcal{D}_t)^T \frac{(\mathcal{D} - \mathcal{D}_t)}{\|\mathcal{D} - \mathcal{D}_t\|_F} \right) \tag{4.40}$$

$$+ c_1 \|\mathcal{D}_{t+1} - \mathcal{D}_t\|_F^2. \tag{4.41}$$

Proof. Fix a sequence $(b_t)_{t\geq 1}$ such that $0 < b_t \leq c' w_t$ for all $t \geq 1$. Write $\mathcal{D}_t = [U_t^{(1)}, \ldots, U_t^{(n)}]$ for $t \geq 1$ and denote

$$\hat{f}_{t+1;i} : U \mapsto \hat{f}_{t+1}(U_{t+1}^{(1)}, \ldots, U_{t+1}^{(i-1)}, U, U_t^{(i+1)}, \ldots, U_t^{(n)}) \tag{4.42}$$

for $U \in \mathbb{R}^{I_i}$ and $i = 1, \ldots, n$. Recall that $U_{t+1}^{(i)}$ is a minimizer of $\hat{f}_{t+1;i}$ over the convex set $\mathcal{C}_{t+1}^{(i)}$ defined in (4.17). Fix arbitrary $\mathcal{D} = [U^{(1)}, \ldots, U^{(n)}] \in \mathcal{C}^{\text{dict}}$ such that $\|\mathcal{D} - \mathcal{D}_t\|_F \leq b_{t+1}$. Then $\|U^{(i)} - U_t^{(i)}\|_F \leq b_{t+1}$ for all $1 \leq i \leq n$. By convexity of $\mathcal{C}^{(i)}$, note that for each $U^{(i)} \in \mathcal{C}^{(i)}$, $U_t^{(i)} + a(U^{(i)} - U_t^{(i)}) \in \mathcal{C}^{(i)}$ for all $a \in [0, 1]$. Then by the definition of $\mathcal{C}_{t+1}^{(i)}$ and the choice of $U_{t+1}^{(i)}$, we have that for all $t \geq 1$,

$$\hat{f}_{t+1;i}(U_{t+1}^{(i)}) - \hat{f}_{t+1;i}(U_t^{(i)}) \leq \hat{f}_{t+1;i}\left(U_t^{(i)} + a(U^{(i)} - U_t^{(i)})\right) - \hat{f}_{t+1;i}(U_t^{(i)}). \tag{4.43}$$

Recall that $\nabla \hat{f} = [\nabla \hat{f}_{t+1;1}, \ldots, \nabla \hat{f}_{t+1;n}]$ is Lipschitz with uniform Lipschitz constant $L > 0$. Hence by Lemma 4 in Appendix 4.10, there exists a constant $c_1 > 0$ such that for all $t \geq 1$,

$$\operatorname{tr}\left(\nabla \hat{f}_{t+1;i}(U_t^{(i)})^T (U_{t+1}^{(i)} - U_t^{(i)})\right) - \frac{L}{2}\|U_{t+1}^{(i)} - U_t^{(i)}\|^2 \tag{4.44}$$

$$\leq a \operatorname{tr}\left(\nabla \hat{f}_{t+1;i}(U_t^{(i)})^T (U^{(i)} - U_t^{(i)})\right) + \frac{La^2 \|U^{(i)} - U_t^{(i)}\|}{2}. \tag{4.45}$$

Adding up these inequalities for $i = 1, \ldots, n$, we get

$$\operatorname{tr}\left(\left[\nabla \hat{f}_{t+1;1}(U_t^{(1)}), \ldots, \nabla \hat{f}_{t+1;n}(U_t^{(n)})\right]^T (\mathcal{D}_{t+1} - \mathcal{D}_t)\right) \tag{4.46}$$

$$\leq a \operatorname{tr}\left(\left[\nabla \hat{f}_{t+1;1}(U_t^{(1)}), \ldots, \nabla \hat{f}_{t+1;n}(U_t^{(n)})\right]^T (\mathcal{D} - \mathcal{D}_t)\right) \tag{4.47}$$

$$+ \frac{L}{2}\|\mathcal{D}_{t+1} + \mathcal{D}_t\|_F^2 + \frac{La^2}{2}\|\mathcal{D} - \mathcal{D}_t\|_F^2. \tag{4.48}$$

Since $\nabla \hat{f}_{t+1}$ is L-Lipschitz, using Cauchy-Schwarz inequality,

$$\operatorname{tr}\left(\nabla \hat{f}_{t+1}(\mathcal{D}_{t+1})^T (\mathcal{D}_{t+1} - \mathcal{D}_t)\right) \tag{4.49}$$

$$\leq a \operatorname{tr}\left(\nabla \hat{f}_{t+1}(\mathcal{D}_t)^T (\mathcal{D} - \mathcal{D}_t)\right) + a\|\mathcal{D}_{t+1} - \mathcal{D}_t\|_F \|\mathcal{D} - \mathcal{D}_t\|_F \tag{4.50}$$

$$+ \frac{3L}{2}\|\mathcal{D}_{t+1} - \mathcal{D}_t\|_F^2 + \frac{La^2}{2}\|\mathcal{D} - \mathcal{D}_t\|_F^2 \tag{4.51}$$

$$\leq a \operatorname{tr}\left(\nabla \hat{f}_{t+1}(\mathcal{D}_t)^T (\mathcal{D} - \mathcal{D}_t)\right) + a\|\mathcal{D}_{t+1} - \mathcal{D}_t\|_F \|\mathcal{D} - \mathcal{D}_t\|_F \tag{4.52}$$

$$+\frac{3L}{2}\|\mathcal{D}_{t+1}-\mathcal{D}_t\|_F^2+ca^2\|\mathcal{D}-\mathcal{D}_t\|_F^2 \tag{4.53}$$

for some constant $c>0$ for all $t\geq 1$. Recall that the above holds for all $a\in[0,1]$. Note that since $\|\nabla\hat{f}_t\|$ is uniformly bounded and $\mathcal{D}^{\text{dict}}$ is compact (see (A2)), the last expression above, viewed as a quadratic function in a, is strictly increasing in a for all $t\geq 1$ when $c>0$ is sufficiently large. We make such choice for c_3. Hence, the above holds for all $a\geq 0$. Now we may choose $a=b_{t+1}/\|\mathcal{D}-\mathcal{D}_t\|$ and bound the last expression by its first term plus $c_1(\|\mathcal{D}_{t+1}-\mathcal{D}_t\|_F+b_{t+1})^2$ for some constant $c_1>0$. Finally, by the radius restriction $\|\mathcal{D}_{t+1}-\mathcal{D}_t\|_F\leq c'w_{t+1}$, we may choose $b_{t+1}=\|\mathcal{D}_{t+1}-\mathcal{D}_t\|_F$. Then the assertion follows by dividing both sides of the resulting inequality by $\|\mathcal{D}_{t+1}-\mathcal{D}_t\|$. □

Proposition 7. *Assume (A1)-(A3). Suppose there exists a subsequence $(\mathcal{D}_{t_k})_{k\geq 1}$ such that either*

$$\sum_{k=1}^{\infty}\|\mathcal{D}_{t_k}-\mathcal{D}_{t_k+1}\|_F=\infty \quad or \quad \liminf_{k\to\infty}\left|\operatorname{tr}\left(\nabla\hat{f}_{t_k+1}(\mathcal{D}_{t_k+1})^T\frac{\mathcal{D}_{t_k}-\mathcal{D}_{t_k+1}}{\|\mathcal{D}_{t_k}-\mathcal{D}_{t_k+1}\|_F}\right)\right|=0. \tag{4.54}$$

There exists a further subsequence $(s_k)_{k\geq 1}$ of $(t_k)_{k\geq 1}$ such that $\mathcal{D}_\infty:=\lim_{k\to\infty}\mathcal{D}_{s_k}$ exists and is a stationary point of Λ.

Proof. By Proposition 5, we have

$$\sum_{k=1}^{\infty}\|\mathcal{D}_{t_k}-\mathcal{D}_{t_k+1}\|_F\left|\operatorname{tr}\left(\nabla\hat{f}_{t_k+1}(\mathcal{D}_{t_k+1})^T\frac{\mathcal{D}_{t_k}-\mathcal{D}_{t_k+1}}{\|\mathcal{D}_{t_k}-\mathcal{D}_{t_k+1}\|_F}\right)\right|<\infty. \tag{4.55}$$

Hence the former condition implies the latter condition in (4.54). Thus it suffices to show that this latter condition implies the assertion. Assume this condition, and let $(s_k)_{k\geq 1}$ be a subsequence of $(t_k)_{k\geq 1}$ for which the liminf in (4.54) is achieved. By taking a subsequence, we may assume that $\mathcal{D}'_\infty=\lim_{k\to\infty}\mathcal{D}_{s_k}$ and $\hat{f}_\infty:=\lim_{k\to\infty}\hat{f}_{s_k}$ exist.

Now suppose for contradiction that $\mathcal{D}_\infty$ is not a stationary point of $\hat{f}_\infty$ over $\mathcal{C}^{\text{dict}}$. Then there exists $\mathcal{D}^\star\in\mathcal{C}^{\text{dict}}$ and $\delta>0$ such that

$$\operatorname{tr}\left(\nabla\hat{f}_\infty(\mathcal{D}_\infty)^T(\mathcal{D}^\star-\mathcal{D}_\infty)\right)<-\delta<0. \tag{4.56}$$

By triangle inequality, write

$$\|\nabla \hat{f}_{s_k+1}(\mathcal{D}_{s_k})^T(\mathcal{D}^\star - \mathcal{D}_{s_k}) - \nabla \hat{f}_\infty(\mathcal{D}_\infty)^T(\mathcal{D}^\star - \mathcal{D}_\infty)\|_F \tag{4.57}$$

$$\leq \|\nabla \hat{f}_{s_k+1}(\mathcal{D}_{s_k}) - \nabla \hat{f}_\infty(\mathcal{D}_\infty)\|_F \cdot \|\mathcal{D}^\star - \mathcal{D}_{s_k}\|_F + \|\nabla \hat{f}_\infty(\mathcal{D}_\infty)\|_F \cdot \|\mathcal{D}_\infty - \mathcal{D}_{s_k}\|_F. \tag{4.58}$$

Noting that $\|\mathcal{D}_t - \mathcal{D}_{t-1}\|_F = O(w_t) = o(1)$, we see that the right hand side goes to zero as $k \to \infty$. Hence for all sufficiently large $k \geq 1$, we have

$$\operatorname{tr}\left(\nabla \hat{f}_{s_k+1}(\mathcal{D}_{s_k})^T(\mathcal{D}^\star - \mathcal{D}_{s_k})\right) < -\delta/2. \tag{4.59}$$

Then by Proposition 6, denoting $\|\mathcal{C}^{\text{dict}}\|_F := \sup_{\mathcal{D},\mathcal{D}'\in\mathcal{C}^{\text{dict}}}\|\mathcal{D} - \mathcal{D}'\|_F < \infty$,

$$\mathop{lim\,inf}_{k\to\infty}\ \operatorname{tr}\left(\nabla \hat{f}_{s_k+1}(\mathcal{D}_{s_k+1})^T \frac{\mathcal{D}_{s_k} - \mathcal{D}_{s_k+1}}{\|\mathcal{D}_{s_k} - \mathcal{D}_{s_k+1}\|_F}\right) \leq -\frac{c_1\delta}{2\|\mathcal{C}^{\text{dict}}\|_F} < 0, \tag{4.60}$$

which contradicts the choice of the subsequence $(\mathcal{D}_{s_k})_{k\geq 1}$. This shows the assertion. □

Recall that during the update $\mathcal{D}_{t-1} \mapsto \mathcal{D}_t$ in (4.18) each factor matrix of $\mathcal{D}_{t-1}$ changes by at most w_t in Frobenius norm. For each $t \geq 1$, we say $\mathcal{D}_t$ is a *long point* if none of the factor matrices of $\mathcal{D}_{t-1}$ change by w_t in Frobenius norm and *short point* otherwise. Observe that if $\mathcal{D}_t$ is a long point, then imposing the search radius restriction in 4.17 has no effect and $\mathcal{D}_t$ is obtained from $\mathcal{D}_{t-1}$ by a single cycle of block coordinate descent on $\hat{f}_t$ over $\mathcal{C}^{\text{dict}}$.

Proposition 8. *Assume (A1)-(A3) hold. If $(\mathcal{D}_{t_k})_{k\geq 1}$ is a convergent subsequence of $(\mathcal{D}_t)_{t\geq 1}$ consisting of long points, then the $\mathcal{D}_\infty = \lim_{k\to\infty}\mathcal{D}_{s_k}$ is stationary.*

Proof. For each $A \in \mathbb{R}^{R\times R}$, $B \in \mathbb{R}^{I_1\times\cdots\times I_n\times b}$, $\mathcal{D} = [U^{(1)}, \dots, U^{(n)}] \in \mathbb{R}^{I_1\times R}\times\cdots\times\mathbb{R}^{I_n\times R}$, define

$$\hat{g}(A, B, \mathcal{D}) = \operatorname{tr}(A\,((U^{(n)})^T U^{(n)} \odot \dots \odot (U^{(1)})^T U^{(1)})) \tag{4.61}$$

$$- 2\operatorname{tr}\left(B^{(n+1)}(U^{(n)} \otimes_{kr} \dots \otimes_{kr} U^{(1)})^T\right), \tag{4.62}$$

where $B^{(n+1)}$ denotes the mode-$(n+1)$ unfolding of B (see also (4.22)). By taking a subsequence of $(t_k)_{k\geq 1}$, we may assume that $A_\infty := \lim_{k\to\infty} A_{t_k}$ and $\mathbf{B}_\infty := \lim_{k\to\infty}\mathbf{B}_{t_k}$ exist.

Hence the function $\hat{g}_\infty := \lim_{k\to\infty} \hat{g}_{t_k} = \hat{g}(A_\infty, \mathbf{B}_\infty, \cdot)$ is well-defined. Noting that since $\nabla \hat{f}_t = \nabla \hat{g}_t$ for all $t \geq 1$ by Proposition 2, it suffices to show that $\mathcal{D}_\infty$ is a stationary point of $\hat{g}_\infty$ over $\mathcal{C}^{\mathrm{dict}}$ almost surely.

The argument is similar to that of [Ber97, Prop. 2.7.1]. However, here we do not need to assume uniqueness of solutions to minimization problems of $\hat{f}_t$ in each block coordinate due to the added search radius restriction. Namely, write $\mathcal{D}_\infty = [U_\infty^{(1)}, \dots, U_\infty^{(n)}]$. Then for each $k \geq 1$,

$$\hat{g}_{t_k+1}(U_{t_k+1}^{(1)}, U_{t_k}^{(2)}, \dots, U_{t_k}^{(n)}) \leq \hat{g}_{t_k+1}(U^{(1)}, U_{t_k}^{(2)}, \dots, U_{t_k}^{(n)}) \tag{4.63}$$

for all $U^{(1)} \in \mathcal{C}^{(1)} \cap \{U \colon \|U - U_{t_k}^{(1)}\|_F \leq c' w_{t_k+1}\}$. In fact, since $\mathcal{D}_{t_k}$ is a long point by the assumption, (4.63) holds for all $U^{(1)} \in \mathcal{C}^{(1)}$. Taking $k \to \infty$ and using the fact that $\|U_{t_k+1}^{(1)} - U_{t_k}^{(1)}\|_F \leq c' w_{t_k+1} = o(1)$,

$$\hat{g}_\infty(U_\infty^{(1)}, U_\infty^{(2)}, \dots, U_\infty^{(n)}) \leq \hat{g}_\infty(U^{(1)}, U_\infty^{(2)}, \dots, U_\infty^{(n)}) \quad \text{for all } U_1 \in \mathcal{C}^{(1)}. \tag{4.64}$$

Since $\mathcal{C}^{(1)}$ is convex, it follows that

$$\nabla_1 \hat{g}_\infty(\mathcal{D}_\infty)^T (U_1 - U_1^{(\infty)}) \geq 0 \qquad \text{for all } U_1 \in \mathcal{C}^{(1)}, \tag{4.65}$$

where ∇_1 denotes the partial gradient with respect to the first block $U^{(1)}$. By using a similar argument for other coordinates of $\mathcal{D}_\infty$, it follows that $\nabla \hat{g}_\infty(\mathcal{D}_\infty)^T(\mathcal{D} - \mathcal{D}_\infty) \geq 0$ for all $\mathcal{D} \in \mathcal{C}^{\mathrm{dict}}$. This shows the assertion. □

Proposition 9. *Assume (A1)-(A3) hold. Suppose there exists a non-stationary limit point $\mathcal{D}_\infty$ of Λ. Then there exists $\varepsilon > 0$ such that the ε-neighborhood $B_\varepsilon(\mathcal{D}_\infty) := \{\mathcal{D} \in \mathcal{C}^{\mathrm{dict}} \,|\, \|\mathcal{D} - \mathcal{D}_\infty\|_F < \varepsilon\}$ with the following properties:*

(a) *$B_\varepsilon(\mathcal{D}_\infty)$ does not contain any stationary points of Λ.*

(b) *There exists infinitely many $\mathcal{D}_t$'s outside of $B_\varepsilon(\mathcal{D}_\infty)$.*

Proof. We will first show that there exists an ε-neighborhood $B_\varepsilon(\mathcal{D}_\infty)$ of $\mathcal{D}_\infty$ that does not contain any long points of Λ. Suppose for contradiction that for each $\varepsilon > 0$, there exists a long point Λ in $B_{\varepsilon}(\mathcal{D}_{\infty})$. Then one can construct a sequence of long points converging to $\mathcal{D}_{\infty}$. But then by Proposition 8, $\mathcal{D}_{\infty}$ is a stationary point, a contradiction.

Next, we show that there exists $\varepsilon > 0$ such that $B_{\varepsilon}(\mathcal{D}_{\infty})$ satisfies **(a)**. Suppose for contradiction that there exists no such $\varepsilon > 0$. Then we have a sequence $(\mathcal{D}_{\infty;k})_{k\geq 1}$ of stationary points of Λ that converges to $\mathcal{D}_{\infty}$. Denote the limiting surrogate loss function associated with $\mathcal{D}_{\infty;k}$ by $\hat{f}_{\infty;k}$. Recall that each $\hat{f}_{\infty;k}$ is parameterized by elements in a compact set (see (A1), Proposition 2, and Lemma 6) in Appendix 4.10. Hence by choosing a subsequence, we may assume that $\hat{f}_{\infty} := \lim_{k\to\infty} \hat{f}_{\infty;k}$ is well-defined. Fix $\mathcal{D} \in \mathcal{C}^{\text{dict}}$ and note that by Cauchy-Schwarz inequality,

$$\nabla \hat{f}_{\infty}(\mathcal{D}_{\infty})^T(\mathcal{D}-\mathcal{D}_{\infty}) \geq -\|\nabla \hat{f}_{\infty}(\mathcal{D}_{\infty}) - \nabla \hat{f}_{\infty;k}(\mathcal{D}_{\infty;k})\|_F \cdot \|\mathcal{D}-\mathcal{D}_{\infty}\|_F \tag{4.66}$$

$$- \|\nabla \hat{f}_{\infty;k}(\mathcal{D}_{\infty;k})\|_F \cdot \|\mathcal{D}_{\infty}-\mathcal{D}_{\infty;k}\|_F \tag{4.67}$$

$$+ \nabla \hat{f}_{\infty;k}(\mathcal{D}_{\infty;k})^T(\mathcal{D}-\mathcal{D}_{\infty;k}). \tag{4.68}$$

Note that $\nabla \hat{f}_{\infty;k}(\mathcal{D}_{\infty;k})^T(\mathcal{D}-\mathcal{D}_{\infty;k}) \geq 0$ since $\mathcal{D}_{\infty;k}$ is a stationary point of $\hat{f}_{\infty;k}$ over $\mathcal{C}^{\text{dict}}$. Hence by taking $k \to \infty$, this shows $\nabla \hat{f}_{\infty}(\mathcal{D}_{\infty})^T(\mathcal{D}-\mathcal{D}_{\infty}) \geq 0$. Since $\mathcal{D} \in \mathcal{D}^{\text{dict}}$ was arbitrary, this shows that $\mathcal{D}_{\infty}$ is a stationary point of $\hat{f}_{\infty}$ over $\mathcal{C}^{\text{dict}}$, a contradiction.

Lastly, from the earlier results, we can choose $\varepsilon > 0$ such that $B_{\varepsilon}(\mathcal{D}_{\infty})$ has no long points of Λ and also satisfies **(b)**. We will show that $B_{\varepsilon/2}(\mathcal{D}_{\infty})$ satisfies **(c)**. Then $B_{\varepsilon/2}(\mathcal{D}_{\infty})$ satisfies **(a)**-**(b)**, as desired. Suppose for contradiction there are only finitely many $\mathcal{D}_t$'s outside of $B_{\varepsilon/2}(\mathcal{D}_{\infty})$. Then there exists an integer $M \geq 1$ such that $\mathcal{D}_t \in B_{\varepsilon/2}(\mathcal{D}_{\infty})$ for all $t \geq M$. Then each $\mathcal{D}_t$ for $t \geq M$ is a short point of Λ. By definition, it follows that $\|\mathcal{D}_t - \mathcal{D}_t\|_F \geq w_t$ for all $t \geq M$, so $\sum_{t=1}^{\infty}\|\mathcal{D}_t - \mathcal{D}_t\|_F \geq \sum_{t=1}^{\infty} w_t = \infty$. Then by Proposition 7, there exists a subsequence $(s_k)_{k\geq 1}$ such that $\mathcal{D}'_{\infty} := \lim_{k\to\infty} \mathcal{D}_{t_k}$ exists and is stationary. But since $\mathcal{D}'_{\infty} \in B_{\varepsilon}(\mathcal{D})$, this contradicts **(a)** for $B_{\varepsilon}(\mathcal{D})$. This shows the assertion. □

We are now ready to give a proof of Lemma 1 **(iv)**.

Proof. **of Lemma 1 (iv)**. Assume (A1)-(A3) hold. Suppose there exists a non-stationary limit point $\mathcal{D}_{\infty}$ of Λ. By Proposition 9, we may choose $\varepsilon > 0$ such that $B_{\varepsilon}(\mathcal{D}_{\infty})$ satisfies the conditions **(a)**-**(b)** of Proposition 9. Choose $M \geq 1$ large enough so that $c' w_t < \varepsilon/4$ whenever $t \geq M$. We call an integer interval $I := [\ell, \ell')$ a *crossing* if $\mathcal{D}_\ell \in B_{\varepsilon/3}(\mathcal{D}_\infty)$, $\mathcal{D}_{\ell'} \notin B_{2\varepsilon/3}(\mathcal{D}_\infty)$, and no proper subset of I satisfies both of these conditions. By definition, two distinct crossings have empty intersection. Fix a crossing $I = [\ell, \ell')$. Then it follows that by triangle inequality,

$$\sum_{t=\ell}^{\ell'-1} \|\mathcal{D}_{t+1} - \mathcal{D}_t\|_F \geq \|\mathcal{D}_{\ell'} - \mathcal{D}_\ell\|_F \geq \varepsilon/3. \tag{4.69}$$

Note that since $\mathcal{D}_\infty$ is a limit point of Λ, $\mathcal{D}_t$ visits $B_{\varepsilon/3}(\mathcal{D}_\infty)$ infinitely often. Moreover, by condition **(a)** of Proposition 9, $\mathcal{D}_t$ also exits $B_\varepsilon(\mathcal{D}_\infty)$ infinitely often. It follows that there are infinitely many crossings. Let t_k denote the k^{th} smallest integer that appears in some crossing. By definition, $\mathcal{D}_{t_k} \in B_{2\varepsilon/3}(\mathcal{D}_\infty)$ for $k \geq 1$. Then $t_k \to \infty$ as $k \to \infty$, and by (4.69),

$$\sum_{k=1}^{\infty} \|\mathcal{D}_{t_k+1} - \mathcal{D}_{t_k}\|_F \geq (\text{\# of crossings})\, \frac{\varepsilon}{3} = \infty. \tag{4.70}$$

Then by Proposition 7, there is a further subsequence $(s_k)_{k\geq 1}$ of $(t_k)_{k\geq 1}$ such that $\mathcal{D}'_\infty := \lim_{k\to\infty} \mathcal{D}_{s_k}$ exists and is stationary. However, since $\mathcal{D}_{t_k} \in B_{2\varepsilon/3}(\mathcal{D}_\infty)$ for $k \geq 1$, we have $\mathcal{D}'_\infty \in B_\varepsilon(\mathcal{D}_\infty)$. This contradicts the condition **(b)** of Proposition 9 for $B_\varepsilon(\mathcal{D}_\infty)$ that it cannot contain any stationary point of Λ. This shows the assertion. □

4.6.4 Proof of the main result

Now we prove the main result in this paper, Theorem 1.

Proof. **of Theorem 1.** Suppose (A1)-(A3) hold. We first show **(i)**. Recall that $\mathbb{E}[\hat{f}_t(\mathcal{D}_t)]$ converges by Lemma 3. Jensen's inequality and Lemma 1 **(iv)** imply

$$|\mathbb{E}[h_{t+1}(\mathcal{D}_{t+1})] - \mathbb{E}[h_t(\mathcal{D}_t)]| \leq \mathbb{E}\left[|h_{t+1}(\mathcal{D}_{t+1}) - h_t(\mathcal{D}_t)|\right] = O(w_{t+1}). \tag{4.71}$$

Since $\mathbb{E}[\hat{f}_t(\mathcal{D}_t)] \geq \mathbb{E}[f_t(\mathcal{D}_t)]$, Lemma 3 **(ii)**-**(iii)** and Lemma 7 in Appendix 4.10 give

$$\lim_{t\to\infty} \mathbb{E}[f_t(\mathcal{D}_t)] = \lim_{t\to\infty} \mathbb{E}[\hat{f}_t(\mathcal{D}_t)] + \lim_{t\to\infty} \left(\mathbb{E}[f_t(\mathcal{D}_t)] - \mathbb{E}[\hat{f}_t(\mathcal{D}_t)]\right) \tag{4.72}$$

$$= \lim_{t\to\infty} \mathbb{E}[\hat{f}_t(\mathcal{D}_t)] \in (1,\infty). \tag{4.73}$$

This shows **(i)**.

Next, we show **(ii)**. Triangle inequality gives

$$|f(\mathcal{D}_t) - \hat{f}_t(\mathcal{D}_t)| \le \left(\sup_{\mathcal{D}\in\mathcal{C}^{\text{dict}}} |f(\mathcal{D}) - f_t(\mathcal{D})| \right) - h_t(\mathcal{D}_t). \tag{4.74}$$

Note that $|h_{t+1}(\mathcal{D}_{t+1}) - h_t(\mathcal{D}_t)| = O(w_{t+1})$ by Lemma 1 **(iii)**. Hence Lemma 3 **(iv)** and Lemma 7 in Appendix 4.10 show that $h_t(\mathcal{D}_t) \to 0$ almost surely. Furthermore, (4.74) and Lemma 9 in Appendix 4.10 show that $|f(\mathcal{D}_t) - \hat{f}_t(\mathcal{D}_t)| \to 0$ almost surely. This completes the proof of **(ii)**.

Lastly, we show **(iii)**. Further assume (A4). Let $\mathcal{D}_\infty \in \mathcal{C}^{\text{dict}}$ be an arbitrary limit point of the sequence $(\mathcal{D}_t)_{t\ge1}$. Recall that $\Sigma_t := (\mathcal{D}_t, A_t, \mathbf{B}_t, r_t)_{t\ge0}$ is bounded by Lemma 5(in Appendix 4.10) and (A1) and (A2). Hence we may choose a random subsequence $(t_k)_{k\ge1}$ so that $\mathcal{D}_{t_k} \to \mathcal{D}_\infty$. By taking a further subsequence, we may also assume that Σ_{t_k} converges to some random element $(\mathcal{D}_\infty, A_\infty, \mathbf{B}_\infty, r_\infty)$ a.s. as $k\to\infty$. Then $\hat{f}_\infty := \lim_{k\to\infty} \hat{f}_{t_k}$ exists almost surely. It is important to note that $\mathcal{D}_\infty$ is a stationary point of $\hat{f}_\infty$ over $\mathcal{C}^{\text{dict}}$ by Lemma 1 **(iv)**.

Recall that $\hat{f}_t(\mathcal{D}_t) - f_t(\mathcal{D}_t) \to 0$ as $t\to\infty$ almost surely by part **(ii)**. By using continuity of $\hat{f}_t$, f_t, f in parameters (see (A4)), it follows that

$$\left|\hat{f}_\infty(\mathcal{D}_\infty) - f(\mathcal{D}_\infty)\right| = \lim_{k\to\infty} \left|\hat{f}_{t_k}(\mathcal{D}_{t_k}) - f_{t_k}(\mathcal{D}_{t_k})\right| \tag{4.75}$$

$$\le \lim_{k\to\infty} \left(\sup_{\mathcal{D}\in\mathcal{C}^{\text{dict}}} |f - f_{t_k}(\mathcal{D})| - h_{t_k}(\mathcal{D}_{t_k}) \right) = 0, \tag{4.76}$$

where the last equality also uses Lemma 9 in Appendix 4.10.

Fix $\varepsilon > 0$ and $\mathcal{D} \in \mathbb{R}^{I_1\times R} \times \cdots \times \mathbb{R}^{I_n\times R}$. Hence, almost surely,

$$\hat{f}_\infty(\mathcal{D}_\infty + \mathcal{D}) = \lim_{k\to\infty} \hat{f}_{s_k}(\mathcal{D}_{s_k} + \mathcal{D}) \ge \lim_{k\to\infty} f_{s_k}(\mathcal{D}_{s_k} + \mathcal{D}) = f(\mathcal{D}_\infty + \mathcal{D}), \tag{4.77}$$

where the last equality follows from Lemma 9. Since $\nabla\hat{f}$ and ∇f are both Lipschitz (see (A4) for the latter), by Lemma 4 in Appendix 4.10, we have

$$\left|\hat{f}_\infty(\mathcal{D}_\infty + \varepsilon\mathcal{D}) - \hat{f}_\infty(\mathcal{D}_\infty) - \operatorname{tr}\left(\nabla\hat{f}_\infty(\mathcal{D}_\infty)^T(\varepsilon\mathcal{D})\right)\right| \le c_1\varepsilon^2\|\mathcal{D}\|_F^2, \tag{4.78}$$

$$\left| f(\mathcal{D}_\infty + \varepsilon\mathcal{D}) - f(\mathcal{D}_\infty) - \operatorname{tr}\left(\nabla f(\mathcal{D}_\infty)^T(\varepsilon\mathcal{D})\right)\right| \le c_1\varepsilon^2\|\mathcal{D}\|_F^2, \tag{4.79}$$

for some constant $c_1 > 0$ for all $\varepsilon > 0$. Recall that $\hat{f}_\infty(\mathcal{D}_\infty) = f(\mathcal{D}_\infty)$ a.s. by (4.75). Hence it follows that there exists some constant $c_2 > 0$ such that almost surely

$$\operatorname{tr}\left(\left(\nabla\hat{f}_\infty(\mathcal{D}_\infty) - \nabla f(\mathcal{D}_\infty)\right)^T(\varepsilon\mathcal{D})\right) \ge -c_2\varepsilon^2\|\mathcal{D}\|_F^2. \tag{4.80}$$

After canceling out $\varepsilon > 0$ and letting $\varepsilon \searrow 0$ in (4.80),

$$\operatorname{tr}\left(\left(\nabla\hat{f}_\infty(\mathcal{D}_\infty) - \nabla f(\mathcal{D}_\infty)\right)^T\mathcal{D}\right) \ge 0 \qquad \text{a.s.} \tag{4.81}$$

Since this holds for all $\mathcal{D} \in \mathbb{R}^{I_1\times R}\cdots\times\mathbb{R}^{I_n\times R}$, it follows that $\nabla\hat{f}_\infty(\mathcal{D}_\infty) = \nabla f(\mathcal{D}_\infty)$ almost surely. But since $\mathcal{D}_\infty$ is a stationary point of $\hat{f}_\infty$ over $\mathcal{C}^{\text{dict}}$ by Lemma 1 **(iv)**, it follows that $\nabla\hat{f}_\infty(\mathcal{D}_\infty)$ is in the normal cone of $\mathcal{C}^{\text{dict}}$ at $\mathcal{D}_\infty$ (see., e.g., [BBV04]). The same holds for $\nabla f(\mathcal{D}_\infty)$. This means that $\mathcal{D}_\infty$ is a stationary point of f over $\mathcal{C}^{\text{dict}}$. Since $\mathcal{D}_\infty$ is an arbitrary limit point of $\mathcal{D}_t$, the desired conclusion follows. □

4.7 Experimental validation

In this section, we compare the performance of our proposed online CPDL algorithm (Algorithm 6) for the standard (offline) NCPD problem (4.8) against the two most popular algorithms of Alternating Least Squares (ALS), which is a special instance of Block Coordinate Descent, and Multiplicative Update (MU) (see [SH05]) for this task. See Algorithms 11 and 12 for implementations of ALS and MU.

We give a more precise statement of the NCPD problem we consider here. Given a 3-mode data tensor $\mathbf{X} \in \mathbb{R}_{\ge 0}^{d_1\times d_2\times d_3}$ and an integer $R \ge 1$, we want to find three nonnegative factor matrices $U^{(k)} \in \mathbb{R}_{\ge 0}^{d_k\times R}$, $k = 1, 2, 3$, that minimize the following CP-reconstruction error:

$$\min_{[U^{(1)},U^{(2)},U^{(3)}]\in\mathcal{C}_M^{\text{dict}}} \left\|\mathbf{X} - \sum_{i=1}^{R}\bigotimes_{k=1}^{3} U^{(k)}(:,i)\right\|_F, \tag{4.82}$$

where $\mathcal{C}_M^{\text{dict}}$ is the subset of $\mathbb{R}_{\geq 0}^{d_1\times R}\times\mathbb{R}_{\geq 0}^{d_2\times R}\times\mathbb{R}_{\geq 0}^{d_3\times R}$ consisting of factor matrices of Frobenius norm bounded by a fixed constant $M \geq \sqrt{R}\|\mathbf{X}\|_F^{1/3}$. Note that the constraint set $\mathcal{C}_M^{\text{dict}}$ is convex and compact, as required in (A2) for Theorem 1 to apply. We claim that the additional bounded norm constraint on the factor matrices does not lose any generality, in the sense that an optimal solution of (4.82) with $M \geq \sqrt{R}\|\mathbf{X}\|_F^{1/3}$ has the same objective value as the optimal solution of (4.82) with $M=\infty$:

$$\left(\text{optimal value of (4.82) for } M \geq \sqrt{R}\|\mathbf{X}\|_F^{1/3}\right) = \left(\text{optimal value of (4.82) for } M=\infty\right). \tag{4.83}$$

In order to maintain the flow, we justify this claim at the end of this section.

We consider one synthetic and three real-world tensor data derived from text data and that were used for dynamic topic modeling experiments in [KKL+21]. Each document is encoded as a 5000 or 7000 dimensional word frequency vector using tf-idf vectorizer [RU11].

1. $\mathbf{X}_{\text{synth}} \in \mathbb{R}_{\geq 0}^{100\times 100\times 100}$ is generated by $\mathbf{X}_{\text{synth}} = 0.01 * \texttt{Out}(V_1, V_2, V_3)$, where the loading matrices $V_1, V_2, V_3 \in \mathbb{R}_{\geq 0}^{100\times 50}$ are generated by sampling each of their entries uniformly and independently from the unit interval $[0,1]$.

2. $\mathbf{X}_{\text{20News}} \in \mathbb{R}_{\geq 0}^{40\times 5000\times 26}$ (41.6MB) is a tensor representing semi-synthetic text data based on 20 Newsgroups dataset [Ren08] synthesized in [KKL+21] for dynamic topic modeling, consisting of 40 stacks of 26 documents encoded in 5000 dimensional word space.

3. $\mathbf{X}_{\text{Twitter}} \in \mathbb{R}_{\geq 0}^{90\times 5000\times 1000}$ (3.6GB) is an anonymized Twitter text data related to the COVID-19 pandemic from Feb. 1 to May 1 of 2020. The three modes correspond to days, words, and tweets, in order. Each day, the top 1000 most retweeted English tweets are collected. The original data was collected in [KKL+21].[2]

4. $\mathbf{X}_{\text{Headlines}} \in \mathbb{R}_{\geq 0}^{203\times 7000\times 700}$ (8.0GB) is a tensor derived in [KKL+21] from news headlines published over a period of 17 years sourced from the Australian news source ABC

[2]For code repository, see https://github.com/lara-kassab/dynamic-tensor-topic-modeling

[Kul18]. The three modes correspond to months, words, and headlines, in order. In each month, 700 headlines are chosen uniformly at random.

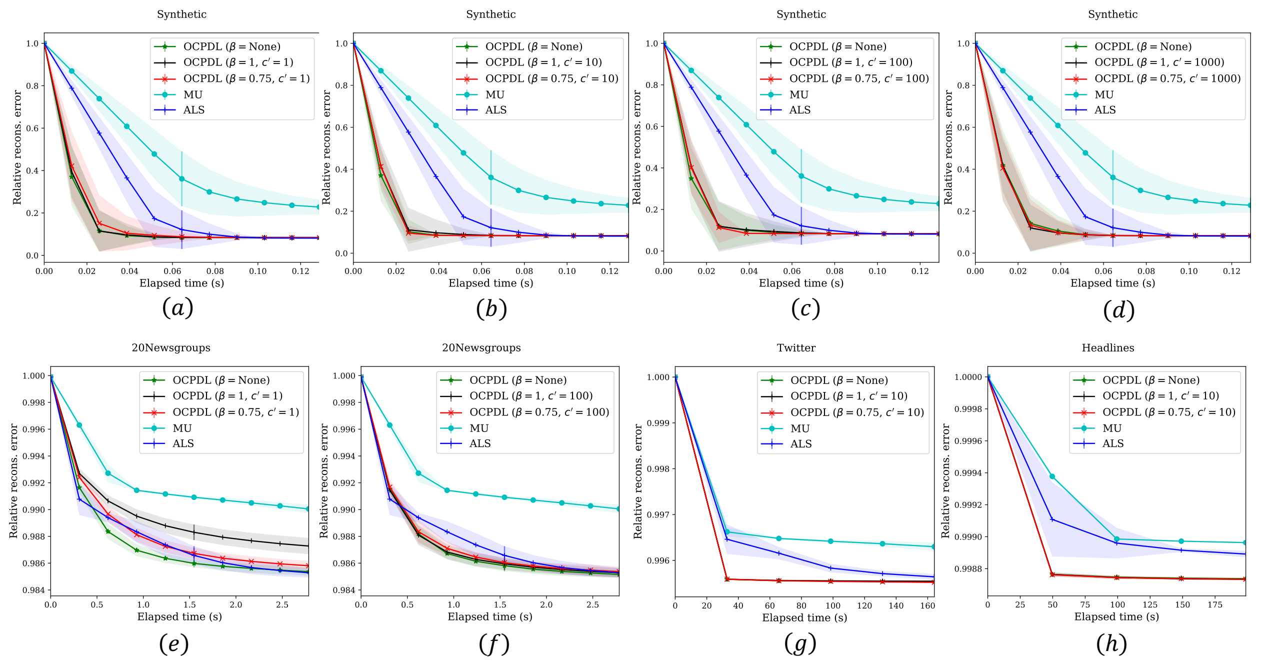

Figure 4.2: Comparison of performance of online CPDL for the nonnegative tensor factorization problem against Alternating Least Squares (ALS) and Multiplicative Update (MU). For each data tensor, we apply each algorithm to find nonnegative loading matrices $U^{(1)}, U^{(2)}, U^{(3)}$ of $R = 5$ columns. We repeat this multiple times (50 for synthetic, 20 for 20Newsgroups, and 10 for the other two) and the average reconstruction error with 1 standard deviation are shown by the solid lines and shaded regions of respective colors.

For all datasets, we used all algorithms to learn the loading matrices $U^{(1)}, U^{(2)}, U^{(3)}$ with $R = 5$ columns, that evolve in time as the algorithm proceeds. The choice of $R = 5$ is arbitrary and is not ideal especially for the real data tensors, but it suffices for the purpose of this experiment as a benchmark of our online CPDL against ALS and MU. We plot the reconstruction error $\|\mathbf{X} - \texttt{Out}(U^{(1)}, U^{(2)}, U^{(3)})\|_F$ against elapsed time in both cases in Figure 4.2. Since these benchmark algorithms are also iterative (see Algorithms 11 and 12), we can measure how reconstruction error drops as the three algorithms proceed. In order to make a fair comparison, we compare the reconstruction error against CPU times with the same machine, not against iteration counts, since a single iteration may have different computational costs across different algorithms. For ALS and MU, we disregarded the bounded norm constraint in (4.82), which makes it only favorable to those benchmark methods so it is still a fair comparison of our method.

We give some implementation details of online CPDL (Algorithm 6) for the offline NCPD problem in (4.82). From the given data tensor $\mathbf{X}$, we obtain a sequence of tensors $\mathbf{X}_1, \dots, \mathbf{X}_T$ obtained by subsampling $1/5$ of the coordinates from the last mode. Hence while ALS and MU require loading the entire tensors into memory, only $1/5$ of the data needs to be loaded to execute online CPDL. In Figure 4.2, OCPDL (β) for for $\beta \in \{0.75,\, 1\}$ denotes Algorithm 6 with weights $w_t = c' t^{-\beta} / \log t$ (with $w_1 = c'$), where 0.75 and 1 for β correspond to the two extreme values that satisfy the assumption (A3) (see also (A3')) of Theorem 1; the case of $\beta =$ None uses $c' = \infty$ and $w_t \equiv t^{-1}/(\log t)$ (with $w_1 = 1$). In all cases, the weights satisfy (A3) so the algorithm is guaranteed to converge to the stationary points of the objective function almost surely by Theorem 1. The constant c' is chosen from $\{1, 10, 100, 1000\}$ for $\beta \in \{0.75, 1\}$. Initial loading matrices for Algorithm 6 are chosen with i.i.d. entries drawn from the uniform distribution on $[0, 1]$.

Note that since the last mode of the full tensors is subsampled, the loading matrices we learn from online CPDL have sizes $(d_1 \times R)$, $(d_2 \times R)$, and $(d_3' \times R)$, where $d_3' < d_3$ equals the size of the last mode of the subsampled tensors. In order to compute the reconstruction error for the full $d_1 \times d_2 \times d_3$ tensor, we recompute the last factor matrix of size $d_3 \times R$ by using the first two factor matrices with the sparse coding algorithm (Algorithm 9). This last step of computing a single loading matrix while fixing all the others is equivalent to a single step of ALS in Algorithm 11.

In Figure 4.2, each algorithm is used multiple times (50 for Synthetic, 20 for 20Newsgroups, and 10 for Twitter and Headlines) for the same data, and the plot shows the average reconstruction errors together with their standard deviation (shading). In all cases except the smallest initial radius $c' = 1$ on the densest tensor $\mathbf{X}_{\text{20News}}$, online CPDL is able to obtain significantly lower reconstruction error much more rapidly than the other two algorithms and maintains low average reconstruction accuracy.

For $\mathbf{X}_{\text{20News}}$ with $c' = 1$ (Figure 4.2 (e)), we observe some noticible difference in the performance of online CPDL depending on β, where larger values of β (faster decaying radii) give slower convergence. This seems to be due to the fact that $\mathbf{X}_{\text{20News}}$ is the densest among the four tensors by orders of magnitude and $c' = 1$ gives too small of an initial radius. Namely, the average Frobeinus norm, $\|\mathbf{X}\|_F/(d_1 d_2 d_3)$ equals 6.26×10^{-5} for $\mathbf{X}_{\text{Synthetic}}$, 1.10×10^{-3} for $\mathbf{X}_{\text{20News}}$, 6.65×10^{-7} for $\mathbf{X}_{\text{Twitter}}$, and 3.78×10^{-7} and $\mathbf{X}_{\text{Headlines}}$. However, we did not observe any significant difference in all other cases. In general, when c' is large enough, it appears that the radius restriction in Algorithm 6 enables the theoretical convergence guarantee in Theorem 1 without any compromise in practical performance, which did not depend significantly on the decay rate paramter β. In our experiments, $c' = \|\mathbf{X}\|_F$ was sufficiently large, where $\|\mathbf{X}_{\text{Synthetic}}\|_F = 62.61$, $\|\mathbf{X}_{\text{20News}}\|_F = 32.74$, $\|\mathbf{X}_{\text{Twitter}}\|_F = 299.37$, and $\|\mathbf{X}_{\text{Synthetic}}\|_F = 376.38$.

Proof. **of claim (4.83).** Suppose $\mathcal{D}_\infty := [U^{(1)}, U^{(2)}, U^{(3)}]$ is an optimal solution of (4.82) without norm restriction (i.e., $M = \infty$). Fix a column index $i \in \{1, \dots, R\}$ and positive scalars $\alpha_1, \alpha_2, \alpha_3$ such that $\alpha_1\alpha_2\alpha_3 = 1$. The objective in (4.82) is invariant under rescaling the three respective columns: $U^{(k)}(:, i) \mapsto \alpha_k U^{(k)}(:, i)$ for $k \in \{1, 2, 3\}$. At the optimal factor matrices, the objective in (4.82) should be at most $\|\mathbf{X}\|_F$. Since all factors are nonnegative, it follows that

$$\prod_{k=1}^{3} \|\alpha_k U^{(k)}(:, i)\|_F = \left\| \bigotimes_{k=1}^{3} \alpha_k U^{(k)}(:, i) \right\|_F \le \|\mathbf{X}\|_F. \tag{4.84}$$

Then we can choose α_k's in a way that $\|\alpha_k U^{(k)}(:, i)\|_F$ is constant in k[3], in which case $\|\alpha_k U^{(k)}(:, i)\|_F \le \|\mathbf{X}\|_F^{1/3}$. This argument shows that we can rescale the ith columns of the optimal factor matrices in $\mathcal{D}_\infty$ in a way that the objective value does not change and the columns have norms bounded by $\|\mathbf{X}\|_F^{1/3}$. This holds for all columns i, so we can find a tuple of factor matrices $[V^{(1)}, V^{(2)}, V^{(3)}]$ in $\mathcal{C}_M^{\text{dict}}$ that has the same objective value as $\mathcal{D}_\infty$ as long as $M \ge R\|\mathbf{X}\|_F^{1/3}$. □

[3]e.g., $\alpha_k = a_j a_l / a_k^2$, where $a_k := \|U^{(k)}(:, i)\|_F$ and $j, k, l \in \{1, 2, 3\}$ are distinct

4.8 Applications

For all our applications in this section, we take the constraint sets $\mathcal{C}^{\text{code}}$ and $\mathcal{C}^{\text{dict}}$ in Algorithm 6 to consists of *nonnegative* matrices so that the learned CP-dictionary gives a "parts-based representation" of the subject data as in classical NMF (see [LS99, LS01, LYC09]). In all our experiments in this section, we used the balanced weight $w_t = 1/t$, which satisfies the assumption (A3).

4.8.1 Reshaping tensors before CP-decomposition to preserve joint features

Before we discuss our real-world applications of the online CPDL method, we first give some remarks on reshaping tensor data before factorization and why it would be useful in applications.

One may initially think that concatenating some modes of a tensor into a single mode before applying CP-decomposition loses joint features corresponding to the concatenated modes. In fact, if we *undo the unfolding* after the decomposition, it actually *preserves* the joint features. Hence in practice, one can exploit the tensor structure in multiple ways before CP-decomposition to disentangle a select set of features in the desired way, which we demonstrate through analyzing a diverse set of examples from image, video, and time-series in Section 4.8.

To better illustrate our point, suppose we have three discrete random variables X_1, X_2, X_3, where X_i takes n_i distinct values for $1 \leq i \leq 3$. Denote their 3-dimensional joint distribution as a 3-mode tensor $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$. Suppose we have its CP-decomposition $\mathbf{X} \approx \texttt{Out}(U^{(1)}, U^{(2)}, U^{(3)})$. We can interpret this as the sum of R product distributions of the marginal distributions given by the respective columns in the three factor matrices. In an extreme case of a $R = 1$ CP-decomposition, any kind of joint features among multiple random variables will be lost in the single product distribution.

On the other hand, consider combining the first two random variables (X_1, X_2) into a single random variable, say, Y_1, which takes $n_1 n_2$ distinct values. Then the joint distribution of (Y_1, X_3) will be represented as a 2-dimensional tensor $\mathbf{X}^{(12)} \in \mathbb{R}^{n_1 n_2 \times n_3}$, which corresponds to the tensor obtained by concatenating the first two modes of $\mathbf{X}$. Suppose we have its CP-decomposition $\mathbf{X}^{(12)} \approx \texttt{Out}(V^{(1)}, V^{(2)})$, where $V^{(1)} \in \mathbb{R}^{n_1 n_2 \times R}$ and $V^{(2)} \in \mathbb{R}^{n_3 \times R}$. Then we can reshape each column $V^{(1)}(:, i)$ to a 2-dimensional tensor $V^{(1)}_{n_1 \times n_2}(:, i) \in \mathbb{R}^{n_1 \times n_2}$ by using the ordering of entries in $[n_1] \times [n_2]$ we used to concatenate X_1 and X_2 into Y_1. In this way, we have approximated the full joint distribution $\mathbf{X}$ as the sum of the product between two- and one-dimensional distributions $V^{(1)}_{n_1 \times n_2}(:, i) \otimes V^{(2)}(:, i)$. In this factorization, the joint features of X_1 and X_2 can still be encoded in the 2-dimensional joint distributions $V^{(1)}_{n_1 \times n_2}(:, i)$, and only the joint features between (X_1, X_2) and X_3 are disentangled.

For instance, the tensor for the mouse brain activity video in Subsection 4.8.3 has four modes $(\texttt{time}, \texttt{horizontal}, \texttt{vertical}, \texttt{color})$. There is almost no change in the shape of the brain in the video and only the color changes indicate neuronal activation in time. Hence, we do not want to disentangle the horizontal, vertical, and color modes, but instead, concatenate them to maintain the joint feature of the spatial activation pattern (see Figure 4.4). See also Figures 4.3 and 4.5 for the effect of various tensor reshaping before factorization in the context of image and time-series data.

4.8.2 Image processing applications

We first apply our algorithm to patch-based image processing. A workflow for basic patch-based image analysis is to extract small overlapping patches from some large images, vectorize these patches, apply some standard dictionary learning algorithm, and reshape back. Dictionaries obtained from this general procedure have a wide variety of uses, including image compression, denoising, deblurring, and inpainting [Ela10, DLZS11, PRSE17, MMYZ13].

Although this procedure has produced countless state-of-the-art results, a major drawback to such methods is that vectorizing image patches can greatly slow down the learning process by increasing the effective dimension of the dictionary learning problem. Moreover, by respecting the natural tensor structure of the data, we find that our learned dictionary

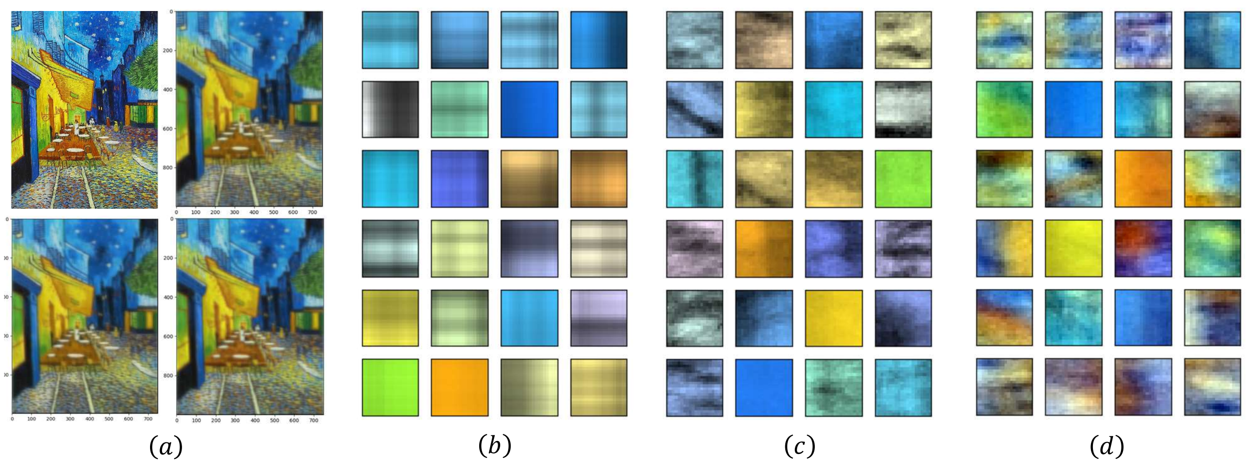

Figure 4.3: Color image reconstruction by online CPDL. The original image is shown in the top left of (a). The top right reconstruction in (a) is derived from the dictionary learned from the unmodified tensor decomposition of color image patches, which is exemplified in (b). The bottom left reconstruction in (a) uses the dictionary in (c) learned by tensor decomposition of color image patches whose spatial modes are vectorized. The bottom right reconstruction in (a) uses the dictionary learned by a tensor decomposition of fully vectorized color image patches, which is shown in (d).

atoms display a qualitative difference from those trained on reshaped color image patch data. We illustrate this phenomenon in Figure 4.3. Our experiment is as follows. Figure 4.3 (a) top left is a famous painting (Van Gogh's *Café Terrace at Night*) from which we extracted 1000 color patches of shape ($\texttt{vertical} \times \texttt{horizontal} \times \texttt{color}$) $= (20 \times 20 \times 3)$. We applied our online CPDL algorithm (Algorithm 6 for 400 iterations with $\lambda = 1$) to various reshapings of such patches to learn three separate dictionaries, each consisting of 24 atoms.

The first dictionary, displayed in Figure 4.3 (b), is obtained by applying online CPDL without reshaping the patches. Due to the rank-1 restriction on the atoms as a 3-mode tensor, the spatial features are parallel to the vertical or horizontal axes, and also color variation within each atom is only via scalar multiple (a.k.a. 'saturation'). The second dictionary, Figure 4.3 (c), was trained by vectorizing the color image patches along the spatial axes, applying online CPDL to the resulting 2-mode data tensors of shape ($\texttt{space} \times \texttt{color}$) $=$ (400×3) , and reshaping back. Here, the rank-1 restriction on the atoms as a 2-mode tensor separates the spatial and color features, but now the spatial features in the atoms are more 'generic' as they do not have to be parallel to the vertical or horizontal axes. Note that the color variation within each atom is still via a scalar multiple. Lastly, the third dictionary, Figure 4.3 (d), is obtained by applying our online CPDL to the fully vectorized image patch data. Here the features in the atoms do not have any rank-1 restriction along with any mode so that they exhibit 'fully entangled' spatial and color features. Although dictionary (b) requires much less storage, the reconstructed images from all three dictionaries shown in Figure 4.3 (a) show that it still performs adequately for the task of image reconstruction.

4.8.3 Learning spatial and temporal activation patterns in cortex

In this subsection, we demonstrate our method on video data of brain activity across a mouse cortex, and how our online CPDL learns dictionaries for the spatial and temporal activation patterns simultaneously. The original video is due to Barson et al. [BHS+20] by using genetically encoded calcium indicators to image brain activity transcranially. Simultaneous cellular-resolution two-photon calcium imaging of a local microcircuit as well as mesoscopic widefield calcium imaging of the entire cortical mantle in awake mice are used to capture the video (see [BHS+20] for more details.)

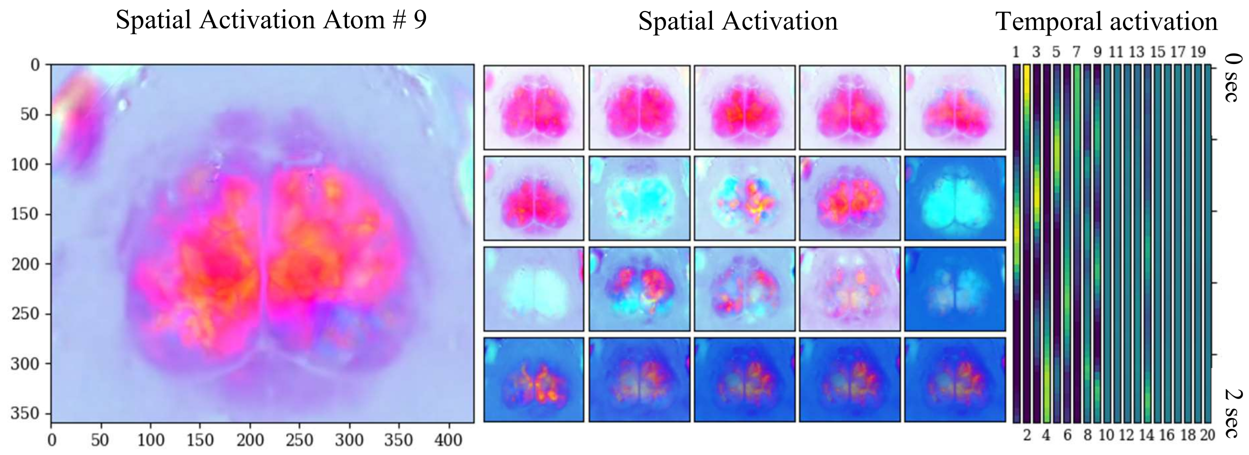

Figure 4.4: Learning 20 CP-dictionary atoms from video frames on brain activity across the mouse cortex.

The original video frame is a tensor of shape ((1501, 360, 426, 3) corresponding to the four modes (`time`, `horizontal`, `vertical`, `color`), where frames are 0.04 sec apart, which spans total 60.04 seconds. We intend to learn weakly periodic patterns of spatial and temporal activation patterns of duration at most 2 seconds. To this end, we sample 50-frame (2 sec. long) clips uniformly at random for 200 times. Each sampled tensor is reshaped into $(\texttt{time}, \texttt{space} * \texttt{color}) = (50,\ 360*426*3)$ matrix, and then sequentially fed into the online CPDL algorithm with $w_t = c'/t$, $\lambda = 2$, and $c' = 10^5$. Note that we vectorize the horizontal, vertical and color modes into a single mode before factorization in order to *preserve* the spatial structure learned in the loading matrix. Namely, for spatial activation patterns, we desire dictionary atoms of the form of Figure 4.3 (d) rather than (b) or (c).

Our algorithm learns a CP-dictionary in the space-color mode that shows spatial activation patterns and the corresponding time mode shows their temporal activation pattern, as seen in Figure 4.4. Due to the nonnegativity constraint, spatial activation atoms representing localized activation regions in the cortex are learned, while the darker ones represent the background brain shape without activation. On the other hand, the activation frequency is simultaneously learned by the temporal activation atoms shown in Figure 4.4 (right). For instance, the spacial activation atom # 9 (numbered lexicographically) activates three times in its corresponding temporal activation atom in the right, so such activation pattern has an approximate period of 2/3 sec.

4.8.4 Joint time series dictionary learning

A key advantage of online algorithms is that they are well-suited to applications in which data are arriving in real-time. We apply our algorithm to a weather dataset obtained from [Ben17]. Beginning with a $(36 \times 2998 \times 4)$ tensor where the first mode corresponds to cities, the second mode to time in hours, and the third mode to weather data such that the frontal slices correspond to temperature, humidity, pressure, and wind speed. We regularized the data by taking a moving average over up to four hours (in part to impute missing data values), and by applying a separate rescaling of each frontal slice to normalize the magnitudes of the entries.

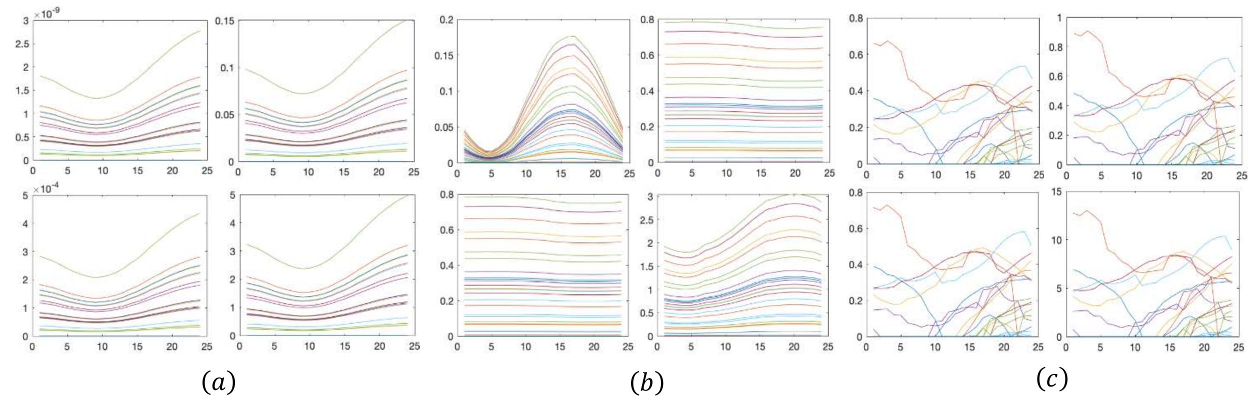

Figure 4.5: Display of one atom from three different dictionaries of 25 atoms which were obtained from online CPDL on weather data: (a) no reshaping, (b) data which was reshaped to $36 \times (24 \times 4)$, and (c) data which was reshaped to $(36 \times 24) \times 4$. For each subplot, the four subplots represent the evolution of four measurements (temperature (top left), humidity (bottom left), pressure (top right), and wind speed (bottom right)) in time for 24 hours (horizontal axis) in 36 cities (in different colors).

From this large data tensor, we sequentially extracted smaller $(36 \times 24 \times 4) = (\texttt{cities} \times \texttt{time} \times \texttt{measurements})$ tensors by dividing time into overlapping segments of length 24 hours, with overlap size 4 hours. Our experiment consisted of applying the online CPDL (Algorithm 6) to this dataset to learn a single CP-dictionary atom ($R = 1$), say $\mathbf{D} \in \mathbb{R}^{36\times24\times4}$, with three different reshaping schemes to preprocess the input tensors of shape $(36 \times 24 \times 4)$: no reshaping $(\texttt{cities} \times \texttt{time} \times \texttt{measurements})$ (Figure 4.5 (a)); concatenating time and measurements $(\texttt{cities} \times (\texttt{time} * \texttt{measurements}))$ (Figure 4.5 (b)); and concatenating cities and time $((\texttt{cities} * \texttt{time}) \times \texttt{measurements})$ (Figure 4.5 (c)). (See Subsection 4.8.1 for a discussion on reshaping and CP-dictionary learning). Roughly speaking, the single CP-dictionary atom we learn is a 3-mode tensor of shape $(36 \times 24 \times 4)$ that best approximates the evolution of four weather measurements during a randomly chosen 24-hour period from the original 2998-hour-long data subject to different constraints on the three modes depending how we reshape the input tensors.

In Figure 4.5, for each atom, the top left corner represents the first frontal slice (temperature), the bottom left the second frontal slice (humidity), the top right the third frontal slice (pressure), and the bottom right the fourth frontal slice (wind speed). The horizontal axis corresponds to time (in hours), each individual time series to a “row" in the first mode, and the vertical axis to the value of the corresponding entry in the CP-dictionary atom.

We emphasize the qualitative difference in the corresponding learned dictionaries. In the first example in Figure 4.5 (a), the CP-constraint is applied between all modes, so the single CP-dictionary atom $\mathbf{D} \in \mathbb{R}^{36\times 24\times 4}$ is given by the outer product of three marginal vectors, one for each of the three mode (analogous to Figure 4.3 (a)). Namely, let $\mathbf{D} = u_1 \otimes u_2 \otimes u_3$, where $u_1 \in \mathbb{R}^{36}$, $u_2 \in \mathbb{R}^{24}$, and $u_3 \in \mathbb{R}^4$. Then u_2 represents a 24-hour long time series, and for example, the humidity of the first city is approximated by $(u_1(1)u_3(2))u_2$. This makes the variability of the time-series across different cities and measurements restrictive, as shown in Figure 4.5 (a).

Next, in the second example in Figure 4.5 (b), the CP-constraint is applied only between cities and the other two modes combined, so the single CP-dictionary atom $\mathbf{D} \in \mathbb{R}^{36\times 24\times 4}$ is given by $\mathbf{D} = u_1 \otimes U_{23}$, where $u_1 \in \mathbb{R}^{36}$ and $U_{23} \in \mathbb{R}^{24\times 4}$. Thus, for each city, the 24-hour evolution of the four measurements need not be some scalar multiple of a single time evolution vector as before, as we can use the full 24×4 entries in U_{23} to encode such information. On the other hand, the variability of the joint 24-hour evolution of the four measurements across the cities should only be given by a scalar multiple, as we can observe in Figure 4.5 (b).

Lastly, in the third example in Figure 4.5 (c), the CP-constraint is applied only between the measurements (last mode) and the other two modes combined, so the single CP-dictionary atom $\mathbf{D} \in \mathbb{R}^{36\times 24\times 4}$ is given by $\mathbf{D} = U_{12} \otimes u_3$, where $U_{12} \in \mathbb{R}^{36\times 24}$ and $u_3 \in \mathbb{R}^4$. Thus, the 24-hour evolution of a latent measurement of the 36 cities can be encoded by the 36×24 matrix U_{12} without rank restriction. For each of the four measurements, this joint evolution pattern encoded in U_{12} is multiplied by a scalar. For example, the temperature evolution across 36 cities is modeled by $u_3(1)U_{12}$.

4.9 Background on Markov chains and MCMC

4.9.1 Markov chains

Here we give a brief account on Markov chains on countable state space (see, e.g., [LP17]). Fix a countable set Ω. A function $P : \Omega^2 \to [0, \infty)$ is called a *Markov transition matrix* if every row of P sums to 1. A sequence of Ω-valued random variables $(X_t)_{t\geq 0}$ is called a *Markov chain* with transition matrix P if for all $x_0, x_1, \dots, x_n \in \Omega$,

$$\mathbb{P}(X_n = x_n \,|\, X_{n-1} = x_{n-1}, \dots, X_0 = x_0) = \mathbb{P}(X_n = x_n \,|\, X_{n-1} = x_{n-1}) = P(x_{n-1}, x_n). \tag{4.85}$$

We say a probability distribution π on Ω a *stationary distribution* for the chain $(X_t)_{t\geq 0}$ if $\pi = \pi P$, that is,

$$\pi(x) = \sum_{y\in\Omega} \pi(y)P(y, x). \tag{4.86}$$

We say the chain $(X_t)_{t\geq 0}$ is *irreducible* if for any two states $x, y \in \Omega$ there exists an integer $t \geq 0$ such that $P^t(x, y) > 0$. For each state $x \in \Omega$, let $\mathcal{T}(x) = \{t \geq 1 \,|\, P^t(x, x) > 0\}$ be the set of times when it is possible for the chain to return to starting state x. We define the *period* of x by the greatest common divisor of $\mathcal{T}(x)$. We say the chain X_t is *aperiodic* if all states have period 1. Furthermore, the chain is said to be *positive recurrent* if there exists a state $x \in \Omega$ such that the expected return time of the chain to x started from x is finite. Then an irreducible and aperiodic Markov chain has a unique stationary distribution if and only if it is positive recurrent [LP17, Thm 21.21].

Given two probability distributions μ and ν on Ω, we define their *total variation distance* by

$$\|\mu - \nu\|_{TV} = \sup_{A\subseteq\Omega} |\mu(A) - \nu(A)|. \tag{4.87}$$

If a Markov chain $(X_t)_{t\geq 0}$ with transition matrix P starts at $x_0 \in \Omega$, then by (4.85), the distribution of X_t is given by $P^t(x_0, \cdot)$. If the chain is irreducible and aperiodic with stationary distribution π, then the convergence theorem (see, e.g., [LP17, Thm 21.14]) asserts that the distribution of X_t converges to π in total variation distance: As $t \to \infty$,

$$\sup_{x_0 \in \Omega} \|P^t(x_0, \cdot) - \pi\|_{TV} \to 0. \tag{4.88}$$

See [MT12, Thm 13.3.3] for a similar convergence result for the general state space chains. When Ω is finite, then the above convergence is exponential in t (see., e.g., [LP17, Thm 4.9])). Namely, there exists constants $\lambda \in (0,1)$ and $C > 0$ such that for all $t \geq 0$,

$$\max_{x_0 \in \Omega} \|P^t(x_0, \cdot) - \pi\|_{TV} \leq C\lambda^t. \tag{4.89}$$

Markov chain mixing refers to the fact that, when the above convergence theorems hold, then one can approximate the distribution of X_t by the stationary distribution π.

Remark 1. Our main convergence result in Theorem 3.1 assumes that the underlying Markov chain Y_t is irreducible, aperiodic, and defined on a finite state space Ω, as stated in (A1). This can be relaxed to countable state space Markov chains. Namely, Theorem 3.1 holds if we replace (A1) by

(A1)'. *The observed data tensors $\mathbf{X}_t$ are given by $\mathbf{X}_t = \varphi(Y_t)$, where Y_t is an irreducible, aperiodic, and positive recurrent Markov on a countable and compact state space Ω and $\varphi : \Omega \to \mathbb{R}^{d \times n}$ is a bounded function. Furthermore, there exist constants $\beta \in (3/4, 1]$ and $\gamma > 2(1-\beta)$ such that*

$$w_t = O(t^{-\beta}), \qquad \sup_{\mathbf{y} \in \Omega} \|P^t(\mathbf{y}, \cdot) - \pi\|_{TV} = O(t^{-\gamma}), \tag{4.90}$$

where P and π denote the transition matrix and unique stationary distribution of the chain Y_t.

Note that the polynomial mixing condition in (A1)' is automatically satisfied when Ω is finite due to (4.89). Polynomial mixing rate is available in most MCMC algorithms used in practice.

4.9.2 Markov chain Monte Carlo Sampling

Suppose we have a finite sample space Ω and probability distribution π on it. We would like to sample a random element $\omega \in \Omega$ according to the distribution π. *Markov chain Monte Carlo (MCMC)* is a sampling algorithm that leverages the properties of Markov chains we mentioned in Subsection 4.9.1. Namely, suppose that we have found a Markov chain $(X_t)_{t\geq 0}$ on state space Ω that is irreducible, aperiodic[4], and has π as its unique stationary distribution. Denote its transition matrix as P. Then by (4.89), for any $\varepsilon > 0$, one can find a constant $\tau = \tau(\varepsilon) = O(\log \varepsilon^{-1})$ such that the conditional distribution of $X_{t+\tau}$ given X_t is within total variation distance ε from π regardless of the distribution of X_t. Recall such $\tau = \tau(\varepsilon)$ is called the *mixing time* of the Markov chain $(X_t)_{t\geq 1}$. Then if one samples a long Markov chain trajectory $(X_t)_{t\geq 1}$, the subsequence $(X_{k\tau})_{k\geq 1}$ gives approximate i.i.d. samples from π.

We can further compute how far the thinned sequence $(X_{k\tau})_{k\geq 1}$ is away from being independent. Namely, observe that for any two nonempty subsets $A, B \subseteq \Omega$,

$$
\begin{aligned}
&|\mathbb{P}(X_{k\tau} \in A,\, X_\tau \in B) - \mathbb{P}(X_{k\tau} \in A)\mathbb{P}(X_\tau \in B)| && (4.91)\\
&\quad = |\mathbb{P}(X_{k\tau} \in A) - \mathbb{P}(X_{k\tau} \in A \,|\, X_\tau \in B)|\, |\mathbb{P}(X_\tau \in B)| && (4.92)\\
&\quad \leq |\mathbb{P}(X_{k\tau} \in A) - \mathbb{P}(X_{k\tau} \in A \,|\, X_\tau \in B)| && (4.93)\\
&\quad \leq |\mathbb{P}(X_{k\tau} \in A) - \pi(A)| + |\pi(A) - \mathbb{P}(X_{k\tau} \in A \,|\, X_\tau \in B)| \leq \lambda^{k\tau} + \lambda^{(k-1)\tau}. && (4.94)
\end{aligned}
$$

Hence the correlation between $X_{k\tau}$ and X_τ is $O(\lambda^{(k-1)\tau})$.

For the lower bound, let us assume that X_t is *reversible* with respect to π, that is, $\pi(x)P(x,y) = \pi(y)P(y,x)$ for $x, y \in \Omega$ (e.g., random walk on graphs). Then $\tau(\varepsilon) = \Theta(\log \varepsilon^{-1})$ (see [LP17, Thm. 12.5]), which yields $\sup_{x\in\Omega}\|P^t(x,\cdot) - \pi\|_{TV} = \Theta(\lambda^t)$. Also, $\mathbb{P}(X_\tau \in B) > \delta > 0$ for some $\delta > 0$ whenever τ is large enough under the hypothesis. Hence

$$|\mathbb{P}(X_{k\tau} \in A,\, X_\tau \in B) - \mathbb{P}(X_{k\tau} \in A)\mathbb{P}(X_\tau \in B)| \qquad (4.95)$$

[4]Aperiodicity can be easily obtained by making a given Markov chain lazy, that is, adding a small probability ε of staying at the current state. Note that this is the same as replacing the transition matrix P by $P_\varepsilon := (1-\varepsilon)P + \varepsilon I$ for some $\varepsilon > 0$. This 'lazyfication' does not change stationary distributions, as $\pi P = \pi$ implies $\pi P_\varepsilon = \pi$.

$$\geq \delta^{-1} \left| \mathbb{P}(X_{k\tau} \in A) - \mathbb{P}(X_{k\tau} \in A \,|\, X_\tau \in B) \right| \tag{4.96}$$

$$\geq \Big| \left|\mathbb{P}(X_{k\tau} \in A) - \pi(A)\right| - \left|\pi(A) - \mathbb{P}(X_{k\tau} \in A \,|\, X_\tau \in B)\right| \Big| \geq c\lambda^{(k-1)\tau} \tag{4.97}$$

for some constant $c > 0$. Hence the correlation between $X_{k\tau}$ and X_τ is $\Theta(\lambda^{(k-1)\tau})$. In particular, the correlation between two consecutive terms in $(X_{k\tau})_{k\geq 1}$ is of $\Theta(\lambda^\tau) = \Theta(\varepsilon)$. Thus, we can make the thinned sequence $(X_{k\tau})_{k\geq 1}$ arbitrarily close to being i.i.d. for π, but if X_t is reversible with respect to π, the correlation within the thinned sequence is always nonzero.

In practice, one may not know how to estimate the mixing time $\tau = \tau(\varepsilon)$. In order to empirically assess that the Markov chain has mixed to the stationary distribution, multiple chains are run for diverse mode exploration, and their empirical distribution is compared to the stationary distribution (a.k.a. multistart heuristic). See [BGJM11] for more details on MCMC sampling.

4.10 Auxiliary lemmas

Lemma 4 (Convex Surrogate for Functions with Lipschitz Gradient). *Let $f : \mathbb{R}^p \to \mathbb{R}$ be differentiable and ∇f be L-Lipschitz continuous. Then for each $\theta, \theta' \in \mathbb{R}^p$,*

$$\left| f(\theta') - f(\theta) - \nabla f(\theta)^T(\theta' - \theta) \right| \leq \frac{L}{2}\|\theta - \theta'\|_F^2. \tag{4.98}$$

Proof. This is a classical Lemma. See [Nes98, Lem 1.2.3]. □

For each $\mathcal{X} \in \mathbb{R}^{I_1\times\cdots\times I_n\times b}$ and $\mathcal{D} \in \mathbb{R}^{I_1\times R} \times \dots \mathbb{R}^{I_n\times R}$, denote

$$H^\star(\mathcal{X}, \mathcal{D}) \in \operatorname*{arg\,min}_{H\in\mathcal{C}^{\mathrm{code}}} \ \ell(\mathcal{X}, \mathcal{D}, H). \tag{4.99}$$

Recall Assumption (A1)'. For each subset S of a Euclidean space, denote $\|S\|_F = \sup_{x\in S}\|x\|_F$. The following boundedness results for the codes H_t and aggregate tensors $A_t, \mathbf{B}_t$ are easy to derive.

Lemma 5. *Assume (A1)' and (A2). Then the following hold:*

(i) *For all $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_n \times b}$ and $\mathcal{D} \in \mathcal{C}^{\text{dict}}$,*

$$\|H^{\star}(\mathcal{X}, \mathcal{D})\|_F^2 \leq \lambda^{-2} \|\varphi(\Omega)\|_F^4 < \infty. \tag{4.100}$$

(ii) *For any sequences $(\mathcal{X}_t)_{t\geq 1}$ in $\mathbb{R}^{I_1 \times \cdots \times I_n \times b}$ and $(\mathcal{D}_t)_{t\geq 1}$ in $\mathcal{C}$, define A_t and $\mathbf{B}_t$ recursively as in Algorithm 6. Then for all $t \geq 1$, we have*

$$\|A_t\|_F \leq \lambda^{-2} \|\varphi(\Omega)\|_F^4, \qquad \|\mathbf{B}_t\|_F \leq \lambda^{-1} \|\varphi(\Omega)\|_F^3. \tag{4.101}$$

Proof. Omitted. See [LNB20, Prop. 7.2]. □

The following lemma shows Lipschitz continuity of the loss function $\ell(\varphi(\cdot), \cdot)$ defined in (4.10). Since Ω and $\mathcal{C}^{\text{code}}$ are both compact, this also implies that $\mathcal{D} \mapsto \hat{f}_t(\mathcal{D})$ and $\mathcal{D} \mapsto f_t(\mathcal{D})$ are L-Lipschitz for some $L > 0$ uniformly for all $t \geq 0$.

Lemma 6. *Suppose (A1)' and (A2) hold, and let $M = 2\|\varphi(\Omega)\|_F + 2\|\mathcal{C}^{\text{dict}}\|_F \|\varphi(\Omega)\|_F^2 / \lambda$. Then for each $Y_1, Y_2 \in \Omega$ and $\mathcal{D}_1, \mathcal{D}_2 \in \mathcal{C}^{\text{dict}}$,*

$$|\ell(\varphi(Y_1), \mathcal{D}_1) - \ell(\varphi(Y_2), \mathcal{D}_2)| \leq M \left(\|Y_1 - Y_2\|_F + \lambda^{-1} \|\varphi(\Omega)\|_F \|\mathcal{D}_1 - \mathcal{D}_2\|_F \right). \tag{4.102}$$

Proof. Omitted. See [LNB20, Prop. 7.3]. □

The following deterministic statement on converging sequences is due to [MBPS10].

Lemma 7. *Let $(a_n)_{n\geq 0}$ and $(b_n)_{\geq 0}$ be non-negative real sequences such that*

$$\sum_{n=0}^{\infty} a_n = \infty, \qquad \sum_{n=0}^{\infty} a_n b_n < \infty, \qquad |b_{n+1} - b_n| = O(a_n). \tag{4.103}$$

Then $\lim_{n\to\infty} b_n = 0$.

Proof. Omitted. See [Mai13b, Lem. A.5]. □

Lemma 8. *Under the assumptions (A1)' and (A2),*

$$\mathbb{E}\left[\sup_{W \in \mathcal{C}^{\text{dict}}} \sqrt{t} \left| f(\mathcal{D}) - \frac{1}{t} \sum_{s=1}^{t} \ell(\mathcal{X}_s, \mathcal{D}) \right| \right] = O(1). \tag{4.104}$$

Furthermore, $\sup_{W\in\mathcal{C}} \left| f(\mathcal{D}) - \frac{1}{t} \sum_{s=1}^{t} \ell(\mathcal{X}_s, \mathcal{D}) \right| \to 0$ almost surely as $t \to \infty$.

Proof. Omitted. See [LNB20, Lem. 7.8]. □

In the following lemma, we generalize the uniform convergence results in Lemma 8 for general weights $w_t \in (0,1)$ (not only for the 'balanced weights' $w_t = 1/t$). The original lemma is due to Mairal [Mai13b, Lem B.7], which originally extended the uniform convergence result to weighted empirical loss functions with respect to i.i.d. input signals. A similar argument gives the corresponding result in our Markovian case (A1)', which was also used in [LNB20]. In Lemma 8 below, we also generalize the statement for the weights w_t satisfy the required monotonicity property $w_{t+1}^{-1} - w_t^{-1} \le 1$ only asymptotically. See Remark 2 for more discussion.

Lemma 9. *Suppose (A1)'-(A2) hold, and assume that there exist an integer $T \ge 1$ such that $w_{t+1}^{-1} - w_t^{-1} \le 1$ for all $t \ge T$. Also assume that there are some constants $c > 0$ and $\gamma \in (0,1]$ such that $w_t \ge ct^{-\gamma}$ for all $t \ge 1$. Further assume that if $T \ge 1$ and $\gamma = 1$, then $c \ge 1/2$. Then there exists a constant $C = C(T) > 0$ such that*

$$\mathbb{E}\left[\sup_{W\in\mathcal{C}^{\text{dict}}} |f(\mathcal{D}) - f_t(\mathcal{D})|\right] \le C w_t \sqrt{t}. \tag{4.105}$$

Furthermore, if $\sum_{t=1}^{\infty} w_t = \infty$, $\sum_{t=1}^{\infty} w_t^2\sqrt{t} < \infty$, then $\sup_{\mathcal{D}\in\mathcal{C}^{\text{dict}}} |f(\mathcal{D}) - f_t(\mathcal{D})| \to 0$ almost surely as $t \to \infty$.

Proof. Fix $t \in \mathbb{N}$. Recall the weighted empirical loss $f_t(\mathcal{D})$ defined recursively using the weights $(w_s)_{s\ge 0}$ in (4.11). For each $0 \le s \le t$, denote $w_s^t = w_s \prod_{j=s}^{t}(1-w_j)$ and set $w_0^t = 0$. Then for each $t \in \mathbb{N}$, we can write $f_t(\mathcal{D}) = \sum_{s=1}^{t} \ell(X_s, W) w_s^t$. Moreover, note that $w_1^t, \dots, w_t^t > 0$ and $w_1^t + \cdots + w_t^t = 1$. Define $F_i(\mathcal{D}) = (t-i+1)^{-1} \sum_{j=1}^{t} \ell(X_i, W)$ for each $1 \le i \le t$. By Lemma 8, there exists a constant $c_1 > 0$ such that

$$\mathbb{E}\left[\sup_{W\in\mathcal{C}} |F_i(\mathcal{D}) - f(\mathcal{D})|\right] \le \frac{c_1}{\sqrt{t-i+1}} \tag{4.106}$$

for all $t \ge 1$ and $1 \le i \le t$. Noting that $[w_1^t, \dots, w_t^t]$ is a probability distribution on $\{1, \dots, t\}$, a simple calculation shows the following important identity

$$f_t - f = \sum_{i=1}^{t} (w_i^t - w_{i-1}^t)(t-i+1)(F_i - f), \tag{4.107}$$

with the convention of $w_0^t = 0$. Also, suppose $T \geq 1$ is such that $w_k^{-1} - w_{k-1}^{-1} \leq 1$ for $k \geq T$. Note that for $i \geq 2$, $w_{i-1}^t \leq w_i^t$ if and only if $w_{i-1}(1 - w_i) \leq w_i$ if and only if $w_i^{-1} - w_{i-1}^{-1} \leq 1$. Hence for each $n > T$ and $k \geq T$, we have $w_k^t \leq w_{k+1}^t \leq \cdots \leq w_t^t = w_t$. Then observe that

$$\begin{aligned}
\mathbb{E}\left[\sup_{\mathcal{D}\in\mathcal{C}^{\text{dict}}} |f_t(\mathcal{D}) - \bar{\psi}(\mathcal{D})|\right] &\leq \mathbb{E}\left[\sum_{i=1}^{t} |w_i^t - w_{i-1}^t|(t-i+1) \sup_{W\in\mathcal{C}^{\text{dict}}} |f_i(\mathcal{D}) - f(\mathcal{D})|\right] && (4.108)\\
&= \sum_{i=1}^{t} |w_i^t - w_{i-1}^t|(t-i+1)\, \mathbb{E}\left[\sup_{\mathcal{D}\in\mathcal{C}^{\text{dict}}} |f_i(\mathcal{D}) - f(\mathcal{D})|\right] && (4.109)\\
&\leq \sum_{i=1}^{t} |w_i^t - w_{i-1}^t| c_1 \sqrt{t-i+1} && (4.110)\\
&\leq c_1\sqrt{t}\left(\sum_{i=1}^{T} |w_i^t - w_{i-1}^t| + \sum_{i=T}^{t} (\hat{w}_i^t - \hat{w}_{i-1}^t)\right) && (4.111)\\
&\leq c_1\sqrt{t}\left(w_t + \sum_{i=1}^{T} w_i^t\right). && (4.112)
\end{aligned}$$

By using Lemma 10, we have $\sum_{i=1}^T w_i^t = O(1/n)$. Furthermore, since $w_{t+1}^{-1} - w_t^{-1} \leq 1$ for all $t \geq T$, we deduce $w_t^{-1} - w_T^{-1} \leq t - T$ for all $t \geq T$, so $w_t \geq \frac{1}{t-T+w_T^{-1}}$. Thus for some constant $c > 0$, we have $w_t \geq \frac{1}{t+c}$ for all $t \geq T$. Thus the last displayed expression above is of $O(w_t\sqrt{t})$. This shows (4.105). We can show the part by using Lemma 7 in Appendix 4.10, following the argument in the proof of [Mai13b, Lem. B7]. See the reference for more details. □

The following lemma was used in the proof of Lemma 9.

Lemma 10. *Fix a sequence $(w_n)_{n\geq 1}$ of numbers in $(0,1]$. Denote $w_k^n := w_k \prod_{i=k+1}^n (1-w_i)$ for $1 \leq k \leq n$. Suppose $w_n^{-1} - w_{n-1}^{-1} \leq 1$ for all sufficiently large $n \geq 1$. Fix $T \geq 1$. Then for all $n \geq T$,*

$$\sum_{i=1}^{T} w_i^n = O(1/n). \tag{4.113}$$

Proof. Suppose $w_n^{-1} - w_{n-1}^{-1} \leq 1$ for all $n \geq N$ for some $N \geq 1$. It follows that $w_n^{-1} - w_N^{-1} \leq n - N$, so $w_n \geq \frac{1}{n-N+w_N^{-1}}$. Hence for some constant $c > 0$, $w_n \geq \frac{1}{n+c}$ for all $n \geq N$. Denote

$a \vee b = \max(a,b)$. Then note that

$$w_k^n = w_k \exp\left(\sum_{i=k+1}^{n} \log(1-w_i)\right) \le \exp\left(-\sum_{i=k+1}^{n} w_i\right) \tag{4.114}$$

$$\le \exp\left(-\int_{N\vee(k+1)}^{n} \frac{1}{x+c}\,dx\right) = \frac{[N\vee(k+1)]+c}{n+c}, \tag{4.115}$$

where the second inequality uses $w_k \le 1$ and the following inequality uses $\log(1-a) \le -a$ for $a < 1$. Hence for each fixed $1 \le T \le n$, we have

$$\sum_{k=1}^{T} w_k^n \le T\left((N\vee(T+1))+c\right)\frac{1}{n+c}. \tag{4.116}$$

This shows the assertion. □

Remark 2. In the original statement of [Mai13b, Lem B.7], the assumption that $w_{t+1}^{-1} - w_t^{-1} \le 1$ for sufficiently large t was not used, and it seems that the argument in [Mai13b] needs this assumption. To give more detail, the argument begins with writing the empirical loss $f_t(\cdot) = \sum_{k=1}^{t} w_k^t\,\ell(\mathcal{X}_k,\cdot)$, where $w_k^t := w_k(1-w_{k-1})\cdots(1-w_t)$, and proceeds with assuming the monotonicity $w_1^t \le \cdots \le w_t^t$, which is equivalent to $w_k \ge w_{k-1}(1-w_k)$ for $2 \le k \le t$. In turn, this is equivalent to $w_k^{-1} - w_{k-1}^{-1} \le 1$ for $2 \le k \le t$. Note that this condition implies $w_k^{-1} \le (k-1) + w_1^{-1}$, or $w_k \ge \frac{1}{k-1+w_1^{-1}}$, where $w_1 \in [0,1]$ is a fixed constant. This means that, asymptotically, w_k cannot decay faster than the balanced weight $1/k$, which gives $w_k^t \equiv 1/t$ for $k \in \{1,\dots,t\}$. Note that we proved Lemma 10 with requiring $w_{t+1}^{-1} - w_t^{-1} \le 1$ for all sufficiently large t.

Next, we will argue that (A3') implies (A3). It is clear that if the sequence $w_t \in (0,1]$ satisfies (A3'), then $\sum_{t=1}^{\infty} w_t = \infty$ and $\sum_{t=1}^{\infty} w_t^2\sqrt{t} < \infty$. So it remains to verify $w_t^{-1} - w_{t-1}^{-1} \le 1$ for sufficiently large t. Suppose $w_t = \Theta(t^{-\beta}(\log t)^{-\delta})$ for some $\beta \in [0,1]$ and $\delta \ge 0$. Let $c_1, c_2 > 0$ be constants such that $w_t t^{\beta}(\log t)^{\delta} \in [c_1, c_2]$ for all $t \ge 1$. Then by the mean value theorem,

$$w_{t+1}^{-1} - w_t^{-1} \le c_2\left((t+1)^{\beta}(\log(t+1))^{\delta} - t^{\beta}(\log t)^{\delta}\right) \tag{4.117}$$

$$\leq c_2 \sup_{t \leq s \leq t+1} \left(\beta s^{\beta-1} (\log s)^{\delta} + \delta s^{\beta-1} (\log s)^{\delta-1} \right) \tag{4.118}$$

$$\leq c_2 \sup_{t \leq s \leq t+1} s^{\beta-1} (\log s)^{\delta-1} \left((\log s) + \delta \right). \tag{4.119}$$

Since $t \geq 1$, the last expression is of $o(1)$ if $\beta < 1$. Otherwise, $w_t = t^{-1}$ for $t \geq 1$ by (A3'). Then $w_{t+1}^{-1} - w_t^{-1} \equiv 1$ for all $t \geq 1$.

4.11 Bounded memory implementation of Algorithm 6

In this section, we introduce an alternative implementation of Algorithm 6 that uses bounded memory that is independent of the number T of minibatches of data tensors being processed. This will be done by replacing the step for computing the surrogate loss function $\hat{f}_t$ with computing two 'aggregate tensors' based on our deterministic analysis in Proposition 2. The total amount of information fed in to the algorithm is $O(T \prod_{i=1}^{n} I_n)$ and $T \to \infty$, whereas Algorithm 7 stores only $O(R \prod_{i=1}^{n} I_n)$ (recall that R is the number of dictionary atoms to be learned and T is the number of minibatches of data tensors that have arrived). This is an inherent memory efficiency of online algorithms against non-online algorithms (see, e.g., [MBPS10]).

We describe how Algorithm 7 is derived and why it is equivalent to Algorithm 6. By the time that the new data tensor $\mathcal{X}_t$ arrives, the algorithm have computed previous loading matrices $U_{t-1}^{(1)}, \dots, U_{t-1}^{(n)}$ and two aggregate tensors $A_{t-1} \in \mathbb{R}^{R \times R}$ and $\mathbf{B}_{t-1} \in \mathbb{R}^{I_1 \times \cdots \times I_n \times R}$. Then one computes the code matrix $H_t \in \mathcal{C}^{\text{code}} \subseteq \mathbb{R}^{R \times b}$ by solving the convex optimization problem in (4.120), and then updates the aggregate tensors $A_t \leftarrow A_{t-1}$ and $\mathbf{B}_t \leftarrow \mathbf{B}_{t-1}$. In order to perform the block coordinate descent to update the loading matrices $U_i^{(t)}$ in eq. (4.18) of Algorithm 6, we appropriately recompute intermediate aggregate matrices $\overline{A}_i$ and $\overline{B}_i$ using Algorithm 8 so that we are correctly minimizing the surrogate loss function $\hat{f}_t$ in (4.21) marginally according to Proposition 2 **(ii)**.

Algorithm 7 Online CP-Dictionary Learning (Bounded Memory Implementation)

```
0: Input: $(\mathcal{X}_t)_{1\le t\le T}$ (minibatches of data tensors in $\mathbb{R}_{\ge 0}^{I_1\times\cdots\times I_n\times b}$); $[U_0^{(1)},\dots,U_0^{(n)}] \in \mathbb{R}_{\ge 0}^{I_1\times R}\times\cdots\times\mathbb{R}_{\ge 0}^{I_n\times R}$ (initial loading matrices); $c'>0$ (search radius constant);
0: Constraints: $\mathcal{C}^{(i)}\subseteq\mathbb{R}^{I_i\times R}$, $1\le i\le n$, $\mathcal{C}^{\text{code}}\subseteq\mathbb{R}^{R\times b}$ (e.g., nonnegativity constraints)
0: Parameters: $R\in\mathbb{N}$ (# of dictionary atoms); $\lambda\ge 0$ ($\ell_1$-regularizer); $(w_t)_{t\ge1}$ (weights in $(0,1]$);
0:   Initialize aggregate tensors $A_0\in\mathbb{R}^{R\times R}$, $\mathbf{B}_0\in\mathbb{R}^{I_1\times\cdots\times I_n\times R}$;
0:   For $t=1,\dots,T$ do:
0:     Coding: Compute the optimal code matrix
           $$H_t \leftarrow \operatorname*{arg\,min}_{H\in\mathcal{C}^{\text{code}}\subseteq\mathbb{R}^{R\times b}} \ell(\mathcal{X}_t, U_{t-1}^{(1)},\dots,U_{t-1}^{(n)}, H); \quad \text{(using Algorithm 9)} \quad (4.120)$$
0:     Update aggregate tensors:
           $$A_t \leftarrow (1-w_t)A_{t-1} + w_t H_t H_t^T \in \mathbb{R}^{R\times R}; \quad (4.121)$$
           $$\mathbf{B}_t \leftarrow (1-w_t)\mathbf{B}_{t-1} + w_t(\mathcal{X}_t \times_{n+1} H_t^T) \in \mathbb{R}^{I_1\times\cdots\times I_n\times R}; \quad (4.122)$$
0:     Update dictionary:
0:       For $i=1,\dots,n$ do:
0:         $\overline{A}_{t;i}\in\mathbb{R}^{R\times R}$, $\overline{B}_{t;i}\in\mathbb{R}^{I_i\times R}$
             $\leftarrow$ Algorithm 8 with input $A_t, \mathbf{B}_t, U_t^{(1)},\dots,U_t^{(i-1)}, U_t^{(i)}, U_{t-1}^{(i+1)},\dots,U_{t-1}^{(n)}, i$;
0:         $\mathcal{C}_t^{(i)} \leftarrow \left\{ U\in\mathcal{C}^{(i)} \,\middle|\, \|U-U_{t-1}^{(i)}\|_F \le c'w_t \right\}$;   (Restrict the search radius by $w_t$)
0:         $U_t^{(i)} \leftarrow \arg\min_{U\in\mathcal{C}_t^{(i)}} \left[\operatorname{tr}(U\overline{A}_{t;i}U^T) - 2\operatorname{tr}(U\,\overline{B}_{t;i}^T)\right]$;   (Using Algorithm 10)
0:       End for
0:   End for
0: Return: $[U_T^{(1)},\dots,U_T^{(n)}]\in\mathcal{C}^{(1)}\times\cdots\times\mathcal{C}^{(n)}$; =0
```

Algorithm 8 Intermediate Aggregation

```
0: Input: $A \in \mathbb{R}^{R\times R}$, $B \in \mathbb{R}^{I_1\times\cdots\times I_n\times R}$, $[U_1,\dots,U_n] \in \mathbb{R}^{I_1\times R}\times\dots\times\mathbb{R}^{I_n\times R}$, $1 \le j \le n$
0: Do:
       $$\overline{A}_i = A \odot U_1^T U_1 \odot \dots \odot U_{i-1}^T U_{i-1} \odot U_{i+1}^T U_{i+1} \odot \dots \odot U_n^T U_n \in \mathbb{R}^{R\times R} \qquad (4.123)$$
0:     For $r = 1,\dots,R$ do:
           $$B(,r) := \text{mode-}(n+1) \text{ slice of } B \text{ at coordinate } r \qquad (4.124)$$
           $$b_{i;r} = B(,r) \times_1 U_1(:,r) \times_2 \cdots \times_{i-1} U_{i-1}(:,r) \times_{i+1} U_{i+1}(:,r) \times_{i+2} \cdots \times_n U_n(:,r) \in \mathbb{R}^{I_i} \qquad (4.125)$$
           $$\overline{B}_{t;i} = I_i \times R \text{ matrix whose } r\text{th column is } b_{i;r} \qquad (4.126)$$
0:     End for
0: Return:
       $$\overline{A}_i = \overline{A}_i(A, U_1,\dots,U_{i-1},U_{i+1},\dots,U_n)$$
       $$\overline{B}_i = \overline{B}_i(B, U_1,\dots,U_{i-1},U_{i+1},\dots,U_n)$$
   =0
```

4.12 Auxiliary Algorithms

In this section, we give auxiliary algorithms that are used to solve convex sub-problems in coding and loading matrix updates for the main algorithm (Algorithm 1 for online CPDL). We denote by Π_S the projection operator onto the given subset S defined on the respective ambient space. For each matrix A, denote by $[A]_{\bullet i}$ (resp., $[A]_{i\bullet}$) the ith column (resp., row) of A.

Algorithm 9 Coding

```
0: Input: $X \in \mathbb{R}^{M\times b}$: data matrix, $W \in \mathbb{R}^{M\times R}$: dictionary matrix
0:     $\lambda \geq 0$: sparsity regularizer
0:     $\mathcal{C}^{\text{code}} \subseteq \mathbb{R}^{R\times b}$: constraint set of codes
0: Repeat until convergence:
0:     Do
           $$H \leftarrow \Pi_{\mathcal{C}^{\text{code}}}\left(H - \frac{1}{\text{tr}(W^TW)}(W^TWH - W^TX + \lambda J)\right), \quad (4.127)$$
       where $J \subseteq \mathbb{R}^{R\times b}$ is all ones matrix.
0: Return $H \in \mathcal{C}^{\text{code}} \subseteq \mathbb{R}^{R\times b}$ =0
```

Algorithm 10 Loading matrix update

```
0: Variables:
0:     $U \in \mathcal{C}^{(i)} \subseteq \mathbb{R}^{I_i\times R}$: previous $j$th loading matrix
0:     $(\overline{A}_i, \overline{B}_{t;j}) \in \mathbb{R}^{R\times R} \times \mathbb{R}^{R\times(I_1\ldots I_n)}$: intermediate loading matrices computed previously
0: Repeat until convergence:
0:     For $i = 1$ to $R$:
           $$[U]_{\bullet i} \leftarrow \Pi_{\mathcal{C}^{(i)}}\left([U]_{\bullet i} - \frac{1}{[\overline{A}_t]_{ii} + 1}(U[\overline{A}_i]_{\bullet i} - [\overline{B}_{t;j}^T]_{\bullet i})\right) \quad (4.128)$$
0: Return $U \in \mathcal{C}^{(i)} \subseteq \mathbb{R}^{I_i\times R}$ =0
```

Algorithm 11 Alternating Least Squares for NCPD

```
0: Input: $\mathbf{X} \in \mathbb{R}_{\geq 0}^{I_1 \times \cdots \times I_m}$ (data tensor); $R \in \mathbb{N}$ (rank parameter); $\mathcal{D}_0 = (U_0^{(1)}, \cdots, U_0^{(m)}) \in$
   $\mathbb{R}_{\geq 0}^{I_1 \times R} \times \cdots \times \mathbb{R}_{\geq 0}^{I_m \times R}$ (initial loading matrices); $N$ (number of iterations);
0:   for $n = 1, \ldots, N$ do:
0:     Update loading matrices $\mathcal{D}_n = [U_n^{(1)}, \cdots, U_n^{(m)}]$ by
0:       For $i = 1, \cdots, m$ do:
           $\mathbf{A} \leftarrow \texttt{Out}(U_{n-1}^{(1)}, \ldots, U_{n-1}^{(i-1)}, U_{n-1}^{(i+1)}, \ldots, U_{n-1}^{(m-1)})^{(m)} \in \mathbb{R}^{(I_1 \times \cdots \times I_{i-1} \times I_{i+1} \times \cdots \times I_m) \times R}$   (4.129)
           $B \leftarrow \text{unfold}(\mathbf{A}, m) \in \mathbb{R}^{(I_1 \cdots I_{i-1} I_{i+1} \cdots I_m) \times R}$   (4.130)
           $U_n^{(i)} \in \arg\min_{U \in \mathbb{R}_{\geq 0}^{I_i}} \|\text{unfold}(\mathbf{X}, i) - B(U^{(i)})^T\|^2$   (4.131)
           $\triangleright$ (unfold$(\cdot, i)$ denotes the mode-$i$ tensor unfolding (see [KB09]))   (4.132)
0:       end for
0:   end for
0: output: $\mathcal{D}_N$ =0
```

Algorithm 12 Multiplicative Update for NCPD

```
0: Input: $\mathbf{X} \in \mathbb{R}_{\geq 0}^{I_1 \times \cdots \times I_m}$ (data tensor); $R \in \mathbb{N}$ (rank parameter); $\mathcal{D}_0 = (U_0^{(1)}, \cdots, U_0^{(m)}) \in$
   $\mathbb{R}_{\geq 0}^{I_1 \times R} \times \cdots \times \mathbb{R}_{\geq 0}^{I_m \times R}$ (initial loading matrices); $N$ (number of iterations);
0:   for $n = 1, \ldots, N$ do:
0:     Update loading matrices $\mathcal{D}_n = [U_n^{(1)}, \cdots, U_n^{(m)}]$ by
0:       For $i = 1, \cdots, m$ do:
           $\mathbf{A} \leftarrow \texttt{Out}(U_{n-1}^{(1)}, \ldots, U_{n-1}^{(i-1)}, U_{n-1}^{(i+1)}, \ldots, U_{n-1}^{(m-1)})^{(m)} \in \mathbb{R}^{(I_1 \times \cdots \times I_{i-1} \times I_{i+1} \times \cdots \times I_m) \times R}$   (4.133)
           $B \leftarrow \mathrm{unfold}(\mathbf{A}, m) \in \mathbb{R}^{(I_1 \cdots I_{i-1} I_{i+1} \cdots I_m) \times R}$   (4.134)
           $U_n^{(i)} \leftarrow U_{n-1}^{(i)} \odot (\mathrm{unfold}(\mathbf{X}, i))^T B \oslash (U B^T B)$   (4.135)
           ▷ (unfold$(\cdot, i)$ denotes the mode-$i$ tensor unfolding (see [KB09]))   (4.136)
           ▷ ($\odot$ and $\oslash$ denote entrywise product and division)   (4.137)
0:       end for
0:   end for
0: output: $\mathcal{D}_N$ =0
```

CHAPTER 5

Conclusion

This final chapter consists of a summary of the work that has been accomplished in this thesis, as well as an extensive collection of future research directions.

5.1 Reflections

The enormity and multi-modality of modern data have made the task of extracting meaningful insights extremely difficult. More often than not, a dataset of interest will consist of data points of such high dimension, and consisting of so many samples, that it is impossible to even store the dataset in memory. Additionally, many datasets are multi-modal, being composed of quite distinct data sources such as text, image, audio, and video. To address the challenges presented by such data, we have introduced our algorithm Online Nonnegative CP-Dictionary Learning (OCPDL). Being an online algorithm, OCPDL is adept at learning latent patterns in large datasets by processing data in batches and aggregating information derived from each batch so as to ensure that previously processed batches influence future dictionary updates, without the need to store said batches. Additionally, the tensorial character of our learned dictionaries ensures that one can effectively handle multi-modal data. In fact, given that a CP-dictionary is constructed in terms of individual factor matrices, each corresponding to a different modality, our algorithm is capable of effectively separating the different modalities.

Throughout this thesis, we have presented applications of online nonnegative matrix factorization (ONMF) and our OCPDL which highlight the ability of online matrix and tensor dictionary learning algorithms to uncover latent structure in data. Our principle method of encoding a given data point using our learned dictionary and computing the normalized code weights allows us to determine which dictionary atoms contributed to the given data point by determining the weights with the greatest magnitude.

On the theoretical side, we have established convergence guarantees for OCPDL in the setting of Markovian data. This is a considerable extension of the original work of [MBPS10], which proved analogous guarantees in the i.i.d. matrix setting, as opposed to our vastly more encompassing Markovian, tensor setting.

5.2 Future Work

On the theoretical end, much work remains. Although we were able to establish convergence guarantees for OCPDL which mirror those obtained in the ONMF with i.i.d. data first proved in [MBPS10], even in this simplified setting it is unknown whether the limiting dictionary achieves a local maximum or minimum of the expected loss; rather, it is only known that one has convergence to the set of critical points of said loss.

An additional direction would be to develop and analyze hierarchical variants of our algorithm. Such variants, like their matrix counterparts, are useful for uncovering additional structure in data, with higher-level factors corresponding to “coarse-scale" features, lower levels corresponding to “fine-scale" features. These might be compared to hierarchical features obtained by training a deep neural network. In general, such networks obtain feature vectors which are dense and thus difficult to interpret, especially when compared to the generally sparse features obtained from nonnegative matrix and tensor dictionary learning algorithms.

Our algorithm is quite general purpose; although we had to make some technical assumptions about the data stream in order to prove our theoretical guarantees, these assumptions did not enforce any additional structure in the data. In practice, one often encounters three-mode tensors such that (without loss of generality) the first mode corresponds to a “population" of data points, the second mode corresponds to features of members of said population, and the third mode corresponds to time. The intrinsic properties of these distinct modes invite us to consider several different specializations of our algorithm. For instance, one can include an additional step in our algorithm which iteratively clusters members of the population, obtains separate factor updates for each subpopulation, and concatenate these to obtain a dictionary has learned atoms attuned to each cluster. For the feature mode, one might consider dropping or appending additional features, applying OCPDL, and comparing the first and third factor matrices for different feature selections. Alternatively, one could include a “selection matrix" which multiplies the second factor, and is updated according to an objective function which enforces that it be column-sparse. Finally, variants of our algorithm which enforce that the learned atoms are reliable for making predictions (either by the basic “restrict and predict" method discussed in chapter 2 or the more elaborate recursive extrapolation technique) could be derived by modifying the original objective function and following a derivation similar to that of OCPDL.

Bibliography

[AAB$^+$16] Dario Amodei, Rishita Anubhai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Jingdong Chen, Mike Chrzanowski, Adam Coates, Greg Diamos, et al. End to end speech recognition in english and mandarin. 2016.

[AES17] Monagi H Alkinani and Mahmoud R El-Sakka. Patch-based models and algorithms for image denoising: a comparative review between patch-based images denoising methods for additive noise reduction. *EURASIP Journal on Image and Video Processing*, 2017(1):1–27, 2017.

[AGJ15] Animashree Anandkumar, Rong Ge, and Majid Janzamin. Learning overcomplete latent variable models through tensor methods. In *Conference on Learning Theory*, pages 36–112, 2015.

[AGM14] Sanjeev Arora, Rong Ge, and Ankur Moitra. New algorithms for learning incoherent and overcomplete dictionaries. In *Conference on Learning Theory*, pages 779–806, 2014.

[AGMM15] Sanjeev Arora, Rong Ge, Tengyu Ma, and Ankur Moitra. Simple, efficient, and neural algorithms for sparse coding. In *Conference on learning theory*, pages 113–149. PMLR, 2015.

[APM19] Abhishek Agarwal, Jianhao Peng, and Olgica Milenkovic. Online convex dictionary learning. In *Advances in Neural Information Processing Systems*, pages 13242–13252, 2019.

[BB05] Michael W Berry and Murray Browne. Email surveillance using non-negative matrix factorization. *Computational & Mathematical Organization Theory*, 11(3):249–264, 2005.

[BBK18] Casey Battaglino, Grey Ballard, and Tamara G Kolda. A practical randomized cp tensor decomposition. *SIAM Journal on Matrix Analysis and Applications*, 39(2):876–901, 2018.

[BBL+07] Michael W Berry, Murray Browne, Amy N Langville, V Paul Pauca, and Robert J Plemmons. Algorithms and applications for approximate nonnegative matrix factorization. *Computational statistics & data analysis*, 52(1):155–173, 2007.

[BBV04] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[BCD10] David Blei, Lawrence Carin, and David Dunson. Probabilistic topic models: A focus on graphical model design and applications to document and image analysis. *IEEE signal processing magazine*, 27(6):55, 2010.

[Ben13] Yoshua Bengio. Deep learning of representations: Looking forward. In *International Conference on Statistical Language and Speech Processing*, pages 1–37. Springer, 2013.

[Ben17] David Beniaguev. Historical hourly weather data 2012-2017, version 2. *https://www.kaggle.com/selfishgene/historical-hourly-weather-data*, 2017.

[Ber97] Dimitri P Bertsekas. Nonlinear programming. *Journal of the Operational Research Society*, 48(3):334–334, 1997.

[Ber99] Dimitri P Bertsekas. *Nonlinear programming*. Athena scientific Belmont, 1999.

[BGJM11] Steve Brooks, Andrew Gelman, Galin Jones, and Xiao-Li Meng. *Handbook of markov chain monte carlo*. CRC press, 2011.

[BHS+20] Daniel Barson, Ali S Hamodi, Xilin Shen, Gyorgy Lur, R Todd Constable, Jessica A Cardin, Michael C Crair, and Michael J Higley. Simultaneous mesoscopic and two-photon imaging of neuronal activity in cortical circuits. *Nature methods*, 17(1):107–113, 2020.

[BKS15] Boaz Barak, Jonathan A Kelner, and David Steurer. Dictionary learning and tensor decomposition via the sum-of-squares method. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 143–151, 2015.

[BMB$^+$15] Rostyslav Boutchko, Debasis Mitra, Suzanne L Baker, William J Jagust, and Grant T Gullberg. Clustering-initiated factor analysis application for tissue classification in dynamic brain positron emission tomography. *Journal of Cerebral Blood Flow & Metabolism*, 35(7):1104–1111, 2015.

[BPL10] Y-Lan Boureau, Jean Ponce, and Yann LeCun. A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 111–118, 2010.

[Bra08] Fred Brauer. Compartmental models in epidemiology. In *Mathematical epidemiology*, pages 19–79. Springer, 2008.

[CC70] J Douglas Carroll and Jih-Jie Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of "Eckart-Young" decomposition. *Psychometrika*, 35(3):283–319, 1970.

[CWS$^+$11] Yang Chen, Xiao Wang, Cong Shi, Eng Keong Lua, Xiaoming Fu, Beixing Deng, and Xing Li. Phoenix: A weight-based network coordinate system using matrix factorization. *IEEE Transactions on Network and Service Management*, 8(4):334–347, 2011.

[Den14] Li Deng. A tutorial survey of architectures, algorithms, and applications for deep learning. *APSIPA Transactions on Signal and Information Processing*, 3, 2014.

[DLH$^+$13] Li Deng, Jinyu Li, Jui-Ting Huang, Kaisheng Yao, Dong Yu, Frank Seide, Michael Seltzer, Geoff Zweig, Xiaodong He, Jason Williams, et al. Recent advances in deep learning for speech research at microsoft. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8604–8608. IEEE, 2013.

[DLZS11] Weisheng Dong, Xin Li, Lei Zhang, and Guangming Shi. Sparsity-based image denoising via dictionary learning and structural clustering. In *Conference on Computer Vision and Pattern Recognition*, pages 457–464. IEEE, 2011.

[DZLZ18] Yishuai Du, Yimin Zheng, Kuang-chih Lee, and Shandian Zhe. Probabilistic streaming tensor decomposition. In *2018 IEEE International Conference on Data Mining*, pages 99–108. IEEE, 2018.

[EA06] Michael Elad and Michal Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image processing*, 15(12):3736–3745, 2006.

[EAH99] Kjersti Engan, Sven Ole Aase, and John Hakon Husoy. Frame based signal compression using method of optimal directions (mod). In *Proceedings of the IEEE International Symposium on Circuits and Systems VLSI*, volume 4, pages 1–4. IEEE, 1999.

[EHJ$^+$04] Bradley Efron, Trevor Hastie, Iain Johnstone, Robert Tibshirani, et al. Least angle regression. *The Annals of statistics*, 32(2):407–499, 2004.

[Ela10] Michael Elad. *Sparse and redundant representations: from theory to applications in signal and image processing.* Springer Science & Business Media, 2010.

[Fis65] Donald L Fisk. Quasi-martingales. *Transactions of the American Mathematical Society*, 120(3):369–389, 1965.

[GBB$^+$15] José Henrique de M Goulart, Maxime Boizard, Rémy Boyer, Gérard Favier, and Pierre Comon. Tensor cp decomposition with structured factor matrices: Algorithms and performance. *IEEE Journal of Selected Topics in Signal Processing*, 10(4):757–769, 2015.

[Gil14] Nicolas Gillis. The why and how of nonnegative matrix factorization. *Regularization, optimization, kernels, and support vector machines*, 12(257):257–291, 2014.

[GS99] Luigi Grippof and Marco Sciandrone. Globally convergent block-coordinate techniques for unconstrained optimization. *Optimization methods and software*, 10(4):587–637, 1999.

[GS00] Luigi Grippo and Marco Sciandrone. On the convergence of the block nonlinear gauss–seidel method under convex constraints. *Operations research letters*, 26(3):127–136, 2000.

[GSSB17] Mohsen Ghassemi, Zahra Shakeri, Anand D Sarwate, and Waheed U Bajwa. Stark: Structured dictionary learning through rank-one tensor recovery. In *Proceedings of the IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing*, pages 1–5. IEEE, 2017.

[GSSB19] Mohsen Ghassemi, Zahra Shakeri, Anand D Sarwate, and Waheed U Bajwa. Learning mixtures of separable dictionaries for tensor data: Analysis and algorithms. *IEEE Transactions on Signal Processing*, 68:33–48, 2019.

[GTLY12] Naiyang Guan, Dacheng Tao, Zhigang Luo, and Bo Yuan. Online nonnegative matrix factorization with robust stochastic approximation. *IEEE Transactions on Neural Networks and Learning Systems*, 23(7):1087–1099, 2012.

[H$^+$70] Richard A Harshman et al. Foundations of the parafac procedure: Models and conditions for an “explanatory” multimodal factor analysis. 1970.

[HCC$^+$14] Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, et al. Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*, 2014.

[HNHA15] Furong Huang, UN Niranjan, Mohammad Umar Hakeem, and Animashree Anandkumar. Online tensor methods for learning latent variable models. *The Journal of Machine Learning Research*, 16(1):2797–2835, 2015.

[Hoy04] Patrik O Hoyer. Non-negative matrix factorization with sparseness constraints. *Journal of machine learning research*, 5(Nov):1457–1469, 2004.

[Juv17] Markus Juvonen. Patch-based image representation and restoration. 2017.

[KB09] Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

[KE05] Matt J Keeling and Ken TD Eames. Networks and epidemic models. *Journal of the Royal Society Interface*, 2(4):295–307, 2005.

[KKL$^+$21] Lara Kassab, Alona Kryshchenko, Hanbaek Lyu, Denali Molitor, Deanna Needell, and Elizaveta Rebrova. Detecting short-lasting topics using nonnegative tensor decomposition (preprint). *arXiv preprint arXiv:2010.01600*, 2021.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[Kul18] Rohit Kulkarni. A Million News Headlines, 2018.

[KWSR17] Alec Koppel, Garrett Warnell, Ethan Stump, and Alejandro Ribeiro. D4l: Decentralized dynamic discriminative dictionary learning. *IEEE Transactions on Signal and Information Processing over Networks*, 3(4):728–743, 2017.

[LBRN07] Honglak Lee, Alexis Battle, Rajat Raina, and Andrew Y Ng. Efficient sparse coding algorithms. In *Advances in neural information processing systems*, pages 801–808, 2007.

[LHK05] Kuang-Chih Lee, Jeffrey Ho, and David J Kriegman. Acquiring linear subspaces for face recognition under variable lighting. *IEEE Transactions on pattern analysis and machine intelligence*, 27(5):684–698, 2005.

[LKVP21] Hanbaek Lyu, Yacoub H Kureh, Joshua Vendrow, and Mason A Porter. Learning low-rank latent mesoscale structures in networks. *arXiv preprint arXiv:2102.06984*, 2021.

[LMNS20] Hanbaek Lyu, Georg Menz, Deanna Needell, and Christopher Strohmeier. Applications of online nonnegative matrix factorization to image and time-series data. In *Information Theory and Applications Workshop*, pages 1–9. IEEE, 2020.

[LMS19] H Lyu, F Memoli, and D Sivakoff. Sampling random graph homomorphisms and applications to network data analysis. *arXiv:???*, 2019.

[LNB19] H. Lyu, D. Needell, and L. Balzano. Online matrix factorization for markovian data and applications to network dictionary learning. 2019. Submitted.

[LNB20] Hanbaek Lyu, Deanna Needell, and Laura Balzano. Online matrix factorization for markovian data and applications to network dictionary learning. *Journal of Machine Learning Research*, 21(251):1–49, 2020.

[LP17] David A Levin and Yuval Peres. *Markov chains and mixing times*, volume 107. American Mathematical Soc., 2017.

[LS99] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[LS00] Michael S Lewicki and Terrence J Sejnowski. Learning overcomplete representations. *Neural computation*, 12(2):337–365, 2000.

[LS01] Daniel D Lee and H Sebastian Seung. Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems*, pages 556–562, 2001.

[LSMN20] Hanbaek Lyu, Christopher Strohmeier, Georg Menz, and Deanna Needell. Covid-19 time-series prediction by joint dictionary learning and online nmf, 2020.

[LSN22] Hanbaek Lyu, Christopher Strohmeier, and Deanna Needell. Online nonnegative cp-dictionary learning for markovian data, 2022.

[LYC09] Hyekyoung Lee, Jiho Yoo, and Seungjin Choi. Semi-supervised nonnegative matrix factorization. *IEEE Signal Processing Letters*, 17(1):4–7, 2009.

[Lyu20] Hanbaek Lyu. Convergence of block coordinate descent with diminishing radius for nonconvex optimization. *arXiv preprint arXiv:2012.03503*, 2020.

[Mai13a] Julien Mairal. Optimization with first-order surrogate functions. In *International Conference on Machine Learning*, pages 783–791, 2013.

[Mai13b] Julien Mairal. Stochastic majorization-minimization algorithms for large-scale optimization. In *Advances in Neural Information Processing Systems*, pages 2283–2291, 2013.

[MBPS10] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research*, 11(Jan):19–60, 2010.

[MMTV17] Arthur Mensch, Julien Mairal, Bertrand Thirion, and Gaël Varoquaux. Stochastic subsampling for factorizing huge matrices. *IEEE Transactions on Signal Processing*, 66(1):113–128, 2017.

[MMYZ13] Liyan Ma, Lionel Moisan, Jian Yu, and Tieyong Zeng. A dictionary learning approach for poisson image deblurring. *IEEE Transactions on medical imaging*, 32(7):1277–1289, 2013.

[MSS16] Tengyu Ma, Jonathan Shi, and David Steurer. Polynomial-time tensor decompositions with sum-of-squares. In *IEEE 57th Annual Symposium on Foundations of Computer Science*, pages 438–446. IEEE, 2016.

[MT12] Sean P Meyn and Richard L Tweedie. *Markov chains and stochastic stability*. Springer Science & Business Media, 2012.

[MYW18] Congbo Ma, Xiaowei Yang, and Hu Wang. Randomized online CP decomposition. In *Proceedings of International Conference on Advanced Computational Intelligence*, pages 414–419. IEEE, 2018.

[Nes98] Yurii Nesterov. Introductory lectures on convex programming volume i: Basic course. *Lecture notes*, 3(4):5, 1998.

[OF97] Bruno A Olshausen and David J Field. Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision research*, 37(23):3311–3325, 1997.

[PRSE17] Vardan Papyan, Yaniv Romano, Jeremias Sulam, and Michael Elad. Convolutional dictionary learning via local processing. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5296–5304, 2017.

[Rao69] K Murali Rao. Quasi-martingales. *Mathematica Scandinavica*, 24(1):79–92, 1969.

[Ren08] J. Rennie. 20 Newsgroups, 2008.

[Ris73] J Rissanen. Algorithms for triangular decomposition of block hankel and toeplitz matrices with application to factoring positive matrix polynomials. *Mathematics of computation*, 27(121):147–154, 1973.

[RLH19] Sirisha Rambhatla, Xingguo Li, and Jarvis Haupt. Noodl: Provable online dictionary learning and sparse coding. *In 7th International Conference on Learning Representations*, 2019.

[RLH20] Sirisha Rambhatla, Xingguo Li, and Jarvis Haupt. Provable online cp/parafac decomposition of a structured tensor via dictionary learning. *Advances in Neural Information Processing Systems*, 33, 2020.

[RPZ$^+$18] Bin Ren, Laurent Pueyo, Guangtun Ben Zhu, John Debes, and Gaspard Duchêne. Non-negative matrix factorization: robust extraction of extended structures. *The Astrophysical Journal*, 852(2):104, 2018.

[RU11] Anand Rajaraman and Jeffrey David Ullman. *Mining of massive datasets*. Cambridge University Press, 2011.

[SBS16] Zahra Shakeri, Waheed U Bajwa, and Anand D Sarwate. Minimax lower bounds for kronecker-structured dictionary learning. In *IEEE International Symposium on Information Theory*, pages 1148–1152. IEEE, 2016.

[SG07] Mark Steyvers and Tom Griffiths. Probabilistic topic models. *Handbook of latent semantic analysis*, 427(7):424–440, 2007.

[SGH02] Arkadiusz Sitek, Grant T Gullberg, and Ronald H Huesman. Correction for ambiguous solutions in factor analysis using a penalized least squares objective. *IEEE transactions on medical imaging*, 21(3):216–225, 2002.

[SH05] Amnon Shashua and Tamir Hazan. Non-negative tensor factorization with applications to statistics and computer vision. In *Proceedings of the 22nd international conference on Machine learning*, pages 792–799. ACM, 2005.

[SHSK18] Shaden Smith, Kejun Huang, Nicholas D Sidiropoulos, and George Karypis. Streaming tensor factorization for infinite data sources. In *Proceedings of the SIAM International Conference on Data Mining*, pages 81–89. SIAM, 2018.

[SLLC17] Will Wei Sun, Junwei Lu, Han Liu, and Guang Cheng. Provable sparse tensor decomposition. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 3(79):899–916, 2017.

[SS17] Tselil Schramm and David Steurer. Fast and robust tensor decomposition with applications to dictionary learning. *arXiv preprint arXiv:1706.08672*, 2017.

[SSB18] Zahra Shakeri, Anand D Sarwate, and Waheed U Bajwa. Identifiability of kronecker-structured dictionaries for tensor data. *IEEE Journal of Selected Topics in Signal Processing*, 12(5):1047–1062, 2018.

[SSB$^+$19] Zahra Shakeri, Anand D Sarwate, Waheed U Bajwa, M Rodrigues, and Y Eldar. Sample complexity bounds for dictionary learning from vector-and tensor-valued data. In *Information Theoretic Methods in Data Science*. Cambridge Univ. Press Cambridge, UK, 2019.

[SSY18] Tao Sun, Yuejiao Sun, and Wotao Yin. On markov chain gradient descent. In *Advances in Neural Information Processing Systems*, pages 9896–9905, 2018.

[Sta06] United States. Noaa online weather data (nowdata): Interactive data query system : public fact sheet. *Washington, D.C. : National Oceanic and Atmospheric Administration.*, 2006.

[SV17] Vatsal Sharan and Gregory Valiant. Orthogonalized als: A theoretically principled tensor decomposition algorithm for practical use. In *International Conference on Machine Learning*, pages 3095–3104. PMLR, 2017.

[Tib96] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[TN12] Leo Taslaman and Björn Nilsson. A framework for regularized non-negative matrix factorization, with application to the analysis of gene expression data. *PloS one*, 7(11):e46331, 2012.

[TS15] Gongguo Tang and Parikshit Shah. Guaranteed tensor decomposition: A moment approach. In *International Conference on Machine Learning*, pages 1491–1500. PMLR, 2015.

[Tuc66] Ledyard R Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, 1966.

[VB12] Anja Voss-Böhme. Multi-scale modeling in morphogenesis: a critical analysis of the cellular potts model. *PloS one*, 7(9), 2012.

[VMV15] Nick Vannieuwenhoven, Karl Meerbergen, and Raf Vandebril. Computing the gradient in optimization algorithms for the cp decomposition in constant memory through tensor blocking. *SIAM Journal on Scientific Computing*, 37(3):C415–C438, 2015.

[VRCB18] Don Van Ravenzwaaij, Pete Cassey, and Scott D Brown. A simple introduction to markov chain monte–carlo sampling. *Psychonomic bulletin & review*, 25(1):143–154, 2018.

[WEG87] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

[XY13] Yangyang Xu and Wotao Yin. A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion. *SIAM Journal on imaging sciences*, 6(3):1758–1789, 2013.

[ZEB18] Shuo Zhou, Sarah Erfani, and James Bailey. Online cp decomposition for sparse tensors. In *IEEE International Conference on Data Mining*, pages 1458–1463. IEEE, 2018.

[ZH05] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.

[ZTX17] Renbo Zhao, Vincent Tan, and Huan Xu. Online nonnegative matrix factorization with general divergences. In *Artificial Intelligence and Statistics*, pages 37–45. PMLR, 2017.

[ZVB$^+$16] Shuo Zhou, Nguyen Xuan Vinh, James Bailey, Yunzhe Jia, and Ian Davidson. Accelerating online cp decompositions for higher order tensors. In *Proceedings* *of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1375–1384, 2016.