

A Symbolic Representation of Time Series Employing Key-Sequences and a Hierarchical Approach

Qiang Wang¹, Guo Li¹, Vasileios Megalooikonomou¹, Christos Faloutsos²
¹Dept. of Computer and Information Sciences ²Dept. of Computer Science
Temple University Carnegie Mellon University
1805 N. Broad St., 5000 Forbes Avenue
Philadelphia, PA 19027 Pittsburgh, PA 15213-3891
{qwang,gli}@lucas.cis.temple.edu vasilis@cis.temple.edu christos@cs.cmu.edu

Abstract

Efficiently and accurately searching for similarities among time series and discovering interesting patterns is an important and non-trivial problem. There is a lot of prior work e.g., F-index introduced by Agrawal et al, ST-index proposed by Faloutsos et al, and PAA suggested by Keogh et al. In this paper we suggest a new method: HFVQA (Hierarchical Frequency-based Vector Quantized Approximation), which is frequency-based and uses a histogram model to calculate the similarity between two time series.

The novelty of our method is that it keeps both local and global information about the original time series in a hierarchical mechanism, processing the original time series at multiple resolutions. Moreover, the proposed representation is symbolic employing key-sequences potentially allowing the application of text-based retrieval techniques into the similarity analysis of time series. Our method is fast and scales linearly with the size of database and the dimensionality. Contrary to the vast majority in the literature that uses the Euclidean distance, it is the first (or "among the few ones") that uses a multi-resolution/hierarchical distance function. We demonstrate the utility and efficiency of HFVQA on multiple real and synthetic data.

1. Introduction

The problem of efficient retrieval of similar time series has received a lot of attention due to its many applications in different domains. Briefly, this problem can be stated as follows:

Given a query sequence q , a database S of N sequences, S_1, S_2, \dots, S_N , a distance measure D and a tolerance threshold ε , find the set of sequences R in S that are within distance ε from q . More precisely find: $R = \{S_i \in S \mid D(q, S_i) \leq \varepsilon\}$.

To compare two given time series, a suitable measure of similarity should be given. Naive approaches for comparing time sequences generally take polynomial time in the length of the sequences, typically linear or quadratic time. These approaches are not useful for large time series databases. Promising techniques include those that are based on the reduction of dimensionality of the original sequences. In this case, the sequences can be represented as multidimensional vectors and similar sequences can be retrieved in sublinear time.

There may be several different criteria to evaluate a method, but generally speaking, a good one should be

- fast and scalable
- accurate (according to some ground truth).

In this paper, we introduce a new method which satisfies these requirements. Our method is called HFVQA: Hierarchical Piecewise Vector Quantized Approximation, and it has the following characteristics:

- 1) it's the first one (or one of the few) that uses time-tested 'vector quantization' methods to discover a 'vocabulary' of sub-sequences;
- 2) It's the first one (or one of the few) that takes multiple resolution into account – this improves both the speed and the accuracy;
- 3) It's the first one (or one of the few) that utilizes text-based techniques (tf / idf method) from Information Retrieval, to weight down uninteresting matches, thus improving the accuracy.

As Agrawal et al [2] proposed, compared with Euclidean distance, a more intuitive idea is that two series should be considered similar if they have enough non-overlapping time-ordered pairs of subsequences that are similar. In this paper, instead of calculating the Euclidean distance, we first extract key-subsequences utilizing the Vector Quantization (VQ) [9] technique and encode each time series based on the frequency of appearance of each key-subsequence. We then apply a histogram model to calculate the dissimilarities among time series. This method can be very meaningful in many domains, for example, when comparing two stocks during a long

period, we may want to find out during how many months the stocks have similar movements, though the same trend may appear in different months for different stocks. This application is similar to mining motifs in massive time series databases [20].

While the histogram metric can record the local information very well, it loses much global information of the time series, since it doesn't keep track of the order of appearance of different key-subsequences. To deal with this problem, we propose to apply a hierarchical mechanism: Original time series are processed at several different resolutions, and similarity analysis is performed using a weighed distance metric combining all the resolution levels. For example, given a time series representing a stock price movement, we know that subsequences of different length have different real meanings. If the length is 5, the sub sequence stands for a weekly trend of the stock, while we can find the monthly trend when the length is 20 (5 * 4 weekdays).

The rest of this paper is organized as follows. Section 2 provides a survey of previous methods and related background knowledge. Section 3 introduces the framework of the new dimensionality reduction technique we propose. Experiment results are presented in Section 4. Directions for future work and concluding remarks are presented in Section 5 and Section 6.

2. Survey - Background

2.1 Related work

Considering the comparison between two given time series, many approaches and techniques have been suggested in the past decade [1, 2, 4, 6, 7, 10, 11, 13, 15, 16, 18, 19, 23, 27, 28].

To deal with dimensionality reduction, the solution to extract a signature from each sequence and to index the signature space was originally proposed by Faloutsos et al [6, 7]. To guarantee completeness (i.e., no false dismissals) the admissibility criterion that the distance function used in the signature space must underestimate the true distance measure (bounding lemma) was also proposed [7].

Obeying the admissibility criterion, many methods have been suggested and proved useful in different fields, such as F-index introduced by Agrawal et al [1], ST-index proposed by Faloutsos et al [7], and PAA suggested by Keogh et al [16, 19].

For these methods in which the distance metric lower bounds the Euclidean distance, one of the most significant characteristics is the avoidance of false dismissals, though there may be a lot of false alarms. However, in some cases, the existence of too many false alarms may

decrease the efficiency of retrieval. At the same time, as many researchers have mentioned in their work [12,25], the Euclidean distance is not always the optimal distance measure. For example, in some time series, different parts have different levels of significance in their meaning. Also, Euclidean distance doesn't allow shifting in time axis, which is not unusual in real life applications.

In order to extract high-level features out of time series, Nick Koudas et al. [24] formalized problems of identifying various "representative" trends in time series data.

2.2 Background

To make the presentation of the proposed work clear, here, we give descriptions of various concepts and definitions used in the paper. We start with the definition for a time sequence and its subsequences.

Definition 1. Time Sequence: A sequence (ordered collection) of real values. $X = x_1, x_2, \dots, x_n$, where n can be very large.

Definition 2. Subsequence: Given a time sequence $X = x_1, x_2, \dots, x_n$, of length n , a subsequence S of X is a sampling of length $m < n$ continuous positions from X , i.e., $S = x_k, x_{k+1}, \dots, x_{k+m-1}$; $1 \leq k \leq n-m+1$.

In similarity analysis, we need to define a metric for the similarity, that is, a measure of the distance between two time series. The most popular metric used in various applications is the Euclidean Distance.

Given two time sequences, $X = x_1, x_2, \dots, x_n$, $Y = y_1, y_2, \dots, y_n$, their distance, D , is defined in general as

$$L_p = \left(\sum_{i=1}^n |x_i - y_i|^p \right)^{1/p} \quad (1)$$

(For an L_p norm, when $p=2$, the distance L_2 stands for Euclidean distance).

Obviously, the simplest way of calculating the similarity (or distance) among time series is to compute the Euclidean distance directly, i.e., on the original series. For a small dataset this may be feasible, however, for large data sets the efficiency is a problem, since the time complexity is $O(N*n)$, where n is the number of features that need to be represented for each time series and N is the number of time series in the dataset. In order to compute efficiently while keeping the accuracy not significantly affected, many techniques of dimensionality reduction have been suggested, such as the Discrete Fourier Transform (DFT), the Discrete Wavelet Transform (DWT), the Piecewise Aggregate Approximation (PAA), etc.

Besides the computation complexity, we can't always be sure that the nearest neighbors in Euclidean space are the most similar ones. This is because that the point-based

information model (computing similarity based on every point) contains only low-level features of the time series and it is vulnerable to different kinds of shape transformations, such as shifting and scaling. Under such circumstances, it will be better if we can find some high-level features and apply a more robust information retrieval model for time series analysis.

Based on this idea, we introduce a *key-sequence* framework (in analogy to *key-blocks* in image retrieval [29]) to facilitate the retrieval of similar time series. This framework consists of the following main components:

- 1) Codebook generation;
- 2) Time series encoding;
- 3) Time series feature representation and retrieval.

This framework is analogous to the *keyblock* framework suggested by Aidong Zhang et al [29] for information retrieval in image domain, which provides a practical solution for content-based image retrieval.

For codebook generation in vector quantization the Generalized Lloyd Algorithm (GLA) [21, 22] is usually applied (this is done during a training phase). The main structure of GLA is given in the flow chart in Figure 1.

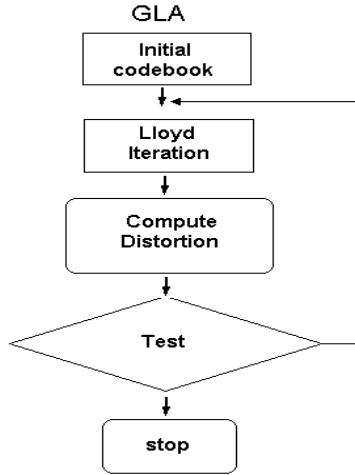


Figure 1. The Generalized Lloyd Algorithm (GLA).

Starting with an initial codebook, the GLA algorithm repeats the Lloyd iteration until the fractional drop of the distortion becomes less than a given threshold. This process is guaranteed to converge since from the necessary conditions for optimality each application of the Lloyd iteration must reduce or leave unchanged the average distortion [9].

In VQ-based retrieval [29], two of the most popular models that have been proposed are the Boolean Model (BM) and the Vector Model (VM). The latter has a special case, which is called Histogram Model (HM). We present these models in the context of time series analysis:

1. **Vector Model (VM):** compute the similarity between the frequency-based representations of 2 time series. Use the following formula:

$$S_{vm}(q, t) = \frac{\sum_{i=1}^s f_{i,t} * f_{i,q}}{\sqrt{\sum_{i=1}^s f_{i,t}^2} * \sqrt{\sum_{i=1}^s f_{i,q}^2}} \quad (2)$$

In the formula, $f_{i,t}$ means the frequency of codeword i in time series t .

2. **Boolean model (BM):** compute the similarity of the Boolean models of the codeword representation of 2 time series.

$$S_{BM}(q, t) = n_{11} * w_{11} + n_{00} * w_{00} \quad (3)$$

where n_{11} is the number of identical indices and n_{00} is the number of indices of the code words that do not exist in both of the representations, while w_{11} and w_{00} are the weights assigned to these frequencies.

3. **Histogram Model (HM):**

$$S_{HM}(q, t) = \frac{1}{1 + dis(q, t)^2} \quad (4)$$

$$\text{where } dis(q, t) = \sum_{i=1}^s \frac{|f_{i,t} - f_{i,q}|}{1 + f_{i,t} + f_{i,q}}$$

3. Proposed method: FVQA, HFVQA

We propose to reduce the dimensionality in time series data by applying a piecewise approximation using VQ encoding at different resolutions. For each resolution, the proposed method called Frequency Vector Quantized Approximation (FVQA) allows a time series of arbitrary length n to be reduced to a much smaller dimension s ($s \ll n$).

With FVQA, a codebook with s codewords $C = \{c_1, c_2, \dots, c_s\}$ is created for a given dataset, and then each time series in the dataset is encoded according to the codebook. In the encoding process, each time series X is partitioned into many equal-size subsequences (the same size as the codeword) and for every subsequence the most similar codeword is found. By counting the appearance frequency for each codeword in the time series, we obtain a new representation of X as: $X' = f'_1, f'_2, \dots, f'_s$.

In FVQA, the length of the codewords is a very important parameter. Using different resolutions, we get totally different codebooks and different representations of the original time series, since each resolution level pays different attention to local and global information. In

order to keep both local and global information as much as possible, we suggest combining different resolution levels in similarity calculation, assigning appropriate weight to each of them. We call this new method Hierarchical Frequency-based Vector Quantization Approximation (HFVQA). Table 1 gives a brief description of the notation we use in the rest of this paper. In the following subsections, we introduce each of the components of our method.

Table 1. Symbol Table

X	Original time series, $X = x_1, x_2, \dots, x_n$ of length n
X'	Encoded form of the original time series $X' = f_1, f_2, \dots, f_s$
N	Number of time series in the dataset
n	Length of original time series
C	Codebook: a set of codeword $\{c_1, \dots, c_k, \dots, c_s\}$
s	Size of the codebook
l	Length of codeword

3.1 Codebook Generation

For each given dataset, a codebook is first generated with a clustering algorithm (such as GLA); each codeword in the codebook corresponds to a *key-subsequence*. We apply the GLA algorithm to generate a codebook of size s based on the dataset T of time series. Each time series in T is partitioned into a number of segments each of length l and each segment forms a sample that is used to generate the codebook. Each codeword in the codebook stands for a *key-subsequence* which is an approximation for a certain group of subsequences of length l . All the time series in the database are then encoded using the codebook.

The version of GLA we use requires a partition split mechanism to solve the initial codebook generation problem. The algorithm starts with a codebook containing only one codeword, the centroid of the whole data set. In each repetition and before the application of the Lloyd iteration, it doubles the number of codewords (and cells) from the previous iteration by splitting the most populous cells. In Table 2 we present an example of a codebook ($s = 16$; $l = 20$) of key-sequences.


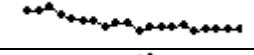
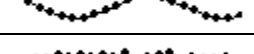
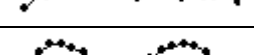
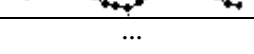

3.2 Data Encoding

After a codebook is generated, we can acquire a new representation of each time series in the dataset. In the process of encoding, every series is decomposed into subsequences of length l (length of the codeword). The

closest (based on a distance metric) entry, i.e., codeword, in the codebook for each subsequence is then found and the corresponding index is stored. After finding the corresponding codeword index for each segment, the appearance frequency of each codeword is counted. The new representation of a time series will be a vector (f_1, f_2, \dots, f_s) showing the appearance frequency of every codeword.

By applying this new encoding form, we can easily deal with time series with arbitrary large number of points, since we can always reduce their dimensionality to a rather small number which is the size of codebook (s).

Table 2. An example of a codebook of key-sequences

1	
2	
3	
4	
5	
...	...
16	

3.3 Distance measures

Based on the frequency of appearance of *key-sequences* within time series, the features of time series are extracted and content-based information retrieval is carried out accordingly. After encoding the time series using FVQA, we get a new representation with a rather small dimensionality, which records the frequency of appearance of *key-sequences* within time series. We still need a distance measure appropriate for this new representation.

As introduced in Section 2.2, there are several information retrieval models for VQ-based techniques. We choose the Histogram Model as the distance metric in FVQA, and all the experimental results presented in Section 4 are based on it.

3.4 Applying a hierarchical mechanism

By applying the histogram model, it is not difficult to identify the time series that are similar to a given query (i.e., that have similar frequent patterns). However, using only one codebook (analysis at a single resolution), introduces some problems that cannot be ignored.

First, although the local information of a time series is kept after the encoding process, the new representation of a time series is not recording the order among the indices of different codewords. This implies that some important global information of the time series is lost. This correspondingly increases the number of false alarms reducing the performance of the method. In Figure 2, we see several different time series whose FVQA representations are the same (2, 1, 1).

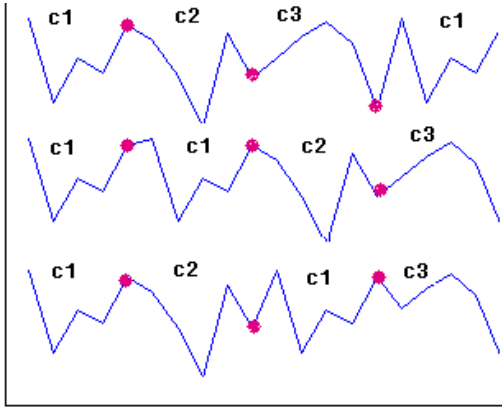


Figure 2. Different series with the same FVQA representation

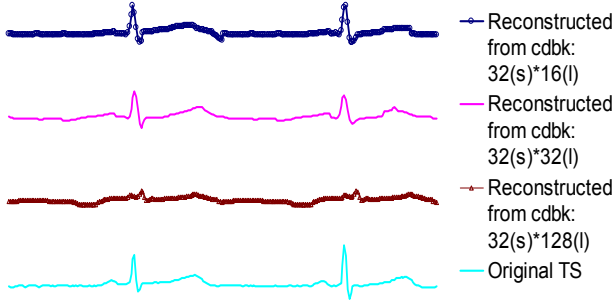


Figure 3. Reconstruction of time series using different resolutions

On the other hand, in real applications, it is not always easy to find a suitable resolution (correspondingly, a suitable codeword length). Moreover, an inappropriate codeword length may reduce the efficiency.

In order to solve these potential problems with FVQA, we suggest HFVQA (Hierarchical FVQA), which improves FVQA by involving different resolutions for encoding. While the encoding form of higher resolution pays more attention to the detail of local information, that of lower resolution has more global information. Figure 3 shows a time series and its reconstruction series using different resolutions. (For different resolution levels, the

sizes of codebooks are the same, 32, and the lengths of codewords are 128, 32, 16, respectively.)

By assigning reasonable weights to different resolutions, we define a new weighted similarity metric: Hierarchical Histogram Model:

$$S_{HHM}(q, d_j) = \sum_{i=1}^c w_i * S_{HMi}(q, d_j) \quad (5)$$

where c is the number of resolution levels.

Experiments show that HFVQA brings much higher accuracy than FVQA at single resolution. The only price for this improvement is slightly more computation, since we have to calculate the similarity on each resolution level before we can finally compute S_{HHM} . In the following experimental analysis we study the behavior of the hierarchical approach with different weights assigned to each resolution level. Using equal weights for all resolutions provides better results in most of the experiments we performed. The selection of the weights could also depend on the particular application. The proposed method provides the ability to include some prior knowledge about the domain in the selection of the weights.

4. Experiments

In time series similarity analysis, best matches retrieval and clustering are two of the most common and important applications. We performed experiments to analyze the ability and efficiency of our method in these two applications, and also address the following issues:

- How accurate is our method.
- How it compares to alternatives.
- How fast and scalable it is.

4.1 Best Match Searching

4.1.1 Experiment design. Many time series applications involve the best match searching. That is, given a query sequence, find the best k matches in the database (i.e., having the lowest dissimilarity with the query) or find all the time series whose dissimilarity with the query is below some predefined threshold.

In order to evaluate the performance of different approaches in best match searching, we need an evaluation metric.

Definition 3. For a given query, the set of time series which are actually within the same class as the query (given our prior knowledge) is taken as the standard set ($std_set(q)$), and the results found by different approaches ($knn(q)$) are compared with this set. The matching accuracy is defined as:

$$\text{Accu} = \frac{|\text{knn}(q) \cap \text{std_set}(q)|}{k} \times 100\% \quad (6)$$

In the definition above, $\text{knn}(q)$, is the k nearest neighbors for the query found by a certain method, while $\text{std_set}(q)$ is the prior knowledge about the dataset. In our experiments, every time series in the dataset is treated as a query, and the best k matches (k nearest neighbors) are sought within the whole dataset. The average accuracy (Accu) of a certain method is then calculated based on the matching results taking each time series as a query. The actual value of k we use depends on the number of time series within the same class.

Since we apply a hierarchical mechanism in our approach, an important parameter is the number of hierarchical levels. As shown in experiments, using too many levels does not bring much improvement, therefore we keep this number reasonably small (in our experiments, this number is 5). Based on this number and the length of time series in the dataset, the length of *key-subsequences* (or codewords) in different codebooks can be decided accordingly.

As for the sizes (the number of codewords) of different codebooks, there are two ways to make choices. One way is to decide the sizes of codebooks with the help of some prior knowledge, and another way is just to use an arbitrary small number as the number of *key-sequences* in all hierarchical levels.

In order to avoid the effects of scaling and shifting in analysis, Golding and Kanellakis [5] formalized an intuitive notions of exact and approximate similarity between time series patterns where the variance of time series is involved. In our work, before we actually perform any experiment, we preprocess the datasets with *zero-mean* normalization. That is, each time series X was

normalized as: $X = \frac{(X - \bar{X})}{\sigma(X)}$ where \bar{X} is the mean value

of X and $\sigma(X)$ is the standard deviation of X .

In our experiments, the value of k can vary, but for the purpose of demonstration, we just show the results when k is set to the number of time series within the same class.

4.1.2 Experiments on a synthetic dataset. In this section, we show the results of the experiments performed on SYNDATA dataset.

SYNDATA is a synthetic dataset which is downloadable from the UCI KDD archive [26]. This dataset contains 600 examples of control charts (each has 60 points) synthetically generated by the process in Alcock and Manolopoulos [3]. The time series belong to six different classes of control charts: Normal, Cyclic, Increasing trend, Decreasing trend, Upward shift, and Downward shift, with each class having 100 time series.

In the first half of the experiments, we considered prior knowledge about the dataset, i.e., that the number of

clusters is 6, and the size of the codebook corresponding to a higher resolution is larger (since there are more training samples available for that resolution). For the second half we did not take any prior knowledge into account and kept all the sizes of different codebooks the same small number (32 in this case). The important experiment parameters are listed in Table 3.

In addition to the above parameters, we need to assign appropriate weight to each hierarchical level in the dissimilarity calculation. In our experiments we tried several typical weight vectors.

Table 3. Experiment parameters for SYNDATA

Level	HFVQA with prior knowledge		HFVQA without prior knowledge	
	l	s	l	s
1	60	6	60	32
2	30	16	30	32
3	20	32	20	32
4	10	64	10	32
5	5	128	5	32

The experimental results on SYNDATA are shown in Table 4. The first element in the weight vector represents the weight assigned to the first level, the second element the weight assigned to the second level, and so on (e.g., with a weight vector [1 0 0 0], only the first level is involved in distance calculation). Accu1 is the matching accuracy (see Eq.(6)) when we apply some prior knowledge in training process, and Accu2 is the matching accuracy when no prior knowledge is involved.

The experimental results clearly demonstrate the effect of using a hierarchical mechanism: when more levels are involved, the retrieval accuracy improves. Also, with some prior knowledge about the dataset, we can extract more accurate feature information, and correspondingly, improve the matching accuracy. However, even without the prior knowledge, we can get reasonably good results. The hierarchical mechanism clearly helps in both cases.

Table 4. Experiment results on SYNDATA

HFVQA Weight Vector	Accu1	Accu2
[1 0 0 0 0]	0.5531	0.4135
[0 1 0 0 0]	0.7025	0.6072
[0 0 1 0 0]	0.5177	0.5117
[0 0 0 1 0]	0.4468	0.4807
[0 0 0 0 1]	0.3907	0.4564
[1 1 1 1 1]	0.8250	0.7461
[1 2 4 8 16]	0.7868	0.6964
[16 8 4 2 1]	0.8228	0.7187

Using the Naïve method over the same dataset, which directly use Euclidean Distance as the dissimilarity metric, we get an average accuracy of only 0.5112. Comparing with the results in Table 4, we can conclude that: for this dataset, the Naïve method did worse even than a single level FVQA, and HFVQA provides a much better matching accuracy besides the dimensionality reduction.

4.1.3 Experiments on GENE data. The GENE dataset is a subset of the NCI60 gene expression data from the National Cancer Institute. This dataset can be downloaded from [30].

Each series in this dataset consists of the gene expression values of 1375 genes. From the cell lines that are available we considered 41 cancer cell lines of 6 kinds of cancers: 6 central nervous system, 7 colon, 6 leukemia, 8 melanoma, 6 ovarian and 8 renal cancer cell lines. The ground truth is available for this data.

We performed similar experiments as with SYNDATA dataset. The experiment parameters and results are shown in Tables 5 and 6 respectively.

Table 5. Experiment parameters for GENE data

Level	HFVQA With prior knowledge		HFVQA Without prior knowledge	
	l	s	l	S
1	1375	6	1375	32
2	275	16	275	32
3	55	32	55	32
4	25	64	25	32
5	5	128	5	32

Table 6. Experiment results on GENE data

Weight Vector	Accu1	Accu2
[1 0 0 0 0]	0.5894	0.3252
[0 1 0 0 0]	0.8049	0.7317
[0 0 1 0 0]	0.6789	0.6781
[0 0 0 1 0]	0.7602	0.5813
[0 0 0 0 1]	0.5569	0.5285
[1 1 1 1 1]	0.8049	0.7805
[1 2 4 8 16]	0.8293	0.7480
[16 8 4 2 1]	0.7927	0.7764

From Table 6, it is clear that for this GENE dataset, the hierarchical mechanism also helps to improve the accuracy and HFVQA results in much better results than a single level FVQA.

Comparing with the average accuracy of Naïve method, which is 0.8455, the retrieval accuracy of HFVQA is a little worse, but still acceptable, especially considering that the retrieval efficiency has been

significantly improved because of the reduced dimensionality.

4.2 Comparing with other methods

In order to compare the efficiency and accuracy of HFVQA in similarity searches we considered other alternative methods including the Discrete Fourier Transform (DFT), straight Euclidean (Naïve), Dynamic Time Warping (DTW) and symbolic PAA [20].

For evaluation and comparison, every time series in the dataset is taken as a query, and the precision and recall pairs corresponding to the top 1,2,3... k retrieved time series are calculated. Then the average value of precision and recall are computed for the whole dataset. The actual k value is different for different methods.

For DFT, symbolic PAA and HFVQA, some parameters need to be set up for the experiments. For DFT, we take the first 6 non-zero coefficient; for symbolic PAA, codebook size is 16 and number of segments is 6; and for HFVQA we take the codebook size as 16 for each of the 5 resolution levels and use [1 1 1 1 1] as the weight vector.

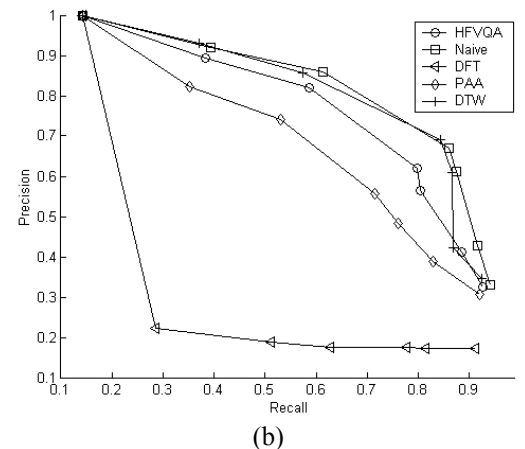
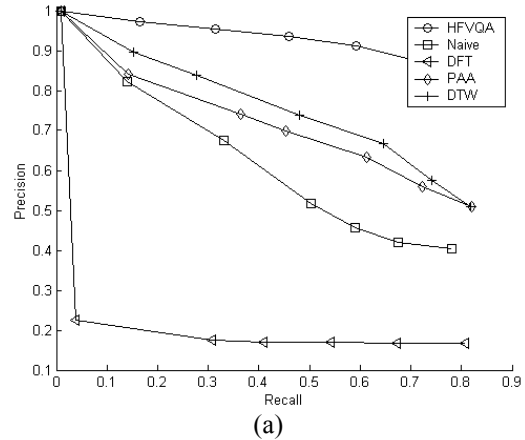


Figure 4. Precision-recall for different methods (a) On SYNDATA dataset (b) On GENE dataset

Figure 4(a),(b) shows the results on SYNDATA and GENE dataset respectively. Notice that for a fixed recall ratio, the fewer time series are retrieved the better, and subsequently the higher the precision is. Figure 4 shows that for the SYNDATA and GENE datasets the Naïve method and DTW have nearly the same performance while the DFT and symbolic PAA, two dimensionality reduction methods based on Euclidean distance, demonstrate performance worse than that of the Naïve method. As a frequency-based dimensionality reduction method, HFVQA achieves the best performance on SYNDATA and it is comparable to the Naïve and DTW methods on GENE data.

Besides accuracy, other considerations for a good method should include speed and scalability. In the application of information retrieval, for a given query, we need to scan the whole dataset and calculate the distance between the query and each of the time series in the dataset before the best matches are located. Depending on the method, an extra step of encoding the query may be required.

Figure 5 shows the processing time of different methods on datasets with various sizes. The experiment settings for different methods are the same as before.

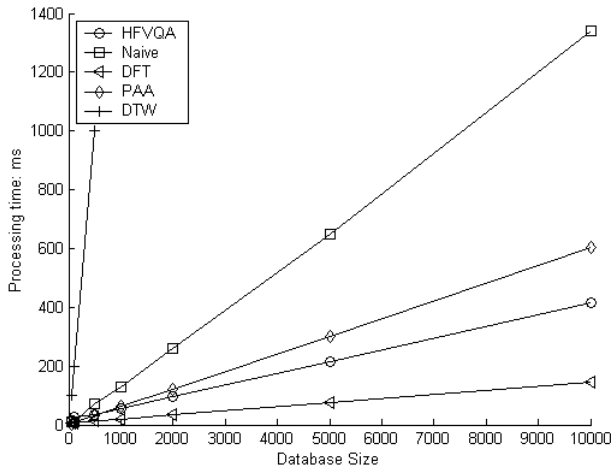


Figure 5. Processing time and scalability

In Figure 5, DFT shows the best processing efficiency with the shortest time, but considering the poor accuracy result shown in figure 4, it should not be taken as a good candidate.

In comparison to the other methods we considered here, although the encoding of the query consumes some time, HFVQA outperforms them all in speed when the database size is not too small.

4.3 Clustering experiments

4.3.1 Experiment design. For time series clustering, we conducted experiments on both synthetic and real datasets. The PAM (Partitioning Around Medoids) clustering algorithm was used to cluster the original time-series in every dataset. Different approaches applied in distance calculation will result in different distance matrix of the time series, and subsequently different clustering results.

In order to evaluate the clustering accuracy and quality of our approach, a cluster similarity metric was used. Given two clusterings $G = G_1, G_2, \dots, G_K$ (the true clusters), and $A = A_1, A_2, \dots, A_k$ (clustering result by a certain method), the clustering accuracy is evaluated with the cluster similarity defined as

$$\text{Sim}(G, A) = \frac{\sum_i \max_j \text{Sim}(G_i, A_j)}{k} \quad (7)$$

$$\text{where } \text{Sim}(G_i, A_j) = \frac{2 |G_i \cap A_j|}{|G_i| + |A_j|}$$

This metric was introduced in [8] to evaluate clustering results and was also used in [14]. The metric value ranges between 0 and 1, and it takes the maximal, i.e. 1, when the clustering result is perfect.

For each dataset, we took the same experiment parameters as what we used in section 4.1. In the first half of the experiments, prior knowledge about the dataset, i.e., the number of clusters is involved, while in the second half no such knowledge is applied. Considering the stochastic nature of the PAM algorithm, given a set of parameters, the experiment was repeated 10 times, and the average result was reported. For the purpose of comparison, clustering results with other methods are also provided.

4.3.2 Experiments on SYNDATA dataset. Taking the same parameters as shown in table 3, experiments of clustering were performed on SYNDATA dataset. The experiment results are listed in Table 7. Table 8 also shows the clustering results of some other methods.

Table 7. Clustering results of HFVQA on SYNDATA

Weight	HFVQA With prior knowledge		HFVQA Without prior knowledge	
	Mean	Std	Mean	Std
[1 0 0 0]	0.6953	0.0183	0.4707	0.0255
[0 1 0 0]	0.7035	0.0605	0.5800	0.0365
[0 0 1 0]	0.5030	0.0377	0.5092	0.0446
[0 0 0 1]	0.4318	0.0315	0.5345	0.0500
[0 0 0 1]	0.3954	0.0500	0.5133	0.0619
[1 1 1 1]	0.7940	0.0331	0.7543	0.0549
[1 2 4 8 16]	0.7655	0.0599	0.7480	0.0473
[16 8 4 2 1]	0.7886	0.0383	0.6726	0.0350

Table 8. Clustering results on SYNDATA

	DFT	PAA	HFVQA	NAIVE
Ave	0.2469	0.6507	0.7940	0.5536

It is clear that for this dataset, we cannot achieve satisfying performance using the Euclidean Distance as the distance metric, while the frequency-based method is very promising. The performance achieved by several single resolution levels is better than that of the Naïve method. By combining different resolution levels, the clustering result is further improved.

4.3.3 Experiments on GENE dataset. Similarly to the previous experiments, we used the same parameters as shown in Table 5 and carried out experiments on the GENE dataset. The results are shown in Table 9. Clustering results with different methods are also provided in Table 10.

Table 9. Clustering results of HFVQA on GENE

Weight	HFVQA with prior knowledge		HFVQA without prior knowledge	
	<i>Mean</i>	<i>Std</i>	<i>Mean</i>	<i>std</i>
[1 0 0 0 0]	0.6908	0.0456	0.3607	0.0485
[0 1 0 0 0]	0.7265	0.0594	0.6619	0.0344
[0 0 1 0 0]	0.6691	0.0510	0.6632	0.0551
[0 0 0 1 0]	0.7131	0.0535	0.6928	0.0440
[0 0 0 0 1]	0.5560	0.0574	0.5775	0.0394
[1 1 1 1 1]	0.8089	0.0411	0.7483	0.0449
[1 2 4 8 16]	0.8122	0.0396	0.7351	0.0493
[16 8 4 2 1]	0.7881	0.0345	0.7282	0.0474

Table 10. Clustering results on GENE

	DFT	PAA	HFVQA	NAIVE
Ave	0.2974	0.6548	0.8122	0.8202

For this dataset, even though the performance of the frequency-based method is slightly worse than that of the Naïve, it is still very satisfying. Note the obvious improvement when applying the hierarchical mechanism over the different resolution levels.

5. Discussion

The HFVQA approach that we proposed for reducing the high dimensionality in time series data to make their analysis more efficient is a natural extension of the piecewise constant approximation schemes proposed earlier. By applying Vector Quantization technique to extract high-level feature of the data and involving a hierarchical mechanism we were able to improve

performance and efficiency in time series similarity retrieval, especially in some domains where we could not get a good result using the Euclidean distance as the similarity metric. Here, we briefly present some directions in which our work can be extended.

First of all, even though under most circumstances, the involvement of hierarchical mechanism brings a better performance, it doesn't always do. So it will be interesting to go further to find out under what circumstance we can't involve hierarchical mechanism. Also, the assignment of weights to different resolution levels is very important, and it is a very interesting topic to figure out a more theoretical way to define the optimal weight assignment. Another interesting problem is related to the size of the codebook. When we train the dataset to generate the codebooks at different resolutions, what should be the appropriate number of codewords (codebook size)?

6. Conclusion

In this paper we introduced a new dimensionality reduction method, FVQA and its extension, HFVQA, for time series similarity analysis. By partitioning a sequence into equal-length segments and using vector quantization to represent each sequence by appearance frequencies of key-sequences, FVQA provides a more meaningful similarity metric for many domains, besides the improvement in efficiency because of the dimensionality reduction. Inheriting the benefits of FVQA, HFVQA involves a hierarchical mechanism to record both the local and global information of the original time series. Even though it requires a little more calculation than FVQA on single resolution level, HFVQA improves the retrieval and clustering accuracy a lot. Experiments we performed on both real and simulated datasets show that HFVQA brings much improvement to the single level FVQA. The proposed transformation on time series is very fast to process long time series, since the length of new representation is only related to the size of codebook. While the experiment results presented here mainly focus on similarity analysis and clustering, our approach can also be easily adjusted to some other applications, such as frequent pattern retrieval (i.e., motif discovery), association rule mining, and other data mining applications.

Acknowledgements

The authors would like to thank Eamonn J. Keogh for providing several datasets and source code through the UCR Time Series Data Mining Archive [17] as well as for encouraging us on this work. This work was supported in part by the Pennsylvania Department of

Health. The funding party specifically disclaims responsibility for any analyses, interpretations and conclusions.

References

- [1] Agrawal, R., Faloutsos, C. & Swami, A. (1993). Efficient similarity search in sequence databases. Proceedings of the 4th Int'l Conference on Foundations of Data Organization and Algorithms. Chicago, IL, Oct 13-15. pp. 69-84.
- [2] Agrawal, R., Lin, K. I., Sawhney, H. S. & Shim, K. (1995). Fast similarity search in the presence of noise, scaling, and translation in time-series databases. Proceedings of the 21st Int'l Conference on Very Large Databases. Zurich, Switzerland, Sept. pp. 490-501.
- [3] Alcock R.J. & Manolopoulos Y. (1999). "Time-Series Similarity Queries Employing a Feature-Based Approach", Proceedings 7th Hellenic Conference on Informatics. Ioannina, Greece, Aug. 27-29. pp.III.1-9.
- [4] Baeza-Yates, R.A. & Gonnet, G.H. (1999). A fast algorithm on average for all-against-all sequence matching. Proceedings of the String Processing and Information Retrieval Symposium, pp. 16-23.
- [5] D.Q. Goldin, and P.C. Kanellakis. On similarity queries for time-series data: Constraint specification and implementation. In Proceedings of Constraint Programming 95.Marseilles, France, 1995.
- [6] Faloutsos, C., Jagadish, H., Mendelzon, A. & Milo, T. (1997). A signature technique for similarity-based queries. Proceedings of the Int'l Conference on Compression and Complexity of Sequences. Positano-Salerno, Italy, Jun 11-13.
- [7] Faloutsos, C., Ranganathan, M. & Manolopoulos, Y. (1994). Fast subsequence matching in time-series databases. Proceedings of the ACM SIGMOD Int'l Conference on Management of Data. Minneapolis, MN, May 25-27. pp. 419-429.
- [8] Gavrilov, M., Anguelov, D., Indyk, P. & Motwani, R. (2000). Mining the stock market: Which measure is best? Proceedings of the International Conference on Data Mining and Knowledge Discovery (KDD '00), pp. 487-496.
- [9] Gersho, A. & Gray R. M. (1992). Vector Quantization and Signal Compression. Kluwer Academic Publishers.
- [10] Gusfield, D. (1997). Algorithms on Strings, Trees and Sequences. Cambridge University Press.
- [11] Hetlrad, M. L. (2002). A survey of recent methods for efficient retrieval of similar time sequences. <http://www.hetland.org/research/>
- [12] Höppner, F. (2001). Discovery of temporal patterns – learning rules about the qualitative behavior of time series. In Proceedings of the 5th European Conference on Principles and Practice of Knowledge Discovery in Databases, Freiburg, Germany, pp. 192-203.
- [13] Huhtala, Y., Kärkkäinen, J. & Toivonen, H. (1999). Mining for similarities in aligned time series using wavelets. Data Mining and Knowledge Discovery: Theory, Tools, and Technology, SPIE Proceedings Series, Vol. 3695. Orlando, FL, Apr. pp. 150-160.
- [14] Kalpakis, K., Gara, D. & Puttagunta, V.(2001). Distance Measures for Effective Clustering of ARIMA Time-Series. Proceedings of the 2001 IEEE International Conference on Data Mining, San Jose, CA, Nov 29-Dec 2. pp. 273-280
- [15] Keogh, E., Chakrabarti, K., Pazzani, M. & Mehrotra, S. (2001). Locally adaptive dimensionality reduction for indexing large time series databases. Proceedings of ACM SIGMOD Conference on Management of Data. Santa Barbara, CA, May 21-24. pp 151-162.
- [16] Keogh, E., Chakrabarti, K., Pazzani, M. & Mehrotra, S. (2000). Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases. Journal of Knowledge and Information Systems.
- [17] Keogh, E. & Folias, T. (2002). The UCR Time Series Data Mining Archive <http://www.cs.ucr.edu/~eamonn/TSDMA/index.html>. Riverside CA. University of California - Computer Science & Engineering Department
- [18] Keogh, E. & Pazzani, M. (1998). An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback. Proceedings of the 4th Int'l Conference on Knowledge Discovery and Data Mining. New York, NY, Aug 27-31. pp. 239-241.
- [19] Keogh, E. & Pazzani, M. (2000). A simple dimensionality reduction technique for fast similarity research in large time series databases. Proceedings of the Fourth Pacific-Asia Conference on Knowledge Discovery and Data Mining, Kyoto, Japan.
- [20] Lin, J., Keogh, E., Patel, P. & Lonardi, S. (2002). Finding motifs in time series. The 2nd Workshop on Temporal Data Mining, at the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. July 23 - 26. Edmonton, Alberta, Canada.
- [21] Linde, S., Buzo, A. & Gray, A. (1980). An algorithm for vector quantizer design, IEEE Transactions on Communications, vol. 28, pp. 84-95.
- [22] Lloyd, S. P. (1982). Least squares quantization in PCM. IEEE Transactions on Information Theory, IT(28), pp. 127-135.
- [23] Park, S., Chu, W.W., Yoon, J. & Hsu, C. (2000). Efficient search for similar subsequences of different lengths in sequence databases. Proceedings of the ICDE, pp. 23-32.
- [24] Piotr Indyk, Nick Koudas, S. Muthukrishnan: Identifying Representative Trends in Massive Time Series Data Sets Using Sketches. VLDB 2000: 363-372
- [25] Rafiei, D. (1999). On similarity-based queries for time series data. In Proceedings of the 15th International Conference on Data Engineering (ICDE), Sydney, Australia, pp. 410-417.

- [26] UCI KDD Archive. <http://kdd.ics.uci.edu>
- [27] Wu, Y., Agrawal, D. & El Abbadi, A. (2000). A comparison of DFT and DWT based similarity search in time-series databases. Proceedings of the 9th ACM CIKM Int'l Conference on Information and Knowledge Management. McLean, VA, Nov 6-11. pp. 488-495.
- [28] Yi, B-K & Faloutsos, C. (2000). Fast Time Sequence Indexing for Arbitrary L_p Norms. Proceedings of the VLDB, Cairo, Egypt, Sept..
- [29] Zhu, L., Rao, A. & Zhang A. (2002). Theory of Keyblock-based Image Retrieval. ACM Transactions on Information Systems, 20(2), pp. 224-257.
- [30] <http://genome-www.stanford.edu/nci60>