Abstract

Period detection from time series is an important problem with many real-world applications such as weather forecast, stock market predictions, electrocardiogram analysis, periodic disease outbreak. In this work, we present a novel approximate period detection method for time series. The simplicity of our algorithm and its adaptability for high dimensional datasets using renowned tools and techniques such as locality sensitive hashing (LSH) and MapReduce (using the Hadoop framework for example) make it easier to implement for practical purposes. We performed experiments to compare our technique with a classic period detection technique and two state-of-the-art techniques in terms of accuracy and noise-resilience.

1 Introduction

Periodicity is the repetition of a pattern at regular intervals; such regular interval is referred to as a period. We are concerned with the detection of both the period and the periodic pattern in this work. Periodicity can be found in medicine, agriculture, financial market and day-to-day human activities. Examples include a commuter’s travel schedule, seasonal sales data, regional sunspot cycle, utility (water) consumption, electrocardiogram. Mining the periodicity in such phenomena can provide useful insights, help make better predictions, help determine structural similarity [17] and detect anomalies among other things. Most of the existing methods for time series periodicity detection assume perfect periodicity and can only detect single periods, which is hardly the case in most natural phenomena. Much of the world’s datasets contain mixed (multiple) periods in addition to being noisy and incomplete. This results in periodic patterns that are not identical, and the intervals between them may not be exactly the same lengths. The noisy characteristic of real datasets warrants more robust periodicity mining techniques.

A period detection method like the one presented in this work can be regarded as robust if it is able to detect mixed imperfect (approximate) periodicity with relatively high accuracy. In general, three types of periodic patterns can be found in literature: partial periodicity, symbol periodicity, and segment periodicity [14]. We adapt the definitions and describe the following types of periodicity in real-valued time series data. We focus on the discovery of segment periodicity in this work.

- Periodicity is partial if at least one data point in the period, in addition to at least one variable data point is periodic. For instance, in time series $S = [0, 1, 5, 1, 0, 1, 7, 7, 0, 1, 1, 9, 0, 1, 5, 7]$, the sequence $[0, 1]$ is periodic with period $p = 4$; and a partial periodic pattern $[1, 0, *, *, *]$ exists in $S$, where * denotes a variable symbol.

- A time series exhibits segment periodicity if the entire pattern is periodic. For instance, the time series $S = [0, 1, 2, 0, 1, 2, 0, 1, 2]$ has a segment period $p = 3$. The periodic segment is $[0, 1, 2]$.

Most existing periodicity mining techniques are designed for discrete sequences [13]. In previous work [13], we proposed an approximate period detection algorithm that utilizes SAX (Symbolic Aggregate approXimation) [11], a commonly used time series discretization technique, and grammar induction. While discretization reduces data complexity and simplifies computation, it also results in information loss. In this work, we propose a novel algorithm, APT, that bypasses the discretization step and detects multiple, full-cycle, approximate periodic patterns from the original time series directly. Our distance-based algorithm returns pruned candidate periods and patterns in a time series, and ranks them in the order of significance.

The remainder of the paper is organized as follows. Section 2 discusses related work. We outline preliminaries in Section 3. We describe our approach in Section 4. Experimental evaluation is presented in Section 5. We conclude and discuss future work in Section 6.
2 Related Work

Autocorrelation and Fourier Transforms are perhaps two of the most popular periodicity detection techniques. Autocorrelation-based technique is able to detect short and long periods, but has difficulty in identifying the true period due to the fact that the multiples of the true period will have the same power as the true period. On the other hand, Fourier transforms suffer from a number of problems: spectral leakage, which causes a lot of false positives in the periodogram, and poor estimation of long periods due to issues with low frequency regions or sparseness in data [10]. Some methods combine both autocorrelation and Fourier transforms [2, 4, 5, 8].

Some existing algorithms detect only the primary period [4, 8] while others detect all candidate periods [14, 17]. Some methods detect all three types of periodicity listed in the previous section [4, 15, 17, 18] while others detect only a subset. As an example, the methods proposed by Han et al. [7] and Elfeky et al. [4, 5], respectively, detect only segment periodicity. Yang et al. [18] proposed a linear-time distance-based technique to detect periods in a subsection of the time series otherwise referred to as partial periodicity. Techniques proposed in [2, 13] also detect partial periodicity. Most of the aforementioned techniques suffer from noise sensitivity. Rasheed et al. [17] proposed a noise-resilient algorithm to detect periodicity in time series using suffix trees. The time complexity for their proposed method can rise to the order of $O(n^2)$ and the multiple candidate periods returned are not ranked. WARP [5] was also developed to be noise-resilient but it has large space and time complexity.

3 Preliminaries

In this section we define periodicity, approximate periodicity and the problem addressed in this work.

Definition 1. Let $S = [t_0, t_1, \ldots, t_{n-1}]$ be a time series of length $n$, i.e. $|S| = n$ and $t_i \in \mathbb{R}$ for $i \geq 0$ and $i < n$. $S$ is said to be periodic if $S(t) = S(t + p)$, where $t < n - p$. $T$ is a periodic subsequence of $S$ such that, $T = [t_0, t_1, \ldots, t_{p-1}]$, i.e. $|T| = p, p \geq 1$ and $p \leq \frac{n}{2}$. |T| is called the period.

Definition 2. Let $S$ be $n$-long sequence of real numbers and $p$ be a period of $S$. We assume the periodic sequence $T$ to be the first $p$ points in $S$. We generate a periodic sequence $S_T$ by concatenating $T^{n-(n \mod p)}$ times and to the first $(n \mod p)$ data points of $T$.

Definition 3. Let $S$ be $n$-long sequence of real numbers. Let $r$ be a ranking function defined on real numbers. $T$ is a periodic pattern of $S$ with $\epsilon$ error on $S$ if there exists a subsequence $T'$ such that $r(S, S_T) = \epsilon$ i.e. $r$ evaluates the rank of assuming $T$ is the correct period of $S$ relative to other candidate periods. The subsequence $T'$ is called an approximate periodic pattern of $S$ with error $\epsilon$. If there is another periodic pattern $Q, T$ is a more significant periodic sequence if $r(S, S_T) < r(S, S_Q)$ and vice versa.

Definition 4. We use vector notation for all occurrences of a set of periodic subsequences in a sequence of real numbers. Let vectors $v_1, v_2, \ldots, v_z$ represent $z$ such sub patterns. We define the average periodic pattern, $v_a$ as the mean of the $z$ vectors i.e.

$$v_a = \frac{v_1 + v_2 + \ldots + v_z}{z}$$  

We can now define our approximate period detection problem:

Definition 5. Given a function $r$, and time series $S$ of length $n$ over real numbers, compute the approximate periodic patterns $T$ and corresponding periods $|T|$ under the function $r$.

4 Our Approach

We will describe the fundamental premise of our approach and our algorithm in this section. Assume $T$ is a perfect periodic pattern of a given time series. Let us also assume $Q$ and $P$ are approximate periods of $S$ where $Q$ is a more significant period. We can generate periodic time series $S_T$, $S_Q$ and $S_P$ respectively according to definition 2. The fundamental premise of our approach is that the distance between $S$ and $S_T$ is zero, assuming that $T$ starts at the beginning of the time series. Analogous to the previous statement, $\text{Distance}(S, S_Q) < \text{Distance}(S, S_P)$. Algorithm 1 shows the pseudocode of our technique and we describe it subsequently in this section.

We iterate over period values from 2 to \(\frac{n}{2}\) in Line 6. Periodic sequences for each period is generated on each iteration, e.g. if $S = [0, 1, 2, 0, 1, 2, 0, 1]$ and the current iteration has $i = 2$, the periodic time series of $S$ on $|T| = 2$ is $[0, 1, 0, 1, 0, 1, 0, 1]$. The distance between each generated sequence and $S$ is computed on each iteration. The period values are then ranked according to their distances from $S$. The period corresponding to the smallest distance is the most significant. We use Euclidean distance as the distance function in this experiment due to its widely acclaimed success and simplicity. We implement early abandonment [9] to speed up the distance computations. More specifically, the distance function in Line 7 accepts a threshold (dist) as a third parameter and terminates the distance computation if the distance computed so far equals to or is greater than the dist value. Algorithm 2 shows the algorithm listing for the euclideanDist function called in Line 7 of Algorithm 1. The data structure, patt, initialized in Line 2.
Algorithm 1 APT algorithm

1: procedure APT(S) \( S = [t_0, t_1, ..., t_n-1] \)
2: \( \text{patt} \leftarrow \text{new list of period objects} \)
3: \( \text{dist} \leftarrow \infty \)
4: \( \text{period} \leftarrow 2 \)
5: \( \text{cDist} \leftarrow 0 \)
6: for \( i \leftarrow 2, \frac{n}{2} \) do \( \triangleright \) get candidate periods in this loop
7: \( \text{cDist} \leftarrow \text{euclideanDist}(S, i, \text{dist}) \)
8: if \( \text{cDist} < \text{dist} \) then
9: \( \text{dist} \leftarrow \text{cDist} \)
10: if \( i = \text{period} + 1 \) then \( \triangleright \) prune candidates
11: \( \text{patt}.\text{remove}(\text{patt}.\text{size}() - 1) \)
12: end if
13: \( \text{period} \leftarrow i \)
14: \( \text{patt}.\text{add}((\text{newPeriod} \text{dist}, \text{period}, \text{null})) \)
15: end if
16: end for \( \triangleright \) now iterate over the candidate periods
17: for \( x \leftarrow 1, \ y \) do \( \triangleright \) y is the # of candidate periods
18: compute the average periodic pattern for each candidate according to equation \[1\]
19: end for
20: sort \( \text{patt} \) \( \triangleright \) sort on dist field in ascending order
21: return \( \text{patt} \) \( \triangleright \) return ranked candidate periods
22: end procedure

Algorithm 2 euclideanDist algorithm

1: procedure \text{EUCLIDEANDIST}(S,i,\text{dist})
2: \( \text{newdist} \leftarrow 0 \)
3: for \( j \leftarrow 1, n \) do
4: \( \text{newdist} \leftarrow \text{newdist} + (t_j - t_{j \text{mod } i})^2 \)
5: if \( \text{newdist} \geq (\text{dist})^2 \) then
6: return \( \text{dist} \)
7: end if
8: end for
9: return square root of \( (\text{newdist}) \)
10: end procedure

4.1 APT AND IMPLICATIONS FOR HIGH DIMENSIONAL TIME SERIES

As stated earlier, our algorithm is distance based in the Euclidean space. If we store the original time series and the generated periodic sequences for each potential period according to definition \[2\] we can reduce the period detection problem to the popular approximate nearest neighbor problem (ANN). The original time series will be the query point while all the periodic sequences will constitute the search pool and the periods of the nearest periodic sequences will be candidate periods. A number of methods have been developed for the ANN problem in high dimension. For example, the method in \[3\] involves the use of the well-acclaimed locality sensitive hashing (LSH) and the method in \[1\] uses kd-trees, box-decomposition trees, and other search strategies. These methods also support Euclidean distance computations. Implementations of these methods are readily available and they have been widely used with success. This makes APT suitable for efficient period detection in high dimension. This adaptation for period detection in high dimension as described here leads to increased memory requirement to hold all the periodic sequences but reduced time complexity. Much of the speed will be gained by not performing the distance computation on the original high dimension data. Instead, the ANN technique adopted computes the nearest neighbor efficiently in a lower dimension size. It is worth mentioning that there are cases where accuracy is important. In such cases, exact solution is desired. In fact, after performing experiments, Ferhan et al. \[16\] found that a brute-force approach should not be readily dismissed especially when high recall is desired. Speed could be further improved by implementing our algorithm in a high performance-computing paradigm such as MapReduce. The nearest neighbour search nature of our algorithm makes it suitable for MapReduce as seen in \[13\]. Space and avoidance of redundancy preclude the inclusion of such details in this work. Hence, we direct the interested reader to consult the references \[1, 3, 16\] if necessary and we will consider such experiments in the future.
5 Experiment

In this section we evaluate APT, a classic technique (Fast Fourier Transforms - FFT) and two state-of-the-art techniques (WARP [5] and MBPD [13]) on synthetic, and real datasets. Experiments were performed on a 2.7GHz, Intel Core i7, MAC OS X version 10.7.5 with 8GB memory.

With FFT, the frequency with the highest spectral power from FFT of the dataset is converted into time domain and considered as the most significant period. The period corresponds to the minimum warping cost in WARP as described in [11]. WARP required more memory space than what was available on our machine; hence we down sampled the datasets for WARP where necessary. To achieve this down sampling, we limited the size of all datasets to 1000 points and the periods of synthetic datasets to 100.

5.1 Datasets

We used 11 datasets to demonstrate the various factors that can affect the performance of a periodicity detection algorithm. Those factors include the type of periodicity, noise (insertion, deletion or modification), and lengths in our experimental evaluation. We ensured the length of each dataset is a power of 2 to avoid introducing bias by padding the dataset with zeros in order to use FFT for comparison.

Seven of the datasets are synthetic (S_ONE - S_SEVEN) with 65636 data points. We modified S_SIX and S_SEVEN to include mixed periods. The size of the real datasets range from 2048 - 262144 data points. More information about the datasets can be found in [13].

<table>
<thead>
<tr>
<th>Datasets</th>
<th>APT</th>
<th>WARP</th>
<th>FFT</th>
<th>MBPD</th>
</tr>
</thead>
<tbody>
<tr>
<td>S_ONE</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>S_TWO</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>S_THREE</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>S_FOUR</td>
<td>0.0001</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>S_FIVE</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>S_SIX</td>
<td>0.0000</td>
<td>-</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>S_SEVEN</td>
<td>0.0000</td>
<td>-</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>POWER</td>
<td>0.0012</td>
<td>-</td>
<td>0.0001</td>
<td>0.0000</td>
</tr>
<tr>
<td>MFCC</td>
<td>0.0000</td>
<td>-</td>
<td>0.0009</td>
<td>0.0000</td>
</tr>
<tr>
<td>SOLAR</td>
<td>0.0010</td>
<td>-</td>
<td>0.0001</td>
<td>0.0000</td>
</tr>
<tr>
<td>SUNSPOT</td>
<td>0.0024</td>
<td>-</td>
<td>0.0005</td>
<td>0.0000</td>
</tr>
</tbody>
</table>

5.2 Results

We evaluated the performance of our method against FFT, WARP and MBPD with respect to the ranking error rates on both the synthetic and real datasets as shown in Table 1.

Table 2: # of false positives on synthetic and real datasets

<table>
<thead>
<tr>
<th>Datasets</th>
<th>APT</th>
<th>WARP</th>
<th>FFT</th>
<th>MBPD</th>
</tr>
</thead>
<tbody>
<tr>
<td>S_ONE</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.5000</td>
<td>0.0005</td>
</tr>
<tr>
<td>S_TWO</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.5000</td>
<td>0.0007</td>
</tr>
<tr>
<td>S_THREE</td>
<td>0.0001</td>
<td>0.0000</td>
<td>0.5000</td>
<td>0.0056</td>
</tr>
<tr>
<td>S_FOUR</td>
<td>0.0001</td>
<td>0.0000</td>
<td>0.5000</td>
<td>0.0050</td>
</tr>
<tr>
<td>S_FIVE</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.5000</td>
<td>0.0011</td>
</tr>
<tr>
<td>S_SIX</td>
<td>0.0004</td>
<td>-</td>
<td>0.5000</td>
<td>0.0005</td>
</tr>
<tr>
<td>S_SEVEN</td>
<td>0.0008</td>
<td>-</td>
<td>0.5000</td>
<td>0.0003</td>
</tr>
<tr>
<td>POWER</td>
<td>0.0004</td>
<td>-</td>
<td>0.5000</td>
<td>0.0049</td>
</tr>
<tr>
<td>MFCC</td>
<td>0.0000</td>
<td>-</td>
<td>0.5000</td>
<td>0.0018</td>
</tr>
<tr>
<td>SOLAR</td>
<td>0.0001</td>
<td>-</td>
<td>0.5000</td>
<td>0.0023</td>
</tr>
<tr>
<td>SUNSPOT</td>
<td>0.0030</td>
<td>-</td>
<td>0.5000</td>
<td>0.0166</td>
</tr>
</tbody>
</table>

Table 3: Period error rate on synthetic datasets

<table>
<thead>
<tr>
<th>Datasets</th>
<th>APT</th>
<th>WARP</th>
<th>FFT</th>
<th>MBPD</th>
</tr>
</thead>
<tbody>
<tr>
<td>S_ONE</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0637</td>
<td>0.0000</td>
</tr>
<tr>
<td>S_TWO</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0637</td>
<td>0.0000</td>
</tr>
<tr>
<td>S_THREE</td>
<td>0.0005</td>
<td>0.0000</td>
<td>0.0637</td>
<td>0.0009</td>
</tr>
<tr>
<td>S_FOUR</td>
<td>0.0006</td>
<td>0.0000</td>
<td>0.0637</td>
<td>0.0011</td>
</tr>
<tr>
<td>S_FIVE</td>
<td>0.0001</td>
<td>-</td>
<td>0.0637</td>
<td>0.0000</td>
</tr>
<tr>
<td>S_SIX</td>
<td>0.0000</td>
<td>-</td>
<td>0.0923</td>
<td>0.0000</td>
</tr>
<tr>
<td>S_SEVEN</td>
<td>0.0000</td>
<td>-</td>
<td>0.0800</td>
<td>0.0000</td>
</tr>
</tbody>
</table>

Table 4: Period values detected real datasets

<table>
<thead>
<tr>
<th>Datasets</th>
<th>APT</th>
<th>WARP</th>
<th>FFT</th>
<th>MBPD</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOLAR</td>
<td>872</td>
<td>-</td>
<td>910.22</td>
<td>870.60</td>
</tr>
<tr>
<td>MFCC</td>
<td>36460</td>
<td>-</td>
<td>37449</td>
<td>36340</td>
</tr>
<tr>
<td>POWER</td>
<td>328</td>
<td>516</td>
<td>334.37</td>
<td>135.85</td>
</tr>
<tr>
<td>SUNSPOT</td>
<td>133</td>
<td>516</td>
<td>136.53</td>
<td>135.85</td>
</tr>
</tbody>
</table>

1. Ranking error rate in this work is computed in terms of the size of the time series i.e. the number of falsely ranked candidate periods divided by the size of the time series. In Table 2 we record false positive rates. The false positive rate is computed as a ratio of the number of false positives to the size of the dataset.

\[
\text{error rate} = \frac{|\text{expected value} - \text{actual value}|}{\text{actual value}}
\]

In Table 3 we show the period error rates of the period value detected on the synthetic datasets by three techniques. The period error rates are computed with equation 2. Since we do not know the exact periods in the real datasets, we did not evaluate the error rate of the period values. Table 4 contains the periods detected on the real datasets. The estimates for the real datasets are SOLAR: 870 - 875; MFCC: 35K - 38K; POWER: 328 - 338; SUNSPOT: 132 - 137. As seen in Table 1 APT has the desired lower false positive
rates compared to FFT and WARP on the real datasets but MBPD performs better in this regard. A low false positive rate indicates the ability to retain only relevant results and saves user the time to look through irrelevant results. From the results in Tables[2][3] and[4], APT detects the period more accurately than FFT, WARP and MBPD, albeit MBPD performed very competitively. We did not record some of the results for WARP in cases where the results were unreasonable e.g. WARP returned the first value tested (2) on the SOLAR dataset. We also did not record some of the results for WARP when the size of the dataset caused an out of memory exception and down sampling could cause the experiment to lose the validity, as is the case with the real datasets. WARP could not accurately detect the period in some of the real datasets and the synthetic datasets with mixed periods (S_SIX and S_SEVEN). FFT, APT and MBPD were able to detect these periods. With all things considered, APT is shown to be more accurate than FFT and WARP. APT performs competitively with MBPD.

FFT has high false positive rate. The false positive rates are approximately half the size of the real time series input. This is because the first $\frac{n}{2} - 1$ energy components are typically considered since the second half of the FFT (coefficients from $\frac{n}{2} + 1$ to $n - 1$) can be ignored due to the complex conjugate symmetry with the first $\frac{n}{2} - 1$ coefficients, for a real time series of size $n$. The coefficient at $\frac{n}{2}$ represents energy at the Nyquist frequency, but this is also ignored by attenuation.

6 Conclusion

We present an approximate period detection scheme in this work. From the results of the experiments performed, our technique detects periods more accurately compared to a classic method and two other state-of-the-art methods. Our method detects mixed periods. As future work, we intend to investigate on approaches to increase the efficiency and accuracy of our technique. We would like to perform experiments on high dimensional datasets using efficient high dimensional similarity computation techniques such as locality sensitive hashing (LSH) and high performance techniques such as MapReduce. We would also like to make our algorithm detect partial or subsection periodicity. One limitation of our technique is that the periodic pattern is assumed to start in the beginning of the dataset. We would like to relax this requirement in the future without accruing too much computational burden on the algorithm.

References

[10] Li, Z., Wang, J., and Han, J. (2012). Mining event periodicity from incomplete observations. KDD.