Summarization Project

QITONG WANG and THEMIS PALPANAS, Université de Paris, France

1 INTRODUCTION

In this project, we will explore one important technique of time series analysis, named summarization. There is another optional task of similarity search, which is one essential application of summarization.

2 DATASET

2.1 Description

We provide two datasets for you to explore, which could be found under this Google Drive shareable folder. The first one is a synthetic dataset, which is relatively simple to summarize. Another dataset is seismic data, which is harder. Both the datasets contain 50,000 univariate time series. Each time series consists of 256 (IEEE 754) float numbers.

The synthetic time series are generated from random walks. This is not important for our project, but in case you are interested, the basic procedure is the following:

(1) Initialize the first value as 0.
(2) To get the ith number, add a new random number out of $N(0, 1)$ to the $i-1$th number.
(3) For every l numbers, z-normalize them and write out to a file stream. $l = 256$ is the length of each sequence.

Finally, we z-normalize every series: we compute the mean and standard deviation of all the (256) values of the series. After that, each value of the time series is subtracted by its mean and then divided by its standard deviation. Therefore, after z-normalization, every time series has mean of 0 and standard deviation of 1.

Figure 1 illustrates the first 2 time series of synthetic dataset.

The seismic dataset is sampled from real world. The provided seismic dataset has also been z-normalized as stated in the above procedure. Figure 2 illustrates the first 2 time series of seismic dataset.

For the optional task of similarity search, we provide another 100 time series for both the synthetic dataset and the seismic dataset.

2.2 Usage

Here we gave two code examples of how to read/write this dataset in Python.

The basic method to read (or write) this binary file is to read (or write) in bytes, unpack (or pack) every 4 bytes to get a float value (since according to the IEEE 754 standard, a float value is represented using 4 bytes), and then group every 256 points (using reshape), as in the following code example.

```
import struct
import numpy as np

with open(filename, 'rb') as in_file:
    time_series = np.array(struct.unpack('f' * 50000 * 256, in_file.read()))
                    .reshape(-1, 256)
```

Authors' address: Qitong Wang, qitong.wang@etu.parisdescartes.fr; Themis Palpanas, themis@mi.parisdescartes.fr, LIPADE, Université de Paris, 45 Rue des Saints-Pères, Paris, 75006, France.

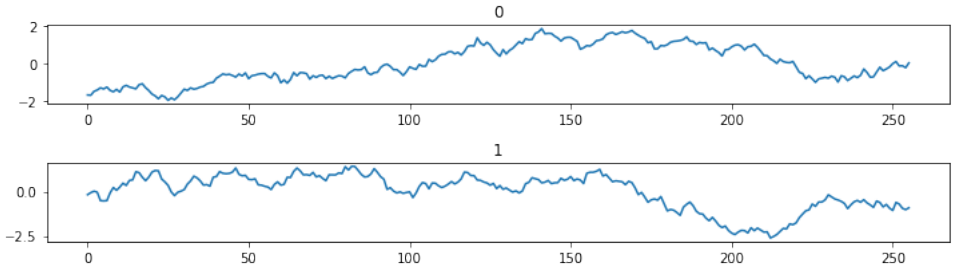

Fig. 1. The first 2 time series of synthetic dataset.

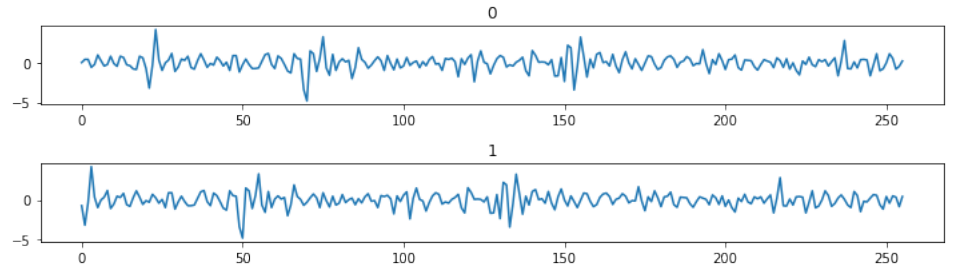

Fig. 2. The first 2 time series of seismic dataset.

The simplest way in Python to read the dataset is to use the numpy.fromfile method, and then "reshape" it to [50000, 256], i.e., an array of 50000 series, where each series is composed of 256 values:

```
import numpy as np

time_series = np.fromfile(filename, dtype=np.float32).reshape(-1, 256)
```

3 TASKS

The main task of our project is to develop a time series summarization technique, and then use it to solve the similarity search problem.

3.1 Summarization

Our first task is to provide a summarization for all the provided 50k time series of both synthetic dataset and seismic dataset.

Informally, summarization means to use less memory to represent the original dataset. This means usually summarization could bring space efficiency. For example, the size of our original dataset is 4 byte / float * 256 float numbers * 50000 series = 51.2 MB, with each series occupying 1024 bytes.

Our summarization should be using less memory. In this project, we will need to summarize each series (of 1024 bytes) using

- 32 bytes
- 64 bytes
- 128 bytes

We will evaluate the effectiveness of our summarizations for these 3 different cases.

3.2 Similarity Search (Optional)

Our second and optional task is to find the closest time series in terms of Euclidean distance in the dataset for all the provided 100 queries of both synthetic dataset and seismic dataset, i.e., similarity search.

Informally, similarity search is that given a query time series, to provide the most similar time series in your dataset. How similar two time series are is defined using distance measures. Euclidean distance is one of the most important and broadly used distance measures. Given two time series X, Y of length n, the Euclidean distance is defined in the following equation:

$$ED(X, Y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{1}$$

Then the question is how to utilize a summarization method to perform similarity search faster. The observation is that since the summarizations are shorter than the original time series, calculating Euclidean distances using the summarizations is much faster than calculating Euclidean distances between the original time series.

A simple and general procedure of using our summarizations to accelerate similarity search is as follows (you may come up with more elaborate methods):

(1) In the preprocessing phase, calculate the summarization $\overline{X}_i$ of every time series X_i in the database.
(2) When there comes a new query time series Y, calculate its summarization $\overline{Y}$ as well.
(3) Initialize the initial smallest Euclidean distance $D_{closest}$ as positive infinity.
(4) Iterate over all the time series in the database. For each time series X_i, calculate its $D_{summarization}(\overline{X}_i, \overline{Y})$ with the query. If $D_{summarization}(\overline{X}_i, \overline{Y}) > D_{closest}$, then X_i is skipped without further checking. Otherwise, $ED(X_i, Y)$ on the original time series values is calculated and compared with $D_{closest}$. If $ED(X_i, Y) < D_{closest}$, set $ED(X_i, Y)$ as the new $D_{closest}$ and continue to the next time series X_{i+1} .
(5) Return the corresponding time series X_j (and $D_{closest}$ if necessary).

Of course one question we need to answer is whether performing step 4 above with our summarization is safe. Step 4 will prune some of the series based on the distance of the summarizations (will never examine the distance of the corresponding original series): are the series we prune correctly pruned? Some summarizations always prune correctly, some others do not. This also depends on the function $D_{summarization}(\overline{X}_i, \overline{Y})$ you use for computing the distance between two time series summarizations.

4 EVALUATION METHOD

The evaluation metric deployed in the summarization task is the reconstruction error. For the optional task of similarity search, our evaluation metric is pruning ratio.

4.1 Reconstruction error

One essential requirement of the time series summarization is the ability of reconstructing the original time series. However, the reconstruction is usually not lossless. Reconstruction error is used to evaluate how precise the summarization could reconstruct the original time series.

The formal definition of reconstruction error deployed in this project is the mean square error between the original time series and the reconstruction time series, i.e.,

$$E(X, \hat{X}) = \sqrt{\sum_{i=1}^{n} (x_i - \hat{x}_i)^2} \tag{2}$$

where $\hat{X}$ is the reconstructed time series.

How to reconstruct the original series from the summarization is depending on the design of the summarization.

For the evaluation of our project, the reported reconstruction errors should be averaged over all the 50k time series from both datasets.

4.2 Pruning Ratio (Optional)

As we've shown in the query procedure of similarity search, some time series X_i in the database might be skipped without checking its Euclidean distance with the query. This is the case when we say this time series X_i is pruned. Informally, the pruning ratio is the fraction of pruned (or skipped) time series in the whole dataset. A larger pruning ratio implies that the designed summarization is more efficient as more time series have been skipped without further checking.

For the evaluation of our project, the reported pruning ratio should be averaged over all the 100 queries from both datasets.