Information Sciences 476 (2019) 337–356

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

A fast LSH-based similarity search method for multivariate time series

Chenyun Yu[a], Lintong Luo[b], Leanne Lai-Hang Chan[c], Thanawin Rakthanmanon[d,e,*], Sarana Nutanong[e]

[a] Department of Computer Science, City University of Hong Kong, Hong Kong, China
[b] Department of Computer Science, University of California, Irvine, USA
[c] Department of Electrical Engineering, City University of Hong Kong, Hong Kong, China
[d] Department of Computer Engineering, Kasetsart University, Thailand
[e] School of Information Science and Technology, VISTEC, Thailand

ARTICLE INFO

Article history:
Received 24 November 2017
Revised 15 October 2018
Accepted 18 October 2018
Available online 19 October 2018

Keywords:
Similarity search
Query processing
Multivariate time series
Locality sensitive hashing
Dynamic time warping

ABSTRACT

Due to advances in mobile devices and sensors, there has been an increasing interest in the analysis of multivariate time series. Identifying similar time series is a core subroutine in many data mining and analysis problems. However, existing solutions mainly focus on univariate time series and fail to scale as the number of dimensions increase. Although, dimensionality reduction can reduce the impact of noisy information, the number of dimensions may still be too large. In this paper, an efficient approximation method is proposed based on locality sensitive hashing. It is a two-step solution which firstly retrieves candidate time series and then exploits their hash values to compute distance estimates for pruning. To probabilistically guarantee the result accuracy, an extensive error analysis has been conducted to determine appropriate LSH parameters. In addition, we also apply the proposed method to the PkNN classification and hierarchical clustering workloads. Finally, extensive experiments are conducted using both the real multivariate time series and the high-dimensional representations generated from univariate datasets in different query processing and data analysis workloads. Empirical results have verified the findings from the error analyses and demonstrated their benefits in terms of query efficiency when dealing with a collection of multivariate time series.

1. Introduction

In recent years, increasing research attention is being dedicated to addressing multivariate time series [7,19,29,34,35]. Benefiting from the development of Internet of Things (IoT) and sensor technologies, large scale of multivariate time series are being generated from a variety of sensor networks. For example, electroencephalography (EEG) and electrocardiography (ECG) outputs are multivariate in nature. In addition, following advances in deep learning research [25], a deep neural network like a *recurrent neural network (RNN)* [32] can be used to convert univariate time series to multivariate ones. For instance, Längkvist et al. [24] have shown that converting univariate time series dataset into a multivariate one results in a classification accuracy improvement in a speech recognition problem.

* Corresponding author.
E-mail addresses: chenyunyu4-c@my.cityu.edu.hk (C. Yu), thanawin.r@ku.ac.th (T. Rakthanmanon).

https://doi.org/10.1016/j.ins.2018.10.026
0020-0255/

The identification of similar time series is a basic operation in many time series data mining applications. To accelerate the processing of similarity search, a major strategy is to reduce the number of time series comparisons. From this viewpoint, existing solutions can be categorized into three types. The first is to use a cheap-to-compute lower bound function to quickly prune dissimilar time series [6,20,22,44]. Since pruning power depends on the tightness of the lower bound, performance degenerates when the number of dimensions increases. The second approach is to formulate an indexing technique in conjunction with a lower bound function. For instance, work [20] proposes an R-tree index [12] based on the lower bound. However, the R-tree-based method often degenerates into sequential scanning when the number of dimensions is increased [17]. The third research approach is to derive a new data representation for time series and then build indexes or adopt a suitable traditional techniques [5,23,38,39,42]. However, this approach is also necessary to deal with the following issues: (i) the cost of translating time series into a new representation; (ii) the cost of estimating distance between time series based on the new representations; (iii) the information loss caused by using new representations.

Despite the increasing prevalence of multivariate time series data, there has been a paucity of efficient solutions to handling multivariate time series with arbitrary dimensionality. Most existing indexing techniques are designed for the low-dimensional time series [21,43], so their performance degrades when the number of dimensions is large. Since a greater noise is usually associated with multivariate time series compared to the univariate setting, there is less discrimination in the former. This makes distance comparisons meaningless, which results in reduced pruning power. Some researchers have applied dimensionality reduction techniques [1,7,45] to mitigate the impact of noisy information and then performed a similarity search based on the useful subset of dimensions. However, the number of dimensions often remains too large to handle.

Objective. In this investigation, we focus on addressing the query efficiency problem for multivariate time series. To effectively handle an arbitrary number of dimensions, we apply an approximation method called *locality sensitive hashing* (LSH) [4,10,17]. LSH is a well-known method for solving the problem of approximate similarity search in high-dimensional spaces [17]. The method uses a family of *"locality preserving"* hash functions to map similar points into the same bucket with a significantly higher probability than those that are far apart. Given a d-dimensional dataset which contains n vectors, when searching the nearest neighbor using LSH with respect to a query point, the time complexity is sublinear with respect to n and polynomial with respect to d [31].

Challenges. Although applying LSH techniques to accelerate similarity search is straightforward, when dealing with multivariate time series there are several challenges that we need to address. Firstly, DTW, a commonly-used measure, does not satisfy triangle inequality, making indexing difficult. Secondly, since a multivariate time series is a collection of vectors rather than a single data point in high-dimensional spaces, a completely new strategy for candidate generation is needed while applying LSH to time series data. Finally, to probabilistically guarantee the result accuracy and control the cost of filtering the false positives, proper LSH parameters need to be determined based on the proposed candidate generation strategy.

Proposed solution. We propose a novel LSH-based method for multivariate time series retrieval. The objective is to reduce the number of exact time series comparisons which is often the main bottleneck. Specifically, LSH is applied in two different steps of the proposed solution. In the first step, we define a type of envelop for the query series Q to limit the search space, and then LSH is used to identify candidate time series that are confined inside the envelop. In the second step, the LSH hash values are exploited for candidate pruning in order to reduce the number of exact distance computations.

Since LSH is an approximation technique, controlling the accuracy level of the results is a mater of major concern. Therefore, we conduct an extensive analysis on the property of the state-of-the-art LSH scheme, i.e., QALSH [16]. By performing an error analysis based on the candidate generation strategy, we derive the relationship between the accuracy of range query and the LSH parameters. The findings from this analysis are then used to help determine the LSH parameters according to a specified accuracy requirement.

In most time series application domains, DTW is regarded as the best similarity measure [6]. However, the Euclidean distance (ED) is an acceptable measure in some applications. This is because: (i) the classification errors of NN-DTW and NN-ED tend to converge as the size of dataset increases [36]; (ii) calculating the Euclidean distance is much less expensive than DTW. In the present work, we provide details of using the proposed method based on DTW and ED, as well as the parameter determination and experimental evaluation.

Experimental studies. Extensive experimental studies have been conducted using real multivariate time series datasets obtained from two different sources: (i) an array of sensors; (ii) RNN structure learning outputs. The experimental studies include three different types of workloads: kNN query, nearest neighbor classification, and hierarchical clustering.

Assessments on both efficiency and accuracy are included in the experimental studies. As for accuracy assessment, experimental results show that the proposed method can achieve a high accuracy level (kNN recall > 0.92) with exact DTW computations being needed for only 10% of the time series in the EEG dataset. On the other hand, the best competitor (the UCR suite [36]) has to perform exact DTW computations for more than 32% of the time series in the same dataset.

In addition, we verify that the classification results based on RNN structure learning outputs provide an improvement with respect to a method which directly operates on univariate time series. Specifically, the LSTM neural network [11,32] is applied on the univariate time series dataset[1] (ElectricDevices dataset) to generate 16-dimensional time series data and

[1] http://www.cs.ucr.edu/~eamonn/time_series_data/

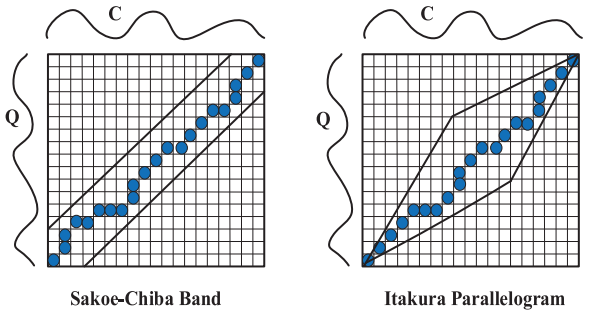


Fig. 1. Limiting the warping path with global constraints.

improve the accuracy from 74% to 83% on the P*k*NN classification workload type. This finding conforms with the results reported by Längkvist et al. [24].

Contribution summary. This work makes the following contributions:

(i) A fast LSH-based similarity search method for *multivariate* time series,
(ii) An error analysis on the proposed method,
(iii) An approach for parameter determination to probabilistically guarantee the result accuracy,
(iv) An extensive array of empirical evaluations of the proposed method in comparison to existing methods.

Structure. The rest of this paper is organized as follows. Section 2 provides background information on multivariate time series and locality sensitive hashing. Section 3 presents related work about time series similarity search. Section 4 focuses on DTW and describes the LSH method we propose. Section 5 presents an error analysis and an LSH parameter setting approach for the method proposed in Section 4. Section 6 discusses details of generalizing the proposed method to support the ED-based multivariate time series query processing. In Section 7, we apply the proposed method to the P*k*NN classification and hierarchical clustering problems. In Section 8, results from extensive experimental studies are reported. Section 9 concludes the paper.

2. Definitions and background

2.1. Time series

Definition 1. (Univariate Time Series) A single dimensional time series $X = x_1, x_2, \ldots, x_T$ is a set of real values, where each x_i is a real number and T represents the length of the time series.

Definition 2. (Multivariate Time Series) A *multivariate time series (MTS)* is a series of high-dimensional vectors. Specifically, a d-dimensional MTS with a length T can be represented as $[< x_{1,1}, x_{1,2}, \ldots, x_{1,d} >, \ldots, < x_{T,1}, x_{T,2}, \ldots, x_{T,d} >]$.

2.2. Dynamic time warping

Dynamic time warping (DTW) is a more reliable and robust similarity measure than the Euclidean distance. This is because, DTW involves computing the optimal alignment between two time series [6]. To align two time series Q and C with a length of m, an $m \times m$ matrix is built first. The (ith, jth) element is the distance between q_i and c_j ($d(q_i, c_j) = (q_i - c_j)^2$). As shown in Fig. 1, a warping path P is a set of K elements in the matrix: $P = p_1, p_2, \ldots, p_K$. P is required to meet three conditions, namely (i) it starts from (q_1, c_1) and finishes at (q_m, c_m); (ii) the points in the warping path must be monotonic; (iii) it has to be continuous, i.e., the steps are limited to adjacent cells. We are interested in the warping path which minimizes the warping cost: $DTW(Q, C) = \min\left\{\sqrt{\sum_{k=1}^{K} p_k}\right\}$. To improve the accuracy by preventing pathological warping, global constraints are added to restrict the warping path. Two typical global constraints are Sakoe-Chiba band [37] and Itakura parallelogram [18] as shown in Fig. 1. The Sakoe-Chiba band constrains the warping path inside a band of fixed width around the diagonal path while the latter uses a parallelogram.

In a multivariate setting, DTW-D [40] is a commonly used measure which considers the dependency among all dimensions.

Definition 3. (DTW-D) Given two multivariate time series Q and C, DTW-D returns the optimal alignment where the distance function $d(q_i, c_j)$ is generalized to the high-dimensional case. Assuming that i and j are the index matching along the warping path, $d(q_i, c_j)$ is defined as $\sqrt{\sum_{m=1}^{d}(q_{i,m} - c_{j,m})^2}$, where m refers to the mth dimension at each time point of a time series.

As stated in the introduction section, existing algorithms can be extended to search multivariate time series by using DTW-D. However, their performance reduces as the number of dimensions increases. In this investigation, we focus on designing a practical LSH-based solution to accelerate multivariate time series query processing.

2.3. Locality sensitive hashing

2.3.1. Research overview: LSH

Locality sensitive hashing (LSH) [17] is a well-known approximation technique for similarity search in a high dimensional space. In this method, a family of LSH functions are used to organize data objects into buckets. For each LSH function, the collision probability of any two objects depends on the distance between them. A closer distance represents a greater probability. Specifically, we can define the properties of a family of LSH functions as follows.

Definition 4. (LSH family) Given a distance function $D()$ in a vector space S, an LSH family $H = \{h : S \to U\}$ maps the vector space, S, to another space, U. A hash family H is said to be (r, cr, p_1, p_2)-sensitive for S if it satisfies the following constraints: For any $\mathbf{v}_1, \mathbf{v}_2 \in S$, (i) if $D(\mathbf{v}_1, \mathbf{v}_2) \le r$, then $\Pr[\mathbf{v}_1$ and $\mathbf{v}_2$ collide under $h] \ge p_1$; (ii) if $D(\mathbf{v}_1, \mathbf{v}_2) \ge cr$, then $\Pr[\mathbf{v}_1$ and $\mathbf{v}_2$ collide under $h] \le p_2$, where $c > 1$ and $p_1 > p_2$.

The LSH method was first proposed by Indyk et al. [17] in the Hamming space (l_1 norm). Subsequently, Datar et al. [4] constructed LSH functions based on the p-stable distribution and proposed the E2LSH method for the Euclidean space. To save on space consumption and improve the reusability of hash values, many LSH variants have been proposed. A detailed introduction to them is available in [10,41,46].

2.3.2. The query-aware LSH mehtod

Recently, Huang et al. [16] proposed a query-aware LSH scheme called QALSH which works with any $c > 1$ in high dimensional spaces. For any vector $\mathbf{v}$ in the Euclidean space, a QALSH function can be defined as $h(\mathbf{v}) = \mathbf{a} \cdot \mathbf{v}$, where $\mathbf{a}$ is a d-dimensional vector randomly chosen from a standard normal distribution. Using this hash function, vector $\mathbf{v}$ is projected onto a real number along a line identified by $\mathbf{a}$. For a specified bucket size ω, we may say that object x collides with object q under $h()$ if $|h(q) - h(x)| \le \frac{\omega}{2}$, i.e., x and q fall in the same bucket.

In the QALSH method, L query-aware LSH functions are used to compute hash values for the points in the dataset. Given a collision threshold, T_s, and query point q, if any object x collides with q under at least T_s hash functions, x will be added into the candidate neighbor set. Next, the real neighbors are identified through distance comparison between the candidates and q.

For two vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ whose Euclidean distance is s, their collision probability $p(s)$ is defined as follows.

$$p(s) = Pr\left[|h(\mathbf{v}_1) - h(\mathbf{v}_2)| \le \frac{\omega}{2}\right] = \int_{-\frac{\omega}{2s}}^{\frac{\omega}{2s}} f(t)\, dt \tag{1}$$

where $f(t) = \frac{1}{\sqrt{2\pi}} e^{\frac{-t^2}{2}}$. For a fixed ω, the collision probability $p(s)$ decreases monotonically as the distance s increases.

3. Related work

3.1. Univariate time series retrieval

Great success has been achieved in dealing with univariate time series retrieval over the past two decades [9,20,27,36]. Ding et al. [6] have shown that DTW is the best similarity measure between time series in most application domains. Since DTW incurs a high computational cost, many methods have been proposed to accelerate the query processing by simplifying the implementation of DTW [28,47]. Another efficiency improvement strategy is to reduce the number of time series comparisons, which can be sub-categorized into 3 different approaches. First, we use a hierarchy of lower bounds to directly scan time series dataset and prune irrelevant sequences. Second, a lower bound function is used in conjunction with an index. Third, approximate results are produced with a predictable loss of accuracy.

I: Lower bound scan. The lower bounds of DTW [20,22,44] can be used in a sequential scan for pruning. A tighter lower bound provides a stronger pruning power. Rakthanmanon et al. have proposed an efficient method called UCR Suite [36] which can prune more than 99% of DTW calculations in a large-scale NN search processing. However, for multivariate time series retrieval, the UCR method has reported a reduced pruning power. This is because, its pruning ability depends on the lower bound of distance. As the number of dimensions increases, the lower bound becomes loose. We have demonstrated the validity of this claim in Tables 4 and 5 (Section 8.2).

II: Indexing. Keogh et al. have proposed a method for indexing time series based on the LB-Keogh lower bounding function [20]. However, this method is also affected by the curse of dimensionality due to the fact that it is R-tree-based [17]. Lin et al. [28] have come up with a symbolic representation for a time series, called SAX, and constructed indices based on their iSAX-MINDIST lower bound function. The method is superior to the R-tree-based method in the univariate case in terms of scalability. However, as the number of dimensions increases, the space requirement and query processing cost increase exponentially. This is because the symbol-based index structure is hierarchical and contains non-overlapping regions,

Table 1
Algorithm comparisons.

Method	Recall & Precision	Index	Multivariate Support
LB-based scan (UCR Suite [36])	100% recall, Precision depends on lower bound	-	Limited to small number of dimensions Reason: loose lower bound
R-tree-based [20]	100% recall, Precision depends on lower bound	R-tree	Degenerate to linear scan
iSAX [39]	High recall High precision	Symbolic tree	Limited to small number of dimensions Reason: *SAX* index table is exponential wrt. d
SSH [30]	Depending on the parameters	Weighted minhash	parameters are difficult to determine in multivariate setting
Dimensionality reduction [7,45]	Depending on the search method	-	the number of dimensions may be still too large
LSH-based (Proposed)	High recall High precision	LSH	Polynomial wrt. d [31]

where each index value can be considered as a reference to a subspace generated by grid partitioning [28,39]. As a result, with the increase of dimensions, the number of nearby subspaces around the query time series increases exponentially. Consequently, generalizing SAX for multivariate time series can entail high costs while query processing.

III: Approximation. Shieh and Keogh [39] have proposed a novel multi-resolution symbolic representation called iSAX to support both fast exact search and fast approximate search for a given time series. Schafer represents time series as a string of symbols using iSAX and applies the LSH method to solve the string similarity problem [38]. Although they report a high recall and precision, extending the iSAX index to multivariate time series still remains to be explored. Recently, Luo et al. [30] have proposed a hash indexing scheme called SSH for the query processing over massive-scale time series. The scheme is efficient in the top-k query for low-dimensional time series, especially when the sequence is long and k is small. However, how to efficiently apply SSH to multivariate time series is not well studied. Specifically, there is *no* theoretical analysis for parameter determination that has been provided.

3.2. Multivariate time series retrieval

In recent years, there has been increasing interest in dealing with multidimensional time series due to developments in mobile devices and sensors. To accelerate the query processing, Vlachos et al. have proposed an efficient index structure for trajectory data [43]. Since their index structure is R-tree-based, it is affected by the curse of dimensionality. Another direction is to perform a similarity search using the useful sub-dimensionality [1,2,7,45], i.e., dimensionality reduction. Although query efficiency can be improved by reducing the impact of noisy dimensionality, the number of dimensions can still be overly large.

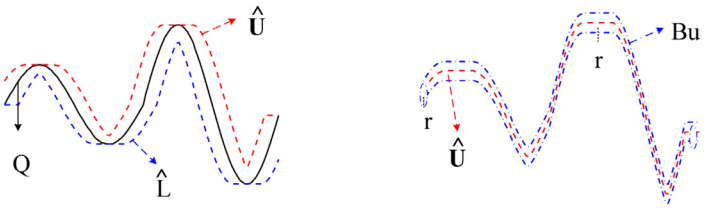
3.3. Summary

The characteristics of our solutions and well-known existing techniques are summarized in Table 1. Even though the recall of the R-tree-based method and the lower bounds pruning techniques is 100%, we may need to calculate the exact distance between the query and many false positives that arise when the number of dimensions are large. The iSAX approach allows for both exact search and approximate search. They report a high recall and precision, but for multivariate time series, the iSAX method may end up with an expensive query processing cost. Despite the fact that performing similarity search after dimensionality reduction can help improve query efficiency, the number of useful dimensions can still be large.

The solution we propose in this paper is an approximation method which attempts to reduce the number of candidates by allowing a small number of false negatives. In terms of the multivariate support, since the proposed method is based on LSH, it is scalable as the number d of dimensions increases. In addition, our method can be used in conjunction with many other dimensionality reduction methods [1,7,45] to achieve superior efficiencies while working with all useful dimensions for similarity search.

4. Proposed solution

As stated in the Introduction section, DTW is a similarity measure which is commonly used in time series applications. In this section, we mainly address the efficiency problem and introduce the proposed method for the DTW-based similarity search over multivariate time series. Since the Euclidean distance is also required in some application domains, we modify the LSH-based method to support the ED-based multivariate time series retrieval in Section 6.



(a) Bounding time series $\hat{U}$ and $\hat{L}$ of time series Q. (b) r-envelop of $\hat{U}$

Fig. 2. Envelops of a time series *Q*.

4.1. Overview

In solving a time series retrieval problem, we usually need to identify the top-k nearest neighbors of the query time series Q, or find those sequences whose distances to Q are smaller than a threshold θ. As stated in Section 3, the indexing techniques based on lower bound functions [20,22,44] and symbolic representation [28,39] are affected by an increase in the number of dimensions [19]. To help accelerate the processing of similarity search, we now examine two approaches: (i) finding an efficient way to apply locality sensitive hashing; (ii) reducing the number of time series comparisons.

Intuitively, if two time series are similar, the locations of their data points are likely to be close to each other [20,40]. Therefore, those sequences which are faraway from the query time series can be safely ignored. In the proposed solution, envelops around the query series Q are defined to limit the search space. Unlike the envelop in the UCR suite [36] which is used for lower bound calculation, the proposed envelop is used to limit the position of the similar time series of Q.

In the DTW-based query processing, three envelops are created in two steps. As shown in Fig. 2(a), we first build bounding time series $\hat{U}$ and $\hat{L}$ for the query time series Q (as Keogh et al. did in their work [20]). Let s denote the warping range. We define two new d-dimensional multivariate time series $\hat{U}$ and $\hat{L}$. The vectors at time point i of $\hat{U}$ and $\hat{L}$ are as follows:

$$\hat{U}_i = \langle \max(q_{i-s,1}, q_{i+s,1}), \ldots, \max(q_{i-s,d}, q_{i+s,d}) \rangle$$

$$\hat{L}_i = \langle \min(q_{i-s,1}, q_{i+s,1}), \ldots, \min(q_{i-s,d}, q_{i+s,d}) \rangle$$

Next, an r-range envelop is defined around each of the time series Q, $\hat{U}$ and $\hat{L}$ respectively. Assume that B_u is the r-envelop of $\hat{U}$ as shown in Fig. 2(b). At any time point i, the Euclidean distance between $\hat{U}_i$ and $B_{u,i}$ is equal to r ($ED(\hat{U}_i, B_{u,i}) = r$). If the time series C is similar to Q, it is likely to be confined inside the space formed by at least one of the three r-envelops.

We utilize the properties of LSH and propose an efficient strategy to identify the candidate neighbors of Q (Section 4.3). Since the retrieved candidate set may include irrelevant time series, the cost of DTW computation in the filtering process can be high. To improve the efficiency, two pruning techniques are provided in Section 4.4 to reduce the number of DTW computations. For the purposes of determining proper range query threshold θ and the envelop size r, we discuss the sample selection and statistical analysis over the dataset in Section 5.1. In order to ensure the quality of results, an error analysis based on the proposed LSH method is described in Section 5.2. The results from this analysis are then used to determine suitable LSH parameters when an error rate is specified. Similarly, we also introduce the details of applying the proposed method for ED-based multivariate time series retrieval in Section 6, as well as the pruning techniques themselves and parameter determination.

4.2. Framework description

To accelerate the DTW-based similarity search for large-scale multivariate time series, (i) we create three envelops around the query time series to limit the search space; (ii) we quickly generate candidate neighbors which are confined inside at least one envelop of the query time series; (iii) then dissimilar time series in the candidate set are filtered using efficient pruning techniques; and finally (iv) results are outputted by exact DTW calculation.

Fig. 3 provides an overview of our similarity search framework which uses an LSH method to identify neighbors for multivariate time series. This framework can be considered as a pipeline with the following four steps: statistical analysis, parameter setting, preprocessing, and querying.

(a) *Statistical analysis.* (i) Sampling a subset D' to cover the entire dataset D using Algorithm 2. (ii) For every time series pair (X'_i, X'_j) in set D', compute the DTW-Euclidean ratio $\frac{DTW(X'_i, X'_j)}{ED(X'_i, X'_j)}$. Then calculate the average a and the standard deviation SD of the ratios of all pairs (X'_i, X'_j). (iii) Compute the distribution of the DTW distance between all pairs (X'_i, X'_j) in D'.

(b) *Parameter setting.* (i) Determine the distance threshold θ and envelop size r for the similarity search in D based on the statistical analysis (Section 5.1.2). (ii) Determine the LSH parameters K and L according to the false negative rate δ for identifying the candidate neighbors of Q. Details concerning parameter determination are given in Section 5.2.

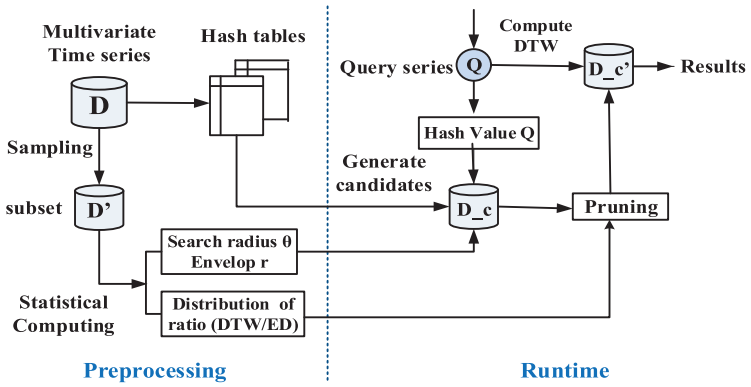


Fig. 3. Framework for similarity search over multivariate time series.

(c) *Preprocessing.* Calculate hash values and store them in hash tables for the time series in set D (Section 4.3.1).
(d) *Runtime Query Processing.* (i) Once a query time series Q has been provided, its hash values are calculated for comparison. (ii) The candidate time series are then obtained through the hash value comparison (Section 4.3.2). (iii) We then use the proposed pruning method and the best first search principle to return the final result set based on exact DTW computation.

4.3. LSH-Based multivariate time series retrieval

In this subsection, an LSH-based scheme is proposed to quickly identify candidate time series confined inside at least one envelop of query time series Q. We use compound QA-LSH functions [46] and redefine the meaning of collision for hash values and compound hash values. The collisions are then exploited to help identify candidate time series.

4.3.1. Computing hash values.

A multivariate time series is defined as a series of d-dimensional vectors: $X = x_1, \ldots, x_T$. With an LSH function $h()$, the d-dimensional vector at each time point of X is projected onto a real value. As a result, $h()$ transforms the time series X into a series of real values as $h(X) = h(x_1), \ldots, h(x_T)$.

In the proposed LSH method, K hash functions h_i are combined into a compound hash function G_i, and L compound hash functions are then used to compute the hash values for a given time series. The structure of hash functions are defined as: $\{G_1 = (h_{1,1}, \ldots, h_{1,k}), \ldots, G_L = (h_{L,1}, \ldots, h_{L,k})\}$. We can see that, while using L compound hash functions, the hash values of time series X can be regarded as a tensor hash$[L][K][T]$.

If the values of K and L are too small, the results would not be stable since the number of random projections are insufficient. To ensure the stability of query results, we apply different hash functions for the vector at each timepoint.

4.3.2. Strategy for generating candidate series.

As stated in Section 4.1, we want to find the time series inside the envelop of query sequence for later pruning and comparisons. Intuitively, if two multivariate time series are similar, it is likely that most of their vectors at the corresponding time point would also be similar [20,40]. As a result, similar multivariate time series would likely have a large number of colliding LSH values.

Next, we provide new definitions of LSH collisions for multivariate time series and design a new strategy to identify candidate time series. *Note that the way we define LSH collisions here is different from that of the classical LSH [4,10].* This is because, a time series is a collection of points rather than one single data point in a high-dimensional space.

For clarity, the hash value of a multivariate time series Q is defined as $h(Q) = h(q_1), \ldots, h(q_T)$, where $h(q_i)$ represents the projection of a vector at time point i of Q. Similarly, we compute the hash values for the bounding multivariate time series $\hat{U}$ and $\hat{L}$ of sequence Q. A compound hash value of Q is defined as $G(Q) = h_1(Q), \ldots, h_K(Q)$.

- ***Projection collision.*** Consider a time series C, at any time point i if $|h(q_i) - h(c_i)| \leq \frac{\omega}{2}$ or $|h(\hat{U}_i) - h(c_i)| \leq \frac{\omega}{2}$ or $|h(\hat{L}_i) - h(c_i)| \leq \frac{\omega}{2}$, there is a *projection collision* between Q and C at timepoint i under hash function $h()$.
- ***Hash collision.*** If the number of projection collisions between Q and C are larger than a given threshold T_s, there is a *hash collision* between Q and C.
- ***Group hash collision.*** Let the compound hash values of time series Q and C be $G_i(Q) = h_1(Q), \ldots, h_K(Q)$ and $G_i(C) = h_1(C), \ldots, h_K(C)$, respectively. If the hash collision between Q and C happens under all K hash functions in a compound hash function $G_i()$, there is a *group hash collision* between Q and C.

Setting the value of projection collision threshold T_s: The projection collision threshold T_s can be determined by using the expected number of their projection collision over T timepoints. That is, $T_s = \alpha T$, where α denotes the projection colli-

Algorithm 1: r-envelop candidate generation.

Input: hashSET[M][L][K][T], hashQ[L][K][T], hashU[L][K][T], hashL[L][K][T], and the projection collision threshold T_s
Output: candidate neighbor set *Cset*

```
1  Cset = ∅
2  for m = 1; m < M; m + 1 do
3      group_collision=false
4      for l = 1; l < L; l + 1 do
5          hash_collision=0
6          for k = 1; k < K; k + 1 do
7              projection_collision=0
8              for t = 1; t < T; t + 1 do
9                  if abs(hashQ[l][k][t] − hashSET[m][l][k][t]) ≤ ω/2
10                 OR abs(hashU[l][k][t] − hashSET[m][l][k][t]) ≤ ω/2
11                 OR abs(hashL[l][k][t] − hashSET[m][l][k][t]) ≤ ω/2 then
12                     projection_collision+=1
13                 end
14             end
15             if projection_collision ≥ Ts then
16                 hash_collision+=1
17             end
18         end
19         if hash_collision==K; then
20             group_collision=true
21         end
22     end
23     if group_collision==true; then
24         Cset = Cset ∪ m
25     end
26 end
27 return Cset
```

Algorithm 2: Sample selection.

Input: Dataset $D[M]$, sampling interval R
Output: Subset D'

```
1  D' = {};
2  for m = 1; m < M; m + 1 do
3      state=1;
4      for each i ∈ D' do
5          if Euclidean_dist(D[i], D[m]) ≤ R then
6              state=0;
7              break;
8          end
9      end
10     if state == 1 then
11         D' = D' ∪ {m};
12     end
13 end
14 return D';
```

sion percentage and T represents the length of time series. In Section 5.2.3, we will discuss in details how to determine α and the LSH parameters.

Identifying candidate time series: As for the strategy of identifying candidate time series confined inside the envelop of query time series, we combine the concepts of the E2LSH [4] and C2LSH [10] methods.

For two multivariate time series Q and C, if there is at least one "group hash collision" among L compound hash functions, series C should be considered as a candidate time series with respect to Q. We assume that we have already computed the hash value matrix hashQ[L][K][T] for query series Q, and the hash value matrix hashU[L][K][T] for the upper bounding time series $\hat{U}$, and the hash value matrix hashL[L][K][T] for the lower bounding time series $\hat{L}$, and hashSET[M][L][K][T] for all series in dataset D, where M denotes the total number of time series in the dataset, L denotes the number of compound hash functions, K denotes the number of projections in each compound hash function, and T denotes the length of each time series.

The structure of the candidate generation process is given in Algorithm 1 . The loop between lines 4 to 22 is used to check whether there is a group hash collision between Q and the current time series. Specifically, The loop from lines 8 to 14 counts the number of projection collisions, if the projection collision counting is greater than T_s, a hash collision happens

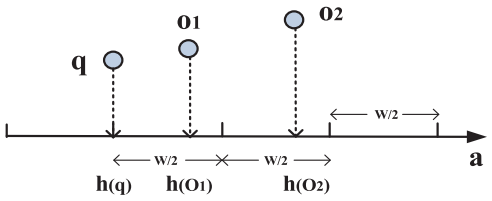


Fig. 4. Hash projection in the Euclidean space.

between Q and the current time series. If the hash collision happens at all K hash projections of a compound hash function, a group hash collision is reported (lines 19–21). Therefore, the current time series is inserted into the candidate neighbor set (lines 23 to 25).

4.3.3. Space cost analysis

Assume there are M multivariate time series in the dataset and the vector at each timepoint is d-dimensional. If the length of time series is T, a space with a size of $O(MTd)$ is required to store the dataset. Apart from this space cost, an extra space cost is required to maintain the hash values in the proposed method. As stated in Section 4.3.1, K query-aware hash functions are combined into a compound hash function $G_i(h_1(), \ldots, h_K())$, and L compound hash functions $G_1(), \ldots, G_L()$ are used to compute the hash values for all multivariate time series. Since a hash function projects the vector at each timepoint into a real value, the space cost is $O(MLKT)$. Therefore, the total space cost is $O((d + LK)MT)$. If $d \geq LK$, the total space consumption of the proposed method is $O(MTd)$.

4.4. Efficiency improvement: pruning techniques

While using LSH, many dissimilar time series may be included in the candidate set. To reduce the number of DTW calculations, we introduce an additional pruning step based on the following observation. If the order of DTW comparison is determined based on their similarity, we can safely stop the filtering process when the current DTW distance is much greater than the threshold. Consequently, the efficiency can be improved by allowing an early termination.

Specifically, the pruning steps include: (i) approximating the Euclidean distance between two series based on their hash values; (ii) computing pseudo bounds of DTW according to the approximate Euclidean distance; (iii) defining measures for ranking candidate series based on the estimated DTW bounds; (iv) applying the best-first search approach to return correct results.

4.4.1. Pruning technique 1: LB-Pru

The first pruning technique is based on the observation that if the distance between two points is small, the difference between their hash values is likely to be small. Specifically, each LSH function is a linear projection in the Euclidean space. As a result, we can use the difference between the hash values at all time points to estimate the Euclidean distance between two time series.

Fig. 4 illustrates the relationship between the hash value differences and the actual Euclidean distances between points. Given a bucket size ω, a hash function $h()$ selected from (r, cr, p_1, p_2)-sensitive LSH family projects points o_1 and o_2 along a line **a**. When a query q arrives, we compute the projection $h(q)$ and check the interval $[h(q) - \frac{\omega}{2}, h(q) + \frac{\omega}{2}]$. Since the difference between the projections of o_1 and q are smaller than $\frac{\omega}{2}$, we call q and o_1 collides under hash function $h()$.

According to the definition of query-aware LSH family, when the Euclidean distance between vectors v_1 and v_2 is less than a specified distance r, they are likely to collide with each other. That is, if $0 \leq ED(v_1, v_2) \leq r$, the difference between their hash values (projections) $|h(v_1) - h(v_2)|$ is likely to be within the range of $[0, \frac{\omega}{2}]$. Similarly, if $nr \leq ED(v_1, v_2) \leq (n+1)r$, the difference between their hash values is likely to be $\frac{n\omega}{2}$ to $\frac{(n+1)\omega}{2}$. Then we have $n \leq |\frac{2(h(v_1)-h(v_2))}{\omega}| \leq (n+1)$.

Approximate Euclidean distance: The Euclidean distance between time series Q and C can be estimated by considering the difference between hash values at each time point over L group hash functions. The lower bound and upper bound of their estimated Euclidean distance can be calculated as follows:

$$LB\_ED(Q, C) = \sum_{i=1}^{L} \sum_{j=1}^{K} \sqrt{\sum_{n=1}^{T} (\frac{2|(h_{i,j}(q_n) - h_{i,j}(c_n))|}{\omega})^2 r^2} \tag{2}$$

$$UB\_ED(Q, C) = \sum_{i=1}^{L} \sum_{j=1}^{K} \sqrt{\sum_{n=1}^{T} (\frac{2|(h_{i,j}(q_n) - h_{i,j}(c_n))|}{\omega} + 1)^2 r^2} \tag{3}$$

Estimate DTW: To estimate the bounds of the DTW distance, we apply a sampling technique commonly used in the set coverage problem [8]. Specifically, we statistically calculate the relationship between DTW and the Euclidean distance by sampling from the dataset [48] as stated in Section 5.1.1. After computing the ratio of Euclidean and DTW distance for

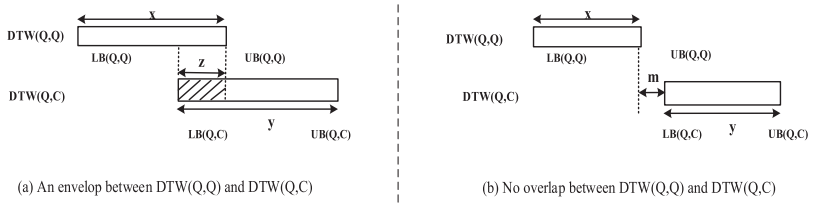


Fig. 5. Pseudo-lower bound and upper bound of DTW distance.

the pairs inside the sampled dataset [3], we calculate the mean a and standard deviation SD of those ratios. For each hash function, we randomly define a value ratio from $[a - SD, a + SD]$ to transform the Euclidean distance into DTW distance. The approximate DTW bounds are as follows:

$$LB(Q,C) = \sum_{i=1}^{L}\sum_{j=1}^{K} \text{ratio}_{i,j} \sqrt{\sum_{n=1}^{T} \left(\frac{2|(h_{i,j}(q_n) - h_{i,j}(c_n))|}{\omega}\right)^2 r^2} \tag{4}$$

$$UB(Q,C) = \sum_{i=1}^{L}\sum_{j=1}^{K} \text{ratio}_{i,j} \sqrt{\sum_{n=1}^{T} \left(\frac{2|(h_{i,j}(q_n) - h_{i,j}(c_n))|}{\omega} + 1\right)^2 r^2} \tag{5}$$

We can see that the two bounds cannot represent the range of DTW exactly since they are statistically calculated according to the estimated Euclidean distance. However, they provide a practical approximate pruning solution for DTW-based similarity search.

Pruning: To identify similar time series based on DTW, the best first search principle is adopt to compute the results in an incremental fashion [13]. Specifically, series in the candidate set are ranked based on the pseudo lower bound of DTW. In the range query, we compute the DTW distance between candidate series and Q, if the value is less than θ, we insert this series into the result set. The process terminates when the DTW distance of consecutive t series with respect to Q become greater than θ.

4.4.2. Pruning technique 2: intersection-Pru

To accurately rank the candidate series based on similarity with respect to the query time series Q, we formulate an improved pruning method by taking both the pseudo lower bound and the upper bound of DTW into consideration. The proposed method is based on the assumption that the estimated DTW distance is an uniform distribution between the distance lower bound and upper bound.

As depicted in Fig. 5, the values of x and y represent the approximate range of $DTW(Q, Q)$ and $DTW(Q, C)$, respectively. Notice that, the approximate range of $DTW(Q, Q)$ is not equal to 0 because we use hash values to approximate the Euclidean distance and then calculate the pseudo bounds of DTW. When two vectors collide under a hash function $h()$, their Euclidean distance is considered to be less than r. For two identical time series, even with the same hash value at each timepoint, we can only ensure the bounds of the distance between them.

Two cases are considered in the analysis: (i) when $LB(Q, C) \le UB(Q, Q)$, let the value of z denote the overlap between these two ranges as shown in Fig. 5(a). (ii) when $LB(Q, C) > UB(Q, Q)$, use the value of m refers to the distance between the two ranges as shown in Fig. 5(b). If C and Q are similar, the overlap z should be great or the distance m should be small. As a result, the similarity probability between series Q and C is defined as:

$$\text{Pro\_sim(Q,C)} = \frac{UB(Q,Q) - LB(Q,C)}{UB(Q,C) - LB(Q,Q)} \tag{6}$$

Using Eq. (6), we can calculate the probability of similarity between candidate series and Q, then sort them in descending order. Similarly, the best first search principle is used to find similar time series based on DTW calculation.

To demonstrate that Eq. (6) is an effective measure for pruning, we need to prove that it can reasonably evaluate the similarity between two time series. For time series Q and C, we first consider four cases:

- **Case 1:** There is an overlap between $DTW(Q, Q)$ and $DTW(Q, C)$ (i.e., $LB(Q, C) \le UB(Q, Q)$), and the size of overlap is z;
- **Case 2:** $LB(Q, C) \le UB(Q, Q)$, the size z' of the overlap is smaller than z;
- **Case 3:** $LB(Q, C) > UB(Q, Q)$, and the difference between $LB(Q, C)$ and $UB(Q, Q)$ is m;
- **Case 4:** $LB(Q, C) > UB(Q, Q)$, the size m' of the difference is greater than m.

Use P_1, P_2, P_3 and P_4 to represent the similarity between Q and C for the four cases, in that order. According to our description and observation, the similarity should be: $P_1 > P_2 > P_3 > P_4$. In the following, we will prove the consistancy in similarity when using the defined measure.

Proof. According to Eq. (6), we obtain $P_1 = \frac{z}{x+y-z}$, $P_2 = \frac{z'}{x+y-z'}$, $P_3 = \frac{-m}{x+y+m}$ and $P_4 = \frac{-m'}{x+y+m'}$. As defined in the four cases, we know that $z > z' > 0$ and $m' > m > 0$. Since $P_1 - P_2 = \frac{(z-z')(x+y)}{(x+y-z)(x+y-z')} > 0$, we have $P_1 > P_2$. Since $P_3 - P_4 = \frac{(m'-m)(x+y)}{(x+y+m)(x+y+m')} > 0$, we have $P_3 > P_4$. Due to that $P_2 > 0$ and $P_3 < 0$, we can prove that $P_1 > P_2 > P_3 > P_4$, which is consistent with the actual situation. □

5. Analysis and parameter determination

In this section, we first describe how to determine the search range θ and the envelope size r. Next, an error analysis is provided based on given θ and r values to help determine LSH parameters.

5.1. Search range parameters

When performing an approximate similarity lookup for an application like the kNN classifier, the number of objects within the search range θ has to be large enough such that it is guaranteed to obtain at least k true positive results. However, a large θ also leads to a greater filtering cost. To achieve a balance, a statistical analysis based on a subset of the dataset is usually applied [48]. We now describe how to apply a statistical analysis to determine the value of θ in this subsection. After exploring the relationship between envelop size r and θ, we provide a method to determine the r value based on a given θ. In addition, this analysis is then extended to explore the relationship between the Euclidean distance and DTW distance in order to formulate a candidate pruning method.

5.1.1. Sample selection

In order to obtain reliable statistical analysis based on a subset of time series, the sample set should reflect the distribution of the time series in the dataset. Therefore, we set a sampling interval R and find a subset D' which covers the entire dataset D. Specifically, we wish to identify a subset $D' \subseteq D$ under two constraints: (i) $\forall X'_i \in D', X'_j \in D'$, the Euclidean distance between X'_i and X'_j is greater than R. (ii) $\forall X'_i \in D', \exists X_j \in D$ such that the Euclidean distance between X'_i and X_j is smaller than R. We can execute the sample selection process in an incremental fashion [8]. For example, we may start with any value of R and then repeatedly increase or decrease the value of R for sample selection using Algorithm 2, until the subset covers about 10%–20% of the dataset.

Algorithm 2 outlines the subset selection process. First, initialize D' to an empty set (line 1). Next, we explain the operations from lines 2 to 13. For each iteration, we consider a time series in D. Specifically, we check whether the Euclidean distance of the time series to each of the existing member of D' is greater than R (lines 4 to 9). If so, the time series is added into D'. Otherwise, the time series is ignored. The process halts when the entire dataset D is considered.

5.1.2. Statistical analysis

In this subsection, we describe how to determine the distance threshold θ and the envelop size r for a query series Q. Assume that the DTW distance of time series pair in D follows a normal distribution. After generating a subset D' using Algorithm 2, we calculate the DTW distance of time series pair in D', then compute the mean $mean\_dtw$ and the standard deviation sd_dtw of those distance. Recall that we want to set a proper distance threshold θ so that the percentage of range query results will be less than β, i.e., $\Pr[Z \le \theta] \le \beta$. The value of Z' can be obtained by using β to check the table of standard normal distribution. Then we have $\frac{\theta - mean\_dtw}{sd\_dtw} = Z'$ and the value of θ can be determined as:

$$\theta = mean\_dtw + Z' \times sd\_dtw. \tag{7}$$

Based on a suitable search range θ, we now consider how to determine the r-range envelop of query series Q. As stated in Section 4.4, we analyze the relationship between the Euclidean distance and DTW distance for the time series in dataset D'. Specifically, for every time series pair (X'_i, X'_j) in subset D', compute the DTW-Euclidean ratio: $\mathrm{DTW}(X'_i, X'_j)/\mathrm{ED}(X'_i, X'_j)$. Next, calculate the mean a and standard deviation SD of the ratios of all pairs (X'_i, X'_j). For a time series C, since the DTW-Euclidean ratio is defined as $\frac{DTW(Q,C)}{ED(Q,C)}$, we have: $\frac{\mathrm{DTW}(Q,C)}{a+SD} \le \mathrm{ED}(Q,C) \le \frac{\mathrm{DTW}(Q,C)}{a-SD}$. If C is confined within the r-range envelop of Q, then $ED(Q,C) \le r\sqrt{T}$ where T denotes the length of a time series in D. Therefore, we have $r \le \frac{\mathrm{DTW}(Q,C)}{(a-SD)\sqrt{T}}$. That is:

$$r = \frac{\theta}{(a - SD)\sqrt{T}}. \tag{8}$$

5.2. LSH Parameters

In order to quickly identify the candidate time series inside the envelop of the query time series, we apply LSH techniques and propose a suitable strategy for processing multivariate time series. Due to the approximate results generated by the LSH-based method, one main challenge is probabilistically guaranteeing the result quality. In the interest of result accuracy, we have to control the number of false negatives, i.e., the time series in the dataset D which are located inside the r-envelop of the query time series Q but are missing from the result set. On the other hand, to reduce the result filtering

cost, we also need to control the number of false positives, i.e., the time series in D which are outside the cr-envelop of Q but are included into the result set.

Since the deployment of the proposed LSH method for multivariate time series is not traditional, existing approaches of error analysis are *not* applicable. In this subsection, we describe an error analysis based on the self-defined candidate generation strategy and provide an appropriate way to determine LSH parameters.

5.2.1. Controlling the number of false positives

Assume that time series O is outside the cr-envelop of query time series Q. The defined candidate generation strategy creates a false positive O if there is a group collision between O and Q. In order to control the number of such false positives, we prefer to have the group collision probability $P_G(Q, O)$ between Q and O to be as small as possible compared to that between Q and a true positive C. That is, we want to maximize the gap between the group collision probability $P_G(Q, C)$ and $P_G(Q, O)$. Therefore, we first formulate the hash collision probability $P_H(Q, C)$ and $P_H(Q, O)$. Then, we formulate the group hash collision probability $P_G(Q, C)$ and $P_G(Q, O)$. Finally, we maximize the gap between $P_G(Q, C)$ and $P_G(Q, O)$ by setting a suitable value of K.

Assume that the LSH family in the proposed method is (r, cr, p_1, p_2)-sensitive, where $p_1 = p(r), p_2 = p(cr)$ and the approximation ratio $c > 1$. Using an LSH hash function $h()$, the d-dimensional vector at each time point of a multivariate time series X is projected onto a hash value, i.e., $h(X) = h(x_1), h(x_2), \ldots, h(x_T)$.

Since C is inside the r-envelop of Q, their Euclidean distance at each timepoint should be smaller than r. Therefore, the probability that $|h(q_i) - h(c_i)| \leq \frac{\omega}{2}$ at each time point i is greater than $p(r)$. Let Num refer to the number of time points which happen a projection collision between two time series. Based on the definition of hash collision in the proposed solution, the hash collision probability $P_H(Q, C)$ between Q and C can be computed as:

$$P_H(Q, C) = \Pr[\text{Num} \geq T_s] = 1 - \sum_{i=0}^{T_s-1} \binom{T_s}{i} p(r)^i \times (1 - p(r))^{T-i}. \tag{9}$$

Since the value of T_s is set to αT, the hash collision probability $P_H(Q, C)$ can be expressed as $\Pr[\text{Num} \geq \alpha T]$. We can simplify the probability $P_H(Q, C)$ using Hoeffding's inequality [14], which is defined as follows. For any $0 < \epsilon < p_1$,

- When $\alpha = p_1 - \epsilon$, $\Pr[Num \leq (p_1 - \epsilon)T] \leq \exp(-2\epsilon^2 T)$;
- When $\alpha = p_1 + \epsilon$, $\Pr[Num \geq (p_1 + \epsilon)T] \leq \exp(-2\epsilon^2 T)$.

Based on this bound, the hash collision probability $P_H(Q, C)$ is given by:

$$\begin{aligned} P_H(Q, C) &= \Pr[Num \geq \alpha T] = 1 - \Pr[Num < \alpha T] \\ &= 1 - \Pr[Num < (p_1 - (p_1 - \alpha))T] \\ &> 1 - \exp(-2(p_1 - \alpha)^2 T). \end{aligned} \tag{10}$$

We set $\alpha = p_2 = p(cr)$, the projection collision threshold T_s is determined by:

$$T_s = \alpha T = p_2 T. \tag{11}$$

Now we have $P_H(Q, C) > 1 - \exp(-2(p_1 - p_2)^2 T)$. Since the expected projection collision number between Q and O is less than $p_2 T$, when setting T_s to $p_2 T$, the hash collision probability $P_H(Q, O)$ would be less than 0.5.

In the proposed method, a group hash collision implies that the hash collision between Q and C happens at all K hash functions of a compound hash function $G()$. Hence, the group hash collision probability $P_G(Q, C)$ between series Q and C can be defined as: $P_G(Q, C) = \Pr[G(Q) = G(C)] = P_H(Q, C)^K$. Similarly, we can compute $P_G(Q, O) = P_H(Q, O)^K$.

Next, we want to set a proper value of K so that the difference between $P_H(Q, C)^K$ and $P_H(Q, O)^K$ can be maximized. Assume that $\eta_1 = P_H(Q, C)$ and $\eta_2 = P_H(Q, O)$. Let $\mu(K) = \eta_1^K - \eta_2^K$, we have $\mu'(K) = \eta_1^K \ln \eta_1 - \eta_2^K \ln \eta_2$. Let $\mu'(K) = 0$, we have: $K^* = \ln \frac{\ln \eta_2}{\ln \eta_1} / \ln \frac{\eta_1}{\eta_2}$. When $0 < K < K^*$, $\mu'(K) > 0$ and when $K > K^*$, $\mu'(K) < 0$. Consequently, when $K = K^*$, $\mu(K) = \eta_1^K - \eta_2^K$ achieves its maximum value. Since K should be a integer, we set:

$$K = \lceil \ln \frac{\ln \eta_2}{\ln \eta_1} / \ln \frac{\eta_1}{\eta_2} \rceil, \tag{12}$$

where $\eta_1 = 1 - \exp(-2(p_1 - p_2)^2 T)$ and $\eta_2 = 0.5$.

5.2.2. Controlling the number of false negatives

According to the candidate generation strategy, a false negative may occur when we fail to identify the time series confined inside the r-range envelop of query time series Q. Specifically, the correct neighbor C could be missing if there is no group hash collision between C and Q across L compound hash functions.

Based on the definition of group hash collision, we have: $P_G(Q, C) = \Pr[G(Q) = G(C)] = P_H(Q, C)^K$. Consequently, the false negative rate FNR that C is not included into the candidate set becomes:

$$FNR = (1 - P_G(Q, C))^L, \tag{13}$$

where $P_G(Q,C) = P_H(Q,C)^K$. To make this probability smaller than δ, the number L of group hash functions should be set as:

$$L \geq \frac{\ln \delta}{\ln(1 - P_H(Q,C)^K)} \tag{14}$$

where δ is an error rate specified by users.

5.2.3. LSH Parameter determination

Based on the error analysis presented in Sections 5.2.1 and 5.2.2, we now describe steps for determining the LSH parameters. Assume a representative subset D' has been obtained from Algorithm , as well as, the search range θ and the envelop size r have been obtained from the statistical analysis explained in Section 5.1.2.

Next, we determine the projection collision probabilities p_1 and p_2. Assuming the default bucket size ω to be $0.75r$, we compute $p_1 = p(r)$ and $p_2 = p(cr)$ using Eq. (1), where c is a user-defined approximation ratio. As for the projection collision percentage α, we set it to p_2. As a result, the projection collision threshold T_s can be determined as αT, where T is the length of the time series. We are now ready to define the number L of compound hash functions and the number K of projections for each compound hash function. Recall that the value of K is given by Eq. (12). Based on the results from Eq. (14), to ensure that the false negative rate FNR does not exceed δ, the value of L should be set as

$$L \geq \frac{\ln \delta}{\ln(1 - (1 - \exp(-2(p_1 - p_2)^2 T))^K)}, \tag{15}$$

6. Similarity search based on euclidean distance

In most application domains, DTW is accepted as the best similarity measure over time series. However, with increasing dataset size, the classification accuracies of NN-DTW and NN-ED tend to converge [36]. Since calculating the Euclidean distance between two time series is much less expensive, ED is also used in some applications. In this section, we demonstrate how to generalize the proposed method to retrieve multivariate time series based on the Euclidean distance. The required changes concern the following four aspects.

(1) Problem Definition: Since time warping is not required when ED is used as the similarity measure, we can only define an r-range envelop for query time series Q to limit the search space. As a result, the candidate neighbors would be the sequences confined inside the envelop of Q.

(2) Strategy of Generating Candidates: For two multivariate time series Q and C, we first calculate their hash values. If $|h(q_i) - h(c_i)| \leq \frac{\omega}{2}$, we say that there is a projection collision between Q and C at timepoint i under hash function $h()$. Except for the definition of projection collision, the remaining parts of generating candidates are the same as described in Section 4.3.2.

(3) Pruning Technique: We use hash values to estimate the bounds of Euclidean distance between two time series, and modify the pruning methods by using Eqs. (2) and (3). For the LB-Pru method, we rank the candidate time series based on the lower bound of Euclidean distance and then use the best first search principle to identify similar sequences based on Euclidean distance calculation. As for the Intersection-Pru method, Eq. (6) is changed to: $\text{Pro\_sim(Q,C)} = \frac{UB\_ED(Q,Q) - LB\_ED(Q,C)}{UB\_ED(Q,C) - LB\_ED(Q,Q)}$. The rest of the operations remain the same as given in Section 4.4.2.

(4) Parameter Determination: When the Euclidean distance is used as the similarity measure, the same procedure can be applied to determine the search range θ. After we calculate the mean *mean_ed* and the standard deviation *sd_ed* of the distance pairs in the sampling set, we can determine the search range θ and the envelop size r. Similarly, we have $\theta = mean\_ed + Z' \times sd\_ed$. If time series C is confined within the r-range envelop of Q , then $r * \sqrt{T} \leq \theta$. Therefore, we have $r \leq \frac{\theta}{\sqrt{T}}$. Next, the LSH parameters can be determined as described in Section 5.2.3.

7. Use cases: machine learning workloads

In this section, we describe how a fast similarity search method can benefit various types of machine learning applications. In particular, we consider two machine learning methods: *probabilistic k nearest neighbor (PkNN) classifier*, and *hierarchical clustering*.

7.1. Probabilistic k*-nearest neighbor classification*

The probabilistic k-nearest neighbor (PkNN) classifier [15] is an effective method for tackling classification problems involving a large number of classes. PkNN is an instance-based learning method which returns the probabilistic distribution over the candidate classes through a comparison with instances stored in memory. The main advantages of this approach are stated as follows: (i) little or no training is required; (ii) the model can represent a complicated target function; (iii) there is no information loss through generalization.

In this investigation, we assess the proposed method when it is used for retrieving multivariate time series in the PkNN classification algorithm. Specifically, for a given time series, we first find the top-k nearest neighbors in a labelled dataset D as the input. Then, a probability mass function (PMF) over a set of possible classes is outputted from the PkNN classifier.

In addition, we verify the claim regarding the benefit of converting a univariate time series dataset into a multivariate one in terms of accuracy improvement [24]. Specifically, the LSTM deep neural network [11,32] is applied to convert a univariate electric-device signal dataset into multivariate representations. After feature learning, the P*k*NN classifier is used to provide a classification result for multivariate time series. An experimental study has also been conducted to assess the performance improvement obtained from the higher dimensional representations. The details of experimental studies are provided in Section 8.3.1.

7.2. Hierarchical clustering

We now study the algorithm performance when the proposed approach is used in a clustering problem. We choose the agglomerative hierarchical clustering algorithm [26] since the method iteratively merges clusters by executing a large number of nearest neighbor queries. The algorithm starts by regarding each single object (i.e., time series) in the dataset as a cluster and the cluster center by itself. In each iteration, the distance between cluster centers are calculated and the clusters corresponding to the nearest pair of cluster centers are combined. The center of the new cluster is then calculated and the process continues until the desired number of clusters have been obtained.

As can be seen, the proposed method can be used to speed up the nearest cluster center identification process. Specifically, while searching the nearest cluster center pair, we first retrieve the top-k nearest cluster center pairs based on the number of projection collisions over $L \times K \times T$ LSH projections. The nearest pair based on DTW calculation is then returned.

8. Performance evaluation

In this section, we perform similarity search and evaluate the performance of the proposed solution against three competitors. The first is the *bruteforce* sequential scan method which performs an exact distance computation between the query and every sequence in the dataset. The second is the UCR suite [36]. The third is the SSH method [30]. Note that the SSH method uses the hash index to generate a preliminary candidate set and then use the UCR suite to achieve further pruning. Hence, the method is called *SSH+UCR* for clarity.

Since our work seeks to contribute to candidate generation and pruning, we evaluate these contributions in the following fashion. First, the proposed LSH component is adopted to generate the candidate set without applying any pruning method. We call this variant *LSH-no-pruning*. Second, we separately apply the pruning methods *LB-Pru* and *Intersection-Pru* to the entire dataset without using the LSH component. These variants are called *LB-Pru* and *Intersection-Pru*, respectively. Third, we separately apply *LB-Pru* and *Intersection-Pru* to the candidate set produced by LSH. We call the methods *LSH-LB-Pru* and *LSH-Intersection-Pru*, respectively.

Finally, we evaluate the proposed method based on both DTW and the Euclidean distance. Experiments were conducted on an `Intel(R) Core(TM)i7-4790 CPU @3.6GHz` dual-processor machine with 8GB main memory. All algorithms and experimental studies were implemented in C++, while the query processing was memory-based.

8.1. Experimental setup

8.1.1. Datasets

Two real world datasets from the UC Irvine KDD repository were used in the experiments.

- The **EEG dataset**[2] was originally collected to study the relationship between EEG patterns and alcoholism. The dataset includes 2,560,000 time points where each time point corresponds to a 64-dimensional vector.
- The **Gas sensor dataset**[3] contains the recordings of 16 chemical sensors exposed to two dynamic gas mixtures at varying concentrations. The data file contains 4,000,000 records (each record contains 16 values) acquired continuously for 12 h.

During data processing, z-normalization is applied for time series. To help readers repeat the experiments, we share the exact data[4]. In order to enlarge the number of time series, we used a sliding window and performed subsequence query processing [33]. The default moving step λ was set to 10. For the EEG dataset, the subsequence length was set to 256 and the total number of subsequences was approximately 2.56×10^5. For the Gas dataset, the subsequence length was set to 200 and the total number of subsequences was approximately 2.0×10^5.

8.1.2. Evaluation metrics

Assessments on both efficiency and accuracy are included in the experimental studies:

- Execution time (Exec. Time): The total time taken to identify the k-nearest time series of the query series.
- Distance computation cost (Dist. Comp. Cost): The number of distance computations. Note that for the bruteforce method, the distance computation cost is always equal to the number of time series in the dataset.
- Bound computation cost (Bound. Comp. Cost): The number of times the estimated distance bound was calculated.
- kNN Recall: The number of correct top-k subsequences returned by the algorithm divided by k.

[2] https://archive.ics.uci.edu/ml/datasets/EEG+Database
[3] http://archive.ics.uci.edu/ml/datasets/Gas+sensor+array+under+dynamic+gas+mixtures
[4] https://drive.google.com/drive/folders/1vz__effBaIOPxJ6XK6fRVNhQQKX7txiZ?usp=sharing

Table 2
Default parameters for each dataset (using DTW).

Dataset	K	L	ω	θ	r	T_s
EEG	3	4	5.8	124	7.75	56
Gas	3	3	0.44	8.32	0.59	44

Table 3
Default parameters for each dataset (using ED).

Dataset	K	L	ω	θ	r	T_s
EEG	3	4	5.4	117	7.3	55
Gas	3	3	0.4	7.54	0.53	43

Table 4
Overview of the experimental results for all datasets (Top-50 query using DTW): d refers to the number of dimensions at each time point, M refers to the number of subsequence in the dataset, and T refers to the length of query series.

Algorithm	EEG Dataset ($d = 64, M = 255975, T = 256$)				Gas Dataset ($d = 16, M = 199980, T = 200$)			
	DTW. Comp. Cost	Bound Comp. Cost	Exec. Time (s)	kNN Recall	DTW. Comp. Cost	Bound Comp. Cost	Exec. Time (s)	kNN Recall
Bruteforce	255,975	/	3,394.8	1	199,980	/	447.9	1
UCR suite	82,591	/	1,197.4	1	5021	/	12.46	1
SSH+UCR	48,332	/	768.6	0.8	3518	/	8.13	0.98
LSH-no-pruning	29,614	/	397.5	0.98	3026	/	7.59	0.98
LB-Pru	61,196	255,975	913.4	0.96	4,322	199,980	10.38	0.98
Intersection-Pru	58,647	255,975	864.7	0.96	4219	199,980	10.13	0.98
LSH-LB-Pru	22,713	29,614	313.8	0.92	2931	3026	7.34	0.98
LSH-Intersection-Pru	20,495	29,614	287.5	0.92	2,784	3026	7.21	0.98

Table 5
Overview of the experimental results for all datasets (Top-50 query using ED): d refers to the number of dimensions at each time point, M refers to the number of subsequence in the dataset, and T refers to the length of query series.

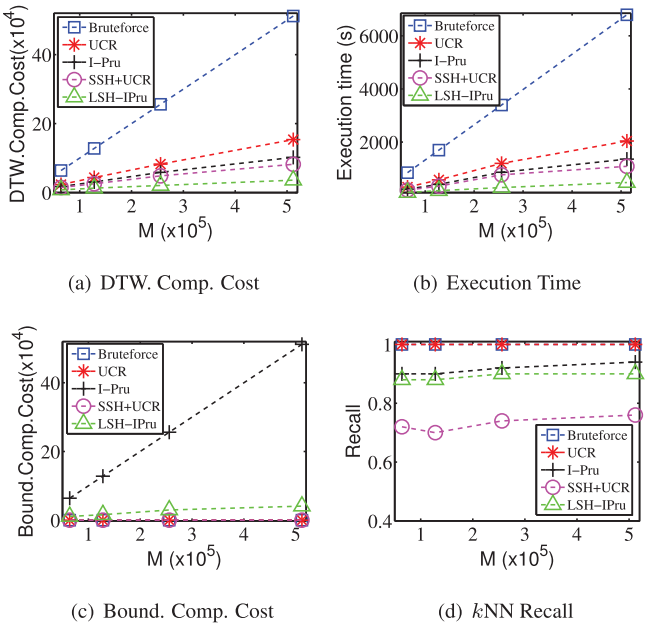
Algorithm	EEG Dataset ($d = 64, M = 255975, T = 256$)				Gas Dataset ($d = 16, M = 199980, T = 200$)			
	ED. Comp. Cost	Bound Comp. Cost	Exec. Time (s)	kNN Recall	ED. Comp. Cost	Bound Comp. Cost	Exec. Time (s)	kNN Recall
Bruteforce	255,975	/	17.52	1	199,980	/	2.72	1
UCR suite	89,373	/	11.64	1	4,523	/	2.36	1
SSH+UCR	42,386	/	7.41	0.8	3092	/	1.61	0.98
LSH-no-pruning	35,025	/	1.42	0.98	2617	/	1.28	0.98
LB-Pru	46,330	255,975	10.55	0.96	3,958	199,980	2.54	0.98
Intersection-Pru	45,821	255,975	10.35	0.96	3897	199,980	2.51	0.98
LSH-LB-Pru	34,872	35,025	2.91	0.92	2535	2617	1.88	0.98
LSH-Intersection-Pru	34,629	35,025	3.42	0.92	2502	2617	2.04	0.98

8.1.3. Parameter setting

For each dataset, the false negative rate δ of r-range similarity search was set to 0.05. For the range query parameter determination, we generated a subset D' using Algorithm 2, and performed the statistical analysis described in Section 5.1.2. During the DTW-based similarity search, the Sakoe-Chiba band was used with a width of 5% of query length, which is a commonly used constraint [20,37]. Assuming that we want to identify the top-50 nearest neighbors, we set the percentage β of range query results to be smaller than 5%, the distance threshold θ and the size of r-range envelop were determined using Eqs. (7) and (8) respectively. In the ED-based similarity search, the values of θ and r were determined as described in Section 6.

While determining the LSH parameters, we set the approximation ratio c to 1.3 and the bucket width ω to $0.75r$. Next, the following parameters were determined according to the analysis given in Section 5: (i) collision probabilities p_1 and p_2 Eq. (1); (ii) projection collision threshold T_s Eq. (11); (iii) the size K of each compound hash function Eq. (12); (iv) the number L of compound hash functions Eq. (14). Tables 2 and 3 provide the default parameters when DTW and ED are used in the experiments.

To evaluate the algorithm performance as the number M of subsequences increases, we varied the moving step λ during subsequence query processing. In addition, we also compared the three methods by varying the query length T, i.e., the number of time points in the query series.



(a) DTW. Comp. Cost (b) Execution Time

(c) Bound. Comp. Cost (d) kNN Recall

Fig. 6. Effect of the size M of dataset.

For the SSH method, we followed their way to determine the parameters. For the EEG dataset, we chose a window size $W = 25$, a shift size $\gamma = 1$ and $n = 10$; for the Gas dataset, we chose $W = 30$, $\gamma = 1$ and $n = 15$. 20 hash functions were used in each of the two datasets.

8.2. Experimental results: k*NN query*

We randomly sampled 100 time series as the queries and evaluated their average performance. An overview of subsequence search based on DTW and ED are provided in Tables 4 and 5 respectively. We can see that the proposed approaches outperformed the three competitors in terms of the distance calculation cost. In particular, the average percentage of DTW computations in each dataset is less than 10%, while the average percentage of ED computations in each dataset is smaller than 12%. The execution time also conforms with the number of distance calculations. As for the measure of accuracy, our methods achieved an average kNN recall of more than 0.92.

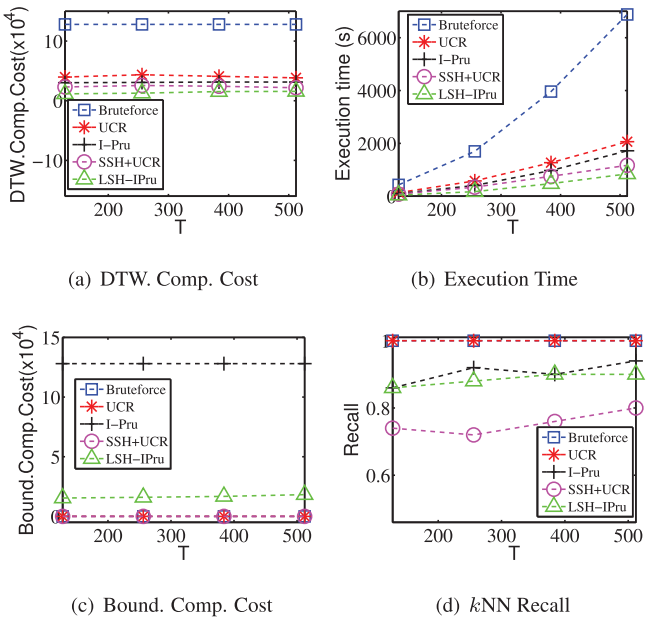
From Table 4, we can also see that for the Gas dataset ($d = 16$), the pruning ratio of the UCR suite is 97.48%. But for the EEG dataset ($d = 64$), the pruning ratio is only about 67.73%. When dealing with multivariate time series, the UCR suite provides a reduced pruning ratio. This is because, the difference between the lower bound and the actual distance becomes greater, resulting in a decrease at pruning power. Although the method which combines SSH and UCR suite improves the pruning ability, the kNN recall is not good enough. The proposed solutions show a good efficiency improvement with a little accuracy sacrifice. If we require a higher accuracy level, we can adjust LSH parameters accordingly.

As for ED-based similarity search, we find that two pruning techniques provide little help (see Table 5). This is because, ED-based similarity search is not compute-intensive, i.e., the cost of calculating the lower bounds outweighs its benefit. Therefore, when applying the proposed method based on ED, it is better to only use the LSH component to provide a pruning benefit.

Let us now compare *LB-Pru* and *Intersection-Pru* with *LSH-LB-Pru* and *LSH-Intersection-Pru*. As shown in Table 4, we can see that the LSH component provides an initial pruning benefit. In particular, for the EEG dataset, the number of bound computations is reduced from 255,975 to 29,614. This results in a speedup factor of 2 in the case of *LSH-Intersection-Pru*. Table 4 also shows that *Intersection-Pru* outperforms *LB-Pru* . This is because Intersection-Pru is considered more aggressive than LB-Pru in terms of ruling out candidate time series. Because of this, we omit the results from *LB-Pru* and *LSH-LB-Pru* when DTW is used as the measure.

8.2.1. Effect of the number M *of subsequences*

We also study how those methods scale as the number M of subsequences in the dataset increases. Specifically, we varied the value of moving step λ from 5 to 40 using the EEG dataset and fixed other parameters to the default values. We kept the brutefoce method as the baseline, while the UCR suite and the SSH method were used as the competitors in later experiments. For brevity, we used *I-Pru* denote our *Intersection-Pru* method, and used *LSH-IPru* represent the *LSH-Intersection-Pru* method. Fig. 6 shows that as the value of M increases, the number of DTW computations and execution



(a) DTW. Comp. Cost (b) Execution Time

(c) Bound. Comp. Cost (d) kNN Recall

Fig. 7. Effect of the length T of time series.

time also increase. This is because, a larger M value results in a larger number of candidates. As shown in the Figs. 6(a) and 6(b), our methods have significantly outperformed the two competitors in terms of the number of DTW computations and execution time. Fig. 6(d) shows that M has little effect on the recall rate since the false negative rate does not depend on this parameter as discussed in Section 5.

8.2.2. Effect of the query length T

We varied the query length T from 128 to 512 time points while using the EEG dataset. As for LSH parameters, we used the default values of ω, K and L, then calculated the collision threshold T_s using the analysis in Section 5. The rest of the parameters were set to default values. As can be seen, as T increases, the execution time of the bruteforce method increases rapidly since the cost of exact DTW calculation increases. Fig. 7 shows that the proposed method has consistently outperformed the competitors in terms of DTW computational cost and execution time. Fig. 7(c) shows that T has a little effect on the number of bound calculation since it is determined by the number of candidate series. From Fig. 7(d), our kNN recall rate is greater than 0.88, which provides a better result accuracy than that of SSH method.

8.3. Performance study of machine learning workloads

To verify that the proposed method can improve the efficiency of mining algorithms which are based on the top-k query, we conducted performance evaluation when the proposed method was applied in two machine learning workloads: PkNN classification and hierarchical clustering.

Recall that we consider multivariate time series from two different sources: (i) an array of sensors, and (ii) LSTM network outputs. Experiments were conducted over both types of datasets. For the first type, we reused the EEG dataset earlier described. For the second type we used the following two univariate time series obtained from the UCR Time Series Classification Archive[5]. The **ElectricDevices dataset** includes 16,637 time series with 96 timepoints and 7 classes. The **StarlightCurves dataset** includes 9236 time series with 1024 time points and 3 classes. For conciseness, we call the two datasets **ElecD** and **StarC** respectively in the experimental studies.

8.3.1. Probabilistic k*-nearest neighbor classification*

In this experimental study, we used DTW as the similarity measure and focused on the best performer *LSH-Intersection-Pru*. Each measurement was the average computed from 100 queries and the number k of nearest time series is 20. We applied the LSTM network [24,32] to convert each univariate time series in the ElecD dataset into 16-dimensional time series and convert the series in StarC dataset into 10-dimensional representations[4]. Then, we performed PkNN classification for the natural multivariate time series and the LSTM generated time series. Table 6 provides a performance comparison for PkNN classification using the dataset EEG and dataset ElecD.

[5] http://www.cs.ucr.edu/~eamonn/time_series_data/

Table 6
Experimental results of P*k*NN classification for dataset EEG and dataset ElecD.

Dataset	P*k*NN Classification	DTW.Comp.Cost	Time (s)	*k*NN
EEG	Bruteforce	255,975	3396.4	1
	UCR suite	82,591	1198.8	1
	SSH+UCR	48,332	770.2	0.8
	Our Method	20,495	289.3	0.92
ElecD	Bruteforce	16,637	11.2	1
	UCR suite	13,249	9.03	1
	SSH+UCR	12,733	8.67	0.77
	Our Method	10,849	7.33	0.93

Table 7
Classification accuracy for univariate time series (k=20).

Dimension	SVM		P*k*NN	
	ElecD	StarC	ElecD	StarC
Univariate	0.62	0.81	0.74	0.87
Multivariate	0.75	0.85	0.83	0.91

Table 8
Experimental results of the hierarchical clustering algorithm for dataset EEG and dataset ElecD.

Dataset	Clustering	DTW.Comp.Cost	Time ($\times 10^3 s$)
EEG	Bruteforce	7,998,000	104.94
	UCR suite	1,664,257	27.84
	SSH+UCR	1,214,907	20.41
	Our Method	60,233	1.29
ElecD	Bruteforce	12,497,500	8.26
	UCR	7,768,537	5.82
	SSH+UCR	6,059,451	4.65
	Our Method	43,108	0.14

As can be seen, our approach outperforms the three competitors at reducing the number of DTW computations. The execution time also conforms with the DTW computation cost. To evaluate the result accuracy of the proposed method, we regarded the classification accuracy of the bruteforce method as 1 and computed the relative accuracy between the proposed approach and the Bruteforce method. Compared with the three competitors, we achieve a relative classification accuracy greater than 97%. That means, the processing of P*k*NN classification can be significantly accelerated with a little accuracy sacrifice.

In the following, we will demonstrate that the classification based on deep neural network structure learning can achieve a better accuracy compared with directly working on the univariate time series. As for classification approaches, the SVM and P*k*NN methods were applied to the original univariate time series and the multivariate representations. The performance comparison is provided in Table 7. We can see that, the multivariate representations yield significantly higher accuracies for both datasets and both classifiers.

8.3.2. Experimental results: hierarchical clustering

As stated in Section 7.2, the proposed LSH-based method can be used to speed up the process of nearest cluster pair identification. In particular, we can find the top-20 nearest cluster pairs based on the number of projection collisions. The nearest pair can be identified by performing exact DTW calculation on these candidate pairs.

We also compared the LSH-based method to the three competitors using the EEG and ElecD datasets. We used the hierarchical clustering algorithm to (i) cluster 4000 objects in EEG dataset to 2 classes, and (ii) cluster 5000 objects in ElecD dataset to 7 classes. The results are reported in Table 8.

Since the LSH-based method has identified 20 candidate pairs, there were at most 20 DTW calculations to be performed during each iteration associated with cluster combination. Therefore, we could significantly reduce the number of DTW computations, as well as the execution time. As shown in Table 8, compared to the brute-force method, we have saved about 98% of time cost for the EEG dataset and ElectD dataset. In terms of accuracy, the proposed method has maintained an accuracy very close to that of the bruteforce method.

Compared with the algorithm performance in the P*k*NN classification, we have achieved a much greater efficiency improvement through hierarchical clustering processing. This is due to the self-join nature of the hierarchical clustering workload. Specifically, the problem of identifying the nearest pair has a quadratic search space with respect to the number of

time series. Therefore, the proposed solution can effectively improve the performance of the machine learning algorithms, especially those that involve joining of two datasets, e.g., finding the closest pair between two large datasets and building a *k*NN graph.

9. Conclusion and future work

We have proposed an efficient LSH-based similarity search method for multivariate time series. To further improve the efficiency, we have designed two pruning techniques based on the approximate bounds of DTW. In addition, an error analysis has been provided to help determine suitable LSH parameters. Experimental results have demonstrated significant improvements compared to the three competitors. As for the future work, we plan to make an improvement by finding a way to apply LSH functions for the whole time series rather than the vector at each timepoint. We also want to extend our method to different similarity measures in variate applications.

Acknowledgment

This research was partially supported by a City University of Hong Kong research grant (CityU Project No. 7200387). We also gratefully acknowledge the Thailand Research Fund (Thailand Research Fund) for the support under the Grand No. MRG6180266.

References

[1] M.T. Bahadori, D.C. Kale, Y. Fan, Y. Liu, Functional subspace clustering with application to time series, in: International Conference on Machine Learning, 2015, pp. 228–237. http://jmlr.org/proceedings/papers/v37/bahadori15.html.
[2] Z. Bankó, J. Abonyi, Correlation based dynamic time warping of multivariate time series, Expert Syst. Appl. 39 (17) (2012) 12814–12823, doi:10.1016/j.eswa.2012.05.012.
[3] Y. Chen, B. Hu, E.J. Keogh, G. Batista, DTW-D: time series semi-supervised learning from a single example, in: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2013, pp. 383–391, doi:10.1145/2487575.2487633.
[4] M. Datar, N. Immorlica, P. Indyk, V.S. Mirrokni, Locality-sensitive hashing scheme based on p-stable distributions, in: Symposium on Computational Geometry, 2004, pp. 253–262, doi:10.1145/997817.997857.
[5] W. Deng, G. Wang, J. Xu, Piecewise two-dimensional normal cloud representation for time-series data mining, Inf. Sci. (Ny) 374 (2016) 32–50, doi:10.1016/j.ins.2016.09.027.
[6] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, E.J. Keogh, Querying and mining of time series data: experimental comparison of representations and distance measures, Proc. VLDB Endowment 1 (2) (2008) 1542–1552.
[7] Y. Ding, E. Keogh, Query suggestion to allow intuitive interactive search in multidimensional time series, in: Proceedings of the 29th International Conference on Scientific and Statistical Database Management, 2017, doi:10.1145/3085504.3085522.
[8] M. Drosou, E. Pitoura, Disc diversity: result diversification based on dissimilarity and coverage, Proc. VLDB Endowment 6 (1) (2012) 13–24. http://www.vldb.org/pvldb/vol6/p13-drosou.pdf.
[9] A.W.-C. Fu, E. Keogh, L.Y. Lau, C.A. Ratanamahatana, R.C.-W. Wong, Scaling and time warping in time series querying, The VLDB Journal-The International Journal on Very Large Data Bases 17 (2008) 899–921, doi:10.1007/s00778-006-0040-z.
[10] J. Gan, J. Feng, Q. Fang, W. Ng, Locality-sensitive hashing scheme based on dynamic collision counting, in: Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data, 2012, pp. 541–552, doi:10.1145/2213836.2213898.
[11] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, J. Schmidhuber, A novel connectionist system for unconstrained handwriting recognition, IEEE Trans. Pattern Anal. Mach. Intell. 31 (5) (2009) 855–868, doi:10.1109/TPAMI.2008.137.
[12] A. Guttman, R-trees: A dynamic index structure for spatial searching, in: Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data, 1984, pp. 47–57, doi:10.1145/602259.602266.
[13] G.R. Hjaltason, H. Samet, Distance browsing in spatial databases, ACM Trans. Database Syst. (TODS) 24 (2) (1999) 265–318, doi:10.1145/320248.320255.
[14] W. Hoeffding, Probability inequalities for sums of bounded random variables, J. Am. Stat. Assoc. 58 (301) (1963) 13–30, doi:10.2307/2282952.
[15] C. Holmes, N. Adams, A probabilistic nearest neighbour method for statistical pattern recognition, J. R. Stat. Soc. 64 (2) (2002) 295–306, doi:10.1111/1467-9868.00338.
[16] Q. Huang, J. Feng, Y. Zhang, Q. Fang, W. Ng, Query-aware locality-sensitive hashing for approximate nearest neighbor search, Proc. VLDB Endowment 9 (1) (2015) 1–12, doi:10.14778/2850469.2850470.
[17] P. Indyk, R. Motwani, Approximate nearest neighbors: Towards removing the curse of dimensionality, in: Proceedings of the thirtieth annual ACM symposium on Theory of computing, 1998, pp. 604–613, doi:10.1145/276698.276876.
[18] F. Itakura, Minimum prediction residual principle applied to speech recognition, IEEE Trans. Acoust. Speech Signal Process. 23 (1) (1975) 67–72, doi:10.1109/TASSP.1975.1162641.
[19] D.C. Kale, D. Gong, Z. Che, Y. Liu, G.G. Medioni, R.C. Wetzel, P. Ross, An examination of multivariate time series hashing with applications to health care, in: 2014 IEEE International Conference on Data Mining, 2014, pp. 260–269, doi:10.1109/ICDM.2014.153.
[20] E.J. Keogh, C.A. Ratanamahatana, Exact indexing of dynamic time warping, Knowl. Inf. Syst. 7 (3) (2005) 358–386, doi:10.14778/1454159.1454226.
[21] S. Kim, B. Jeong, Performance bottleneck of subsequence matching in time-series databases: observation, solution, and performance evaluation, Inf. Sci. (Ny) 177 (22) (2007) 4841–4858, doi:10.1016/j.ins.2007.06.032.
[22] S. Kim, S. Park, W.W. Chu, An index-based approach for similarity search supporting time warping in large sequence databases, in: Proceedings of the 17th International Conference on Data Engineering, 2001, pp. 607–614, doi:10.1109/ICDE.2001.914875.
[23] M. Krawczak, G. Szkatula, An approach to dimensionality reduction in time series, Inf. Sci. (Ny) 260 (2014) 15–36, doi:10.1016/j.ins.2013.10.037.
[24] M. Längkvist, L. Karlsson, A. Loutfi, A review of unsupervised feature learning and deep learning for time-series modeling, Pattern Recognit. Lett. 42 (2014) 11–24, doi:10.1016/j.patrec.2014.01.008.
[25] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444, doi:10.1038/nature14539.
[26] T.W. Liao, Clustering of time series data - a survey, Pattern Recognit. 38 (11) (2005) 1857–1874, doi:10.1016/j.patcog.2005.01.025.
[27] S. Lim, H. Park, S. Kim, Using multiple indexes for efficient subsequence matching in time-series databases, Inf. Sci. (Ny) 177 (24) (2007) 5691–5706, doi:10.1016/j.ins.2007.07.004.
[28] J. Lin, I.E.J. Keogh, L. Wei, S. Lonardi, Experiencing SAX: a novel symbolic representation of time series, Data Min. Knowl. Discov. 15 (2) (2007) 107–144, doi:10.1007/s10618-007-0064-z.
[29] L. Liu, Y. Peng, S. Wang, M. Liu, Z. Huang, Complex activity recognition using time series pattern dictionary learned from ubiquitous sensors, Inf. Sci. (Ny) 340–341 (2016) 41–57, doi:10.1016/j.ins.2016.01.020.

[30] C. Luo, A. Shrivastava, SSH (Sketch, shingle, & hash) for indexing massive-scale time series, in: NIPS 2016 Time Series Workshop NIPS 2016 Time Series Workshop, 2017, pp. 38–58. http://arxiv.org/abs/1610.07328.
[31] Q. Lv, W. Josephson, Z. Wang, M. Charikar, K. Li, Multi-probe LSH: efficient indexing for high-dimensional similarity search, in: Proceedings of the 33rd international conference on Very large data bases, 2007, pp. 950–961.
[32] T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, S. Khudanpur, Recurrent neural network based language model, in: Eleventh Annual Conference of the International Speech Communication Association, 2, 2010, p. 3. http://www.isca-speech.org/archive/interspeech_2010/i10_1045.html.
[33] Y. Peng, R.C. Wong, L. Ye, P.S. Yu, Attribute-based subsequence matching and mining, in: IEEE 28th International Conference on Data Engineering, 2012, pp. 989–1000, doi:10.1109/ICDE.2012.81.
[34] A. Petropoulos, S.P. Chatzis, S. Xanthopoulos, A hidden markov model with dependence jumps for predictive modeling of multidimensional time-series, Information Science 412 (2017) 50–66, doi:10.1016/j.ins.2017.05.038.
[35] S. Pravilovic, M. Bilancia, A. Appice, D. Malerba, Using multiple time series analysis for geosensor data forecasting, Inf. Sci. (Ny) 380 (2017) 31–52, doi:10.1016/j.ins.2016.11.001.
[36] T. Rakthanmanon, B.J.L. Campana, A. Mueen, G. Batista, M.B. Westover, Q. Zhu, J. Zakaria, E.J. Keogh, Searching and mining trillions of time series subsequences under dynamic time warping, in: Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining, 2012, pp. 262–270, doi:10.1145/2339530.2339576.
[37] H. Sakoe, S. Chiba, Dynamic programming algorithm optimization for spoken word recognition, IEEE Transactions on Acoustics, Speech, and Signal Processing 26 (1) (1978) 43–49, doi:10.1109/TASSP.1978.1163055.
[38] P. Schäfer, Scalable time series classification, Data Min. Knowl. Discov. 30 (5) (2016) 1273–1298, doi:10.1007/s10618-015-0441-y.
[39] J. Shieh, E.J. Keogh, *i*SAX: indexing and mining terabyte sized time series, in: Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, 2008, pp. 623–631, doi:10.1145/1401890.1401966.
[40] M. Shokoohi-Yekta, J. Wang, E.J. Keogh, On the non-trivial generalization of dynamic time warping to the multi-dimensional case, in: Proceedings of the 2015 SIAM International Conference on Data Mining, 2015, pp. 289–297, doi:10.1137/1.9781611974010.33.
[41] Y. Tao, K. Yi, C. Sheng, P. Kalnis, Efficient and accurate nearest neighbor and closest pair search in high-dimensional space, ACM Trans. Database Syst. (TODS) 35 (3) (2010) 20.
[42] L. Ulanova, N. Begum, E.J. Keogh, Scalable clustering of time series with U-Shapelets, in: Proceedings of the 2015 SIAM International Conference on Data Mining, 2015, pp. 900–908, doi:10.1137/1.9781611974010.101.
[43] M. Vlachos, M. Hadjieleftheriou, D. Gunopulos, E.J. Keogh, Indexing multi-dimensional time-series with support for multiple distance measures, in: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, 2003, pp. 216–225, doi:10.1145/956750.956777.
[44] B. Yi, H.V. Jagadish, C. Faloutsos, Efficient retrieval of similar time sequences under time warping, in: Proceedings of the 14th International Conference on Data Engineering, 1998, pp. 201–208, doi:10.1109/ICDE.1998.655778.
[45] H. Yoon, K. Yang, C. Shahabi, Feature subset selection and feature ranking for multivariate time series, IEEE Transactions on Knowledge and Data Engineering 17 (9) (2005) 1186–1198, doi:10.1109/TKDE.2005.144.
[46] C. Yu, S. Nutanong, H. Li, C. Wang, X. Yuan, A generic method for accelerating lsh-based similarity join processing, IEEE Trans. Knowl. Data Eng. 29 (4) (2017) 712–726, doi:10.1109/TKDE.2016.2638838.
[47] Z. Zhang, R. Tavenard, A. Bailly, X. Tang, P. Tang, T. Corpetti, Dynamic time warping under limited warping path length, Inf. Sci. (Ny) 393 (2017) 91–107, doi:10.1016/j.ins.2017.02.018.
[48] Q. Zhu, G.E.A.P.A. Batista, T. Rakthanmanon, E.J. Keogh, A novel approximation to dynamic time warping allows anytime clustering of massive time series datasets, in: Proceedings of the 2012 SIAM international conference on data mining, 2012, pp. 999–1010, doi:10.1137/1.9781611972825.86.