# IFABX020 DATA SCIENCE PROJECT: DATA SUMMARIZATION AND SIMILARITY SEARCH ON TIME SERIES

CUI ZHENGUO[1], KENGO BENITO[1], NGOM SEUNE[1]

## CONTENTS

## LIST OF FIGURES

## LIST OF TABLES

[1] *Faculty of Mathematics and Computer Science, Paris Descartes University, Paris, France*

# 1 INTRODUCTION

This project is conducted within the master course *IFABX020 Data Science* in spring semester 2019-2020 at Université de Paris.

In this project, we investigated several time series data summarization techniques and applied these techniques to two datasets of fifty thousand time series. Furthermore, we conducted similarity search with clustering technique on time series.

The rest of this report is organized as follows: Section 2 explains the background of this project. Section 3 introduces related works in time series data summarization. Section 4 represents our implementations and discusses the results. Section 5 concludes the project.

# 2 BACKGROUND

The time series, as a sequence of listed data points, represents a set of observations of a variable ordered in time. Data mining on time series is aiming to extract meaningful information from the shape of entire time series or sub-series.

**Definition 1.** ***Time Series.***
A time series $X$ consists of $n$ data points $x_1, x_2, ..., x_n$ from $R^d$, where $d$ is the dimentionality of the time series.

In this project, two different types of time series have been provided: *synthetic time series* and *seismic time series*.

Every float numbers in synthetic time series are generated by random walks with Gaussian distribution and every time series has been z-normalized which means it has mean of 0 and standard deviation of 1. Figure 1 shows a synthetic time series in synthetic dataset.

The seismic time series dataset is collected from real world and has also been z-normalized. Figure 2 represents a seismic time series.

Both synthetic and seismic datasets contain 50,000 time series and each time series consists of 256 float numbers. The values in datasets follow the IEEE-754 standard, every float is allocated 32 bits (4 bytes) and stored in binary file. In total, both two datasets occupy 51200000 bytes $(= 256 * 4 * 50000)$.

More precisely, every time series has a sequence of 256 samples, every sample is a float value represented by the bitwidth 4, and the time series is univariate (dimensionality 1).

Summarizing a time series, also called *Segmentation*, reduces the dimension of time series data which is one of the most important technique for time series data mining.

**Definition 2.** ***Time Series Summarization.***
Given a Time Series $X$, a sequence of $n$ data points, find a length $k$ sub-sequence $\widehat{X}$ ($n >> k$) and the sub-sequence represents a compact description of the original time series.

As the first task of this project, our target is to reduce the size of every time series from 1024 bytes to 128, 64 and 32 bytes. And the reconstruction from reduced time series (128/64/32 bytes) to initial size time series (1024 bytes) is also required in this task.

The compression-reconstruction process of time series is not lossless, the performance will be evaluated by calculating reconstruction error which is the Root Mean Square Error (RMSE) between the original time series and reconstructed time series. The RMSE is defined as following equation:

**Definition 3.** ***Root Mean Square Error.***
$\mathrm{RMSE}(X, \widehat{X}) = \sqrt{\Sigma_{i=1}^{n}(x_i - \widehat{x_i})^2/n}$
*The RMSE is a type of Euclidean distance.*

The effectiveness of the summarization will be evaluated by RMSE between every original and reconstructed time series. For the entire dataset, the average reconstruction error among 50,000 time series will be evaluated in the following 3 cases:

- 1024 bytes -> 128 bytes -> 1024 bytes
- 1024 bytes -> 64 bytes -> 1024 bytes
- 1024 bytes -> 32 bytes -> 1024 bytes

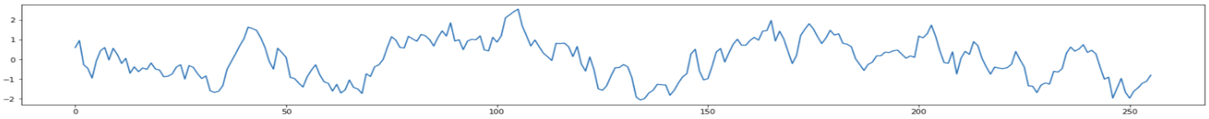

**Figure 1:** An example of synthetic time series

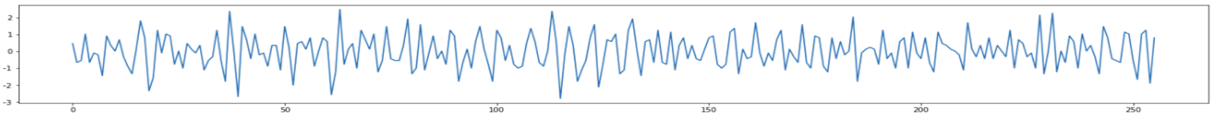

**Figure 2:** An example of seismic time series

The second task of this project is similarity search. Similarity search for a query, is aiming to find the most similar time series among our fifty thousand time series dataset. The similarity of two time series is measured by Euclidean distance. The Euclidean distance is defined as following equation:

**Definition 4.** ***Euclidean Distance.***
$D_{Euc}(X, \widehat{X}) = \sqrt{\Sigma_{i=1}^{n}(x_i - \widehat{x_i})^2}$

The objective is to identify the most closest time series among 50,000 time series dataset, that the closest time series has the minimum Euclidean distance with the query. The most similar time series is defined as following:

**Definition 5.** ***Most similar time series.***
Given a query Q of length $l$, and a set of time series $X_i$ of length $l$ ($i \in n$). The most similar time series of Q among all $X_i$ :
$X_s(Q) = \min(D_{Euc}(Q, X_i)), \forall i \in 0..n-1$

However, the similarity search on original size time series is costly. In order to diminish computational cost, the implemented summarization solution in first task will be used to decompose original size time series in this task.

## 3 RELATED WORK

Agrawal et al.[1] applied Discrete Fourier Transformation (DFT) to summarize time series by mapping points of time series from spatial domain to frequency domain. They generated 400 synthetic time series and conducted data segmentation with different sequence lengths and numbers of Fourier coefficients. Their experimentation on generated synthetic dataset showed the a few Fourier coefficients resulted better performance.

Chaovalit et al.[2] reviewed the literature in Discrete Wavelet Transformation (DWT) on time series data mining. They pointed out that the selection of the mother wavelet and coefficients effects different data reduction performance.

Wu et al.[3] studied the performance of DFT and DWT on similarity search in time series database. They used the historical data of 100 companies in 360 transaction days from stock market. The dataset had been pre-processed into 36000 samples of 128 sub-sequences. They evaluated average errors for 100 queries with Euclidean distance and found out that the results of DFT and DWT were similar.

Keogh et al.[4] introduced Piecewise Aggregate Approximation (PAA) in 2001. Their experimentation proved that a simple data compression method by computing the piecewise averages of every segments of original time series reaches splendid results in similarity search on time series dataset.

Roy et al.[5] conducted experiments to compare the performance of different data compression methods on summarizing bio-medical time series. Their comparative study implemented Discrete Cosine Transformation (DCT), DFT, DWT and Walsh Hadamard Transformation(WHT) to decompose electrocardiogram and photoplethysmography signals. They specified the compression with DCT, when compression ratio reached more than 100, obtained the smallest error than DFT and DWT.

Keogh and Ratanamahatana[6] investigated on lower bounding measures for Dynamic

Time Warping (DTW) in time series indexing. They introduced their lower bounding function with Piecewise Aggregate Approximation (PAA) which provides a tighter bound than other approaches. The experiment showed that their technique obtained a higher pruning performance than others.

Vlachos et al.[7] developed a clustering approach I-kMeans based on DWT and K-Means. They conducted iterative k-Means clustering on different length representations to adjust clusters by re-using the centroids of previous steps. For decomposition of time series, they selected Haar wavelet and stored entire outcome after decomposition, which allowed the reconstruction became a reversible lossless process. After the decomposition, they looped the k-Means clustering from relatively small representation and incrementally increased representation size of time series. The extracted centroids of every representation level will be doubled and used as the initial centroids of next level. They mesured the clustering result of the highest level representation with Euclidean distance. Comparing with general k-Means method, I-kMeans reduced at least half of the running time of clustering and also improved clustering quality.

# 4 IMPLEMENTATION & EXPERIMENTATION

## 4.1 Data summarization

We considered that the data summarization can be divided into two sub-tasks:

1. reduce the dimensionality of time series
2. reduce the storage size of time series

### 4.1.1 *Dimensionality reduction of time series*

Due to Parseval's theorem, the Euclidean distance in the spatial domain keeps the same as in the frequency domain, which is an advantage of applying the spatial-frequency domain transformation for time series data. So we firstly applied Discrete Cosine Transform (DCT) and Discrete Fourier Transform (DFT) to decrease time series dimension by using *Legacy discrete Fourier transforms package* of *SciPy*, scipy.fftpack. We chose 32, 16 and 8 as the length of the transform which satisfied summarization requirements: 128, 64 and 32 bytes because every point requires 4 bytes.

Since approximations are frequently used to create aggregate representation, we also investigated the techniques such as Piecewise Aggregate Approximation (PAA). We utilized the library *tslearn.piecewise* to summarize time series with PAA. Moreover, we implemented Piecewise-Median which calculate the midpoint of every segment, as a benchmark method of PAA, aiming to compare the difference between taking piecewise average and piecewise "middle" value in terms of time series representation.

Furthermore, we implemented a method named PiecewiseSum&SquareSum. The idea of this method came from the Clustering Feature (CF) of BIRCH, the clustering research of Zhang et al.[8]. Every piece(segment) will be represented as the following piecewise feature:
$\mathrm{FEATURE} = \{N, Sum, SquareSum\}$
Every segment is represented by the length of segment *N*, the linear sum of values of the points *Sum* and the square sum points in this segment *SquareSum*. By using this piecewise representation, for example, a segment of 4 points [1, 0, -1, 2] will be summarized by 2 numbers [2, 6]. Here the first value 2 is the sum of [1, 0, -1, 2] and the second is the square sum of this segment. By dividing the segment length *N*, the $Sum/N$ represents the segment level deviation from the origin and the $\sqrt{SquareSum/N}$ stands for the variation of every points from the origin. For a compression from 16 values to 4 values, this technique requires 2 segments of length 8. But using twice enlarged segment might bring more information losses for reconstruction.

The wavelet transform is a tool that cuts up data or functions or operators into different frequency components, and then studies each component with a resolution matched to its scale [9]. The Discrete Wavelet Transform (DWT) is used in functional and digital analysis is a technique of transforming signals into wavelets, to compress and also completely reconstruct the original signal using infinite summations of discrete wavelet coefficients according to the respect of certain criteria. This method was proposed for the similarity search, also called *Indexing*, to reduce dimentionality of time series[10]. We used the library *pywt* to summarize time series with DWT.

We selected built-in Haar wavelet and Daubechies db1 wavelet as the mother wavelet parameters because every time they decompose time series exactly the half of the previous size.

Besides, creating new values based on the linear combinations of original data points can also reduce the dimensionality of original time series. Principal Component Analysis (PCA) is mostly used in this case. By specifying the number of principal components (new values from linear combination of initial values), the time series will be decomposed into expected number of points. We applied this method with the library *sklearn.decomposition.PCA* in our project.

### 4.1.2 *Storage size reduction of time series*

The objective of data summarization in this project is to reduce the size of every time series from 1024 bytes to 128/64/32 bytes and saved in binary file. From another perspective, reducing the size of every points is allowed.

For single precision floating point of IEEE-754, every point of time series is allocated 4 bytes as its storage size. However, if possible, every number can be allocated less bytes by using other data types. For example, the initial time series is 1024 bytes, after data summarization, an expected 32 bytes representation can be 8 *numpy.float32* points or 32 *numpy.int8* points. Obviously 32 data points contain more information than 8 points.

As a starting point of this idea, we tried to map the range of representation into target range, we named this process as range-mapping. Precisely, we converted the representation of 128/64/32 float32 points into the same number of int8 points by mapping the interval of float32 representation into the value range of int8, $[-127, 127]$. We take the maxmimum of absolute values of the representations, then divide 127 by this maximum value, the outcome is denoted as mapping scale *scale_n*. All points of summarized float32 representation have multiplied by *scale_n* and been rounded then saved as integer type int8.

Besides, using Symbolic Aggregate approXimation (SAX) with PAA can also create time series representations which are also able to fit in the value range of 1 byte data type, *numpy.uint8* has value range from [0, 255].

The common part of these two approaches is that both are data adaptive methods. But there are two main differences between our range-mapping and SAX-PAA, one is the hypothesis on data point distribution and the other is the additional storage requirement.

The SAX-PAA approach supposes the data follows a Gaussian distribution, every symbol represents a same proportion of points. The details on data concentrated area can be kept and the representations are more accurate. In the mean time, every symbol requires a lower bound value in order to reconstruct the original time series from summarized representation. In our case, the value range of data type uint8 [0, 255], these 256 symbols require 256 floating points in addition to our compression target. Strictly, at least we exceeded 1024 bytes to store these 256 values.

On the contrary, our range-mapping method does not take into account the data distribution of time series. With a simple multiplication scale *scale_n*, which has been calculated by the equation $127/max(abs(TS_{min}), TS_{max})$, all values are mapped from initial interval $[TS_{min}, TS_{max}]$ into [-127, 127]. The hypothesis is that all data points of time series are located homogeneously in the initial range, as the uniform distribution. But the advantage of this method is, unlike the SAX, that single-byte representation requires only one parameter *scale_n*. In case the requirement of the project haven't allowed any additional bytes, the value can be concatenated at the end of the compressed file name. If so, the reconstruction process of this summarization file shall be: firstly split the concatenated *scale_n* then re-map from int8 to float32 interval before applying the reconstruction methods.

### 4.1.3 *Results of the experiment*

To asses the effectiveness of different approaches, we compared compression-reconstruction process by applying on 2 datasets and measuring the Root Mean Square Error (RMSE) which have been explained in Section 2.

All experiments used a virtual machine on Google Cloud Platform. The virtual machine has 4 vCPUs and 15GB memory (*type n1-standard-4*).

In total, we applied the following 8 methods: Discrete Cosine Transform (DCT), Discrete Fourier Transform (DFT), Piecewise Aggregate Approximation (PAA), Piecewise Middle-point Approximation (PiecewiseMedian), PiecewiseSum&SquareSum, Discrete Wavelet Transform (DWT), Principal Component Analysis (PCA) and Symbolic Aggregate approXimation (SAX). For DWT, we observed that there was no convincing difference between Haar wavelet and Daubechies db1 wavelet, so that we only kept the result of db1 wavelet on the results of experiment.

For each of the 50,000 times series, the reconstruction error is measured by the RMSE between the original time series and the reconstruction time series. With the same compression target storage size, 128/64/32 bytes, we compared the average RMSE of these methods with 32/16/8 float32 representations and 128/64/32 int8 representations. The results of the experiment on synthetic and seismic datasets are shown in following Table 1 and Table 2.

| | DCT | DFT | PCA | DWT | PAA | SAX | PiecewiseMedian | PiecewiseSum&SquareSum |
|---|---|---|---|---|---|---|---|---|
| 32 float32 | 1.084 | 1.010 | 0.169 | 0.217 | 0.217 | n.a. | 0.230 | 1.022 |
| 16 float32 | 1.050 | 1.003 | 0.238 | 0.305 | 0.305 | | 0.317 | 1.080 |
| 8 float32 | 1.027 | 1.001 | 0.330 | 0.425 | 0.425 | | 0.437 | 1.159 |
| 128 int8 | 1.106 | 1.054 | 0.081 | 0.095 | 0.096 | 0.096 | 0.135 | 0.915 |
| 64 int8 | 1.123 | 1.029 | 0.119 | 0.150 | 0.151 | 0.151 | 0.171 | 0.971 |
| 32 int8 | 1.084 | 1.010 | 0.170 | 0.217 | 0.217 | 0.217 | 0.231 | 1.022 |

**Table 1:** Average RMSE among 50k synthetic time series

| | DCT | DFT | PCA | DWT | PAA | SAX | PiecewiseMedian | PiecewiseSum&SquareSum |
|---|---|---|---|---|---|---|---|---|
| 32 float32 | 1.060 | 1.002 | 0.864 | 0.988 | 0.988 | n.a. | 1.038 | 1.391 |
| 16 float32 | 1.030 | 1.000 | 0.933 | 0.997 | 0.997 | | 1.018 | 1.403 |
| 8 float32 | 1.015 | 1.000 | 0.967 | 0.999 | 0.967 | | 1.008 | 1.408 |
| 128 int8 | 1.223 | 1.030 | 0.291 | 0.791 | 0.792 | 0.791 | 1.119 | 1.323 |
| 64 int8 | 1.117 | 1.007 | 0.711 | 0.964 | 0.964 | 0.964 | 1.087 | 1.369 |
| 32 int8 | 1.060 | 1.002 | 0.865 | 0.988 | 0.988 | 0.988 | 1.038 | 1.391 |

**Table 2:** Average RMSE among 50k seismic time series

From Table 1, we observed that the time series summarization with PCA, DWT, PAA and PiecewiseMedian had less reconstruction error than other methods. The averages of every segment (PAA) are more accurately represented than the medians (PiecewiseMean) for original time series. Additionaly, with our range-mapping method(float32 -> int8), the reconstruction error of these methods even reduced 50% on synthetic dataset, for example 0.217 -> 0.096 (128 bytes for DWT). RMSE on the Table 2 showed that PCA, DWT and PAA also reached smaller reconstruction error on seismic dataset. The method we created, PiecewiseSum&SquareSum did not reach good compression-reconstruction performance on both datasets. Beyond these methods, the spatial-frequency domain transformations, DCT and DFT performed deficiently on our z-normalized datasets.

However, the complexity and additional storage space of the best performed methods are disparate. In terms of complexity, PAA, DWT and SAX are the most efficient methods with linear time complexity $\mathcal{O}(n)$. The method with the least RMSE, PCA has an exponential time complexity and requires 2000 bytes per principle component as additional storage requirement besides the compressed dataset. Moreover, the method SAX requires a lower bound value per representation symbol, but acquired same RMSE with range-mapped PAA. In our case, 256 symbols required additional 1024 bytes which exceeded slightly the project requirement.

Though our range-mapping method and SAX improved quality of compression-reconstruction of time series, but these two methods have their limits. For both synthetic and seismic time series dataset, all the values are z-normalized and follow Gaussian distributions with a extremely small deviation. In general, time series contains points with much higher deviation, real-valued data is messy and the distribution is skew.

In brief, we discovered that from the results of experenmentation on two z-normalized datasets, DWT and PAA achieved best performance. These two methods were excellent in time complexity aspect and representation performance. And our range-mapping method and SAX did not improve significantly the data summarization performance.

## 4.2 Similarity search

Similarity search on large time series database is costy. As we mentioned in Section 2, every time series will be compressed into a much smaller size sequence (1024->128/64/32 bytes), and the similarity measure will be applied based on compressed time series to reduce computational costs.

The objective is to skip some of the time series in the database, without checking the Euclidean distance with the query. The skipped time series are described as a pruned time series. The quality of similarity search is measured by pruning ratio, the average of the numbers of skipped time series will be reviewed.

### 4.2.1 *Similarity search with Piecewise K-Means*

Clustering helps for classification without any knowledge of data. The clustering method K-Means is able to partition multidimensional data points into K clusters that the points are closest to the center of a cluster. K-Means emphasizes homogeneity rather than separation[11], the sizes of clusters are close to each other. The similar size clusters can achieve more pruned time series than skewed clusters from general cases.

To find a most similar time series of a query with K-Means, firstly we compared the techniques we applied for data summarization. We generated simple tri-clusters K-Means clustering models with the 128/64/32 bytes compressed time series that decomposed by 8 different compression methods. We figured out that except the PiecewiseSum&SquareSum, all various size representations of other 7 methods obtained very low silhouette score. In addition, we identified that the length of representation even caused worse quality of clustering model. Among the methods, we selected PiecewiseSum&SquareSum approach with 32 bytes representations as our summarized database for K-Means.

Then we tested the number of clusters K with the 32 bytes representation. By benchmarking the number of clusters K from 3 to 10, we noticed that the increase of K even caused worse clustering performance both for synthetic and seismic datasets. In this context, we chose $K = 3$ as our clustering parameters.

After the experiment for representation selection and parameter selection of K-Means, we realized that the performance of whole time series clustering is still insufficient when measuring silhouette score. The clustering model with whole time series matching reached at most 0.3 silhouette score, the poor performance was far from our expectation.

We resumed to the original idea of this summarization methods, the representation comes from the segment level features and every 2 values represent a single segment. Accordingly, we splited 32 bytes summarized representation into 4 segments and investigated on piecewise K-Means clusters. With $K = 3$, all four piecewise clustering models obtained eligible silhouette score, the average score even exceeded 0.5 in some cases.

The similarity search procedure is implemented as follows: (1) Compress the original size database and query time series dataset to 32 bytes representations. Then (2) split the summarized database into 4 segment level representations $seg_i$, $i \in 1..4$, and generate 4 K-Means clustering models based on $seg_i$. (3) Use the models to predict splited summarized database itself, generate piecewise classes for every time series in database, saved as $p\_seg_i$, $i \in 1..4$. (4) Verify a time series $X_j$ its segment level classes $p\_seg_i[j]$ in database with the prediction of a query $q\_class_i$. (5$a_1$) If all segment level classes are equal between a time series and current query, this time series is not pruned and the Euclidean distance will be calculated. (5$a_2$) In case the distance is smaller than previous memorization, we record this distance as the smallest and memorize index of this time series. (5b) In contrast to the case with same classes, when one of 4 classes is different, the time series is pruned. (6) Repeat the steps (2)-(5) till all time series in database are checked.

The procedure is developed as the following algorithm 1:

**Algorithm 1** Similarity Search with Data Summarization and Piecewise K-Means

```
Require: query time series dataset $Q_i$, $i \in 0..m-1$
Require: time series database $X_j$, $j \in 0..n-1$
compress whole dataset X to $X_{com}$
split $X_{com}$ into 4 segment level representations $seg_i$, $i \in 1..4$
generate 4 K-means clustering models $cluster_i$ with $seg_i$, number of clusters $K = 3$, $i \in 1..4$
predict piecewise classes for $X_{com}$ : $p\_seg_1$, $p\_seg_2$, $p\_seg_3$, $p\_seg_4$
compress query dataset Q to $Q_{com}$
initialize NOT pruned counting for query dataset $count = 0$
for i = 0 to m-1 do
    initialize the closest Euclidean distance for original size $D_{ori}$ as $+\infty$
    initialize memorization of the most similar time series index $i\_sim = 0$
    predict the segment level classes $q\_class_1$, $q\_class_2$, $q\_class_3$, $q\_class_4$ for $Q_{i\_com}$
    for j = 0 to n-1 do
        if $p\_seg_k[j] == q\_class_k$ ($k \in 1..4$) then
            $count++$
            $d_{ori} = D_{euc}(Q, X_{rec\_i})$
            if $d_{ori} < D_{ori}$ then
                $D_{ori} = d_{ori}$
                $i\_sim = i$
            end if
        end if
    end for
end for
pruning_ratio = (50000 - count/100) / 50000
return pruning_ratio
```

### 4.2.2 *Results of the experiment*

With piecewise linear sum and square sum as the clustering representation, we were able to create K-Means clustering models on both synthetic and seismic databases. The generated K-Means clustering models are as the following figure 3 and 4 for synthetic database and seismic databases.

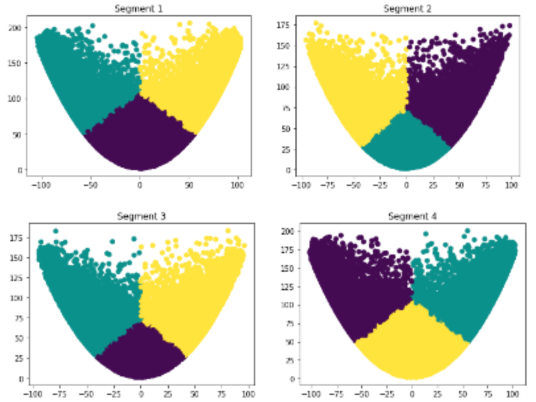

Figure 3: K-Means clusters for synthetic database

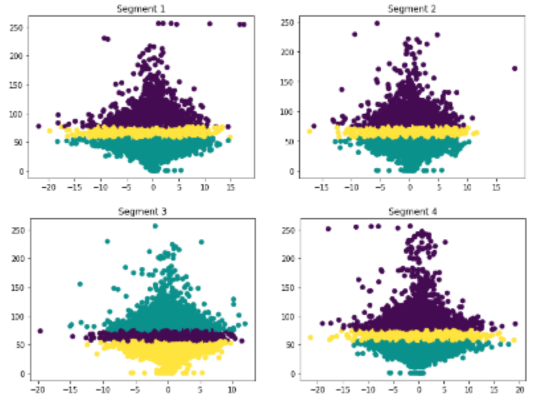

Figure 4: K-Means clusters for seismic database

In terms of the average pruning ratio, we obtained 99.5% for synthetic time series and 96.1% for seismic time series for 100 query time series.

## 5 CONCLUSION

Time series is ubiquitous. Time series data mining techniques are applied in various fields such as economy, biology, medical analysis, etc. Due to the high dimentionality characteristic, the analysis of time series requires enormous computational capacity. The time series summarization, namely representation, is famously used for feature extraction of time series.

In this project, we investigated on different approaches for time series data summarization, we also outlined differences between the approaches, and experimented on small size databases. Furthermore, we implemented time series clustering technique K-Means for purpose of conducting more efficient time series indexing.

## REFERENCES

[1] Rakesh Agrawal, Christos Faloutsos, and Arun Swami. Efficient similarity search in sequence databases. In *International conference on foundations of data organization and algorithms*, pages 69–84. Springer, 1993.

[2] Pimwadee Chaovalit, Aryya Gangopadhyay, George Karabatis, and Zhiyuan Chen. Discrete wavelet transform-based time series analysis and mining. *ACM Computing Surveys (CSUR)*, 43(2):1–37, 2011.

[3] Yi-Leh Wu, Divyakant Agrawal, and Amr El Abbadi. A comparison of dft and dwt based similarity search in time-series databases. In *Proceedings of the ninth international conference on Information and knowledge management*, pages 488–495, 2000.

[4] Eamonn Keogh, Kaushik Chakrabarti, Michael Pazzani, and Sharad Mehrotra. Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and information Systems*, 3(3):263–286, 2001.

[5] Anamitra Bardhan Roy, Debasmita Dey, Bidisha Mohanty, and Devmalya Banerjee. Comparison of fft, dct, dwt, wht compression techniques on electrocardiogram and photoplethysmography signals. In *IJCA Special Issue on International Conference on Computing, Communication and Sensor Network CCSN*, pages 6–11, 2012.

[6] Eamonn Keogh and Chotirat Ann Ratanamahatana. Exact indexing of dynamic time warping. *Knowledge and information systems*, 7(3):358–386, 2005.

[7] Michail Vlachos, Jessica Lin, Eamonn Keogh, and Dimitrios Gunopulos. A wavelet-based anytime algorithm for k-means clustering of time series. In *In proc. workshop on clustering high dimensionality data and its applications*. Citeseer, 2003.

[8] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Birch: an efficient data clustering method for very large databases. *ACM Sigmod Record*, 25(2):103–114, 1996.

[9] Ingrid Daubechies. *Ten lectures on wavelets*, volume 61. Siam, 1992.

[10] Kin-Pong Chan and Ada Wai-Chee Fu. Efficient time series matching by wavelets. In *Proceedings 15th International Conference on Data Engineering (Cat. No. 99CB36337)*, pages 126–133. IEEE, 1999.

[11] Christian Hennig. Clustering strategy and method selection. *arXiv preprint arXiv:1503.02059*, 2015.