Non-Linear Mining of Social Activities in Tensor Streams

Koki Kawabata*
ISIR, Osaka University
koki88@sanken.osaka-u.ac.jp

Yasuko Matsubara*
ISIR, Osaka University
yasuko@sanken.osaka-u.ac.jp

Takato Honda*
ISIR, Osaka University
takato@sanken.osaka-u.ac.jp

Yasushi Sakurai*
ISIR, Osaka University
yasushi@sanken.osaka-u.ac.jp

ABSTRACT

Given a large time-evolving event series such as Google web-search logs, which are collected according to various aspects, i.e., timestamps, locations and keywords, how accurately can we forecast their future activities? How can we reveal significant patterns that allow us to long-term forecast from such complex tensor streams?

In this paper, we propose a streaming method, namely, CubeCast, that is designed to capture basic trends and seasonality in tensor streams and extract temporal and multi-dimensional relationships between such dynamics. Our proposed method has the following properties: (a) it is *effective*: it finds both trends and seasonality and summarizes their dynamics into simultaneous non-linear latent space. (b) it is *automatic*: it automatically recognizes and models such structural patterns without any parameter tuning or prior information. (c) it is *scalable*: it incrementally and adaptively detects shifting points of patterns for a semi-infinite collection of tensor streams. Extensive experiments that we conducted on real datasets demonstrate that our algorithm can effectively and efficiently find meaningful patterns for generating future values, and outperforms the state-of-the-art algorithms for time series forecasting in terms of forecasting accuracy and computational time.

CCS CONCEPTS

• **Information systems → Data mining**.

KEYWORDS

Time series, Tensor analysis, Automatic mining

ACM Reference Format:
Koki Kawabata, Yasuko Matsubara, Takato Honda, and Yasushi Sakurai. 2020. Non-Linear Mining of Social Activities in Tensor Streams . In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '20), August 23–27, 2020, Virtual Event, CA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3394486.3403260

*Artificial Intelligence Research Center, The Institute of Scientific and Industrial Research (ISIR), Osaka University

KDD '20, August 23–27, 2020, Virtual Event, CA, USA

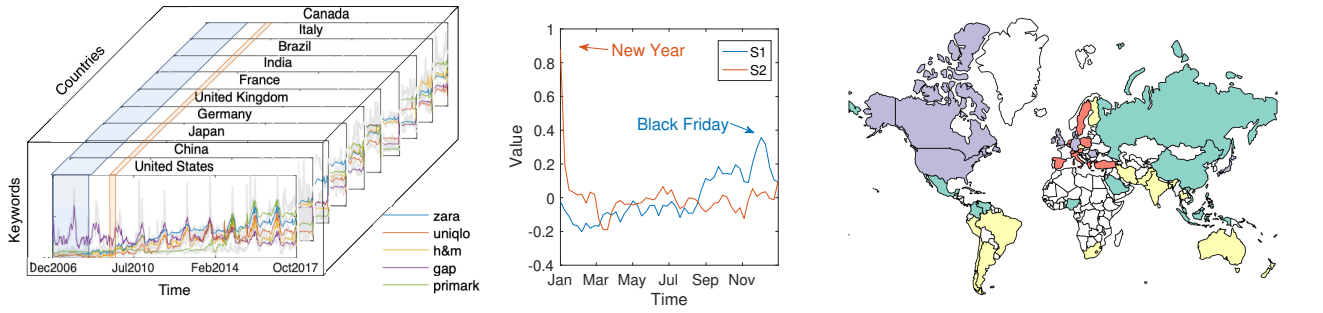
ACM ISBN 978-1-4503-7998-4/20/08…$15.00
https://doi.org/10.1145/3394486.3403260

1 INTRODUCTION

Time series forecasting has been playing a significant role in providing a wide range of applications such as smart decision making [10], automated sensor network monitoring [9, 16] and user activity modeling [19], where analysts are interested in finding useful patterns in collected data with which to predict future phenomena. For example, marketers want to know how many people will react to their products to enable inventory management, new product development, etc. They can avoid wasting human and material resources by accurately forecasting future customer behavior.

The modeling of dynamic patterns has been addressed for several decades but remains a challenging task thanks to the advent of the Internet of Things (IoT) [6, 20], which enables us to access a massive volume and variety of time series, and thus data have multiple domains. For example, when we consider user behavior analysis in relation to web search activities, observations could be of the form (*timestamp*, *location*, *keyword*), which is also called a 3rd-order tensor. Thus, there has been a need for the multi-way mining of tensor streams. Given such large tensor streams, how can we extract the beneficial dynamics from a complex tensor? How can we forecast future activities effectively? The difficulties involved in forecasting tensor streams have the following two causes: *(a) Multiple factors behind observable data:* Many time series data contain several patterns such as trends and seasonality; moreover we cannot know their real characteristics in advance. More importantly, such dynamic patterns appear individually in several groups by location, product category, etc. It is extremely difficult to design an appropriate model for such patterns by hand. The method for tensor streams should therefore be fully automatic as regards estimating model parameters and the number of hidden dynamical patterns. This would enable us to understand data structures and thus save time and human resources. *(b) Patterns that vary over time:* All the factors in time series can change as time progresses for any of a number of reasons, e.g., new product releases. It is important to understand not only trends and seasonality but also their dynamical changes. We refer to individual pattern groups as a *regime*. We want to detect regime changes and reflect the latest information in a model as quickly as possible to realize highly accurate adaptive tensor forecasting.

In this paper, we tackle a challenging problem, namely, the real-time forecasting of tensor streams, and we present CubeCast, which is an effective mining method that can simultaneously capture time-evolving trends and seasonality as well as multiple discrete patterns in tensor streams. Intuitively, the problem we wish to solve is as follows.



(a) Tensor stream (b) Seasonal patterns (c) Location-specific pattern groups

Figure 1: Modeling power of CubeCast for an online search volume tensor stream related to five apparel companies: (a) Given the original tensor (gray lines), CubeCast quickly identifies the non-linear dynamics in the latest tensor (blue), then, continuously forecasts multiple steps ahead values (red), (b) while extracting seasonal patterns common to all countries. (c) It also automatically identifies similar country groups based on their trends and seasonality, which are compressed into compact models (please also see Figure 2).

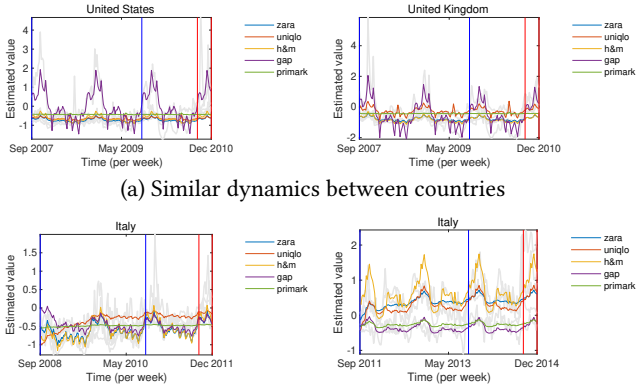
InformalProblem 1. ***Given*** *a tensor stream $\mathcal{X}$ up to the current time point t_c, which consists of elements at d_l locations for d_k keywords, i.e., $\mathcal{X} = \{x_{tij}\}_{t,i,j=1}^{t_c,d_l,d_k}$,*

- ***find*** *trends and seasonal patterns*
- ***find*** *a set of groups (i.e., regimes) of similar dynamics*
- ***forecast*** *l_s-step ahead future values, i.e., $\mathcal{X}^f = \{x_{tij}\}$ where $(t = t_c + l_s;\ i = 1, \ldots, d_l;\ j = 1, \ldots, d_k)$*
- *continuously and automatically in a streaming fashion.*

1.1 Preview of our results

Figure 1 shows the result of CubeCast for online mining over a time-evolving tensor stream. Figure 1 (a) shows a data stream that consists of weekly web-online search volumes in relation to apparel companies in fifty countries. CubeCast retains only recent tensor series (shown in blue) and produces future values (shown in red) by extracting important patterns and dynamics hidden in the tensor. Specifically, our method captures the following properties:

- **Long-term trends:** Figure 1 (a) shows our algorithm, which scans the latest tensor incrementally and captures non-linear trends to realize the long-term forecasting. As shown in the figure, CubeCast successfully captures trends exhibiting an overall increase such as "zara" in the stream for the United States.
- **Seasonality:** Figure 1 (b) shows the seasonalities extracted from the tensor stream using CubeCast. It found the two kinds of yearly patterns resulting from "New year sales" and "Black Friday". To accurately forecast time-evolving dynamics, we must discover and model such periodic patterns.
- **Multi-aspect dynamic patterns:** CubeCast can find multi-aspect (i.e., time and country) patterns. Figure 1 (c) shows the CubeCast clustering result for fifty countries from 2006 to 2008, where the result is plotted in color on a world map. It identifies groups of countries by summarizing their dynamics (i.e., both trends and seasonality) into compact models. For example, Figure 2 shows our (52 : 65)-steps ahead forecasting results when given a tensor for two consecutive years. Given the data between the blue lines, our method generates the future values shown between the red lines. As shown in Figure 2 (a), our algorithm groups the United States and the United Kingdom into the same group because they have similar dynamics including a strong seasonality for "gap". Figure 2 (b) shows the pattern differences between moments for search counts in Italy where the search counts of "h&m" grew hugely. By switching a model for an upcoming pattern, it effectively forecasts tensor streams.



(a) Similar dynamics between countries

(b) Time-varying dynamics in a country

Figure 2: Multi-aspect mining of CubeCast for Google-Trends related to major apparel companies. It automatically detects (a) similar country groups based on dynamics, and (b) changes between discrete dynamics.

Note that our method finds all the above important components in a streaming fashion, without parameter tuning.

1.2 Contributions

In this paper, we propose a streaming method, namely, CubeCast, for the efficient mining and forecasting of co-evolving tensor streams. In summary, our proposed method has the following properties.

- *Effective*: CubeCast operates on large tensor streams and decomposes them into similar groups, i.e., *regimes*, with respect to both time and location, each of which captures latent non-linear dynamics for trends and seasonality.
- *Automatic*: We carefully formulate data encoding schemes to reveal predictable patterns/dynamics, which enables us to analyze complex tensors without any expertise with respect to dataset and parameter tuning.
- *Scalable*: The computational time required by CubeCast is constant with regard to the entire length of the input tensor. Our algorithm outperforms the state-of-the-art algorithms for time series forecasting.

The rest of this paper is organized as follows: In Section 2 we introduce related studies. We present our proposed model and its optimization algorithms in Section 3 and Section 4, respectively. We provide our experimental results in Section 5, followed by our conclusions in Section 6.

2 RELATED WORK

The mining and forecasting of big time series are highly significant topics, especially in relation to data mining and databases [20, 25, 26]. Table 1 shows the relative advantages of our method. Only CubeCast meets all requirements. We roughly separate our descriptions of related work into three categories.

Conventional approaches for time series forecasting are based on statistical models such as auto regression (AR), linear dynamical systems (LDS), Kalman filters (KF), and their extensions. RegimeCast [15] and OrbitMap [16] are real-time forecasting methods that can capture latent non-linear dynamics in multi-dimensional event streams. However, these methods cannot model seasonal patterns in the streams. In recent years, many deep neural network models have been proposed [8, 22]. In particular, Long short-term memory (LSTM) and Gated recurrent units (GRUs) are well known deep structures for capturing long-term temporal dependencies [1], but sensitive parameter tuning and a high training cost are needed if we are to obtain generalized models.

For mining high-order data, tensor decomposition is a powerful technique with which to understand latent factors in data, and it has been extensively studied over the past few decades [3, 12, 33]. For example, AutoCyclone [30] provides the robust factorization needed for separating basic trends, seasonality and outliers from time series by considering the seasonal stacked time series as a tensor. There have also been many online/incremental approaches to the techniques [28, 29, 34]. However, these methods typically focus on data completion, i.e., the recovery of missing values in tensors with their multilinear relationships, rather than the prediction of future data. For time series forecasting, CompCube [18] and PowerCast [27] exploit suitable non-linear equations for their dataset domains. For a more general approach to tensor time series, a multi-linear dynamical system (MLDS) [24] has been proposed as a straightforward extension of LDS, which can jointly learn temporal dependency and tensor factorization, while remaining a linear model.

Clustering and summarizing time series are also relevant to our work [7, 14, 32]. AutoPlait [17] and BeatLex [9] can automatically find multiple distinct dynamical patterns. As with the two methods, the minimum description length (MDL) principle is widely applied to the automation of optimization processes [2, 4, 13, 31]. Unlike previous studies, this work addresses the automatic pattern mining of a specific dimension (e.g., locations) of tensors based on non-linear models. As a consequence, none of the previous studies specifically address the modeling and forecasting of non-linear dynamics in tensor streams, which include seasonality and multiple distinct patterns.

Table 1: Capabilities of approaches. Only CubeCast meets all requirements.

	SARIMA/++	RegimeCast	LSTM/GRU	AutoCyclone	MLDS	CompCube	AutoPlait	CubeCast
Stream processing	-	✓	-	-	-	-	-	✓
Tensor analysis	-	-	✓	-	✓	✓	-	✓
Data compression	-	-	-	✓	-	✓	✓	✓
Non-linear	-	✓	✓	✓	-	✓	-	✓
Seasonality	✓	-	-	✓	-	✓	-	✓
Segmentation	-	✓	-	-	-	-	✓	✓
Forecasting	✓	✓	✓	-	✓	✓	-	✓
Parameter free	-	-	-	✓	-	✓	✓	✓

3 PROPOSED MODEL

In this section, we describe our proposed model namely, CubeCast, for mining time-evolving tensor streams. We first introduce notations and definitions, and then explain the model in detail.

3.1 Problem definition

Table 2 lists the main symbols that we use throughout this paper. We consider a tensor stream to be a 3rd-order tensor, which is denoted by $\mathcal{X} \in \mathbb{R}^{t_c \times d_l \times d_k}$, where t_c is the number of time points, and d_l and d_k show the numbers of locations and keywords, respectively. That is, the element x_{tij} corresponds to the search volume at time point t in the i-th location of the j-th keyword. Our overall aim is to realize the long-term prediction of a tensor $\mathcal{X}$ while adapting to the latest tendencies. We define $\mathcal{X}^c = \{x_{tij}\}_{t,i,j=t_p,1,1}^{t_c,d_l,d_k}$ as the partial tensor of $\mathcal{X}$, whose length is denoted by l_c, i.e., $t_p = t_c - l_c$. That is, l_c denotes the data duration that our method retains. Similarly, let $\mathcal{X}^f = \{x_{tij}\}_{t,i,j=t_s,1,1}^{t_e,d_l,d_k}$ denote a partial tensor from $t_s = t_c + l_s$ to $t_e = t_s + l_e$. Here, we define l_s and l_e as the time points that we forecast and a regular time interval for reports, respectively. Finally, we formally define our problem as follows.

Problem 1 (l_s-step ahead forecasting). ***Given**: a data stream $\mathcal{X}^c = \{x_{tij}\}_{t,i,j=t_p,1,1}^{t_c,d_l,d_k}$, where, t_c is the current time point and $t_p = t_c - l_c$; **Forecast**: l_s-steps ahead future values $\mathcal{X}^f = \{x_{tij}\}_{t,i,j=t_s,1,1}^{t_e,d_l,d_k}$ where $t_s = t_c + l_s$ and $t_e = t_s + l_e$.*

3.2 CubeCast

To capture all the components outlined in the introduction, we present a full model, namely CubeCast, which can model latent dynamic patterns underlying tensor streams. So, how can we build our model so that it summarizes tensor streams in terms of their

Table 2: Symbols and definitions.

Symbol	Definition
d_k, d_l	Number of keywords and locations
t_c	Current time point
$\mathcal{X}$	Tensor stream, i.e., $\mathcal{X} \in \mathbb{R}^{t_c \times d_l \times d_k}$
$\mathcal{X}_{t_s:t_e}$	The partial tensor of $\mathcal{X}$ from time point t_s to t_e
$\mathcal{X}_{:,i}$	Time series for the i-th country, i.e., $\mathcal{X}_{:,i} \in \mathbb{R}^{t_c \times d_k}$
k_z, k_v	Number of latent states for base trends and seasonality
$\mathbf{Z}$	Latent variables for base trends, i.e., $\mathbf{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_t\}$, $\mathbf{z}_i \in \mathbb{R}^{k_z}$
$\mathbf{V}$	Latent variables for seasonality, i.e., $\mathbf{V} = \{\mathbf{v}_1, \ldots, \mathbf{v}_t\}$, $\mathbf{v}_i \in \mathbb{R}^{k_v}$
$\mathbf{A}$, $\mathcal{B}$	Non-linear dynamical system, i.e., $\mathbf{A} \in \mathbb{R}^{k \times k}$, $\mathcal{B} \in \mathbb{R}^{k \times k \times k}$, $k = k_z + k_v$
$\mathcal{W}$	Observation matrix set for base trends, i.e., $\mathcal{W} = \{\mathbf{W}_1, \ldots, \mathbf{W}_m\}$
$\mathcal{U}$	Observation matrix set for seasonality, i.e., $\mathcal{U} = \{\mathbf{U}_1, \ldots, \mathbf{U}_m\}$
p	Period of seasonality
$\mathbf{S}$	Latent seasonal components, i.e., $\mathbf{S} \in \mathbb{R}^{p \times k_v}$
$\mathcal{E}$	Estimated variables, i.e., $\mathcal{E} \in \mathbb{R}^{t_c \times d_l \times d_k}$
m	Number of local groups in a regime
n	Number of regimes
θ	Regime parameter set, i.e., $\theta = \{\mathbf{A}, \mathcal{B}, \mathcal{W}, \mathcal{U}\}$
Θ	Full parameter set, i.e., $\Theta = \{\mathbf{S}, \theta_1, \ldots, \theta_n\}$
$\mathcal{R}$	Regime assignment set, i.e., $\mathcal{R} = \{\mathbf{r}_1, \ldots, \mathbf{r}_n\}$

important components? Specifically, our model should have the following three capabilities:

- **Non-linear latent dynamics:** Our model adopts a non-linear dynamical system to capture complex dynamics in time series.
- **Seasonality:** We extend the non-linear dynamical system to handle seasonality that can also evolve over time.
- **Co-evolving patterns in tensor streams:** Finally, we propose an adaptive model that can describe both temporal and locational differences in tensor streams.

3.2.1 Latent non-linear dynamics in a single location. We first focus on the simplest case, where we have only a single dynamical pattern given a d-dimensional time series, such as search volumes for a single country. In our basic model, we assume that time series have two types of latent activities:

- $\mathbf{z}_t$: k_z-dimensional latent activities at time point t.
- $\mathbf{e}_t$: d-dimensional actual activities observed at time point t.

That is, we can only observe the actual event $\mathbf{e}_t$, while $\mathbf{z}_t$ is an unobservable vector that describes non-linear dynamics evolving over time. The temporal dependency of these activities can be described with the following equations.

$$\begin{aligned} \mathbf{z}_{t+1} &= \mathbf{A}\mathbf{z}_t + \mathcal{B}\mathbf{z}_t \otimes \mathbf{z}_t, \\ \mathbf{e}_t &= \mathbf{W}\mathbf{z}_t, \end{aligned} \tag{1}$$

where $\otimes$ shows the outer product for two vectors. $\mathbf{A} \in \mathbb{R}^{k_z \times k_z}$ and $\mathcal{B} \in \mathbb{R}^{k_z \times k_z \times k_z}$ describe linear/non-linear dynamical activities, respectively. $\mathbf{W} \in \mathbb{R}^{d \times k_z}$ shows the observation projection with which to obtain the estimated event $\mathbf{e}_t$ from the latent activity $\mathbf{z}_t$. Note that we omitted the bias terms of each equation for clarity.

3.2.2 With latent seasonal dynamics. Next, we consider our second goal, namely, modeling seasonal/cyclic patterns in time series by extending our basic model, Equation (1). More specifically, we want to define another latent space for seasonality that can interact with linear/non-linear activities over time. For example, it would allow us to represent intensifying seasonal patterns in conjunction with latent trends. To this end, we additionally assume two types of latent activities:

- $\mathbf{v}_t$: latent seasonal intensity at time point t, i.e., $\mathbf{v}_t \in \mathbb{R}^{k_v}$.
- $\mathbf{S}$: latent seasonality, i.e., $\mathbf{S} \in \mathbb{R}^{p \times k_v}$.

Here, k_v shows the number of dimensions of latent spaces for seasonality, and p is a seasonal period. Consequently, Equation (1) is redefined as follows.

$$\begin{aligned} \begin{bmatrix} \mathbf{z}_{t+1} \\ \mathbf{v}_{t+1} \end{bmatrix} &= \mathbf{A}\begin{bmatrix} \mathbf{z}_t \\ \mathbf{v}_t \end{bmatrix} + \mathcal{B}\begin{bmatrix} \mathbf{z}_t \\ \mathbf{v}_t \end{bmatrix} \otimes \begin{bmatrix} \mathbf{z}_t \\ \mathbf{v}_t \end{bmatrix}, \\ \mathbf{e}_t &= \mathbf{W}\mathbf{z}_t + \mathbf{U}(\mathbf{v}_t \circ \mathbf{S}_{t \bmod p}), \end{aligned} \tag{2}$$

where $\circ$ shows the element-wise product of two vectors. The terms, $\mathbf{A}$ and $\mathcal{B}$ are extended to $\mathbf{A} \in \mathbb{R}^{k \times k}$ and $\mathcal{B} \in \mathbb{R}^{k \times k \times k}$, respectively, where $k = k_z + k_v$. Estimated vectors $\mathbf{e}_t$ are obtained with a projection matrix $\mathbf{U} \in \mathbb{R}^{d \times k_v}$ for seasonal latent activities, $\mathbf{v}_v$ and $\mathbf{S}$, as well as $\mathbf{W} \in \mathbb{R}^{d \times k_z}$ for latent trends, $\mathbf{z}_t$. Once we have the initial states $\mathbf{z}_0$ and $\mathbf{v}_0$, the following latent states can be recursively generated using a single common dynamical system, which allows us to extract the latent interaction between trends and seasonality.

3.2.3 Full model with multiple locations. Our final goal is to answer the most crucial question, namely, how can we describe regime shifts over time in a large tensor stream, where we assume that there are multiple distinct activities in terms of locations. We thus enhance our non-linear dynamical system, Equation (2), so that it can identify both time-changing and location-specific patterns. Here, we assume that we have a 3rd-order tensor $\mathcal{X}$. We specifically want to divide the tensor along the 2nd mode, i.e., d_l locations, into a set of m local groups ($m < d_l$) in order to capture location-specific activities. That is, dynamical patterns at the i-th location can be described by one of the sets of observation matrices $\mathbf{W}_i$ and $\mathbf{U}_i$ where $i \in \{1, \ldots, m\}$, while sharing a single latent space given by $\mathbf{A}$ and $\mathcal{B}$ in Equation (2). This causes similar time series to share similar latent non-linear factors. Let $\mathcal{E} \in \mathbb{R}^{t_c \times d_l \times d_k}$ be an estimated tensor of $\mathcal{X} \in \mathbb{R}^{t_c \times d_l \times d_k}$. If an observation vector $\mathbf{x}_{ti} \in \mathcal{X}$ for the i-th location at time point t is modeled using the j-th observation matrices, the estimated vector $\mathbf{e}_{ti} \in \mathcal{E}$ is described as follows.

$$\begin{aligned} \begin{bmatrix} \mathbf{z}_{t+1} \\ \mathbf{v}_{t+1} \end{bmatrix} &= \mathbf{A}\begin{bmatrix} \mathbf{z}_t \\ \mathbf{v}_t \end{bmatrix} + \mathcal{B}\begin{bmatrix} \mathbf{z}_t \\ \mathbf{v}_t \end{bmatrix} \otimes \begin{bmatrix} \mathbf{z}_t \\ \mathbf{v}_t \end{bmatrix}, \\ \mathbf{e}_{ti} &= \mathbf{W}_j\mathbf{z}_t + \mathbf{U}_j(\mathbf{v}_t \circ \mathbf{S}_{t \bmod p}), \\ &\textit{where } i = 1, \ldots, d_l \textit{ and } j = 1, \ldots, m. \end{aligned} \tag{3}$$

Definition 3.1 (Single regime parameter set). Let θ be the parameter set of a single non-linear dynamical system, namely $\theta = \{\mathbf{A}, \mathcal{B}, \mathcal{W}, \mathcal{U}\}$, where $\mathcal{W}$ and $\mathcal{U}$ are sets of observation matrices for m local groups, i.e., $\mathcal{W} = \{\mathbf{W}_1, \ldots, \mathbf{W}_m\}$ and $\mathcal{U} = \{\mathbf{U}_1, \ldots, \mathbf{U}_m\}$.

Furthermore, we want to detect the regime transitions between distinct latent dynamics. Let n denote the proper number of regimes up to the current time point. Then, a tensor $\mathcal{X}$ is described using a set of n regimes, i.e., $\{\theta_1, \ldots, \theta_n\}$. Consequently, a full model set for a tensor stream is defined as follows.

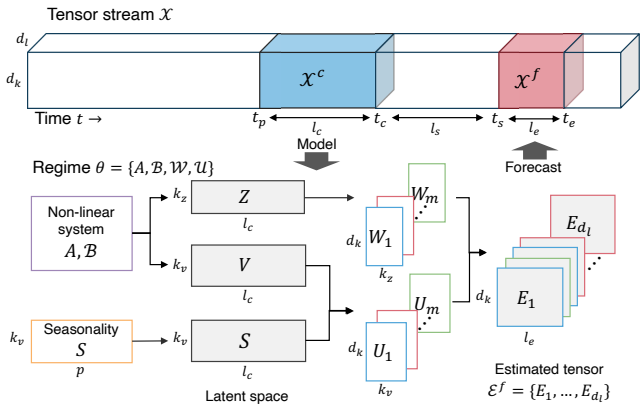


Figure 3: Graphical representation of CubeCast: Given a current tensor $\mathcal{X}^c$, (a) it identifies a regime θ while capturing seasonality with S. (b) It generates latent states Z and V. (c) It reports l_s-steps ahead values $\mathbf{E}_i$ at the i-th location with projection matrices $\mathbf{W}_j$ and $\mathbf{U}_j$, which capture the j-th location-specific pattern.

Definition 3.2 (Full parameter set). Let Θ be a full parameter set, namely, $\Theta = \{\theta_1, \ldots, \theta_n, \mathbf{S}\}$, that describes multiple non-linear patterns with seasonality.

The complete graphical model of CubeCast is shown in Figure 3. Our model uses a different regime $\theta \in \Theta$ that depends on a time-varying pattern. Thus, we also want to determine the assignments of regimes as well as those of their inner local groups.

Definition 3.3 (Regime assignment set). Let $\mathcal{R}$ be a full regime assignment set for Θ, namely, $\mathcal{R} = \{\mathbf{r}_1, \ldots, \mathbf{r}_n\}$, where $\mathbf{r}_i = \{r_1, \ldots, r_j, \ldots, r_{d_l}\}$ is a set of d_l integers for the i-th regime θ_i, and thus $r_j \in \{1, \ldots, m_i\}$ is the local group index to which the j-th country belongs.

Example 3.4. Assume that we have a tensor $\mathcal{X} \in \mathbb{R}^{6\times5\times10}$ consisting of six time points, five locations, ten keywords and with two regimes θ_1 and θ_2, where $\mathbf{r}_1 = \{1, 3, 3, 1, 2\}$ and $\mathbf{r}_2 = \{1, 1, 2, 2, 2\}$. If the algorithm assigns each time point as $\theta_1 \rightarrow \theta_1 \rightarrow \theta_2 \rightarrow \theta_2 \rightarrow \theta_2 \rightarrow \theta_1$, then the tensor $\mathcal{X}$ is divided into $\mathcal{X}_{1:2}$, $\mathcal{X}_{3:5}$, $\mathcal{X}_6$ along the time axis. $\mathcal{X}_{1:2}$ and $\mathcal{X}_6$ are also divided into three local groups, namely, $\mathcal{X}_{1:2,[1,4]}$, $\mathcal{X}_{1:2,[5]}$, $\mathcal{X}_{1:2,[2,3]}$ and $\mathcal{X}_{6,[1,4]}$, $\mathcal{X}_{6,[5]}$, $\mathcal{X}_{6,[2,3]}$, respectively, based on $\mathbf{r}_1$. Similarly, $\mathcal{X}_{3:5}$ is divided into $\mathcal{X}_{3:5,[1:2]}$, and $\mathcal{X}_{3:5,[3:5]}$ based on $\mathbf{r}_2$. Note that S is used to represent seasonality for all these divided tensors.

4 OPTIMIZATION ALGORITHMS

In this section, we present our optimization algorithms for the real-time forecasting of co-evolving tensor streams.

Thus far, we have proposed a model based on non-linear dynamical systems. To forecast future events effectively and accurately with the model, we still have to address two problems, namely, (a) the real-time forecasting of future events while generating and switching regimes adaptively and (b) the automated mining and estimation of multiple non-linear dynamics. Concerning the first goal (a), we need an effective way to incrementally manage the entire model structure Θ so that it can detect regime switching to another known/unknown regime. Moreover, for the second goal (b) we want a criterion with which to determine a well-compressed model that can capture the underlying dynamics of data without any human intervention. We introduce a streaming algorithm, CubeCast, which accomplishes the above goals. Algorithm 1 shows the overall procedure of CubeCast. The basic idea of the algorithm is the tensor encoding scheme we propose. It can update all the components in a model set Θ while processing the current tensor $\mathcal{X}^c$. More specifically, the algorithm consists of:

Algorithm 1 CubeCast $(\mathcal{X}^c, \Theta, \mathcal{R})$

Input: (a) Current tensor $\mathcal{X}^c$
(b) Full parameter set Θ
(c) Regime assignment set $\mathcal{R}$
Output: (a) l_s-steps-ahead future values $\mathcal{E}^f$
(b) Updated full parameter set Θ'
(c) Updated regime assignment set $\mathcal{R}'$

```
/* (I) Estimate a new regime for given data */
{θ, r} ←RegimeEstimation (X^c, S);  // S ∈ Θ
/* (II) Update model set and detect current dynamics */
{Θ', R'} ←RegimeCompression (X^c, Θ, R, θ, r);
/* (III) Generate future values using a current regime */
{θ, r} ← argmin_{θ'∈Θ', r'∈R'} ||X^c − f(θ', r')||  // f(·,·): Equation (3)
E^f ← f(θ, r);  // E^f = {e_tij}_{t,i,j=t_s,1,1}^{t_e,d_l,d_k},
return {E^f, Θ', R'};
```

(1) RegimeEstimation: Estimate a non-linear dynamical system from scratch, namely, θ given a tensor $\mathcal{X}^c$. It also splits the tensor by arranging regime assignment $\mathbf{r}$, and adds sets of observation matrices in $\mathcal{W}$ and $\mathcal{U}$ for θ.
(2) RegimeCompression: Update full parameter set Θ and regime assignment set $\mathcal{R}$ with current $\mathcal{X}^c$ and newly estimated θ and $\mathbf{r}$ for $\mathcal{X}^c$. In this step, the algorithm decides whether or not to employ new regime θ and selects an optimal regime for $\mathcal{X}^c$. After updating $\mathcal{R}$, it also updates seasonality S.
(3) Finally, it generates an l_s-steps future event tensor $\mathcal{E}^f = \{e_{tij}\}_{t,i,j=t_s,1,1}^{t_e,d_l,d_k}$ according to Equation (3) with the most suitable regime θ and regime assignment $\mathbf{r}$ for $\mathcal{X}^c$, which are selected by RegimeCompression.

4.1 Automated tensor summarization

Here, we describe our objective function with respect to the minimum description length (MDL) principle to find the optimal model set Θ automatically. The MDL principle enables us to determine the nature of a good summarization by minimizing the sum of the model description cost and data encoding cost as follows.

$$\Theta = \arg\min_{\Theta'} < \Theta' > + < \mathcal{X} | \Theta' >, \tag{4}$$

where $< \Theta' >$ shows the cost of describing Θ', and $< \mathcal{X} | \Theta' >$ represents the cost of describing the data $\mathcal{X}$ given the model Θ'. In short, it follows the assumption that the more we can compress the data, the more we can learn about its underlying patterns. We thus propose two costs for our model.

4.1.1 Model cost. The class of the model parameter set we should search for is parameterized by the number of latent states for trends

and seasonality as well as the number of regimes. Once we have these numbers, we calculate the description complexity of the entire model with the following terms:

- The dimensionality of a tensor: $< t_c >= \log^*(t_c)$[1], $< d_l >= \log^*(d_l)$, $< d_k >= \log^*(d_k)$.
- The dimensionality of latent components: $< k_z >= \log^*(k_z)$, $< k_v >= \log^*(k_v)$, $< p >= \log^*(p)$.
- Seasonality: $< \mathbf{S} >= |\mathbf{S}| \cdot (\log(p) + \log(k_v) + c_F) + \log^*(|\mathbf{S}|)$.
- Single regime parameter set: $< \theta >=< k_z > + < \mathbf{A} > + < \mathcal{B} > + < \mathcal{W} > + < \mathcal{U} >$.

Here, $|\cdot|$ describes the number of non-zero elements and c_F denotes the floating point cost[2]. The model description cost of each component in $< \theta >$ is defined as follows.

$$
\begin{aligned}
< \mathbf{A} > &= |\mathbf{A}| \cdot (2 \cdot \log(k) + c_F) + \log^*(|\mathbf{A}|),\\
< \mathcal{B} > &= |\mathcal{B}| \cdot (3 \cdot \log(k) + c_F) + \log^*(|\mathcal{B}|),\\
< \mathcal{W} > &= \Sigma_{i=1}^{m} |\mathbf{W}_i| \cdot (\log(d_k) + \log(k_z) + c_F) + \log^*(|\mathbf{W}_i|),\\
< \mathcal{U} > &= \Sigma_{i=1}^{m} |\mathbf{U}_i| \cdot (\log(d_k) + \log(k_v) + c_F) + \log^*(|\mathbf{U}_i|).
\end{aligned}
$$

4.1.2 Data cost. We can encode the data $\mathcal{X}$ using Θ based on Huffman coding [23]. The coding scheme assigns a number of bits to each value in $\mathcal{X}$, which is the negative log-likelihood under a Gaussian distribution with mean μ and variance σ^2, i.e.,

$$
< \mathcal{X} | \Theta >= \sum_{t,i,j=1}^{t_c, d_k, d_l} - \log_2 p_{\mu,\sigma}(x_{tij} - e_{tij}), \tag{5}
$$

where, $e_{tij} \in \mathcal{E}$ shows the reconstruction values of $x_{tij} \in \mathcal{X}$ using Equation (3). Finally, the total encoding cost $< \mathcal{X}; \Theta >$ is written as follows.

$$
\begin{aligned}
< \mathcal{X}; \Theta > &=< \Theta > + < \mathcal{X} | \Theta >\\
&=< t_c > + < d_l > + < d_k > + < p >\\
&\quad + < k_v > + < \mathbf{S} > + \sum_{i=1}^{n} < \theta_i > + < \mathcal{X} | \Theta > . \qquad (6)
\end{aligned}
$$

4.2 RegimeEstimation

It is difficult to find the global optimal solution of Equation (6) due to interdependent components in the model: (a) latent dynamical systems $\mathbf{A}$ and $\mathcal{B}$, (b) observation matrices in $\mathcal{W}$ and $\mathcal{U}$ and (c) seasonal patterns, $\mathbf{S}$. Therefore, we first aim to find the local optima for the components (a) and (b) using a greedy approach, more specifically, we propose RegimeEstimation to minimize Equation (6) for a tensor $\mathcal{X}^c$.

Algorithm 2 shows RegimeEstimation in detail: it first regards current tensor $\mathcal{X}^c$ as a single regime; it searches for discrete local patterns in $\mathcal{X}^c$ by grouping similar dimensions in the target mode of $\mathcal{X}^c$. Our first goal is to estimate the optimal parameters $\theta = \{\mathbf{A}, \mathcal{B}, \mathcal{W}, \mathcal{U}\}$ to minimize the total cost, $< \mathcal{X}^c; \mathbf{S}, \theta, \mathbf{r} >$, while keeping seasonality $\mathbf{S}$ fixed. The first assumption is that there is a single local activity group (i.e., $m = 1$), i.e., $\mathcal{W} = \{\mathbf{W}\}$, $\mathcal{U} = \{\mathbf{U}\}$,

Algorithm 2 RegimeEstimation ($\mathcal{X}^c, \mathbf{S}$)

Input: Current tensor $\mathcal{X}^c$ and seasonality $\mathbf{S}$
Output: Regime parameter set θ and regime assignment $\mathbf{r}$

```
W = φ; U = φ; r = {r_i = 1 | i = 1, ..., d_l};
W* = φ; U* = φ; // candidate observation matrix set
/* Estimate a regime with a single local activity */
{A, B, W, U} ← argmin_{θ'={A',B',W',U'}} < X^c; S, θ', r >;
Push W into W*; Push U into U*;
/* Estimate local activities */
while W* and U* are not empty do
    Pop an entry W_0 from W*; Pop an entry U_0 from U*;
    θ ← {A, B, W_F, U_F}; // W_F = W ∪ W* ∪ {W_0}
    Initialize r*; Initialize W_1, W_2, U_1, U_2;
    θ* ← {A*, B*, W*_F, U*_F}; // A* = A, B* = B
    // W*_F = W ∪ W* ∪ {W_1, W_2}, U*_F = U ∪ U* ∪ {U_1, U_2}
    while < X^c; S, θ*, r* > is improved do
        Estimate r*;
        Estimate W_1, W_2, U_1, U_2;
        Estimate A*, B*;
    end while
    if < X^c; S, θ*, r* > is less than < X^c; S, θ, r > then
        Push {W_1, W_2} into W*; Push {U_1, U_2} into U*;
        A ← A*; B ← B*; r ← r*
    else
        Push W_0 into W; Push U_0 into U;
    end if
end while
return {θ, r}; // θ = {A, B, W, U}
```

and all elements $r_i \in \mathbf{r}$, where $i = 1, \ldots, d_l$, are set at 1. To estimate the number of non-linear activities k_z, the algorithm increases the number from 1, while the total cost is decreasing. For each estimation with k_z, it sets $\mathcal{B} = 0$ and optimizes only linear parameters $\{\mathbf{A}, \mathbf{W}, \mathbf{U}\} \in \theta$ using the expectation-maximization (EM) algorithm. After obtaining the best k_z, it employs the Levenberg-Marquardt (LM) algorithm [21] to optimize the non-linear parameters in $\mathcal{B}$[3]. Note that the initial states $\mathbf{z}_0$ and $\mathbf{v}_0$ are also estimated with θ.

Next, the problem is how to find differences with respect to one of the aspects of a tensor. To avoid considering all candidate combinations of rich attributes, e.g., locations, we propose an efficient stack-based algorithm. Let $\mathcal{W}^*$ and $\mathcal{U}^*$ be stacks containing candidate local activities that could be further divided. While the stacks are not empty, the algorithm pops entries $\{\mathbf{W}_0, \mathbf{U}_0\}$, and then tries to divide the local group into two by generating $\{\mathbf{W}_1, \mathbf{U}_1\}$ and $\{\mathbf{W}_2, \mathbf{U}_2\}$. After initializing the first local activity assignments $\mathbf{r}^*$ for the two candidate local groups, it iterates three procedures to estimate a new parameter set θ^*: (a) minimize reconstruction errors only by updating $\{\mathbf{W}_1, \mathbf{W}_2, \mathbf{U}_1, \mathbf{U}_2\} \in \theta^*$; (b) minimize reconstruction errors only by updating $\{\mathbf{A}^*, \mathcal{B}^*\} \in \theta^*$ and (c) rearrange the regime assignments in $\mathbf{r}^*$ only for the two candidate local groups based on the newly estimated parameters θ^*. The new assignment $r_i^* \in \mathbf{r}^*$ of the i-th country in the divided local group is set to the local group index that minimizes the total cost, $< \mathcal{X}^c_{:,i} | \mathbf{A}, \mathcal{B}, \mathbf{W}_j, \mathbf{U}_j >$, where $j \in \{1, 2\}$. This alternative procedure makes the latent dynamical system more sophisticated in relation to divided activities. Note that it uses all observation matrices $\mathcal{W}_F$ and $\mathcal{U}_F$ in every iteration since updating $\mathbf{A}$ and $\mathcal{B}$ affects the model quality for the entire local groups. Finally, if the coding cost with newly estimated components θ^* and $\mathbf{r}^*$ is less than the cost with

[1] Here, $\log^*$ is the universal code length for integers.
[2] We use $c_F = 32$ bits.
[3] Note that the non-linear activity tensor $\mathcal{B}$ should be sparse to eliminate the complexity of the system, thus we only use the diagonal elements $b_{iii} \in \mathcal{B}$, where $i \in [1, k]$

undivided components θ and $\mathbf{r}$, the algorithm pushes the candidate pairs into the stacks $\mathcal{W}^*$ and $\mathcal{U}^*$ for subsequent iterations. Otherwise, it employs $\mathbf{W}_0$ and $\mathbf{U}_0$ as an optimal local group (i.e., $m = m + 1$).

4.3 RegimeCompression

Here, we answer the final question, namely, how can we realize the good compression of tensor streams while detecting regime shifts? As described in subsection 1.1, real-world applications comprise several discrete phases. We propose RegimeCompression, that makes effective and efficient updating possible so that the approach can detect upcoming dynamical patterns. The main idea is to employ/update regimes when they reduce the total cost of $\mathcal{X}^c$. The overall RegimeCompression algorithm is shown as Algorithm 3. Given a current tensor $\mathcal{X}^c$, it finds an optimal regime based on a previous model set $\{\Theta, \mathcal{R}\}$ and a candidate regime $\{\theta, \mathbf{r}\}$ estimated using RegimeEstimation. The goal is to continue minimizing the total cost of $\mathcal{X}^c$ when given a model set Θ. First, the algorithm searches for an optimal regime $\theta^* \in \Theta$ and $\mathbf{r}^* \in \mathcal{R}$, which minimizes the coding cost $< \mathcal{X}^c | \mathbf{S}, \theta^*, \mathbf{r}^* >$. If a newly estimated θ gives us a lower total cost for $\mathcal{X}^c$ than θ^*, it adds θ into Θ, which indicates that θ represents a good summarization for an additional pattern. Otherwise, it describes $\mathcal{X}^c$ with θ^*. After the algorithm updates the regime shift dynamics $\mathcal{R}$, it updates seasonality $\mathbf{S}$ and current regime θ^* using the LM algorithm. More specifically, it alternately updates one component with the other component fixed. It can find a k_v seasonal component $\mathbf{S}$ that minimizes the reconstruction error $\mathcal{X}^c$.

Before we start online forecasting, we need to initialize the number of seasonal components k_v and seasonality $\mathbf{S}$. Therefore, we estimate the two components based on independent component analysis (ICA). Specifically, we first estimate a regime θ with RegimeEstimation, where $k_v = 0$ and $\mathbf{S} = 0$. Then, we vary $k_v = 1, 2, 3, \ldots$, and determine an appropriate number so as to minimize the total cost $< \mathcal{X}; \mathbf{S}, \theta, \mathbf{r} >$. For each given k_v, we apply ICA to the matrix $\mathbf{X} \in \mathbb{R}^{p \times d}$, which is reshaped from $\mathcal{X}$ for training, and obtain independent components as $\mathbf{S}$.

Lemma 4.1. *The computation time of CubeCast is $O(n\, d_1 d_k)$ per time point, where n is the number of regimes.*

Proof. Please see Appendix A. □

5 EXPERIMENTS

In this section, we describe the performance of CubeCast on real datasets. The experiments were designed to answer the following questions:

- *Q1. Effectiveness*: How well does our method extract latent dynamical patterns?
- *Q2. Accuracy*: How accurately does our method predict future values?
- *Q3. Scalability*: How does our method scale in terms of computational time?

Our experiments were conducted on an Intel Xeon W-2123 3.6GHz quad core CPU with 128GB of memory and running Linux.

Algorithm 3 RegimeCompression ($\mathcal{X}^c, \Theta, \mathcal{R}, \theta, \mathbf{r}$)

Input: (a) Current tensor $\mathcal{X}^c$
(b) Full parameter set Θ and regime assignment set $\mathcal{R}$
(c) Candidate regime θ and regime assignment $\mathbf{r}$
Output: Updated model set Θ^* and regime assignment set $\mathcal{R}^*$

```
/* Search an optimal regime within Θ */
{θ*, r*} ← arg min_{θ'∈Θ, r'∈R} < X^c; S, θ', r' >;
if < X^c; S, θ, r > is less than < X^c; S, θ*, r* > then
    Θ* ← Θ ∪ θ;  R* ← R ∪ r;
    θ* ← θ;  r* ← r;  // Replace an optimal regime with a new regime
else
    Θ* ← Θ;  R* ← R;
end if
while < X^c; S, θ*, r* > is improved do
    Estimate θ*;  // θ* ∈ Θ*
    Estimate S;  // S ∈ Θ*
end while
return {Θ*, R*};
```

Table 3: Dataset description.

ID	Dataset	Query
#1	Apparel	zara, uniqlo, h&m, gap, primark
#2	Chatapps	facebook, LINE, slack, snapchat, twitter, telegram, viber, whatsapp
#3	Hobby	soccer, baseball, basketball, running, yoga, crafts
#4	LinuxOS	debian, ubuntu, centos, redhat, fedora, opensuse, steamos, raspbian, kubuntu
#5	PythonLib	numpy, scipy, sklearn, matplotlib, plotly, tensorflow
#6	Shoes	booties, flats, heels, loafers, pumps, sandals, sneakers

Datasets. We used the 6 real event streams on GoogleTrends[4], which contained the weekly search volumes for keywords from January 1, 2004 to December 31, 2018 (totally 14 years) from 236 countries. The queries of our datasets are described in Table 3. Due to a significant amount of missing data, we selected the top 50 countries in order of their GDP scores[5]. We normalized the values so that each sequence had the same mean and variance (i.e., z-normalization).

Baselines. We used the following baselines, which are state-of-the-art algorithms for modeling and forecasting time series:

- RegimeCast [15] – Real-time forecasting method with multiple discrete non-linear dynamical systems. We set the number of latent states $k = 4$, the model hierarchy $h = 2$, and the model generation threshold $\epsilon = 0.5 \cdot \|\mathcal{X}_c\|$.
- SARIMA [5] – A state space method for capturing seasonal elements of time series. We choose the optimal number of parameters for the model from $\{1, 2, 4, 8\}$ based on AIC.
- MLDS [24] – Multilinear dynamical system (MLDS), which learns the multilinear projection of each dimension of a sequence of latent tensors. We varied the ranks of the latent tensors $\{2, 4\}$ and $\{4, 8\}$.
- LSTM/GRU[1] – RNN-based models for time series. We stacked a 2-layer LSTM/GRU to encode and decode/predict parts, each of which has 50 units. We also applied a dropout rate of 0.5 to the connection of the output layer. In their learning steps, we used Adam optimization [11] and early stopping.

[4] https://trends.google.com/trends/
[5] https://www.imf.org/external/index.htm

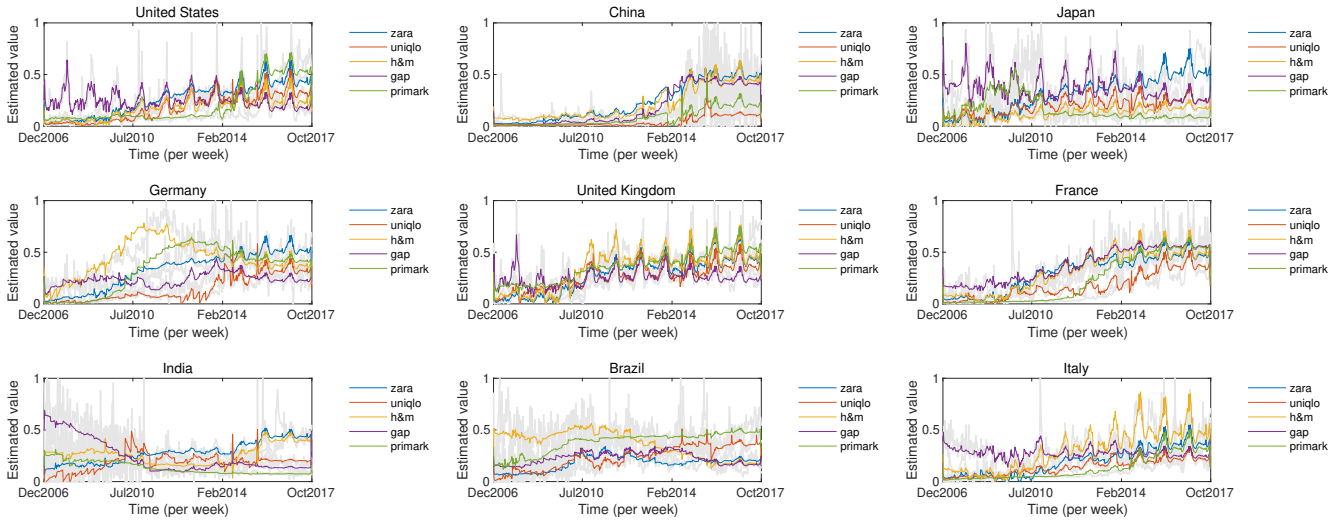


Figure 4: Fitting results of CubeCast for five apparel companies on GoogleTrends (here, we show the results for nine countries). CubeCast incrementally and automatically identifies sudden changes in dynamical patterns including the latent trends, seasonality and structure of groups of similar countries.

5.1 Q1. Effectiveness

We first describe how successfully CubeCast found dynamical patterns and their structural changes over time on co-evolving tensor streams. Some of the results have already been presented in section 1 (i.e., Figure 1 and Figure 2).

Figure 4 shows the additional fitting results of CubeCast for nine countries in the (#1) *Apparel* tensor stream. The original activities are shown as faint lines, and our estimated volumes are shown as solid lines. The results were obtained when our method retained a tensor $\mathcal{X}^c$ of length 104 (i.e., two years) at every 13-th time point (i.e., a quarter of a year). We also assume that our datasets have yearly seasonality, so we set $p = 52$ and initialized seasonal components with tensor streams from 2004 to 2006.

Overall, our proposed model successfully captured latent dynamical patterns for multiple countries and keywords. As shown in Figure 4, the detected change points are near the spots where the trends are suddenly changed. For example, In 2009, Germany had become the biggest market for H&M, and thus its search volume is increasing at that time. This trend is captured in September 2008. In August 2012, we can observe increasing trends in China and countries in Europe. Recently, all the countries have been maintaining their seasonality without eather the growth or decay of trends. Since our method compresses an input tensor into a compact model, it can detect comprehensive pattern changes in web activity tensor streams.

5.2 Q2. Accuracy

Next, we evaluate the forecasting accuracy of CubeCast compared with baselines. Figure 5 shows the average root mean square error (RMSE) between the original tensor and the 52-step (i.e., a year) ahead estimated values using every current tensor $\mathcal{X}^c$ of length 104. A lower value indicates a better forecasting accuracy.

Unsurprisingly, our method outperforms the other time series forecasting methods for all datasets because it can model non-linear dynamics and seasonality simultaneously. RegimeCast is capable of handling multiple discrete non-linear dynamics but misses seasonal patterns. Thus, it cannot forecast future values very well on our online web activity datasets. Note that linear models are unsuitable for long-term (i.e., multiple steps ahead) forecasting.

5.3 Q3. Scalability

Finally, we evaluate the computational time needed by CubeCast for large tensor time series by comparison with its competitors.

Figure 6 shows the average response time for each dataset. As we expected, our method achieves a great improvement in terms of computational time. The RNN-based models require a significant amount of learning time, while our method can identify the current regime including several groups of countries and forecast future events quickly and continuously. Since RegimeCast and SARIMA are incapable of handling tensor data, they need to process a tensor as a large matrix, which incurs high computational costs. Overall, our proposed method is very efficient for the real-time/long-term forecasting of tensor series.

We also evaluated the scalability of CubeCast in more detail. Figure 7 shows the average wall clock time of RegimeEstimation when the tensor size is varied, i.e., the duration and the number of countries on six GoogleTrends datasets. Thanks to our proposed stack-based country-specific pattern identification procedure, the complexity of RegimeEstimation scales linearly with both the duration and the number of countries. Consequently, our method has a desirable property in that it can forecast large tensor streams based on multi-aspect dynamic pattern mining.

6 CONCLUSION

In this paper, we proposed an effective and efficient forecasting method, namely CubeCast, for large time-evolving tensor series. Our method can recognize basic trends and seasonality in input observations by extracting their latent non-linear dynamical systems.

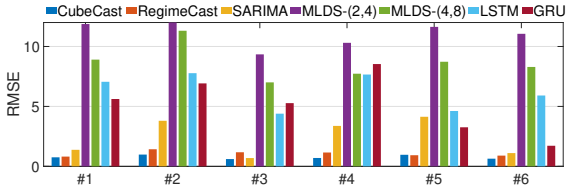


Figure 5: Average forecasting accuracy of CubeCast: our method is consistently superior to its competitors for all datasets (lower is better).

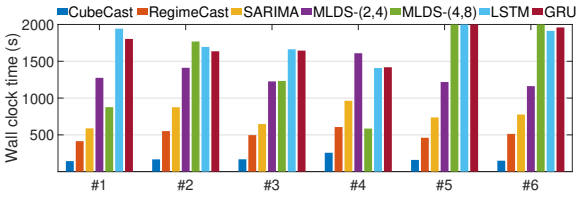


Figure 6: Average wall clock time on GoogleTrends: CubeCast can quickly provide a forecast while detecting regime shifts and important patterns (lower is better).

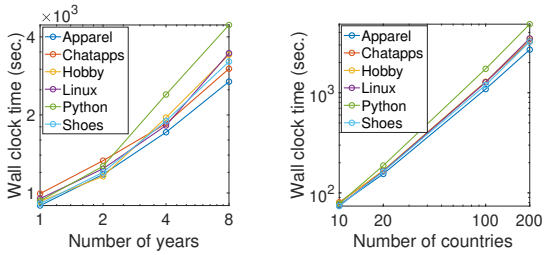


Figure 7: Average wall clock time vs. tensor stream size, i.e., duration (t_c) and number of countries (d_l). CubeCast scales linearly with respect to the time and target mode for division into several groups.

We showed that our method has the following advantages over its competitors for time series forecasting using real Google search volume datasets. It is *Effective*: it effectively captures complex non-linear dynamics for tensor time series when forecasting long-term future values. It is *Automatic*: it automatically recognizes all the components in regimes and their temporal/structural innovations, while requiring no prior knowledge of the data. It is *Scalable*: the computation time of CubeCast is independent of the time series length.

Acknowledgment. The authors would like to thank the anonymous referees for their valuable comments and helpful suggestions. This work was supported by JSPS KAKENHI Grant-in-Aid for Scientific Research Number JP19J11125, JP17J11668, JP17H04681, JP18H03245, JP20H00585, JST-PRESTO JPMJPR1659, JST-Mirai JPMJMI19B3, MIC/SCOPE 192107004, ERCA-Environment Research and Technology Development Fund.

REFERENCES

[1] 2018. Recurrent Neural Networks for Multivariate Time Series with Missing Values. *Scientific Reports* 8, 1 (2018), 6085.
[2] Roel Bertens, Jilles Vreeken, and Arno Siebes. 2016. Keeping it short and simple: Summarising complex event sequences with multivariate patterns. In *KDD*. 735–744.
[3] Yongjie Cai, Hanghang Tong, Wei Fan, Ping Ji, and Qing He. 2015. Facets: Fast Comprehensive Mining of Coevolving High-order Time Series. In *KDD*. 79–88.
[4] Deepayan Chakrabarti, Spiros Papadimitriou, Dharmendra S. Modha, and Christos Faloutsos. 2004. Fully automatic cross-associations. In *KDD*. 79–88.
[5] James Durbin and Siem Jan Koopman. 2012. *Time Series Analysis by State Space Methods* (2 ed.). Oxford University Press.
[6] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. 2013. Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions. *Future Gener. Comput. Syst.* 29, 7 (2013), 1645–1660.
[7] David Hallac, Sagar Vare, Stephen Boyd, and Jure Leskovec. 2017. Toeplitz Inverse Covariance-Based Clustering of Multivariate Time Series Data. In *KDD*. 215–223.
[8] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. 2012. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine* 29, 6 (2012), 82–97.
[9] Bryan Hooi, Shenghua Liu, Asim Smailagic, and Christos Faloutsos. 2017. BeatLex: Summarizing and Forecasting Time Series with Patterns. In *ECML PKDD*. Springer, 3–19.
[10] Emre Kiciman and Matthew Richardson. 2015. Towards Decision Support and Goal Achievement: Identifying Action-Outcome Relationships From Social Media. In *KDD*. 547–556.
[11] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2015).
[12] Tamara G. Kolda and Brett W. Bader. 2009. Tensor Decompositions and Applications. *SIAM Rev.* 51, 3 (September 2009), 455–500.
[13] Jae-Gil Lee, Jiawei Han, and Kyu-Young Whang. 2007. Trajectory clustering: a partition-and-group framework. In *SIGMOD*. 593–604.
[14] Lei Li, James McCann, Nancy S. Pollard, and Christos Faloutsos. 2009. DynaMMo: mining and summarization of coevolving sequences with missing values. In *KDD*. 507–516.
[15] Yasuko Matsubara and Yasushi Sakurai. 2016. Regime Shifts in Streams: Real-time Forecasting of Co-evolving Time Sequences. In *KDD*. 1045–1054.
[16] Yasuko Matsubara and Yasushi Sakurai. 2019. Dynamic Modeling and Forecasting of Time-Evolving Data Streams. In *KDD*. 458–468.
[17] Yasuko Matsubara, Yasushi Sakurai, and Christos Faloutsos. 2014. AutoPlait: Automatic Mining of Co-evolving Time Sequences. In *SIGMOD*.
[18] Yasuko Matsubara, Yasushi Sakurai, and Christos Faloutsos. 2016. Non-Linear Mining of Competing Local Activities. In *WWW*.
[19] Yasuko Matsubara, Yasushi Sakurai, Christos Faloutsos, Tomoharu Iwata, and Masatoshi Yoshikawa. 2012. Fast mining and forecasting of complex time-stamped events. In *KDD*. 271–279.
[20] Gianmarco De Francisci Morales, Albert Bifet, Latifur Khan, Joao Gama, and Wei Fan. 2016. IoT Big Data Stream Mining. In *KDD, Tutorial*. 2119–2120.
[21] Jorge J. Moré. 1978. The Levenberg-Marquardt algorithm: Implementation and theory. In *Numerical Analysis*. 105–116.
[22] Yao Qin, Dongjin Song, Haifeng Cheng, Wei Cheng, Guofei Jiang, and Garrison W. Cottrell. 2017. A Dual-stage Attention-based Recurrent Neural Network for Time Series Prediction. In *IJCAI*. AAAI Press, 2627–2633.
[23] Jorma Rissanen. 1978. Modeling by shortest data description. *Automatia* 14 (1978), 465–471.
[24] Mark Rogers, Lei Li, and Stuart J Russell. 2013. Multilinear Dynamical Systems for Tensor Time Series. In *NIPS*. 2634–2642.
[25] Yasushi Sakurai, Yasuko Matsubara, and Christos Faloutsos. 2015. Mining and Forecasting of Big Time-series Data. In *SIGMOD, Tutorial*. 919–922.
[26] Yasushi Sakurai, Yasuko Matsubara, and Christos Faloutsos. 2016. Mining Big Time-series Data on the Web. In *WWW, Tutorial*. 1029–1032.
[27] Hyun Ah Song, Bryan Hooi, Marko Jereminov, Amritanshu Pandey, Lawrence T. Pileggi, and Christos Faloutsos. 2017. PowerCast: Mining and Forecasting Power Grid Sequences. In *ECML/PKDD*.
[28] Qingquan Song, Xiao Huang, Hancheng Ge, James Caverlee, and Xia Hu. 2017. Multi-Aspect Streaming Tensor Completion. In *KDD*. 435–443.
[29] Jimeng Sun, Dacheng Tao, and Christos Faloutsos. 2006. Beyond Streams and Graphs: Dynamic Tensor Analysis. In *KDD*. 374–383.
[30] Tsubasa Takahashi, Bryan Hooi, and Christos Faloutsos. 2017. AutoCyclone: Automatic Mining of Cyclic Online Activities with Robust Tensor Factorization. In *WWW*. 213–221.
[31] Nikolaj Tatti and Jilles Vreeken. 2012. The long and the short of it: summarising event sequences with serial episodes. In *KDD*. 462–470.
[32] Peng Wang, Haixun Wang, and Wei Wang. 2011. Finding semantics in time series. In *SIGMOD*. 385–396.
[33] Junchen Ye, Leilei Sun, Bowen Du, Yanjie Fu, Xinran Tong, and Hui Xiong. 2019. Co-Prediction of Multiple Transportation Demands Based on Deep Spatio-Temporal Neural Network. In *SIGKDD*. 305–313.
[34] Shuo Zhou, Nguyen Xuan Vinh, James Bailey, Yunzhe Jia, and Ian Davidson. 2016. Accelerating Online CP Decompositions for Higher Order Tensors. In *KDD*. 1375–1384.

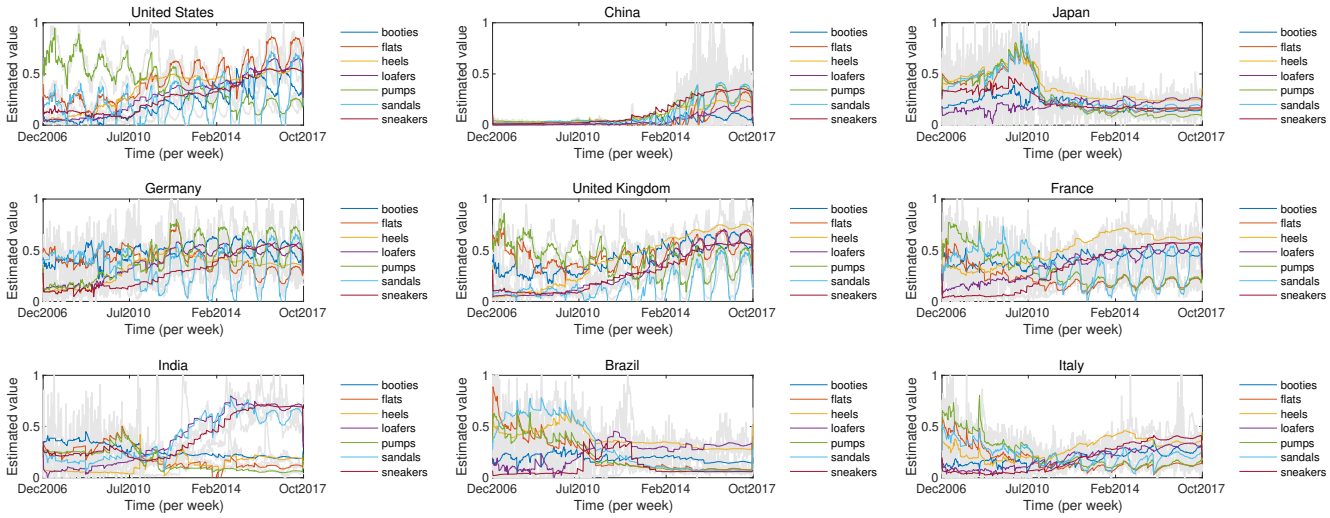


Figure 8: Fitting results of CubeCast for the number of searches of seven shoe type keywords in nine countries. CubeCast detects time-evolving dynamics including various patterns, trends and seasonalities for each country in a complex tensor stream. We emphasize that our algorithm does not need any prior training or knowledge regarding the keywords.

A STREAMING ALGORITHM

Proof of Lemma 4.1. For each time tick, CubeCast performs matrix operations to reconstruct a given tensor $\mathcal{X}^c \in \mathbb{R}^{l_c \times d_l \times d_k}$ with Θ. For d_l dimensions, it needs $O(l_c d_k k_z)$ for trends and $O(l_c d_k k_v + p k_v)$ for seasonality. RegimeEstimation iterates splitting a candidate regime into two local groups, which needs $O(\#iter \cdot d_l \cdot 2\, l_c d_k)$. In RegimeCompression, searching for an optimal regime from n regimes in Θ requires $O(n d_l d_k)$. Since $\#iter$, l_c and the numbers of hidden components k_z, k_v are negligibly small constant values, the total computation time of CubeCast is $O(n d_l d_k)$. □

B EXPERIMENTS

Here we show the additional results relating to the effectiveness and scalability of our method.

B.1 Effectiveness

We examined the modeling power of CubeCast in terms of capturing important patterns, trends and seasonalities of each country in a given tensor stream. Figure 8 shows our real-time fitting results for the search volumes of seven keywords as regards shoe types.

As an example, see the top left of Figure 8 (United States), the dataset has various fashion trends and seasonalities, such as the long-term decrease in the popularity of pumps, the cyclic increase in the popularity of sandals and the decrease in the popularity of booties in the summer. Moreover, these tensor stream characteristics vary by country. Compared with the sequence for the United States, the popularity of pumps is increasing in Germany as seen in the center left of Figure 8. In India, shown in the bottom left of Figure 8, the data have only weak seasonalities because of the country's climate and culture. Our method automatically and effectively identifies multiple trends and seasonalities as well as location-specific features, and then models all behavior in a streaming fashion.

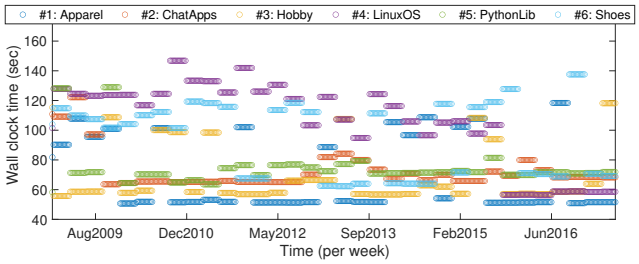


Figure 9: Wall clock time vs. tensor stream length t_c. For each dataset, CubeCast can identify the current dynamics and forecast future values with constant time.

B.2 Scalability

Next, we present additional results regarding the time consumed by CubeCast. Figure 9 shows the computation time of CubeCast at each time interval for reports. As described in Lemma 4.1, the time complexity scales linearly in terms of the number of regimes in the current model set Θ and the size of current tensor $\mathcal{X}^c$. The algorithm works efficiently even if it splits a given tensor into several groups of countries in RegimeEstimation. We also note that our algorithm finally reported the number of regimes n for each dataset as follows. (#1) $n = 6$, (#2) $n = 30$, (#3) $n = 8$, (#4) $n = 19$, (#5) $n = 26$, (#6) $n = 14$. As shown in this figure, the actual running time of CubeCast is dominated by the time for RegimeEstimation, not for searching for an optimal regime from among n regimes in a model set Θ. Our algorithm can perform tensor forecasting continuously, the time required remains almost constant.