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 CYCLONE, 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 CYCLONE 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: CYCLONE 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.

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


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 sensitivity results in that regular patterns include such intensive 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.

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: \(A_{i,j}\) denotes the \((i,j)\)-th element of matrix \(A\), whereas \(A_{:,j}\) denotes the \(j\)-th column of \(A\).
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 \( \mathbf{m} \) is a sensible way for principal component analysis to be robust against outliers.
- Outlier sparsity estimation via group Lasso [26] provides robustification.

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

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

where \( \mathbf{S} \) and \( \mathbf{U} \) are low-rank approximated matrices, \( \mathbf{O} \) is the outliers, \( \lambda_2 \) is the outlier sparsity controlling parameter, \( \| \mathbf{O} \|^2_{2,r} \) is the row-wise group lasso penalty, and \( \mathbf{V} = \{ \mathbf{m}, \mathbf{U}, \mathbf{S} \} \).

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 \( \mathbf{X} \) of dimension \( I \times J \times K \).

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

\[
[a \circ b \circ c]_{i,j,k} = a_i b_j c_k.
\]

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

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

where \( \mathbf{A}_r \) is the \( r \)-th column of matrix \( \mathbf{A} \) and \( \mathbf{a} \) 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 \( \mathbf{a} \).

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 \( \mathbf{X} \) 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.
4. PROPOSED MODEL

In this section, we present our proposed model, namely, CYCLONE M. Consider that we have a collection of activity volumes \( X \) of \( d \) keywords, with duration \( n \). That is, we have \( X = \{x_1, \ldots, x_i, \ldots, x_d\} \), where \( x_i = \{x_i(t)\}_{t=1}^{n} \) is a sequence of activity about keyword \( i \). Given a set of seasonal time series \( X \) having cyclic period \( \ell \), our goal is to (a) capture the dynamics of \( X \), (b) find the hidden cyclic characteristics of \( X \), and (c) distinguish regular cyclic patterns from rare events.

To capture the cyclic dynamics of \( X \) on the view of cyclic nature, we transform \( X \) into a tensor \( \mathbf{X} \), which stacks periodically segmented subsequences of \( X \). We call this transformation from a matrix to the tensor, Cyclic Folding. Then, we propose a cyclic model of the time-series sequences for each period and each attribute. Figure 2 shows the structure of our model. In the above approximation of \( \mathbf{X} \), \( \mathbf{B} \) and \( \mathbf{C} \) capture trends and seasonalties, respectively. Then \( \mathbf{B} \) and \( \mathbf{C} \) represent regular behaviors of \( \mathbf{X} \) 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 is activities which do not follow regular cyclic seasonality, that is \( \mathbf{O} = \mathbf{X} - \mathbf{B} - \mathbf{C} \). Therefore, it takes \( d \times m \times \ell \) space: \( \mathbf{O} \in \mathbb{R}^{d \times m \times \ell} \). However, \( \mathbf{O} \) should capture only abnormal behavior. Assuming such remarkable abnormal behaviors are rare, \( \mathbf{O} \) 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 CYCLONE M. As we mentioned previously, we need to answer (i) How can we robustly find important patterns and seasonal characteristics (i.e., \( \mathbf{B}, \mathbf{C} \)) 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 \( \mathbf{C} \).
- Low-rank approximation with sparse representation for cyclic pattern tensor \( \mathbf{C} \).

Thus, the objective function we solve is:

\[
\mathcal{Z} = \text{arg min}_{\mathbf{X}'} (||\mathbf{X} - \mathbf{X}'||_F^2 + \lambda_1 \sum_{r=1}^{k} (||\mathbf{U}_r||_1 + ||\mathbf{V}_r||_1) + \lambda_2 \sum_{g \in G} ||\mathbf{O}_g||_2)
\]  

where \( \mathcal{Z} = \mathbf{B} + \mathbf{C} + \mathbf{O} \), \( \lambda_1 \) and \( \lambda_2 \) are sparsity control parameters for \( \mathbf{C} \) and \( \mathbf{O} \), respectively. \( g \in G \) is a group which shares the outlier sparsity. The first penalty term enforces sparsity in \( \mathbf{C} \) by the conventional \( L_1 \) (lasso) regularization. The second penalty term enforces sparsity in \( \mathbf{O} \) 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 CYCLONE FACT is:

\[
\mathcal{F} = \{\ell, k, \lambda_1, \lambda_2, \mathbf{B}, \mathbf{C}, \mathbf{O}\}
\]
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 \( B, C \), 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 \( \mathcal{O} \).

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, we encourage the grouped elements to be sparse at the same time.

We can find a compact and reasonable description of \( \mathcal{X} \) based on our CYCLONEFACT model. Specifically, the problem we want to solve is as follows:

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

Thus, the objective function we want to minimize is:

\[
\mathcal{F} = \arg \min_{\mathcal{F}^*} < \mathcal{F}^* > + < i \mathcal{X} | \mathcal{F}^* > .
\]

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 \( B \) for the outlier subtracted tensor \( \mathcal{Z}_\mathcal{O} \). Next, we compute sparse CP-ALS (Algorithm 2) to get sparse compact descriptions of cyclic patterns \( C \) (FactorizeC). To induce sparsity, we use LASSO (\( \ell_1 \)) regularization, which is the standard sparsifying technique. In FactorizeC, we use soft-thresholding. The soft-thresholding update for sparse representation at level \( \lambda_1 \) is:

\[
x = \text{sgn}(x)(|x| - \lambda_1)_+
\]

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

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

\[
o_{\tau, w} = \frac{o_{\tau, w}(|g_{\tau, w}| - \lambda_2/2)_+}{|g_{\tau, w}|}.
\]

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

Algorithm 1 describes the overall procedure of CYCLONEFACT. Given input parameters \( k, \lambda_1, \lambda_2 \), CYCLONEFACT estimates the full component set of CYCLONEM, \( \mathcal{F} \).

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 \( \mathcal{X} \) based on our CYCLONEM model. Specifically, the problem we want to solve is as follows:

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

Thus, the objective function we want to minimize is:

\[
\mathcal{F} = \arg \min_{\mathcal{F}^*} < \mathcal{F}^* > + < i \mathcal{X} | \mathcal{F}^* > .
\]

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

We discuss about model quality by 2 separate parts. First, to choose the best \( k, \lambda_1, \lambda_2 \) and \( \ell \), 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 \( d \times \ell \times C_p \) bits, where \( C_p \) is the number of bits required to encode a floating point number. \( C \) consists of \( [\alpha; U; V; W] \). \( \alpha \) needs \( k \times C_P \) bits. \( \mathbf{U}_r \), which is a sparse vector with size \( d \), contains \( N_{i,r} \) nonzeros, and for each nonzero we need \( \text{log}(d) \) bits to encode its position and \( C_p \) bits to encode its value. Since the number \( \text{We assume } C_p=8 \text{ bits by following [16].} \)
of non-zeros \( N_{i,r} \) ranges 0 to \( d \), it requires \( \log(d+1) \) bits to encode. \( V_i \) is similar. Since \( W \) is non-sparse, we encode it in a non-sparse manner, requiring \( k \times \ell \times C_F \) bits. Outlier tensor \( O \) 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 \( O \) 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 \( \log^*(k) \) bits. The period \( \ell \) requires \( \log(n) \) bits since \( \ell \) is chosen from 1 to \( n \). The description complexity of model parameter set consists of the following terms,

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

The total model description cost \( <F> \) is the sum of the above terms.

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

\[
<\mathcal{X}|F> = 2C_F + \sum_{\tau,w} -\log_2 p_{\text{Gauss}}(\mu,\sigma^2)(\mathcal{X}_{i,\tau,w}-Z_{i,\tau,w})
\]

where \( Z_{i,\tau,w} = B_{i,\tau,w} + C_{i,\tau,w} + O_{i,\tau,w} \), and \( 2C_F \) is coding cost of \( \mu \) and \( \sigma \).

Finally, the total encoding cost \( <\mathcal{X}|F> \) is given by:

\[
<\mathcal{X}|F> = <F> + <\mathcal{X}|F>
= <\ell> + <k> + <B> + <C> + <O> + <\ell>C_F
\]

### 6.2 Quality of PARAFAC Model

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

\[
q_k = k \times \text{CORCO}(C[k]).
\]

That is, we only consider \( k \) values, for which

\[
q_k > q_{k-1}.
\]

Thus we propose to solve the following optimization problem:

\[
\mathcal{F} = \arg \min_{F'} <F'> + <\mathcal{X}|F'> \quad \text{s.t. } q_k > q_{k-1} \text{ and } k \geq 1
\]

Here, \( \log^* \) is the universal code length for integers.

---

**Algorithm 3 AUTOCYCLONE (X)**

**Input:** Time-series \( \mathbf{X} \)

**Output:** Parameter set \( F = \{\ell, k, \lambda_1, \lambda_2, B, C, O\} \)

1. \( c \leftarrow 0 \)
2. /* get candidates of \( \ell / \)
3. \( \ell \leftarrow \text{PeriodogramAnalysis}(\mathbf{X}) \cup \mathcal{L}_0 \)

4. for all \( \ell \in \mathcal{L} \) do

5. \( \mathcal{X} \leftarrow f(X, \ell) \)

6. \( F_c, c_c \leftarrow \text{AC-Fit}(\mathcal{X}) \)

7. if \( c_c < c \) then

8. \( F \leftarrow F_c; c \leftarrow c_c / * c_c is (8) of \mathcal{F} with \ell / * \)

9. end if

10. end for

11. return \( F \)

---

**Algorithm 4 AC-Fit(\mathcal{X})**

**Input:** Cyclic folding tensor \( \mathcal{X} \)

**Output:** Parameter set \( F_c = \{\ell, k, \lambda_1, \lambda_2, B, C, O\} \)

1. \( K \leftarrow \min(\ell, \ell_m, md); c_k \leftarrow \infty; \lambda_2 \leftarrow 0.1(d/2) \)

2. while description cost can be reduced.

3. \( c^* \leftarrow \infty; c_0 \leftarrow \infty; c_0 \leftarrow 0 \)

4. for \( k = 1 : K \) do

5. \( c_k \leftarrow c_0; \lambda_1 \leftarrow c / * c \leftarrow 2 \times 10^{-4} / *

6. while description cost can be reduced.

7. \( F_{k_0} \leftarrow \text{CYCLONEFACT}(\mathcal{X}, k, \lambda_1, \lambda_2) \)

8. \( c_k \leftarrow c_{k_0}; F_{k_0} > /* compute (8) */

9. \( q_k^* = k \times \text{CORCO}(C[k]) \)

10. if \( q_k^* > q_{k-1} \) and \( c_k < c \) then

11. \( c_k \leftarrow c_k; q_k \leftarrow q_k^*; F_k \leftarrow F_{k_0} \)

12. end if

13. \( \lambda_1 \leftarrow \gamma_1 \lambda_1 / * \gamma_1 = 10 / *

14. end while

15. if \( c_k > c_{k-1} \) then

16. break for loop;

17. end if

18. \( c^* \leftarrow c_k; F^* \leftarrow F_k; \lambda_2 \leftarrow \gamma_2 \lambda_2 / * \gamma_2 = 0.1 / *

19. end for

20. if \( c^* < c \) then

21. \( c_k \leftarrow c^*; F_k \leftarrow F^* \)

22. end if

23. end while

24. return \((F_c, c_c)\)

---

6.3 AUTOCYCLONE

We propose a multi-layer optimization framework AUTOCYCLONE (Algorithm 3), which searches for a good parameter set \( F \) 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 \( k, \lambda_1 \) and \( \lambda_2 \) for a fixed \( \ell \). Then we choose \( F \) minimizing (9).

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

3For monthly, weekly and daily records, we set \( \mathcal{L}_0 = \{12, 6\}, \{52, 26\}, \{365, 182\} \), respectively.
Table 3: Datasets.

<table>
<thead>
<tr>
<th>#</th>
<th>Dataset</th>
<th>d</th>
<th>n</th>
<th>record type</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>Google Trends: Sports</td>
<td>3</td>
<td>520</td>
<td>weekly</td>
</tr>
<tr>
<td>#2</td>
<td>Google Trends: Retail companies</td>
<td>6</td>
<td>551</td>
<td>weekly</td>
</tr>
<tr>
<td>#3</td>
<td>Energy Consumption [1]</td>
<td>7</td>
<td>120</td>
<td>monthly</td>
</tr>
<tr>
<td>#4</td>
<td>Sea Surface Temperature [2]</td>
<td>8</td>
<td>3652</td>
<td>daily</td>
</tr>
</tbody>
</table>

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

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 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”).

(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.

Observation 2. (Seasonality and anomalies in retail companies.) Our AutoCyclone spots seasonal patterns (Figure 3c) like “Black Friday” (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 $W$) 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

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

7"Black Friday" is a big, annual sale event at the 4th Friday of each November.
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 $O$ 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 $B$, 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 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 $\ell_X$ 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 utilize 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 (red) is up to 20 times faster than the other methods.
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**: AUTO CYCLONE chooses all parameters of CYCLONEFACT to automatically achieve high accuracy and intuitiveness.

Acknowledgments

The authors would like to thank the anonymous referees 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