AutoCyclone: Automatic Mining of Cyclic Online Activities with Robust Tensor Factorization

Tsubasa Takahashi
NEC Corporation
t-takahashi@nk.jp.nec.com

Bryan Hooi
Carnegie Mellon University
bhooi@andrew.cmu.edu

Christos Faloutsos
Carnegie Mellon University
christos@cs.cmu.edu

ABSTRACT

Given a collection of seasonal time-series, how can we find regular (cyclic) patterns and outliers (i.e. rare events)? These two types of patterns are hidden and mixed in the time-varying activities. How can we robustly separate regular patterns and outliers, without requiring any prior information?

We present CycloneM, a unifying model to capture both cyclic patterns and outliers, and CycloneFact, a novel algorithm which solves the above problem. We also present an automatic mining framework AutoCyclone, based on CycloneM and CycloneFact. Our method has the following properties; (a) *effective*: it captures important cyclic features such as trend and seasonality, and distinguishes regular patterns and rare events clearly; (b) *robust* and *accurate*: it detects the above features and patterns accurately against outliers; (c) *fast*: CycloneFact takes linear time in the data size and typically converges in a few iterations; (d) *parameter free*: our modeling framework frees the user from having to provide parameter values.

Extensive experiments on *4 real* datasets demonstrate the benefits of the proposed model and algorithm, in that the model can capture latent cyclic patterns, trends and rare events, and the algorithm outperforms the existing state-of-the-art approaches. CycloneFact was up to *5 times* more accurate and *20 times* faster than top competitors.

Keywords

time-series; tensor factorization; anomaly detection;

1. INTRODUCTION

As time-series data has increased rapidly in size and availability, so has its potential for real-world application in diverse areas ranging from disaster preparation, electricity grid monitoring, to marketing and the understanding of online user activity. In particular, the task of separating high-level patterns from anomalies is of key importance in allowing us to both accurately understand trends, as well as detecting unusual events of significance.

Given time-varying activities, such as the search volume for the keywords "Swimming", "Running" and "Yoga", how can we find patterns and characteristics to perform marketing research? Especially, how can we robustly detect regular, seasonal patterns of our time series? At the same time, how do we detect remarkable, anomalous occurrences that an analyst might be interested in?

Preview of our results. Figure 1 shows our discoveries related to the sports consisting of $d = 3$ activities: "Swimming" (red), "Running" (blue) and "Yoga" (yellow) taken from Google Trends [1]. The data consists of weekly measurements spanning 2004–2013. AutoCyclone discovered the following:

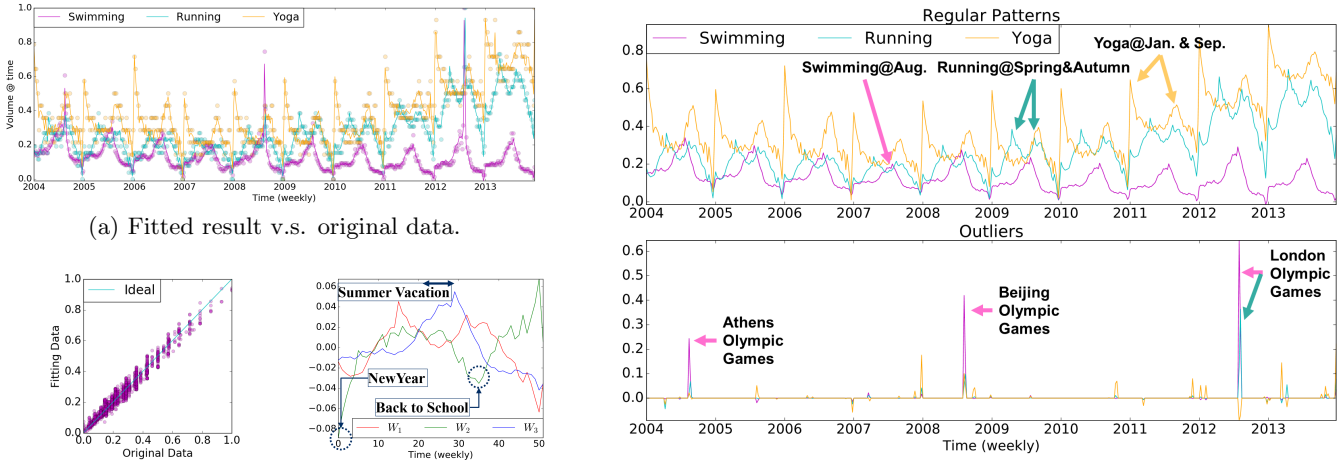
- *Long-term fitting*: Figure 1a shows the original sequences of three activities as circles, and our fitted model as solid lines. Notice that our fitting result is visually very good and smooth, and captures the overall decreasing trend for "Swimming" and the overall increasing trends for "Running" and "Yoga". It also captures three big spikes caused by "Olympic fever" in the years 2004, 2008 and 2012, as shown in Figure 1d.
- *Cyclic patterns and anomaly detection*: Figure 1d shows the seasonal (cyclic) patterns and outliers, on the top and bottom of the figure, respectively. These two are clearly separated. The regular patterns show the cyclic seasonality (e.g., yearly periodicity) and its trend. Figure 1c shows the latent cyclic patterns that repeats every year, which fit the data well and agree with intuition with regard to various holidays and sports seasons. Outliers are very clearly separated from the regular patterns, and include only a few *rarely* and *sparsely* appearing remarkable events (e.g. "Olympic Games").

Contributions. We propose AutoCyclone, a powerful framework which automatically captures cyclic seasonality by distinguishing regular patterns from outliers. AutoCyclone has the following desirable properties:

1. **Effective**: CycloneM captures long-range dynamics and seasonal patterns in a way that allows for easy human interpretation.
2. **Robust and Accurate**: CycloneFact is robust against outliers, thus providing accurate results.
3. **Fast**: Computation of CycloneFact is linear on the input size and converges in a few iterations.

WWW 2017, April 3–7, 2017, Perth, Australia.
ACM 978-1-4503-4913-0/17/04.
http://dx.doi.org/10.1145/3038912.3052595

[1] https://www.google.com/trends/



(a) Fitted result v.s. original data.

(b) Fitting Error.

(c) Cyclic patterns.

(d) AutoCyclone captures regular seasonal behaviors and outliers.

Figure 1: **AutoCyclone is effective for real data**: (a) Our model (solid lines) fits the original data (circles) very well, and (b) fitting values are close to the original data. (c) AutoCyclone finds distinct cyclic patterns, therefore (d) captures smooth regular behaviors having yearly periodicity and detects outliers (e.g. "Olympic fever").

4. **Parameter-free**: AutoCyclone chooses all parameters of CycloneFact to achieve high accuracy and intuitiveness.

Reproducibility. We developed the proposed method in Python 3.5. We will make our code publicly available[2].

2. RELATED WORK

The problem of mining time-series has been studied extensively. Traditional approaches like auto-regression (AR) and Kalman filters and their variants have had a lot of success as essential data mining tools.

Capturing dynamics and segmentation Capturing dynamics and segmentation are very important tasks to understand time-series. The traditional AR, ARIMA and PLiF [11] capture dynamics based on linear equations, but the dynamics we want to capture is *non-linear*. Based on prior knowledge, we can introduce a specific non-linear dynamics model to capture it well. For competitions between co-evolving sequences, EcoWeb [16] captures eco-system-like dynamics. CompCube [17] is a pattern analysis method for capturing local competition. However, because we want to capture the dynamics without any prior information, a *domain independent* model is required.

SWAB [9], pHMM [22], AutoPlait[15] and RegimeShift[14] capture the dynamics of sequences and segment the sequences. Related to the clustering of time-series, an important motif, namely, Shapelet [25] has been introduced. [24] discovers common progression stages in event sequences.

Concept Factorization. Matrix/tensor factorizations are powerful tools to understand latent factors of the target datasets including time-series [21]. There have been many applications and approaches, such as concept discovery [8], network discovery [12] and epidemiology [18]. Marble [5] is a sparse tensor factorization method for understanding concepts in the higher order datasets. Rubik [23], the successor of Marble, scales up and incorporates domain knowledge.

[2] `http://www.cs.cmu.edu/~tsubasat/code/AutoCyclone.zip`

Anomaly Detection Anomaly detection is an important task in data mining. Several existing methods capture both regular patterns and anomalies. JSPCA[7] evaluates the degree of anomalousness for principal components. For noisy multivariate data, several works estimates (regular) latent structure between attributes to detect anomalous behaviors [6] [4]. However, anomaly detection which finds deltas against regular patterns is sensitive to intensive deltas (i.e. spikes). This sensitiveness results in that regular patterns include such intensitve outliers, and the patterns may be skewed. Thus, for mining both regular patterns and outliers, *robust to outliers* is an important property. Robust PCA [13] is a method which robustly detects principal components while capturing outliers.

Contrast with competitors. Table 1 illustrates the relative advantages of our method. "*Seasonality*" means that a method can capture periodic patterns. "*Non-linear*" means that a method can capture dynamics following non-linear differential equations. "*Robust to outlier*" means that a method can clearly separate both regular patterns from outliers. A method which can capture desired patterns in any application domain without any prior information is "*domain independent*". A method is "*parameter free*" if it does not require users to input or tune any parameters. "*Auto. period detection*" refers to methods which automatically determine the periodicity of the time-series. None of the published methods have all the features that AutoCyclone has, as shown in Table 1. In addition to those properties, one of the key innovations of AutoCyclone is the use of "cyclic folding" (Section 4), which allows us to perform tensor factorization in a way that captures seasonal patterns, in a flexible way.

3. PRELIMINARIES

In this section, we briefly describe two essential backgrounds; 1) robust matrix factorization, and 2) tensor factorization. Table 2 lists the symbols used in this paper. For vector, matrix and tensor indexing, we use the Matlab-like notation: $\mathbf{A}_{i,j}$ denotes the (i,j)-th element of matrix $\mathbf{A}$, whereas $\mathbf{A}_{:,j}$ denotes the j-th column of $\mathbf{A}$.

Table 1: Capabilities of approaches.

	AR	Robust PCA[13]	JSPCA[7]	Marble[5]	EcoWeb[16]	CompCube[17]	**AutoCyclone**
Seasonality	✓				✓	✓	✓
Non-linear		✓	✓	✓	✓	✓	✓
Domain independent	✓	✓	✓	✓			✓
Robust to outliers		✓				✓	✓
Parameter free					✓	✓	✓
Auto. period detection							✓

Robust Matrix Factorization with Outlier Rejection. Robust PCA (in short RoPCA) [13] is a novel robust matrix factorization approach which controls outlier sparsity. RoPCA is founded on the following two important assumptions:

- Robust estimation of mean vector $\boldsymbol{m}$ is a sensible way for principal component analysis to be robust against outliers.
- Outlier sparsity estimation via group Lasso [26] provides robusification.

The RoPCA estimates principal components of matrix $\mathbf{X}$ by minimizing the following objective function:

$$\{\mathcal{V}, \mathbf{O}\} = \arg\min_{\mathcal{V},\mathbf{O}} ||\mathbf{X} - \mathbf{1}\boldsymbol{m}^T - \mathbf{S}\mathbf{U} - \mathbf{O}||_F^2 + \lambda_2 ||\mathbf{O}||_{2,r}$$

where $\mathbf{S}$ and $\mathbf{U}$ are low-rank approximated matrices, $\mathbf{O}$ is the outliers, λ_2 is the outlier sparsity controlling parameter, $||\mathbf{O}||_{2,r}$ is the row-wise group lasso penalty, and $\mathcal{V} = \{\boldsymbol{m}, \mathbf{U}, \mathbf{S}\}$.

Tensor and Tensor Factorization. A tensor is a generalization of a matrix. We introduce some important notation for tensors. For more detail, Kolda [10] provided a comprehensive overview of tensor factorization.

First, for the sake of simplicity, consider a three way tensor $\boldsymbol{\mathcal{X}}$ of dimension $I \times J \times K$.

Definition 1. *(Outer Product). The three way outer product of vectors $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$ forms a rank-1 tensor. The outer product is defined as:*

$$[\boldsymbol{a} \circ \boldsymbol{b} \circ \boldsymbol{c}]_{i,j,k} = \boldsymbol{a}_i \boldsymbol{b}_j \boldsymbol{c}_k.$$

Definition 2. *(PARAFAC decomposition). The PARAFAC (also called CP) tensor decomposition of $\boldsymbol{\mathcal{X}}$ in k components is defined as:*

$$\boldsymbol{\mathcal{X}} \approx \sum_{r=1}^{k} \boldsymbol{\alpha}_r \mathbf{A}_r \circ \mathbf{B}_r \circ \mathbf{C}_r = [\![\boldsymbol{\alpha}; \mathbf{A}; \mathbf{B}; \mathbf{C}]\!]$$

where $\mathbf{A}_r$ is the r-th column of matrix $\mathbf{A}$ and $\boldsymbol{\alpha}$ is scaling coefficients. We use $[\![\cdot]\!]$ for a shorthand notation of the sum of outer products of columns of the input matrices. We assume each column of $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ is normalized and the scaling coefficients are stored in $\boldsymbol{\alpha}$.

The most popular algorithm for the PARAFAC decomposition is Alternating Least Squares (CP–ALS). In this paper, we also apply CP–ALS in our proposed algorithm.

We can convert a tensor into a matrix by unfolding it:

Definition 3. *(Tensor Unfolding). The third order tensor $\boldsymbol{\mathcal{X}}$ can be unfolded (i.e. matricized) in the following three ways: $\mathbf{X}_{(1)}$, $\mathbf{X}_{(2)}$, $\mathbf{X}_{(3)}$ of sizes $I \times JK$, $J \times IK$, $K \times IJ$, respectively.*

Table 2: Table of Symbols.

Symbol	Definition
$\boldsymbol{\mathcal{X}}$	a tensor
$\mathbf{X}_{(n)}$	mode-n matricization of a tensor
a	a scalar (lowercase, italic letter)
$\boldsymbol{a}$	a column vector (lowercase, bold letter)
$\mathbf{A}$	a matrix (uppercase, bold letter)
$\mathbf{A}^T$	transpose of $\mathbf{A}$
$\mathbf{A}^\dagger$	pseudo inverse of $\mathbf{A}$
$\otimes$	Kronecker product
$\odot$	Khatri-Rao product
$\circ$	outer product
$*$	element-wise multiplication
$\mathbf{X}$	a time-series (d attributes $\times$ n duration)
ℓ	length of period
${}_\ell\boldsymbol{\mathcal{X}}$	a cyclic folding tensor ($d \times m \times \ell$,$m = \lfloor n/\ell \rfloor$)
$\mathbf{B}$	base trend matrix ($d \times m$)
$\boldsymbol{\mathcal{C}}$	cyclic pattern tensor ($d \times m \times \ell$)
$\boldsymbol{\mathcal{O}}$	outliers ($d \times m \times \ell$)
k	number of components
λ_1	sparsity control parameter for $\boldsymbol{\mathcal{C}}$
λ_2	sparsity control parameter for $\boldsymbol{\mathcal{O}}$

Once a tensor is unfolded into a matrix, we can apply the usual matrix operations to it. We can also fold a matrix to a tensor to express relationships of multiple aspects.

Quality of PARAFAC. CORCONDIA[3] heuristically assesses the quality of a PARAFAC model based on core consistency, that is the closeness between a super-diagonal core tensor and the restricted Tucker 3 decomposition. Since PARAFAC can be seen as a restricted Tucker 3 decomposition with super-diagonal core tensor, if the estimated PARAFAC is good, the core tensor of a Tucker decomposition restricted to use the factors of the PARAFAC decomposition should be as close to super-diagonal as possible. If it is far from the super-diagonal, then this indicates the PARAFAC model is inappropriate, such as decomposition rank or model structure is not appropriate.

We denote the core consistency of tensor $\boldsymbol{\mathcal{X}}$ as $CORCO(\boldsymbol{\mathcal{X}})$. The core consistency score by CORCONDIA is in $[-\infty, 100]$. If the value is 100, it suggests the estimated PARAFAC is completely appropriate. When k=1 and PARAFAC is not penalized (e.g. sparsified), the core consistency score is always 100 because there are no off-super-diagonal core elements. As k increases, the core consistency typically decreases more or less monotonically. [3] suggested that around 50 might be fair and negative values might be invalid. [20] proposed an efficient algorithm for computing CORCONDIA. [19] proposed an automatic mining based on this quality assesment.

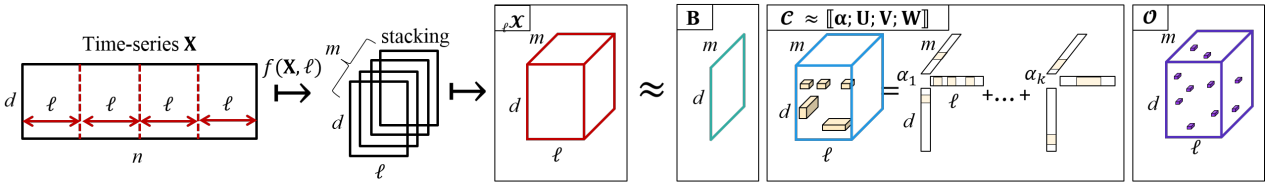


Figure 2: Illustration of AutoCyclone structure.

4. PROPOSED MODEL

In this section, we present our proposed model, namely, CycloneM. Consider that we have a collection of activity volumes $\mathbf{X}$ of d keywords, with duration n. That is, we have $\mathbf{X} = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_i, \cdots, \boldsymbol{x}_d\}$, where $\boldsymbol{x}_i = \{x_i(t)\}_{t=1}^{n}$ is a sequence of activity about keyword i. Given a set of seasonal time series $\mathbf{X}$ having cyclic period ℓ, our goal is to (a) capture the dynamics of $\mathbf{X}$, (b) find the hidden cyclic characteristics of $\mathbf{X}$, and (c) distinguish regular cyclic patterns from rare events.

To capture the cyclic dynamics of $\mathbf{X}$ on the view of cyclic nature, we transform $\mathbf{X}$ into a tensor ${}_{\ell}\boldsymbol{\mathcal{X}}$, which stacks periodically segmented subsequences of $\mathbf{X}$. We call this transformation from a matrix to the tensor, *Cyclic Folding*.

Definition 4. *(Cyclic Folding/Unfolding). The cyclic folding maps $\mathbf{X}$ to 3-mode tensor ${}_{\ell}\boldsymbol{\mathcal{X}}$. ${}_{\ell}\boldsymbol{\mathcal{X}}$ is a periodic tensor of size $d \times \ell \times m$, $m = \lfloor n/\ell \rfloor$. The cyclic folding $f : \mathbf{X} \mapsto {}_{\ell}\boldsymbol{\mathcal{X}}$ can be computed by:*

$$ {}_{\ell}\boldsymbol{\mathcal{X}}_{i,\tau,w} = \mathbf{X}_{i,t} \quad (w = t \bmod \ell, \tau = \lfloor t/\ell \rfloor). \tag{1}$$

Inversely, the unfolding from ${}_{\ell}\boldsymbol{\mathcal{X}}$ into $\hat{\mathbf{X}}$ is described as:

$$ \hat{\mathbf{X}}_{i,t} = {}_{\ell}\boldsymbol{\mathcal{X}}_{i,\tau,w} \quad (t = \ell\tau + w). \tag{2}$$

4.1 CycloneM

In our model, we express the time-series $\mathbf{X}$ as a sum of 3 separate effects:

- **Long-term trends**: the overall changes in the time-series from period to period. This is represented by the base trend matrix $\mathbf{B}$.
- **Seasonal variation**: the regular, seasonal patterns that the time-series exhibits within each period. This is represented by the cyclic pattern tensor $\boldsymbol{\mathcal{C}}$.
- **Outliers**: one-off events that cause anomalous changes to the time-series. This is represented by the outlier tensor $\boldsymbol{\mathcal{O}}$.

In order to capture those characteristics, first we assume the time-series $\mathbf{X}$ is transposed into ${}_{\ell}\boldsymbol{\mathcal{X}}$ by Cyclic Folding (1). Then, we propose a cyclic model CycloneM which has base trend tensor $\boldsymbol{\mathcal{B}}$, cyclic pattern tensor $\boldsymbol{\mathcal{C}}$ and outlier tensor $\boldsymbol{\mathcal{O}}$. The model approximates ${}_{\ell}\boldsymbol{\mathcal{X}}$ by using these three tensors:

$$\begin{aligned} {}_{\ell}\boldsymbol{\mathcal{X}} &\approx \boldsymbol{\mathcal{B}} + \boldsymbol{\mathcal{C}} + \boldsymbol{\mathcal{O}} \\ \boldsymbol{\mathcal{C}} &= [\![\boldsymbol{\alpha}; \mathbf{U}; \mathbf{V}; \mathbf{W}]\!] \\ \boldsymbol{\mathcal{B}} &= \mathbf{B} \circ \mathbf{1}_{\ell} \\ \mathbf{B} &= \mathrm{mean}_{d\times m}({}_{\ell}\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{O}}) \end{aligned} \tag{3}$$

where $\mathrm{mean}_{d\times m}(\cdot) \in \mathbb{R}^{d\times m}$ computes the mean over subsequences for each period and each attribute. Figure 2 shows the structure of our model. In the above approximation of ${}_{\ell}\boldsymbol{\mathcal{X}}$, $\boldsymbol{\mathcal{B}}$ and $\boldsymbol{\mathcal{C}}$ capture trends and seasonalities, respectively. Then $\boldsymbol{\mathcal{B}}$ and $\boldsymbol{\mathcal{C}}$ represent regular behaviors of ${}_{\ell}\boldsymbol{\mathcal{X}}$ without outliers. For interpretability, $\mathbf{U}, \mathbf{V}$ should be sparse. We do not enforce $\mathbf{W}$ being sparse as that would indicate a seasonal pattern that influences a small number of time points within each period, whereas we would actually expect each pattern to affect the entire period in a smooth manner.

A simple definition of outliers are activities which do not follow regular cyclic seasonality, that is $\boldsymbol{\mathcal{O}} = {}_{\ell}\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{B}} - \boldsymbol{\mathcal{C}}$. Therefore, it takes $d \times m \times \ell$ space: $\boldsymbol{\mathcal{O}} \in \mathbb{R}^{d\times m\times \ell}$. However, $\boldsymbol{\mathcal{O}}$ should capture only abnormal behavior. Assuming such remarkable abnormal behaviors are rare, $\boldsymbol{\mathcal{O}}$ should be sparse. Accurate estimation of outliers is extremely important for the overall model's performance and effectiveness.

5. PROPOSED ALGORITHM

In the previous section, we have seen how we can describe the dynamics of multivariate sequences with respect to three properties that we observed in real data. Now, we want to estimate the parameter set of CycloneM. As we mentioned previously, we need to answer (i) How can we robustly find important patterns and seasonal characteristics (i.e., $\mathbf{B}$, $\boldsymbol{\mathcal{C}}$) against outliers? (ii) How can we estimate intuitive patterns?

The basic idea to answer the questions is:

- Controlling outlier sparsity to robustly estimate $\mathbf{B}$ and $\boldsymbol{\mathcal{C}}$.
- Low-rank approximation with sparse representation for cyclic pattern tensor $\boldsymbol{\mathcal{C}}$.

Thus, the objective function we solve is:

$$\begin{aligned} \boldsymbol{\mathcal{Z}} = \arg\min_{\boldsymbol{\mathcal{Z}}'} (&||{}_{\ell}\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{Z}}'||_F^2 \\ &+ \lambda_1 \sum_{r=1}^{k} (||\mathbf{U}_r||_1 + ||\mathbf{V}_r||_1) \\ &+ \lambda_2 \sum_{g\in G} ||\boldsymbol{\mathcal{O}}_g||_2) \end{aligned} \tag{4}$$

where $\boldsymbol{\mathcal{Z}} = \boldsymbol{\mathcal{B}} + \boldsymbol{\mathcal{C}} + \boldsymbol{\mathcal{O}}$, λ_1 and λ_2 are sparsity control parameters for $\boldsymbol{\mathcal{C}}$ and $\boldsymbol{\mathcal{O}}$, respectively. $g \in G$ is a group which shares the outlier sparsity. The first penalty term enforces sparsity in $\boldsymbol{\mathcal{C}}$ by the conventional L_1 (lasso) regularization. The second penalty term enforces sparsity in $\boldsymbol{\mathcal{O}}$ in terms of outlier rejection. Thus, our choices of the groups G influences the model's robustness. We will discuss about how to design the groups in the following subsection.

To recap, the full parameter set of CycloneFact is:

$$\mathcal{F} = \{\ell, k, \lambda_1, \lambda_2, \mathbf{B}, \boldsymbol{\mathcal{C}}, \boldsymbol{\mathcal{O}}\}.$$

5.1 Outlier Rejection for Cyclic Time-series

Let us begin with the first question, namely: how can we robustly find important patterns and characteristics $\mathbf{B}$, $\boldsymbol{\mathcal{C}}$, against outliers.

Since least squares minimization is known to be very sensitive to outliers, the proposed model assuming (4) will be very influenced by outliers. Thus, the important thing is how to control outliers $\boldsymbol{\mathcal{O}}$.

The group lasso regularization for outliers can enhance the robustness of other components. The group lasso update encourages the grouped elements to be sparse at the same time.

We want to robustly estimate patterns at the subsequence and period level; hence it would not make sense for sparsity groups to span across subsequences or periods. Instead, since a single unusual day would be liable to affect all the keywords on that day, the best way is to pack values for different keyword at each tick of subsequence each period:

$$g_{\tau,w} = \{o_{1,\tau,w}, \ldots, o_{d,\tau,w}\},$$

then we employ $\boldsymbol{\mathcal{O}}_{g_{\tau,w}} = \boldsymbol{\mathcal{O}}_{:,\tau,w}$ in (4).

5.2 CycloneFact

Finally, we introduce our proposed algorithm CycloneFact, a robust tensor factorization approach for modeling cyclic time-series. CycloneFact robustly estimates cyclic seasonal characteristics and sparsely represents rare events.

CycloneFact first estimates cyclical characteristics $\mathbf{B}$ for the outlier subtracted tensor $\boldsymbol{\mathcal{Z}}_{\boldsymbol{\mathcal{O}}}$. Next, we compute sparse CP–ALS (Algorithm 2) to get sparse compact descriptions of cyclic patterns $\boldsymbol{\mathcal{C}}$ (FactorizeC). To induce sparsity, we use LASSO (ℓ_1) regularization, which is the standard sparsifying technique. In FactorizeC, we use soft-thresholding. The soft-thresholding update for sparse representation at level λ_1 is:

$$x = sgn(x)(|x| - \lambda_1)_+ \tag{5}$$

where x is a scalar, $(\cdot)_+ = \max(\cdot, 0)$.

We next update the outlier tensor $\boldsymbol{\mathcal{O}}$. We also update it by the soft-thresholding update described as:

$$o_{i,\tau,w} = \frac{o_{i,\tau,w}(||g_{\tau,w}||_2 - \lambda_2/2)_+}{||g_{\tau,w}||_2}. \tag{6}$$

We introduce the notion $\mathcal{S}[{}_\ell\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{B}} - \boldsymbol{\mathcal{C}}, (\lambda_2/2)\boldsymbol{\mathcal{I}}]$ to express the soft-thresholding update.

Algorithm 1 describes the overall procedure of CycloneFact. Given input parameters k, λ_1, λ_2, CycloneFact estimates the full component set of CycloneM, $\mathcal{F}$.

Algorithm 1 CycloneFact $({}_\ell\boldsymbol{\mathcal{X}}, k, \lambda_1, \lambda_2)$

Input: (a) Periodic Folding Tensor ${}_\ell\boldsymbol{\mathcal{X}}$ $(d \times m \times \ell)$, (b) number of components k, (c) Sparsity control parameter λ_1, (d) Outlier sparsity parameter λ_2
Output: Parameter set $\mathcal{F}^* = \{k, \lambda_1, \lambda_2, \mathbf{B}, \boldsymbol{\mathcal{C}}, \boldsymbol{\mathcal{O}}\}$

$\boldsymbol{\mathcal{O}} \leftarrow \emptyset_{d \times m \times \ell}$
$\boldsymbol{\alpha} \leftarrow (1, 1, 1)$
randomly initialize $\mathbf{U}, \mathbf{V}, \mathbf{W}$
$\boldsymbol{\mathcal{C}} \leftarrow [\![\boldsymbol{\alpha}; \mathbf{U}; \mathbf{V}; \mathbf{W}]\!]$
while not converged **do**
  $\boldsymbol{\mathcal{Z}}_{\boldsymbol{\mathcal{O}}} \leftarrow {}_\ell\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{O}}$
  $\mathbf{B} \leftarrow \text{mean}_{d \times m}(\boldsymbol{\mathcal{Z}}_{\boldsymbol{\mathcal{O}}})$; $\boldsymbol{\mathcal{B}} \leftarrow \mathbf{B} \circ \mathbf{1}_\ell$
  $\boldsymbol{\mathcal{C}} \leftarrow \text{FactorizeC}(\boldsymbol{\mathcal{Z}}_{\boldsymbol{\mathcal{O}}} - \boldsymbol{\mathcal{B}}, k, \lambda_1)$
  $\boldsymbol{\mathcal{O}} \leftarrow \mathcal{S}[{}_\ell\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{B}} - \boldsymbol{\mathcal{C}}, (\lambda_2/2)\boldsymbol{\mathcal{I}}_{d \times \ell \times m}]$
end while
return $\mathcal{F}^* \leftarrow \{\ell, k, \lambda_1, \lambda_2, \mathbf{B}, \boldsymbol{\mathcal{C}}, \boldsymbol{\mathcal{O}}\}$

Algorithm 2 FactorizeC($\boldsymbol{\mathcal{X}}$, λ_1, $[\![\boldsymbol{\alpha}; \mathbf{U}; \mathbf{V}; \mathbf{W}]\!]$)

Input: $\boldsymbol{\mathcal{X}}$, λ_1, (optional) PARAFAC $[\![\boldsymbol{\alpha}; \mathbf{U}; \mathbf{V}; \mathbf{W}]\!]$
Output: PARAFAC $[\![\boldsymbol{\alpha}; \mathbf{U}; \mathbf{V}; \mathbf{W}]\!]$

while not converged. **do**
  $\mathbf{U} \leftarrow (\mathbf{W} \odot \mathbf{V})(\mathbf{W}^T\mathbf{W} * \mathbf{V}^T\mathbf{V})^\dagger$
  Normalize columns of $\mathbf{U}$ (storing norms in vector $\boldsymbol{\alpha}$.)
  Compute (5) for all elements of $\mathbf{U}$ /* sparsify $\mathbf{U}$ */
  $\mathbf{V} \leftarrow (\mathbf{W} \odot \mathbf{U})(\mathbf{W}^T\mathbf{W} * \mathbf{U}^T\mathbf{U})^\dagger$
  Normalize columns of $\mathbf{V}$ (storing norms in vector $\boldsymbol{\alpha}$.)
  Compute (5) for all elements of $\mathbf{V}$ /* sparsify $\mathbf{V}$ */
  $\mathbf{W} \leftarrow (\mathbf{V} \odot \mathbf{U})(\mathbf{V}^T\mathbf{V} * \mathbf{U}^T\mathbf{U})^\dagger$
  Normalize columns of $\mathbf{W}$ (storing norms in vector $\boldsymbol{\alpha}$.)
end while
return $[\![\boldsymbol{\alpha}; \mathbf{U}; \mathbf{V}; \mathbf{W}]\!]$

6. AUTOMATED MINING

In this section, we describe our automatic mining framework for periodic time-series using CycloneFact. The framework AutoCyclone can find a compact and reasonable description of $\boldsymbol{\mathcal{X}}$ based on our CycloneM model. Specifically, the problem we want to solve is as follows:

Problem 1. *Given time-series* $\mathbf{X}$*, find a compact description that represents regular cyclic patterns with trends and outliers of* $\mathbf{X}$*, that is* $\mathcal{F} = \{\ell, k, \lambda_1, \lambda_2, \mathbf{B}, \boldsymbol{\mathcal{C}}, \boldsymbol{\mathcal{O}}\}$.

Thus, the objective function we want to minimize is:

$$\mathcal{F} = \arg\min_{\mathcal{F}'} < \mathcal{F}' > + < {}_\ell\boldsymbol{\mathcal{X}} | \mathcal{F}' > . \tag{7}$$

where $< \cdot >$ is description cost of either model or error. $< \mathcal{F} >$ is model description cost of $\mathcal{F}$, $< {}_\ell\boldsymbol{\mathcal{X}} | \mathcal{F} >$ is data coding cost (i.e., error).

We discuss about model quality by 2 separate parts. First, to choose the best k, λ_1, λ_2 and ℓ, we provide a new intuitive coding scheme, which is based on the minimum description length (MDL) principle. Using MDL, we assess the quality of the sparse encoding and errors. Second, to get a high quality model, we assess the modeling quality of the PARAFAC decomposition using a metric based on core consistency. For rank k, we employ those two metrics to choose a model that best summarizes the original time-series.

6.1 Description Cost

MDL follows the assumption of Ockham's razor, which aims to explain the data in a parsimonious way. Thus, based on MDL, we can choose a well compressed model which parsimoniously captures underlying patterns of the data, by minimizing the number of bits needed to describe the model, and to describe the data given the model.

Model description cost. The base trend $\mathbf{B}$ has $d \times \ell$ floating points. Thus it requires $d \times \ell \times C_F$ bits, where C_F[3] is the number of bits required to encode a floating point number. $\boldsymbol{\mathcal{C}}$ consists of $[\![\boldsymbol{\alpha}; \mathbf{U}; \mathbf{V}; \mathbf{W}]\!]$. $\boldsymbol{\alpha}$ needs $k \times C_F$ bits. $\mathbf{U}_r$, which is a sparse vector with size d, contains $N_{i,r}$ nonzeros, and for each nonzero we need $\log(d)$ bits to encode its position and C_F bits to encode its value. Since the number

[3]We assume C_F=8 bits by following [16].

of non-zeros $N_{i,r}$ ranges 0 to d, it requires $\log(d+1)$ bits to encode. $\mathbf{V}_r$ is similar. Since $\mathbf{W}$ is non-sparse, we encode it in a non-sparse manner, requiring $k \times \ell \times C_F$ bits. Outlier tensor $\boldsymbol{\mathcal{O}}$ is also sparse. For each of N_O outliers, we need $\log(d) + \log(\ell) + \log(m)$ bits to encode its position and C_F bits to encode its value. The number of non-zeros in $\boldsymbol{\mathcal{O}}$ requires $\log(d \times \ell \times m + 1)$ bits since it is between 0 and $d \times \ell \times m$. The number of components k requires $\log^*(k)$[4] bits. The period ℓ requires $\log(n)$ bits since ℓ is chosen from 1 to n. The description complexity of model parameter set consists of the following terms,

- $< \mathbf{B} > = d \times \ell \times C_F$
- $< \boldsymbol{\mathcal{C}} > = k \times C_F + k \times \ell \times C_F$
  $+ \sum_{r=1}^{k} \sum_{i=1}^{d} (N_{i,r}(\log(d) + C_F) + \log(d+1))$
  $+ \sum_{r=1}^{k} \sum_{\tau=1}^{m} (N_{\tau,r}(\log(m) + C_F) + \log(m+1))$
- $< \boldsymbol{\mathcal{O}} > = N_O(\log(d) + \log(\ell) + \log(m) + C_F) + \log(d \times \ell \times m + 1)$
- $< k > = \log^*(k)$
- $< \ell > = \log(n)$.

The total model description cost $< \mathcal{F} >$ is the sum of the above terms.

Data coding cost. Once we have decided the full parameter set $\mathcal{F}$, we can encode the cyclic folding tensor ${}_\ell\boldsymbol{\mathcal{X}}$ using Huffman coding. A number of bits is assigned to each value in ${}_\ell\boldsymbol{\mathcal{X}}$, which is negative log-likelihood of its error with respect to the model prediction $\boldsymbol{\mathcal{Z}}$ under a Gaussian error model. The encoding cost of ${}_\ell\boldsymbol{\mathcal{X}}$ given $\mathcal{F}$ is computed by:

$$< {}_\ell\boldsymbol{\mathcal{X}} | \mathcal{F} > = 2C_F + \sum_{i,\tau,w=1}^{d,\ell,m} - \log_2 p_{Gauss(\mu,\sigma^2)}({}_\ell\boldsymbol{\mathcal{X}}_{i,\tau,w} - \boldsymbol{\mathcal{Z}}_{i,\tau,w})$$

where $\boldsymbol{\mathcal{Z}}_{i,\tau,w} = \boldsymbol{\mathcal{B}}_{i,\tau,w} + \boldsymbol{\mathcal{C}}_{i,\tau,w} + \boldsymbol{\mathcal{O}}_{i,\tau,w}$, and 2 C_F is coding cost of μ and σ.

Finally, the total encoding cost $< {}_\ell\boldsymbol{\mathcal{X}}; \mathcal{F} >$ is given by:

$$\begin{aligned} < {}_\ell\boldsymbol{\mathcal{X}}; \mathcal{F} > &= < \mathcal{F} > + < {}_\ell\boldsymbol{\mathcal{X}} | \mathcal{F} > \\ &= < \ell > + < k > + < \mathbf{B} > + < \boldsymbol{\mathcal{C}} > \\ &\quad + < \boldsymbol{\mathcal{O}} > + < {}_\ell\boldsymbol{\mathcal{X}} | \mathcal{F} > . \end{aligned} \quad (8)$$

6.2 Quality of PARAFAC Model

As discussed the above, we can choose k by coding cost, but we obtained very good results when k was chosen so that it would have good $CORCO(\cdot)$ value. Thus, we restrict the value of k, to the ones that show increase in the 'normalized' $CORCO(\cdot)$ score, defined as

$$q_k = k \times CORCO(\boldsymbol{\mathcal{C}}|k).$$

That is, we only consider k values, for which

$$q_k > q_{k-1}.$$

Thus we propose to solve the following optimization problem:

$$\begin{aligned} \mathcal{F} &= \arg\min_{\mathcal{F}'} < \mathcal{F}' > + < {}_\ell\boldsymbol{\mathcal{X}} | \mathcal{F}' > \\ &s.t.\ q_k > q_{k-1} \ and \ k \geq 1 \end{aligned} \quad (9)$$

[4]Here, $\log^*$ is the universal code length for integers.

Algorithm 3 AutoCyclone ($\mathbf{X}$)

Input: Time-series $\mathbf{X}$
Output: Parameter set $\mathcal{F} = \{\ell, k, \lambda_1, \lambda_2, \mathbf{B}, \boldsymbol{\mathcal{C}}, \boldsymbol{\mathcal{O}}\}$

```
c ← 0
/* get candidates of ℓ */
L ← PeriodgramAnalysis(X) ∪ L_0
for all ℓ ∈ L do
    ℓX ← f(X, ℓ)
    F_ℓ, c_ℓ ← AC-Fit(ℓX)
    if c_ℓ < c then
        F ← F_ℓ; c ← c_ℓ /* c_ℓ is (8) of F with ℓ */
    end if
end for
return F
```

Algorithm 4 AC-Fit(${}_\ell\boldsymbol{\mathcal{X}}$)

Input: Cyclic folding tensor ${}_\ell\boldsymbol{\mathcal{X}}$
Output: Parameter set $\mathcal{F}_\ell = \{\ell, k, \lambda_1, \lambda_2, \mathbf{B}, \boldsymbol{\mathcal{C}}, \boldsymbol{\mathcal{O}}\}$

```
K ← min(dℓ, ℓm, md); c_ℓ ← ∞; λ_2 ← 0.1(d/2)
while description cost can be reduced. do
    c* ← ∞; c_0 ← ∞; q_0 ← 0
    for k = 1 : K do
        c_k ← ∞; λ_1 ← ε /* ε = 2 × 10^-4 */
        while description cost can be reduced. do
            F'_k ← CycloneFact(ℓX, k, λ_1, λ_2)
            c'_k ← < ℓX; F'_k >   /* compute (8) */
            q'_k ← k × CORCO(C'_k|k)
            if q'_k > q_{k-1} and c'_k < c_k then
                c_k ← c'_k; q_k ← q'_k; F_k ← F'_k
            end if
            λ_1 ← γ_1 λ_1   /* γ_1 = 10 */
        end while
        if c_k > c_{k-1} then
            break for loop;
        end if
        c* ← c_k; F* ← F_k; λ_2 ← γ_2 λ_2   /* γ_2 = 0.1 */
    end for
    if c* < c' then
        c_ℓ ← c*; F_ℓ ← F*
    end if
end while
return (F_ℓ, c_ℓ)
```

6.3 AutoCyclone

We propose a multi-layer optimization framework AutoCyclone (Algorithm 3), which searches for a good parameter set $\mathcal{F}$ based on minimum description length.

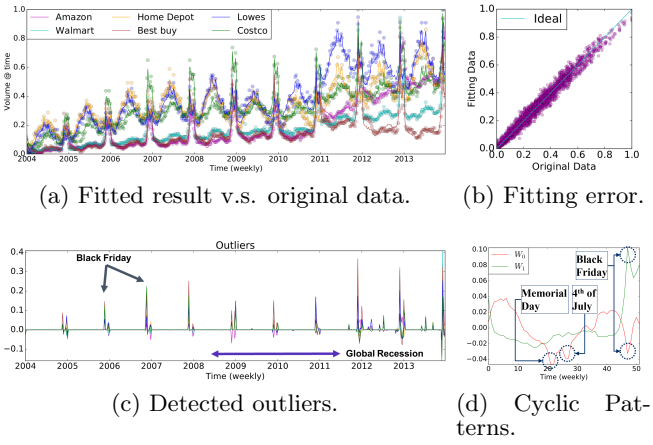
AutoCyclone contains four parameters $\ell, k, \lambda_1, \lambda_2$. In the inner loop of AutoCyclone, AC-Fit (Algorithm 4) finds the best combination of k, λ_1 and λ_2 for a fixed ℓ. Then we choose $\mathcal{F}$ minimizing (9).

After getting the best result from AC-Fit, AutoCyclone searches for the best possible period ℓ with minimum encoding cost. We choose the best period in a set of periods which are detected by Fourier periodogram analysis and universal periods $\mathcal{L}_0$[5] for each record type, such as monthly, weekly and daily.

[5]For monthly, weekly and daily records, we set $\mathcal{L}_0 = \{12, 6\}$, $\{52, 26\}$, $\{365, 182\}$, respectively.

Table 3: Datasets.

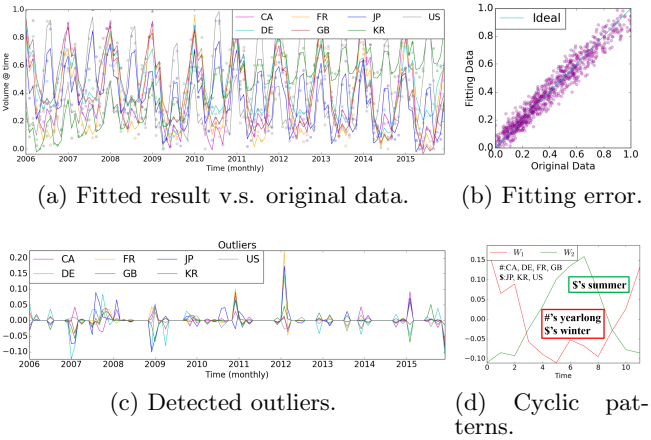
#	Dataset	d	n	record type
#1	Google Trends: Sports	3	520	weekly
#2	Google Trends: Retail companies	6	551	weekly
#3	Energy Consumption [1]	7	120	monthly
#4	Sea Surface Temperature [2]	5	3652	daily



(a) Fitted result v.s. original data. (b) Fitting error.

(c) Detected outliers. (d) Cyclic Patterns.

Figure 3: **AutoCyclone is effective** for online user activities about retail companies.



(a) Fitted result v.s. original data. (b) Fitting error.

(c) Detected outliers. (d) Cyclic patterns.

Figure 4: **AutoCyclone is effective** for general seasonal time-series about energy consumptions.

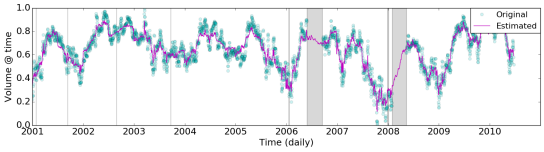


Figure 5: **AutoCyclone is Doubly Robust**: Not only against outliers, but *also* against missing values (gray zones in figures), AutoCyclone predicts smooth sequences on the missing values.

7. EXPERIMENTS

In this section we demonstrate the effectiveness of AutoCyclone with real data. The experiments were designed to answer following questions:

- Q1 *Effectiveness*: How successful is our method in distinguishing regular cyclic patterns from outliers?
- Q2 *Accuracy*: How accurate is our method compared to existing methods?
- Q3 *Scalability*: How does our method scale in terms of computational time?

Datasets. We used 4 real datasets whose characteristics are described in Table 3. *Google Trends* consists of a set of sequences which are activities of keywords from different topics. Datasets #1 and #2[6] are from Google Trends. Energy Consumption dataset (#3), which we use here, contains monthly records between 2006 and 2015. For SST (#4), we downloaded 5 daily records including some missing values duration between 2001 and 2010. Note that the dataset is scaled such that each sequence has a peak volume of 1.0.

7.1 Q1. Effectiveness

We now demonstrate how well our model captures the cyclic characteristics and important patterns. We show the time-series obtained by cyclic unfolding for the result of AutoCyclone. All parameters $\ell, k, \lambda_1, \lambda_2$ are automatically set by AutoCyclone.

The results for the "sports" data (#1) has already been presented in Figure 1.

Observation 1. *(Seasonality and anomalies in Sports.) Our* CycloneFact *captures seasonal patterns (e.g., swimming peaks each July), as well as rare patterns ("Olympics").*

Figure 3 shows the results for the top six retail companies (#2), Amazon (red), Wallmark (blue), Home Depot (yellow), Best buy (brown), Lowes (deep blue), and Costco (green). All activities are well fitted and smooth (Figure 3a and 3b).

Observation 2. *(Seasonality and anomalies in retail companies.) Our* AutoCyclone *spots seasonal patterns (Figure 3c) like "Black Friday"[7] (end of each November).*

The "Black Friday" spikes have downward trends from 2007 to 2009 and upward trends from 2009 to 2012 which is likely due to the global recession between 2008 and 2010. Further, the cyclic pattern tensor (seasonality $\mathbf{W}$) has a small spikes on "Memorial Day" and "4th of July". On "Black Friday", there are spikes having different intensity.

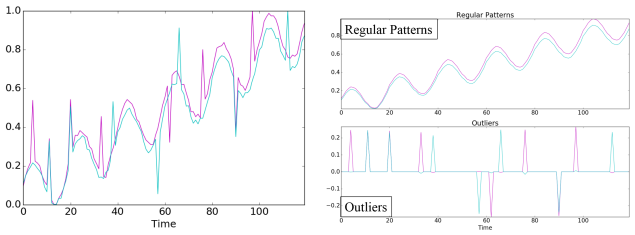
Generality: beyond online user activities. Figure 4 shows the results for the monthly energy consumptions of seven countries.

Observation 3. *(Seasonality in national energy consumptions.)* AutoCyclone *captured yearly periodical dynamics, that is Northern countries (CA, DE, FR and GB) have peaks in winter, while JP, KR and US have peaks in both summer and winter (i.e. combination of W_1 and W_2).*

The up-trend of energy consumptions for KR, and the down-trend for JP were observed in Figure 4a. The detected outliers includes several small spikes and a few remarkable

[6]Dataset #2 was also used in [16].

[7]"Black Friday" is a big, annual sale event at the 4th Friday of each November.



(a) Synthetic time-series. (b) Detected patterns and outliers

Figure 6: AUTOCYCLONE decomposes (a) synthetic time-series including Gaussian noises, and (b) identifies all spikes as outliers.

Table 4: Anomaly Classification.

	precision@15	TPR	FPR
AUTOCYCLONE	1.000	1.000	0.0466
RoPCA	0.467	0.600	0.0466

spikes (Figure 4c). In early 2012, a major cold wave occurred in Europe, which agrees with the large anomalies that we detect in FR and DE.

Robustness in the presence of missing values. We here demonstrate how well our model captures the cyclic characteristics and important patterns against missing values. Here we used SST, which includes some missing values. Figure 5 shows the results for SST.

OBSERVATION 4. *(Recovery of missing values.) Even if there are missing values, the data well-fitted and the missing values are very smoothly completed.*

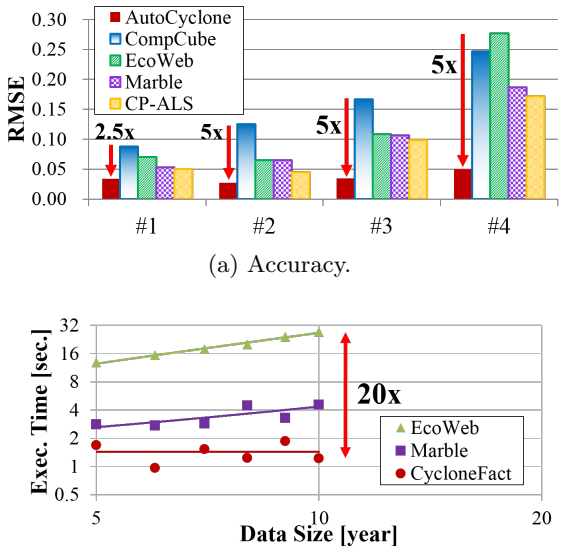
Above all, not only against outliers, but also against missing values, AUTOCYCLONE can estimate smooth sequences.

7.2 Q2. Accuracy

In this section, we discuss the fitting accuracy of AUTOCYCLONE. We measure the classification accuracy of outliers, and the fitting error by RMSE.

First, Table 4 displays the classification accuracy by precision@15, true positive rate (TPR) and false positive rate (FPR) for the synthetic data including 15 spikes shown in Figure 6a. Here, we assume that non zero in $\mathcal{O}$ is positive (anomaly), and higher absolute values are more anomalous. We compared AUTOCYCLONE and RoPCA. Table 4 shows that AUTOCYCLONE distinguished very clearly with high accuracy as well as 6b. Since AUTOCYCLONE considered cyclic patterns by periodic folding, its classification accuracy (i.e., precision@15 and TPR) was better than RoPCA.

Next, in terms of fitting accuracy, we measured RMSE. Here, we compared AUTOCYCLONE, CompCube [17], EcoWeb [16], Marble [5] and CP–ALS. Similarly to our base trend matrix $\mathbf{B}$, Marble employs an augmented tensor to capture overall trends, while EcoWeb and CompCube employ an nonlinear dynamical systems model to capture interactions between the time-series given by different keywords. Marble actually does not minimize least squares error, but we utilized it to measure relative goodness of AUTOCYCLONE. There is a successor of Marble, namely, Rubik, but it is a semi-supervised approach. We thus compare with Marble



(a) Accuracy.

(b) Scalability.

Figure 7: **AutoCyclone outperforms competitors**: (a) AUTOCYCLONE (red) is the most accurate and up to 5 times more accurate than the other methods, (b) CYCLONEFACT (red) is up to 20 times faster than the other methods.

on the unsupervised basis. We employed all competitors developed in Python 3.5.

For EcoWeb and CompCube, they set all parameters by their own self-tuning ways. For tensor factorization methods, Marble and CP–ALS, they factorized $_{\ell}\mathcal{X}$ by using the same parameters as AUTOCYCLONE.

Figure 7a shows the RMSE between original data and estimated data. As shown in the figure, our approach achieved very high accuracy.

Dataset #2 may follow the ecosystems (i.e. competitions) between attributes, which EcoWeb and CompCube assume. Since EcoWeb and CompCube estimated such ecosystem by the specific nonlinear dynamical systems, showing reasonable accuracy, but AUTOCYCLONE still performed better.

Since Marble and CP–ALS cannot capture seasonal characteristics, their error was generally larger than AUTOCYCLONE.

As shown by the above results, AUTOCYCLONE could accurately distinguish regular cyclic patterns from outliers, and estimate original time-series.

7.3 Q3. Scalability

We also measured the scalability of our method. We used the dataset whose size (number of periods) was varied from five to ten years. Because some methods do not have the way of automatic parameter selection, we here do not utlize such automations, but a fixed parameter set. This experiment was run on a machine which has 3.2 GHz quad core CPU, 16GB main memory, 1TB HDD and MacOS 10.10 operating system.

Figure 7b shows the average execution time of CYCLONEFACT. We compared execution times between competitors. We observed that CYCLONEFACT was linear with respect to data length n. Moreover, CYCLONEFACT was up to 3 times faster than Marble, and 20 times faster than EcoWeb.

8. CONCLUSIONS

We presented CycloneM, an intuitive cyclic model for mining large scale co-evolving time series containing seasonal patterns. Our main idea is that time-series having seasonality consists of both cyclic regular patterns and local rare events (outliers). Further, in the proposed algorithm, CycloneFact, robust estimation of the cyclic characteristics by cyclically folding the tensor detecting and removing outliers showed some good results. AutoCyclone also automatically tunes the parameter set. Our proposed method has the following appealing properties:

1. **Effective**: CycloneFact detects important characteristics, such as trends and seasonal patterns, and distinguishes regular patterns from outliers.
2. **Robust and Accurate**: CycloneFact detects the above characteristics and patterns accurately and robustly in the presence of outliers and missing values.
3. **Fast**: CycloneFact is linear on the input size.
4. **Parameter-free**: AutoCyclone chooses all parameters of CycloneFact to automatically achieve high accuracy and intuitiveness.

Acknowledgments

The authors would like to thank the anonymous refrees for their valuable comments and helpful suggestions. This material is based upon work supported by the National Science Foundation under Grants No. IIS-1247489, CNS-1314632 and IIS-1408924, by the Army Research Laboratory under Cooperative Agreement Number W911NF-09-2-0053, and by a Google Focused Research Award. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation, or other funding parties. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

9. REFERENCES

[1] Monthly electricity statistics, international enegy agency. `http://www.iea.org/statistics/monthlystatistics/monthlyelectricitystatistics/`.

[2] Tropical atomosphere ocean project. `http://www.pmel.noaa.gov/tao/data_deliv/deliv.html`.

[3] R. Bro and H. A. Kiers. A new efficient method for determining the number of components in parafac models. *Journal of chemometrics*, 17(5):274–286, 2003.

[4] W. Cheng, K. Zhang, H. Chen, G. Jiang, and W. Wang. Ranking causal anomalies via temporal and dynamical analysis on vanishing correlations. In *KDD*, 2016.

[5] J. C. Ho, J. Ghosh, and J. Sun. Marble: high-throughput phenotyping from electronic health records via sparse nonnegative tensor factorization. In *KDD*, pages 115–124, 2014.

[6] T. Idé, A. C. Lozano, N. Abe, and Y. Liu. Proximity-based anomaly detection using sparse structure learning. In *SDM*, pages 97–108, 2009.

[7] R. Jiang, H. Fei, and J. Huan. Anomaly localization for network data streams with graph joint sparse pca. In *KDD*, pages 886–894, 2011.

[8] U. Kang, E. Papalexakis, A. Harpale, and C. Faloutsos. Gigatensor: scaling tensor analysis up by 100 times-algorithms and discoveries. In *KDD*, pages 316–324, 2012.

[9] E. Keogh, S. Chu, D. Hart, and M. Pazzani. An online algorithm for segmenting time series. In *ICDM*, pages 289–296, 2001.

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

[11] L. Li, B. A. Prakash, and C. Faloutsos. Parsimonious linear fingerprinting for time series. *PVLDB*, 3(1-2):385–396, 2010.

[12] Y.-R. Lin, J. Sun, H. Sundaram, A. Kelliher, P. Castro, and R. Konuru. Community discovery via metagraph factorization. *TKDD*, 5(3):17, 2011.

[13] G. Mateos and G. B. Giannakis. Robust pca as bilinear decomposition with outlier-sparsity regularization. *IEEE Transactions on Signal Processing*, 60(10):5176–5190, 2012.

[14] Y. Matsubara and Y. Sakurai. Regime shifts in streams: Real-time forecasting of co-evolving time sequences. In *KDD*, 2016.

[15] Y. Matsubara, Y. Sakurai, and C. Faloutsos. Autoplait: Automatic mining of co-evolving time sequences. In *SIGMOD*, pages 193–204, 2014.

[16] Y. Matsubara, Y. Sakurai, and C. Faloutsos. The web as a jungle: Non-linear dynamical systems for co-evolving online activities. In *WWW*, pages 721–731, 2015.

[17] Y. Matsubara, Y. Sakurai, and C. Faloutsos. Non-linear mining of competing local activities. In *WWW*, pages 737–747, 2016.

[18] Y. Matsubara, Y. Sakurai, W. G. van Panhuis, and C. Faloutsos. Funnel: automatic mining of spatially coevolving epidemics. In *KDD*, pages 105–114, 2014.

[19] E. E. Papalexakis. Automatic unsupervised tensor mining with quality assessment. In *SDM*, 2016.

[20] E. E. Papalexakis and C. Faloutsos. Fast efficient and scalable core consistency diagnostic for the parafac decomposition for big sparse tensors. In *ICASSP*, pages 5441–5445, 2015.

[21] M. Rogers, L. Li, and S. J. Russell. Multilinear dynamical systems for tensor time series. In *NIPS*, pages 2634–2642, 2013.

[22] P. Wang, H. Wang, and W. Wang. Finding semantics in time series. In *SIGMOD*, pages 385–396, 2011.

[23] Y. Wang, R. Chen, J. Ghosh, J. C. Denny, A. Kho, Y. Chen, B. A. Malin, and J. Sun. Rubik: Knowledge guided tensor factorization and completion for health data analytics. In *KDD*, pages 1265–1274, 2015.

[24] J. Yang, J. McAuley, J. Leskovec, P. LePendu, and N. Shah. Finding progression stages in time-evolving event sequences. In *WWW*, pages 783–794, 2014.

[25] L. Ye and E. Keogh. Time series shapelets: a new primitive for data mining. In *KDD*, pages 947–956, 2009.

[26] M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2006.