On Efficiently Searching Trajectories and Archival Data for Historical Similarities

Reza Sherkat*
IBM Toronto Lab
rsherkat@ca.ibm.com

Davood Rafiei
University of Alberta
drafiei@cs.ualberta.ca

ABSTRACT

We study the problem of efficiently evaluating similarity queries on histories, where a history is a d-dimensional time series for $d \geq 1$. While there are some solutions for time-series and spatio-temporal trajectories where typically $d \leq 3$, we are not aware of any work that examines the problem for larger values of d. In this paper, we address the problem in its general case and propose a class of summaries for histories with a few interesting properties. First, for commonly used distance functions such as the L_p-norm, LCSS, and DTW, the summaries can be used to efficiently prune some of the histories that cannot be in the answer set of the queries. Second, histories can be indexed based on their summaries, hence the qualifying candidates can be efficiently retrieved. To further reduce the number of unnecessary distance computations for false positives, we propose a finer level approximation of histories, and an algorithm to find an approximation with the least maximum distance estimation error. Experimental results confirm that the combination of our feature extraction approaches and the indexability of our summaries can improve upon existing methods and scales up for larger values of d and database sizes, based on our experiments on real and synthetic datasets of 17-dimensional histories.

1. INTRODUCTION

There are many applications where the history of changes to an object or an entity can be described as a sequence of points with each point giving a snapshot of the object at a time point. Consider the observations made about a patient in a hospital. Changes to body temperature, blood pressure, heart beat rate and blood sugar may be recorded regularly over time, producing a chronological history of the patient's conditions. Similarly in the financial sector, the history of a stock may be tracked using indicators such as daily opening and closing prices, trading volume, etc. Also in meteorology, measurements such as temperature, precipitation, wind speed, pressure, moisture and snowfall are regularly collected (e.g. daily or hourly) for many earth surfaces by weather stations. Finding similarities between histories is often useful for exploratory analysis, clustering, and prediction. For instance, the similarities between the histories of two stocks may explain or predict short-term and long-term trends. Finding patients with similar recovery histories may be useful for treatments or the trial of a new drug. Detecting similarities between the weather conditions of two regions may indicate that crops successfully produced in one region may also be tried in the other region.

The problem to be addressed in this paper is efficiently evaluating similarity queries on histories where a history is a sequence of d-dimensional points; a timestamp may also be assigned to each point. In the special case where $d = 1$ and the snapshots are recorded regularly in time, a history becomes a time series, and there are quite a number of works that address this problem (e.g. [1, 12, 15, 26, 20]). In cases where $d = 2$ or 3, a history, often called a trajectory, may describe the position of a moving object in a plane or space, and there are also works that address this problem (e.g. [22, 32, 19, 31]). Since time series and trajectories are usually long sequences, a common approach is to index features extracted from segments using a variant of R-tree structure. While efficient for time series and trajectories, it is generally difficult to scale up many of these solutions for high dimensional histories mainly because the number of features to be indexed increases with the dimensionality of histories. For instance, a Minimum Bounding hyper-Rectangle (MBR) extracted from a segment of a d-dimensional history is a $2(d + 1)$-dimensional vector. We are not aware of any work that addresses the problem for larger values of d, except two works on indexing video sequences; one work does not consider the order of the frames [29] while the other uses techniques specialized to video sequences [21].

As a measure of similarity between histories, many of the distance functions proposed for time series and 2d- or 3d-trajectories may be generalized to high dimensional histories. Example distance functions include L_p-norm [33] (e.g. city-block distance and Euclidean distance), dynamic time warping [4], longest common subsequence [32], edit distance with real penalty [8] and edit distance on real sequences [9]. The choice of a distance function is subjective and a selection may depend on the data domain. In most cases, an index that is built for a particular distance function is useless if another distance function is used. We ideally want our approach to be independent of the distance function used.

*This work was done when the author was at the University of Alberta.

VLDB '08, August 24-30, 2008, Auckland, New Zealand

In this paper, we propose two concise summaries for efficiently processing similarity queries in a filter-and-refinement paradigm. Our first summary, referred to as HSums for short, represents histories as compact time series that can be indexed, hence allowing fast filtering of distant histories with no false negatives. Our second proposed summary gives a fine approximation of histories as a set of MBRs which are derived by minimizing an upper bound of Distance Approximation Error (uDAE[1] for short). Experimental results show that a combination of our feature extraction and the indexability of our HSums can improve upon existing methods and scales with the size of database. Compared to a traditional volume-based splitting, uDAE-based MBR approximations give a more accurate estimate of the true distances. Moreover, uDAE-based MBR approximations are comparable, in terms of distance preservation, to a more costly approximation scheme [2] but are significantly faster to derive. Since uDAE is computed locally for each history, it is more efficient to maintain MBR approximations when either new observations are recorded for histories in the database or new histories are inserted. Our main contributions are as follows:

- We propose techniques for extracting compact HSums from histories with some interesting properties: (1) for a large class of distance functions including all those mentioned in this section, the same distance function that operates on histories can be computed more efficiently using HSums; (2) the distances between HSums can be used to prune some of the histories that are far from a query; (3) HSums can be indexed, hence the pruning and retrievals can be done efficiently. Our techniques for deriving HSums make use of a kernel function that maps a d-dimensional point to a real number. We identify a class of functions that can be used as kernels; hence a spectrum of summaries can be obtained.

- In a finer level of approximation than HSum, we propose uDAE-based MBR approximations of histories to further prune false positives not pruned based on their HSums. Two important features of a uDAE-based MBR approximation are: (1) the maximum distance approximation error is minimized independent of the queries, and (2) it resolves some of the limitations of previous MBR-based techniques for approximating histories in higher dimensions (i.e. large values of d).

- To improve the tightness of uDAE-based MBR approximations while keeping the same space requirements, we propose two adaptive splitting strategies, namely variable splitting and superimposed encoding. Variable splitting targets cases where there is a large difference, in terms of the degree of changes, between dimensions, and accordingly adjusts the splitting policy. Superimposed encoding targets cases where two or more dimensions are correlated, and it might be possible to reduce the dimensionality of MBRs without increasing the maximum distance estimation error.

- Finally, we conduct extensive experiments to evaluate the efficiency and the scalability of our methods on both real and synthetic data.

[1]uDAE is pronounced Yoda as in the Star Wars movie.

Roadmap - Sec.2 presents a generalization of a few distance functions commonly used for time series to compare histories. Our summarization method is presented in Sec.3.2, followed by uDAE-based MBR approximation of histories in Sec.4. Adaptive splitting policies are discussed in Sec.5. Our algorithm to process similarity queries over collection of histories is presented in Sec.6. Experimental results are reported in Sec.7. Related work is surveyed in Sec.8, followed by conclusions and future directions in Sec.9.

2. PRELIMINARIES: DISTANCE BETWEEN HISTORIES

A *history*, denoted by $\langle \vec{a}_1, \ldots, \vec{a}_n \rangle$, is a chronologically ordered sequence of points where $\vec{a}_1, \ldots, \vec{a}_n \in \mathbb{R}^d$ for some d. Let ϕ_d denote a distance function in $\mathbb{R}^d$; a candidate for ϕ_d is the weighted L_p-norm distance defined as

$$\phi_d(\vec{x}, \vec{y}) = \left(\sum_{i=1}^{d} (w_i \cdot |x_i - y_i|)^p \right)^{\frac{1}{p}} \quad (1)$$

where w_i is a real number, referred to as normalizing coefficient. While $w_i = 1$ for $i = 1, \ldots, d$ gives the standard L_p-norm, the normalizing coefficients can be set to other values, for instance, to make the range of variations of all dimensions (or features) equal and thus numerically comparable, or to emphasize the significance of some dimensions over others. The distance of two histories can be formulated as a combination of the pairwise distances of their points.

Let $A = \langle \vec{a}_1 \ldots \vec{a}_n \rangle$ and $B = \langle \vec{b}_1 \ldots \vec{b}_m \rangle$ be two histories. A simple way to measure the distance between A and B in particular when $n = m$ is to use L_p-norm again, i.e.

$$D_p(A, B) = \left(\sum_{i=1}^{n} \phi_d(\vec{a}_i, \vec{b}_i)^p \right)^{\frac{1}{p}}. \quad (2)$$

Two histories may be considered similar, even if they are of different lengths (e.g. when they are collected at different rates) or are out-of-phase (one is shifted in time). The dynamic time warping [4] between two histories extends the histories by replicating some of their points such that the extended histories are of the same lengths. An aggregation of the pairwise distances between the matching points of the extended histories can be used to measure the distance of the two histories (as defined for time series [34]):

$$D_{dtw}(A, B) = \begin{cases} 0 & \text{if both } A \text{ and } B \text{ are empty} \\ \infty & \text{if one of } A \text{ or } B \text{ is empty} \\ \\ \phi_d(head(A), head(B)) + \\ min \begin{cases} D_{dtw}(A, rest(B)), \\ D_{dtw}(rest(A), B), \\ D_{dtw}(rest(A), rest(B)) \end{cases} \\ \text{otherwise.} \end{cases}$$

where $head(A)$ denotes the first point of history A and $rest(A)$ denotes a history which is derived from A by removing $head(A)$. There is also a variant of DTW where two points can be matched only if they are recorded within a time interval, also called *warping range* [4].

An alternative measure to compare two histories is their Longest Common Subsequence Score (LCSS) [32]. LCSS addresses the problems associated with excessive matching in DTW; each point of a history can either be matched with a similar point of the other history or remain unmatched. The similarity is thus proportional to the number of matched

points. Given ϵ as the threshold for matching points, the LCSS of two histories is defined as

$$S_{lcss}(A,B,\epsilon) = \begin{cases} 0 & \text{if } A \text{ or } B \text{ is empty} \\ 1 + S_{lcss}(rest(A), rest(B), \epsilon) & \\ \quad \text{if } \phi_d(head(A), head(B)) \leq \epsilon & \\ max \begin{cases} S_{lcss}(A, rest(B), \epsilon) \\ S_{lcss}(rest(A), B, \epsilon) \end{cases} & \\ \quad \text{otherwise.} \end{cases}$$

S_{lcss} is a similarity score and not a distance function. A distance function based on S_{lcss} may be defined as

$$D_{lcss}(A,B,\epsilon) = 1 - S_{lcss}(A,B,\epsilon)/min(m,n).$$

Also a constraint may be introduced to limit the range of matches in LCSS, as discussed for DTW.

3. SUMMARIZING HISTORIES

Consider searching a large collection of histories for those that are similar to a given history. We are interested in efficient ways of organizing data such that similarity queries can be evaluated efficiently. Our proposed solution constructs summaries and uses them for efficient searching and filtering. In particular, we propose two summaries, HSums (discussed in this section) and MBR approximations (discussed in Sec. 4). But first, we introduce non-expansive mapping, a concept we use to derive our HSums.

3.1 Non-Expansive Mapping

DEFINITION 1 (NON-EXPANSIVE MAPPING). *A function $f : \mathbb{R}^d \to \mathbb{R}$ is a non-expansive mapping if for all points $\vec{x}$ and $\vec{y}$ in $\mathbb{R}^d$, there exists a constant λ, $0 < \lambda \leq 1$, such that*

$$\phi_1\left(f(\vec{x}), f(\vec{y})\right) \leq \lambda \cdot \phi_d(\vec{x}, \vec{y}).$$

If the above condition is satisfied for $0 < \lambda < 1$, then the mapping is said to be *contractive* and the constant λ shows the maximum size of a contraction. Next we present some candidates for f, ϕ_d and ϕ_1 and a discussion of the rationals for choosing these candidates.

Weighted sum - Given a weight vector $\vec{w}$, the weighted sum of $\vec{x}$ is the scalar $\vec{w}^T\vec{x}$. If we assume each weight w_i gives the importance of a variable x_i, then the weighted sum would give a collective assessment of the variables.

LEMMA 1. *Let ϕ_d be the weighted L_p-norm, defined in Eq. 1, and ϕ_1 be the weighted L_1-norm. The weighted sum performs a non-expansive mapping from $\mathbb{R}^d$ to $\mathbb{R}$.*

PROOF. Appears in the appendix. □

The weighted sum has some nice properties. In the case of patients, it may give the overall condition of a patient. The weighted sum may be used to classify points by a classifier, e.g. Perceptron [27]. When $w_i = 1/d$ for $i = 1, \ldots, d$, the weighted sum becomes the average and measures the central tendency of a point[2]. If the weight vector is set to the first principal component [14] of all points in a set of histories, the maximum variance of the points is preserved.

[2]In this case, the weighted sum can be used to construct an optimal approximation of the point in d-dimensional space, in terms of the sum of squared error.

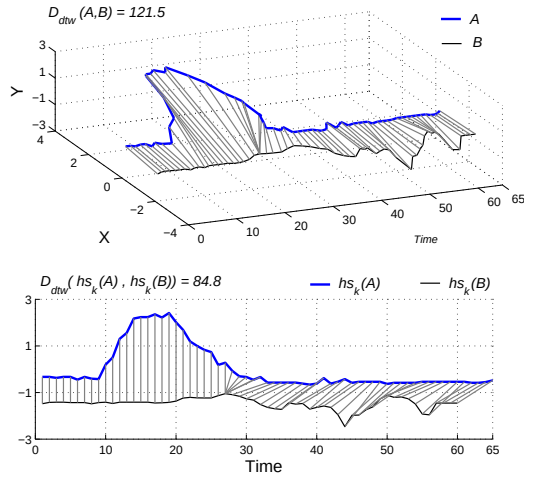


Figure 1: 2-d histories A and B (top) and their HSums as time series (bottom); $\kappa(\vec{x}) = \vec{w}^T\vec{x}$ with $\vec{w} = (0.5, 0.5)^T$.

Metric space embeddings - A large class of metric space embeddings [18] can be used to construct a non-expansive mapping when ϕ_d is a metric distance function. For instance, let $\vec{r}$ be a reference point; a mapping can be defined as $f_1(\vec{x}) = \phi_d(\vec{x}, \vec{r})$. In the case of patients, the reference can be the conditions of a normal healthy person. Given two reference points, a mapping $f_2(\vec{x})$ can be defined as the projection of a point on the line that connects the two reference points [11]. Again in the case of patients, the reference points can be the conditions of two patients: one a normal healthy person and the other a patient in some critical condition. More generally, one can select a set R, a finite and non-empty subset of points in $\mathbb{R}^d$, as reference and define a mapping function with respect to the set R as

$$f_R(\vec{x}) = min_{\vec{y} \in R}\{\phi_d(\vec{x}, \vec{y})\} \quad (3)$$

which is a special case of Lipschitz embedding [5].

LEMMA 2. *If ϕ_1 is set to the L_1-norm: (1) f_1 and f_R perform a non-expansive mapping from $\mathbb{R}^d$ to $\mathbb{R}$ if ϕ_d is set to any metric distance; (2) f_2 performs a non-expansive mapping from $\mathbb{R}^d$ to $\mathbb{R}$ if ϕ_d is set to the Euclidean distance.*

PROOF. Appears in [18]. □

3.2 History Summaries

DEFINITION 2 (HSUM). *The HSum of a history $A = \langle \vec{a}_1, \ldots, \vec{a}_n \rangle$, with respect to a kernel function $\kappa : \mathbb{R}^d \to \mathbb{R}$ is a time series denoted by $hs_\kappa(A)$ and defined as*

$$hs_\kappa(A) = [\kappa(\vec{a}_i)] \quad , \quad i = 1, \ldots, n.$$

Fig.1 illustrates the idea of HSum for 2-dimensional histories. The definition of HSum allows any function from $\mathbb{R}^d$ to $\mathbb{R}$ to be a potential kernel. However, we are interested in kernels that generate summaries that can be used to prune histories, without introducing false negatives. This constrains the distance of two HSums to be a lower-bound of the distance between respective histories.

Table 1: Average tightness

	ASL	Marine	VT1	VT2	Word	Web
D_2	0.54	0.97	0.65	0.79	0.50	0.71
D_{dtw}	0.54	0.96	0.63	0.74	0.45	0.70
D_{lcss}	0.39	0.94	0.97	0.93	0.34	0.61

THEOREM 1. *Let the kernel function κ performs a non-expansive mapping from $\mathbb{R}^d$ to $\mathbb{R}$. The distance between HSums of histories A and B, measured using D_p, D_{dtw}, and D_{lcss} provides a lower-bound for the distance between A and B, respectively measured using D_p, D_{dtw}, and D_{lcss}.*

PROOF. Appears in the appendix. □

The lower-bounding property is good to have as it guarantees the correctness of an index that uses the lower-bound, but it doesn't give the full picture. In particular, it doesn't tell how good is the lowerbound in approximating an actual history. Fig.1 illustrates the lower-bounding property for D_{dtw} where ϕ_d is set to the L_2-norm and ϕ_1 is set to the weighted L_1-norm with a normalizing coefficient of $\sqrt{2}$. Table 1 reports average tightness for 50 randomly chosen histories from real datasets (Sec.7.1). For each distance function, the tightness is the distance between two history summaries divided by the distance between two histories. For D_p and D_{dtw}, we scaled the distance between summaries by $\frac{1}{\lambda}$ to achieve a better lower bound for D_p and D_{dtw}.

To construct an HSum in general, one needs to select a kernel κ and a distance function ϕ_1 in $\mathbb{R}$. The choice of κ and ϕ_1 depends on ϕ_d and the function that aggregates the distances between points to give the distance between two histories. It can be proved by contradiction that a kernel function must perform a non-expansive mapping to guarantee the lower-bounding property. However, the lower-bounding property is not limited to D_p, D_{dtw}, and D_{lcss}.

THEOREM 2. *Let HSums be derived using a kernel which performs a non-expansive mapping. The lower-bounding property is guaranteed to hold if the distance between two histories and their HSums is an aggregation of the distances between their respective points for commonly used aggregation functions including min, max, sum, count and avg.*

It can be shown that the lower-bounding property holds for other previously proposed distance functions for time series when generalized for histories such as ERP [8] and EDR [9][3].

3.3 Pruning Histories using HSums

HSums have two interesting properties for the purpose of pruning. First, because of the lower-bounding property; if the distance between two HSums is not less than a threshold, the distance between their respective histories also cannot be less than a scaled threshold, where the scale is determined by the kernel function used. However, the distance between two HSums is more efficient to compute. For instance, D_p can be computed in $O(dn)$ time for two histories of the same length n, and in $O(n)$ time for their HSums. For two histories of lengths m and n, D_{dtw} and D_{lcss} are computed in $O(mn)$ time for HSums and in $O(dmn)$ time for respective histories. The distance between the HSums of a query history and a data history can be used to avoid computing a more expensive distance between the query and the data history.

Second, since HSum is a time series, each history can be indexed based on its HSum using any indexing technique developed in the domain of time series (and there is a rich collection of such indexes). Though it should be noted that an HSum gives a coarser representation of a history and some patterns may show in the history but not in its HSum. For instance, with the weights set the same for all dimensions, a weighted sum remains unchanged for any permutations of the dimensions. This is a type of distortion that cannot be detected using HSums; it can reduce the pruning effectiveness of HSum and degrade the performance of filtering by increasing the number of false positives (e.g. Word dataset in Table 1). The amount of this distortion directly depends on the kernel function used and the data distribution. Next, we present a finer representation of histories; we will use this finer representation to further prune some of the false positives that cannot be pruned based on their HSums.

4. A FINER APPROXIMATION OF HISTORIES

We consider approximating a history using a set of Minimum Bounding hyper-Rectangles (MBRs) which encloses all points of the history. This representation, also commonly used for organizing spatial and spatio-temporal objects ([24, 12, 22, 16]), provides a concise abstraction for histories. The set of MBRs of a history, in general, preserves trends in individual dimensions of the history with a higher resolution than its HSum. Moreover, for a large class of distance functions, including D_p, D_{dtw}, and D_{lcss}, the distance between two histories can be underestimated efficiently by the distance between their MBR representations (e.g. [22, 20, 31]).

4.1 Notations and problem definition

To approximate a history $A = \langle \vec{a}_1, \ldots, \vec{a}_n \rangle$ as a sequence of k MBRs spread along the time axis, a splitting algorithm must be used to divide A into k consecutive and non-overlapping segments. Let s_i and e_i, respectively, denote the indexes of the first and the last points of segment i. By construction, $s_1 = 1$, $e_k = n$, and $s_{i+1} = e_i + 1$.

Let $a_t[r]$ be the value of coordinate r of vector $\vec{a}_t$. Segment i of the history is approximated by $A_i = (s_i, e_i, \vec{l}_i, \vec{h}_i)$ where

$$l_i[r] \leq a_t[r] \leq h_i[r] \quad s_i \leq t \leq e_i$$

for $1 \leq r \leq d$. MBR A_i is a hyper-rectangle which tightly encloses all points of the history falling in segment i. There are $\binom{n-2}{k-1}$ possible ways to decompose A into k consecutive and non-overlapping segments, and as a result there are that many representations of the history. Among all possible representations, we are interested in the one which can be used more effectively for the purpose of filtering. Let A^k denote an arbitrary representation of history A as a set of k MBRs. Let $D(\cdot)$ denote the distance between two histories. Given a query history Q, A^k is optimized for pruning if it minimizes the distance approximation error defined as

$$DAE(Q, A, A^k) = D(Q, A) - D(Q, A^k) \qquad (4)$$

where $D(Q, A^k)$ is the minimum distance between Q and any history that can be approximated using A^k, therefore $D(Q, A^k)$ is a lower-bound of the true distance of Q and A.

[3]Both ERP and EDR make use of a *gap* symbol which is set to a real number for time series. For histories, the gap symbol would be a vector.

When the query history is provided, one can adjust splitting points to minimize Eq.4 for that query. However, often the splitting is performed in a pre-processing step; therefore the algorithm to derive A^k must be independent of the query. An approach, which has been used extensively for indexing spatial and spatio-temporal objects (e.g. [16, 31, 22]), would consider total volume of the MBRs as a criterion for an optimal splitting. However, this approach produces MBRs that are optimal for indexing but not for pruning.

A more effective solution for this problem was proposed in [2], assuming that queries are selected uniformly at random from the set of histories to be indexed. They propose global distance-based segmentation to preserve all pairwise distances. The splitting is both costly to derive and is only optimal for static collections; it is not possible to predict the effectiveness of this approach in a more realistic setting where queries are not selected from the given dataset.

4.2 uDAE - Our optimality criterion

An optimal solution for Eq.4 cannot be reached at indexing time as it requires the knowledge of Q, which is not available at indexing time. Therefore, we propose an upper-bound for distance approximation error (uDAE for short) which can be formulated independent of the query.

DEFINITION 3. *Given a history A and an MBR approximation A^k, an upper bound of the distance approximation error w.r.t. to a distance function $D(\cdot)$ is $uDAE(A, A^k)$ if for every history Q,*

$$D(Q, A) - D(Q, A^k) \leq uDAE(A, A^k).$$

DEFINITION 4. *A near-optimal approximation of history A is defined as*

$$\arg\min_{A^k} uDAE(A, A^k).$$

The term *near-optimal* is used to indicate that the error bound (and not the actual error) is minimized; the actual error cannot be determined without advance knowledge of Q and its points distribution. Ideally, when uDAE is zero, the near-optimal solution will be the same as the optimal solution. We derive uDAE when $D(\cdot)$ is substituted by L_p-norm, D_{dtw}, and D_{lcss} assuming that ϕ_d is a metric distance.

4.2.1 L_p-norm

Consider Eq.4 and let $D(\cdot)$ be defined as in Eq.2. To simplify our presentation, let $p = 1$.

$$DAE(Q, A, A^k) = \sum_{j=1}^{k} \sum_{i=s_j}^{e_j} \phi_d(\vec{q}_i, \vec{a}_i) - MINDIST(\vec{q}_i, A_j) \tag{5}$$

where $MINDIST(\vec{q}_i, A_j)$ denotes the distance of $\vec{q}_i$ to its closest point on or inside MBR A_j, as depicted in Fig.2. There are two cases: when $\vec{q}_i$ is outside MBR A_j as is shown in the left figure, the metric property of ϕ_d implies that

$$\begin{aligned} \phi_d(\vec{q}_i, a_i) - MINDIST(\vec{q}_i, A_j) &\leq \phi_d(\vec{a}_i, \vec{u}) \\ &\leq MAXDIST(\vec{a}_i, A_j) \end{aligned}$$

where $MAXDIST(\vec{a}_i, A_j)$ denotes the distance of $\vec{a}_i$ to its farthest point in A_j. When $\vec{q}_i$ is on or inside MBR A_j as in Fig.2 right, we have

$$\begin{aligned} \phi_d(\vec{q}_i, a_i) - MINDIST(\vec{q}_i, A_j) &= \phi_d(\vec{q}_i, \vec{a}_i) \\ &\leq MAXDIST(\vec{a}_i, A_j). \end{aligned}$$

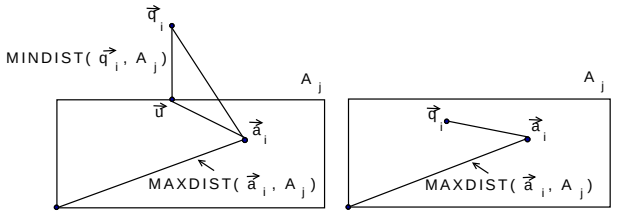


Figure 2: $MINDIST(\vec{q}_i, A_j)$ **and** $MAXDIST(\vec{a}_j, A_j)$ **for points** $\vec{q}_i$ **and** $\vec{a}_i$ **and MBR** A_j

because $MINDIST(\vec{q}_i, A_j)$ is zero. Replacing $\phi_d(\vec{q}_i, \vec{a}_i) - MINDIST(\vec{q}_i, A_j)$ in Eq.5 with its upper-bound derived above gives an upper-bound of the distance approximation error

$$uDAE(A, A^k) = \sum_{j=1}^{k} \sum_{i=s_j}^{e_j} MAXDIST(\vec{a}_i, A_j). \tag{6}$$

THEOREM 3. *When the distance between histories is defined using L_p-norm,*

$$\arg\min_{A^k} uDAE(A, A^k) = \arg\min_{A_1,\ldots,A_k} \sum_{j=1}^{k} \sum_{i=s_j}^{e_j} MAXDIST(\vec{a}_i, A_j).$$

PROOF. (*sketch*) When $p = 1$, the proof is straightforward from Eq. 6. For $p > 1$, $uDAE(.)$ is a polynomial of order p of $MAXDIST(\vec{a}_i, A_j)$ and $MINDIST(\vec{q}_i, A_j)$; the latter can be bounded to $c.MAXDIST(\vec{a}_i, A_j)$ where c is a constant. Since coefficient in the polynomial is a positive number, a solution that minimizes the sum of $MAXDIST(\vec{a}_i, A_j)$ would also minimize $uDAE(.)$. □

4.2.2 D_{dtw} and D_{lcss}

THEOREM 4. *When the distance between histories is defined using D_{dtw} or D_{lcs},*

$$\arg\min_{A^k} uDAE(A, A^k) = \arg\min_{A_1,\ldots,A_k} \sum_{j=1}^{k} \sum_{i=s_j}^{e_j} MAXDIST(\vec{a}_i, A_j).$$

PROOF. Because of the space limitation, we only show this for D_{dtw}. Let $n(\vec{a}_i, Q) \geq 1$ denote the number of points in history Q which are matched with point $\vec{a}_i$ of history A. An upper-bound for Eq.4 can be formulated as a weighted sum of the errors of individual matches. Since the error in a match which involves point $\vec{a}_i$ is at most $MAXDIST(\vec{a}_i, A_j)$, an upper-bound for distance approximation error can be formulated as

$$\sum_{j=1}^{k} \sum_{i=s_j}^{e_j} n(\vec{a}_i, Q) \cdot MAXDIST(\vec{a}_i, A_j). \tag{7}$$

Bounding $n(\vec{a}_i, Q)$ from above would allow us to derive a bound of the distance approximation error. This is a natural constraint because a full length warping is not often desired and might result into unrealistic matches [25]. In the presence of a warping constraint, $n(\vec{a}_i, Q)$ cannot exceed a warping range ω and

$$uDAE(A, A^k) = \omega \sum_{j=1}^{k} \sum_{i=s_j}^{e_j} MAXDIST(\vec{a}_i, A_j).$$

and the rest follows. □

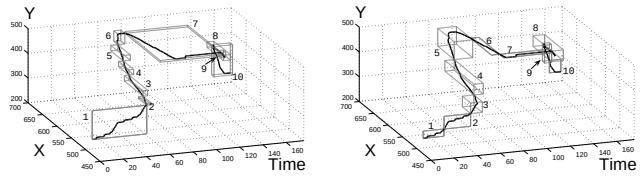


Figure 3: MBRs of a history, one with min. volume (left) and one with min. uDAE (right)

In the rest of the paper, we refer to Eq.6 as uDAE instead of its general definition to make discussions about uDAE concrete.

4.2.3 Computing uDAE

For a history of length n, an optimal splitting in terms of uDAE into k MBRs can be found using a dynamic programming algorithm. Let $A_{s:e}$ denote a subsequence of history A consisting of points $\vec{a}_s \dots \vec{a}_e$, and $A^1_{s:e}$ denote a single MBR which encompasses $A_{s:e}$. Let $U[\ell, k]$ denote the uDAE of an optimal splitting of history $A_{1:\ell}$ using k MBRs. When $k = 1$, there is only one possible splitting of a history which is optimal by default and the corresponding uDAE can be computed directly from Eq.6. For $1 < k \leq n$, $U[n, k]$ can be computed recursively as follows:

$$U[n,k] = Min_{k-1 \leq \ell < n} \left\{ U[\ell, k-1] + uDAE(A_{\ell+1:n}, A^1_{\ell+1:n}) \right\}$$

$U[n, k]$ can be computed in $O(n^2 k)$ time using dynamic programming. The computational complexity for minimizing uDAE is the same as that for minimizing total volume [16]. However, compared to volume, uDAE has some nice properties as discussed next.

4.3 uDAE compared to total MBR volume

Minimizing total volume, in general, does not lead to an optimal representation of a history in terms of approximation error. Fig.3 presents two approximations of the same history, one derived by minimizing total volume and the other derived by minimizing uDAE. If a history shows no change along one or more dimensions within an interval, the total volume for any possible splitting of the history in that interval is the same (i.e. zero). An optimal splitting strategy for the interval with respect to total MBR volume is to assign a single MBR for the whole interval, as happens for MBR 1 in Fig.3(left). This is a serious problem for approximating histories in particular when d is relatively large and points are sparse. Each history often has segments where the points are not distributed along all dimensions, hence the intrinsic (or real) dimensionality of the history within those intervals is less than d. The same intervals are approximated by tighter MBRs when uDAE is minimized in Fig.3(right). This problem is also observed when changes happen in all dimensions of histories but there is a big variance in the degree of the changes; this is shown for MBRs 7 and 10 in Fig.3(left). A representation that minimizes the total volume is expected to give a more accurate description of the changes in dimensions with smaller variance. Unlike volume, uDAE is minimized only when the MBRs are as tight as possible; hence uDAE generally provides a better approximation of histories and it is not affected by intervals which produce trivial splittings when volume is used. A better approximation is expected to improve both the tightness of the lower-bounds and the effectiveness of pruning, as shown in our experiments (Sec.7).

4.4 Splitting policy

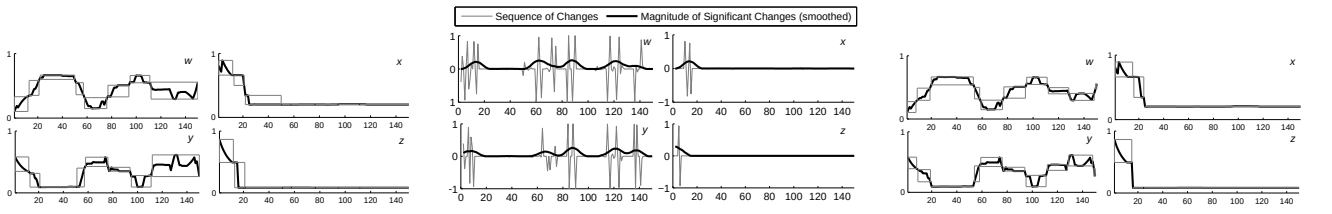
The MBR approximation presented in this section adopts a *fixed splitting* policy where the same splitting intervals are considered for all dimensions of a history. With a fixed splitting, we need to maintain for each segment, the minimum and the maximum values along each dimension, and the ending point of the segment. Because the MBRs of each history are stored sequentially in our scheme, there is no need to keep the starting points. Therefore, an approximation A^k requires $k(2d + 1)$ features to maintain. Naturally, increasing k will result into a better approximation of the history, measured in terms of uDAE. However, given a fixed amount of space for an approximation, a major concern for high-dimensional histories is to make a clever use of the space by finding the best possible approximation. A straightforward approach is to apply lossless data compression techniques [36] to reduce the space requirement of A^k, hence increasing k indirectly. However, if we could find a *better* splitting, without increasing k, we would also benefit from compression. A fixed splitting policy would be a desirable property to keep if MBRs are to be stored in a spatial index structure. For histories of higher dimensionality, MBRs generally cannot be efficiently indexed due to the large number of features [28], and there is no justification for a fixed splitting. A fixed splitting may be ineffective, for instance, when changes are observed only on a subset of the dimensions or the absolute values of the changes on several dimensions are not correlated.

5. ADAPTIVE SPLITTING OF HISTORIES

Given a history and the available space M, the search for an optimal adaptive splitting is a constrained optimization problem where the goal is to find an MBR representation which minimizes uDAE and does not exceed M in space usage. An approach closely related to our adaptive splitting is Adaptive Piecewise Constant Approximation (APCA) [7] where each time series is approximated by a set of constant value segments. The split points could be adjusted to derive an optimal approximation of time series using a dynamic programming algorithm [10]. We consider a more general case for high-dimensional histories where the number and the position of split points can change for different groups of dimensions of a history. Our heuristics consider possible correlation between dimensions and the similarity of change trends to improve upon an optimal fixed splitting scheme in terms of uDAE. Improving uDAE can improve the tightness of the lower-bounds and the effectiveness of pruning.

5.1 Variable Splitting (VS)

When the variances of the changes along all dimensions are not the same, a fixed splitting may over-allocate the split points to dimensions with less or no changes; this is typically for the cost of under-allocating the split points to dimensions that can use more split points. Our first heuristic partitions the set of dimensions and assigns split points to each partition independently; the number of split points can also vary between partitions. Fig.4 illustrates the intuition for a 4-dimensional history where our variable splitting reduces uDAE. The variable splitting assigns 10 MBRs to $\{w, y\}$ and 4 MBRs to $\{x, z\}$ which is a deviation from 8



$P_{fixed} = \{\langle \pi(A, \{w, x, y, z\}), 8\rangle\}$ change trends $P_{vs} = \{\langle \pi(A, \{w, y\}), 10\rangle, \langle \pi(A, \{x, z\}), 4\rangle\}$

Figure 4: Optimal fixed splitting (left) change trends (middle) and optimal variable splitting (right)

MBRs of the fixed splitting; the break down results in allocating 3 MBRs to the last 40 points of $\{w, y\}$ for a better approximation instead of 1 MBR of the fixed splitting. We use the following definition to formalize variable splitting.

DEFINITION 5 (INDUCED HISTORY). *Let $\mathbb{D} = \{1, \ldots, d\}$ denote the set of dimensions of history A, and D_i be a nonempty subset of $\mathbb{D}$. The induced history $\pi(A, D_i)$ is a history derived from A by removing all dimensions in $\mathbb{D} - D_i$.*

Let $\{D_1, \ldots, D_p\}$ be a partitioning of the set $\mathbb{D}$, such that $D_i \cap D_j = \varnothing$ for $i \neq j$ and $\cup_{i=1}^{p} D_i = \mathbb{D}$. A variable splitting of history A is defined as a set $P = \{\langle D_1, k_1\rangle, \ldots, \langle D_p, k_p\rangle\}$ where k_i is a natural number and the pair $\langle D_i, k_i\rangle$, $1 \leq i \leq p$, indicates that the induced history $\pi(A, D_i)$ is approximated using k_i MBRs. A fixed splitting is indeed a variable splitting where $P = \{\langle \mathbb{D}, k\rangle\}$. The split points for each induced history have to be determined independent from other induced histories by a splitting algorithm that minimizes a cost, such as uDAE or volume. A variable splitting P could be used to derive a unique approximation of history A in $O(|P|k_{max})$ time where k_{max} is the maximum number of MBRs dedicated to the induced histories derived by P. The optimal variable splitting policy must minimize $uDAE$ subject to a space constraint[4]. Because the number of features required to store k_i MBRs of the induced history $\pi(A, D_i)$ is $k_i(2|D_i| + 1)$, the space constraint can be formulated as

$$\sum_{\langle D_i, k_i\rangle \in P} k_i(2|D_i| + 1) \leq M. \tag{8}$$

LEMMA 3. *An optimal adaptive splitting is guaranteed not to do worse than a fixed splitting policy.*

PROOF. (*sketch*) Since the search space for an optimal adaptive splitting policy is a superset of the space searched for the fixed splitting, the rest follows. □

An exhaustive search to find an optimal variable splitting is computationally prohibitive, since all possible partitioning of $\mathbb{D}$ into nonempty subsets need to be constructed and for each partitioning, the optimal number of splittings of each partition must be determined. The number of possible partitioning of $\mathbb{D}$ into non-empty subsets is B_d, the bell number[5]. Because performing an exhaustive search becomes inefficient for large databases when $d \geq 10$ ($B_{10} = 115,975$), we exploit a simple heuristic which considers only one partitioning of $\mathbb{D}$. To form this partitioning, we cluster dimensions that are likely to benefit from the same splitting points. We use the similarity of change trends of dimensions as the criterion for clustering (Fig.4 middle).

[4]More detail about space allocation in Sec.7.4.

[5]$B_0 = B_1 = 1$ and for $d \geq 1$, $B_{d+1} = \sum_{i=0}^{d} B_i \binom{d}{i}$

Extracting change trends. Extracting change trends involves three steps. First, for each dimension, the sequence of changes is extracted as a time series. The value of this time series at time t_i is the value of the change for the corresponding dimension from time t_i to t_{i+1}. Second, a change at time t_i is considered significant if the magnitude (i.e. absolute value) of change is α standard deviation greater than the average magnitude of changes in a window of length w centered at t_i; otherwise we set the change to zero. Finally, significant changes are smoothed using a moving average window of length w to derive change trends. We set $w = 7$ (to resemble a week) and set $\alpha = 1.5$ in our experiments.

Finding an optimal assignment. After extracting the change trends for all dimensions, any clustering algorithm such as K-means can be used to partition the dimensions of each history, based on the similarity of change trends, into p subsets. When $p = 1$, all dimensions of the history are in the same group and a variable splitting is equivalent to a fixed splitting. When $p = d$, dimensions are split independently. For a partitioning of $\mathbb{D}$, we want to assign the number (and the position) of splitting points for each induced history. A brute-force approach would consider all possible assignments of k_i, $i = 1, \ldots, p$, which satisfy the space constraint. For each assignment a dynamic programming algorithm must be performed to find the positions of the splits. After the split points for each induced history is determined, uDAE is computed to identify and keep an optimal variable splitting. As finding k_i MBRs of $\pi(A, D_i)$ requires $O(n^2 k_i)$ time, the brute-force approach is not efficient since it requires to find optimal split for every assignment of k_i, $i = 1, \ldots, p$. However, the search can be formulated as a dynamic programming algorithm where the optimal splitting of $\pi(A, D_i)$ into $k_i - 1$ MBRs can be computed directly from the matrix which was computed for finding the optimal splitting of $\pi(A, D_i)$ using k_i points. By induction, only one splitting is required for each induced history $\pi(A, D_i)$, with k_i set to

$$\left\lfloor \frac{M - \sum_{j=1}^{p} (2|D_j| + 1)}{2|D_i| + 1} \right\rfloor \quad , \quad j \neq i \tag{9}$$

which is the maximum number of MBRs that could be allocated to $\pi(A, D_i)$, when only one MBR is assigned to every other induced history $\pi(A, D_j)$, $j \neq i$. A branch-and-bound algorithm could be developed to find an optimal assignment of MBRs for a given partitioning. This, combined with our heuristic used to examine only one partitioning of $\mathbb{D}$, gives a suboptimal variable splitting of a history.

5.2 Superimposed encoding (SE)

When there is a high similarity among the dimensions of an induced history, this similarity could be used to significantly reduce the size of an encoding. Such similarity can

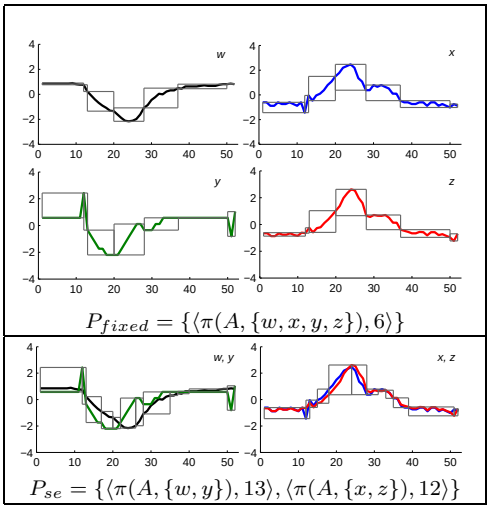


Figure 5: Optimal fixed splitting (top) superimposed encoding (bottom)

Input: A d-dimensional history Q as query
An index constructed on HSums
Output: k most similar histories to Q
Pre-processing:
Apply kernel function κ on Q to extract $hs_\kappa(Q)$.
Obtain $\tilde{Q}$; MBR approximation of Q.
Search:
(1) Find k-NN of $hs_\kappa(Q)$ using the index.
(2) Let $\mathcal{R}_1$ be all records in the result set.
(3) Let r be the largest value of $D(H,Q)$, $H \in \mathcal{R}_1$.
(4) Perform a range search for $hs_\kappa(Q)$ and range r.
(5) Let $\mathcal{R}_2$ be all records in the result set.
(6) Initialize Topk list to the first k records in $\mathcal{R}_2$.
(7) For each history H in $\mathcal{R}_2$
(8) Read $\tilde{H}$; MBR approximation of H
(9) If $D_{lb}(\tilde{H},\tilde{Q}) \geq$ Topk.dist
(10) Prune H
(11) Else
(12) Read H the full history corresponding to $\tilde{H}$.
(13) Compute $D(H,Q)$; update Topk if needed
(14) EndIf

Figure 6: Algorithm for k-NN search

have a low support over the whole dataset, and traditional dimensionality reduction techniques (e.g. PCA) might not consider it significant. However, such a rather local similarity can be used to reduce the space required to encode the MBRs of induced histories by imposing similar dimensions into one representative group. Each segment of a superimposed set of dimensions is associated with a time interval and is approximated by the minimum and the maximum values observed in the interval. Hence a set of superimposed dimensions is encoded as a sequence of *one*-dimensional MBRs. The saving in the number of features could increase the number of splits and potentially reduce uDAE while keeping the space requirement the same. In Fig.5, for instance, dimensions $\{w,y\}$ and similarly $\{x,z\}$ are grouped together and the MBR of each superimposed segment is encoded using a temporal range and a pair of minimum and maximum. As a result, dimensions $\{w,y\}$ and $\{x,z\}$ are respectively split into 13 and 12 segments, using the same amount of space as splitting the history into 7 segments, to reduce uDAE.

The approach discussed for variable splitting can be easily modified to implement our superimposed encoding; since $|D_i|$ dimensions are superimposed into one, $|D_i|$ in Eq.8 must be replaced by one, to derive the space constraint. The setting of maximum value for k_i in Eq.9 should be modified, alternatively, to $\lfloor \frac{M-3(|P|-1)}{3} \rfloor$ and the partitioning of $\mathbb{D}$ must be performed based on the similarities of dimensions instead of their change trends. Note that our superimposed encoding has similarities with the concept of Skyline Bounding Regions (SBR) of Li et al. [23], but the two are different. While SBRs are built on multiple similar time series with no other relationships, superimposed encoding is done on a partition, which includes similar dimensions of one history. Also Li et al. use the area of SBRs to find best approximations while we use uDAE because of its advantages for pruning as discussed earlier.

6. SIMILARITY SEARCH ON HISTORIES

We are now ready to present our algorithm for processing nearest-neighbors queries. Our algorithm uses an index over HSums which are 1-d time series. Leaf nodes of the index contain both HSums and uDAE-MBR approximations of data histories, whereas internal nodes are built based on HSums. Fig.6 gives an example of a multi-step nearest-neighbors search algorithm [28] which performs filtering using the index (line 1-5) and pruning based on uDAE-MBRs (line 9). The algorithm first retrieves k histories with the most similar HSums to the HSum of the query. For each retrieved history, r is computed as an upper-bound of the distance of the query and potential candidates. Given the non-expansive property of the kernel function, a history H is a candidate for original nearest-neighbors query, only if $D(hs_\kappa(H), hs_\kappa(Q)) \leq r$. Therefore, the algorithm performs a range query on the index on HSums to retrieve a superset of the qualifying histories (line 4). Some false positives are pruned by computing a lower-bound[6] of true distance in line 9; true distance is computed in line 13 to prune false positives not pruned by HSum index and uDAE-MBRs.

7. EXPERIMENTAL EVALUATIONS

First we compare our approach to a related and recently proposed splitting algorithm [2] on the basis of efficiency of splitting and quality of generated MBRs. Second, we compare our uDAE-based splitting with the traditional volume based splitting in terms of the tightness of lower bounds. Third, we investigate the effectiveness of our adaptive splitting heuristics in improving the quality of MBRs measured by uDAE. Finally, we study the efficiency of our algorithm in terms of pruning power, running time, and scalability. Experiments are performed on a machine with a single AMD XP2600 CPU, 512MB RAM, running Red Hat Linux.

7.1 Datasets and Settings

Real1 - This dataset was provided to us by the authours of [2] and has 41 real datasets from UCR time series archive[7]. Each dataset contains 50 time series of length 512.

Real2 - Table 2 provides a summary of the datasets in this group, which are 2-4 dimensional histories. These datasets have been used in related work (e.g. [31]).

[6] $D_{lb}(\cdot)$ can be any function that lower-bounds $D(\cdot)$
[7] http://www.cs.ucr.edu/∼eamonn/TSDMA/

Table 2: Summary of Real2 datasets

	ASL	Marine	VT1	VT2	Word
Dimension	3	2	2	2	4
Size	6,756	4	15	23	2,381
Avg. length	58	128	151	543	178

Web - This dataset contains the history of a sample of the Web as a collection of 17-dimensional histories. We used Google Directory[8] to get a sample of highly ranked web pages. Google Directory organizes web sites by their categories in a hierarchical structure. Each node in the structure contains a set of links to other nodes, as well as a list of web pages and a descriptive text for each page. In each node, the web pages are ordered according to their PageRank. We crawled the first five levels of this directory and extracted a set of descriptive terms for each of the 17 categories (e.g. Art, Business, Sports, etc.). The set of descriptive terms for each category included all terms that appeared in the text description of any node that descended from the category within the crawled data. From the crawled data, we collected the URL of 11,328 web sites; these are links to external web sites within the first five levels of Google Directory. We checked the change history of these pages in Internet Archive[9]. For most of the web pages, either the page did not change in the specified period or few versions of it (less than 50) were stored in Internet Archive. To focus our experiments on pages that changed more often, we decided to crawl those with at least 50 different versions in the first six months of 2004 from Internet Archive. This provided us with 1,191 histories of web pages. We crawled all versions of these pages and mapped each version into a point in a 17-dimensional space. The mapping showed the degree of overlap between the content of each version of a page and the descriptive terms of each category. The result after this mapping was a set of 17-dimensional histories that showed the change patterns of 1,191 pages over this interval. The average number of versions for each web site was 81.

Synthetic - In order to investigate the scalability of our approach, we constructed a large but realistic synthetic dataset. We used the Web dataset as a seed set and generated multiple copies of the histories in the seed by applying a combination of four operations: *permutation, time shifting, compression* and *insertion of new points*, thus increasing the number of histories in the dataset. Permutation randomly changes the order of dimensions of a history. Time shifting, introduces a random shift τ in time. Compression, selects a random segment and replaces it with a single point which is the average value of that segment. New points are inserted at index t. The inserted point was set to the average of the points at index $[t-w, t)$. A combination of compression and insertion can increase or decrease the length of histories. We selected τ from $[1, 5]$, t from 1 to the history length, and w from $[1, 10]$, all uniformly at random.

Settings - In the pre-processing step, all histories were normalized so the mean for each dimension was zero. We set ϕ_d to Euclidean distance, ϕ_1 to weighted L_1-norm with a normalizing coefficient of $\sqrt{2}$ according to Lemma 1. The kernel function κ was set to *average* for D_{euc}, D_{dtw}, and D_{lcss}. The number of splits for each history was set to 10% of the length of the history. For tightness, we computed the ratio of estimated distance over true distance; a tightness closer to one indicates a more distance preserving splitting. To estimate distance between two histories, we derived an MBR approximation of the two histories and used the same lower-bounds used in [31] for Euclidean distance, dynamic time warping, and longest common subsequence. To measure Euclidean distance between histories of different lengths, we chopped the longer history; other possible alternative would be up-sampling (down-sampling) the shorter (longer) history[10]. For D_{dtw}, the warping range was set to 5% of query length and for D_{lcss}, ϵ was set to 25% of the query standard deviation, as both suggested in [31].

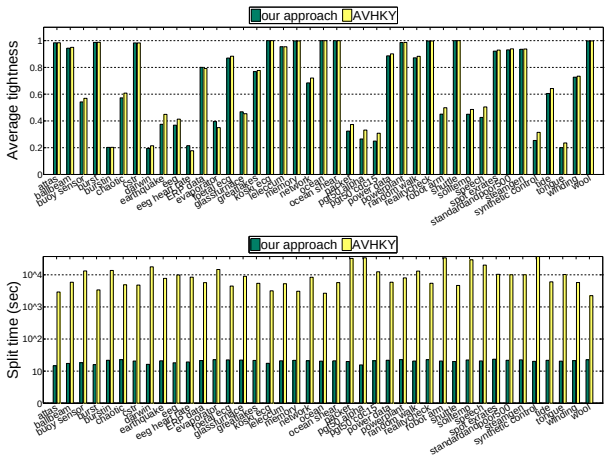


Figure 7: Average tightness and split time

7.2 uDAE vs. Distance-based Splitting

We investigated how a splitting algorithm which minimizes uDAE could be compared with a related algorithm [2] which performs a global distance-based segmentation of histories, aiming at maximizing pairwise distance preservation. Since the exact solution of the proposed approach could not be applied to even a database of moderate size, due to its intractable complexity, Anagnostopoulos et al. propose a greedy solution that closely preserves the sum of pairwise distances; we refer to this approach as AVHKY and compare it with a dynamic programming algorithm which minimizes total uDAE for histories (our approach). To be in-line with the experiments performed in [2], we used the same datasets and distance function i.e. Euclidean distance.

We partitioned each dataset and considered a quarter as query data and the rest as data to be indexed. Similar to AVHKY, we set the number of MBRs for the collection to be split equal to 10% of the total sum of the length of the histories in the collection. We wished to split each query using the same approach used to split the corresponding dataset. Since AVHKY could not be used to independently split a query, we split each query by minimizing volume, to avoid being biased to any of the two methods to be compared. For each query and data history pair, we measured tightness and report average tightness and splitting time in Fig.7.

Our approach performed very close to AVHKY in terms of average tightness but was significantly better in terms of running time. This is because uDAE is a local measure,

[8]http://directory.google.com/

[9]http://www.archive.org/

[10]Either approach can produce error in k-NN classification.

Table 3: Avg. tightness of lower bounds

		Asl	Marine	VT1	VT2	Word
D_{euc}	volume	0.42	0.73	0.87	0.84	0.50
	uDAE	0.49	0.75	0.90	0.89	0.56
D_{dtw}	volume	0.50	0.64	0.76	0.75	0.44
	uDAE	0.54	0.65	0.77	0.77	0.46
D_{lcss}	volume	0.53	0.69	0.82	0.70	0.48
	uDAE	0.58	0.71	0.82	0.75	0.54

computed for each history independently, unlike AVHKY where the distances of the segments of all pairs of histories in the dataset are computed to make a decision on merging consecutive segments of each history. It should be mentioned that all-pairs distance computation of AVHKY cannot be performed for updates whereas ours can.

7.3 uDAE vs. Total Volume

Tightness of lower-bounds We investigated how a uDAE-based splitting scheme could be compared with a volume-based approach [16], w.r.t. the tightness of the lower-bounds proposed in [31]. Other than the volume-based approach in [31], the only similar comparison which we are aware of, is reported for time series and Euclidean distance [2]; we consider high dimensional histories and more flexible distance functions here. Table 3 reports average tightness computed for fifty histories selected uniformly at random from each dataset when uDAE(volume) is minimized. Even though uDAE minimizes an upper bound (instead of the exact value) of distance approximation error, our observation is that for most datasets and distance functions, using uDAE makes estimated distances closer to true distances, thus it is expected to be more effective for pruning.

Right number of MBRs Finding the right number of MBRs is an important issue in MBR approximation of histories. A heuristic is proposed in [17] which finds a number beyond which increasing the number of MBRs is not beneficial as it does not decrease total volume. The same heuristic can be used to find a proper value for the number of MBRs when uDAE is used because uDAE, like volume, is a monotonically decreasing function of the number of MBRs. In this experiment, we investigated how history approximation improved with k, the number of MBRs, for Web and Word (Real2) datasets. Let $v_1(u_1)$ be the total volume(uDAE) of the histories in the dataset when a single MBR is assigned to each history. We increased k from 20 to 450 and for each k, derived optimal approximation of the histories using k MBRs, where optimality was measured using total volume and total uDAE. Fig.8 shows the total volume and uDAE, normalized by $v_1(u_1)$ for 20 randomly picked histories of Word and Web datasets, varying the number of MBRs.

As expected, both volume and uDAE decreased with k. However, volume decreased with a faster rate. In particular, for Web dataset, total volume was zero when as few as 30 MBRs were assigned to *all* histories (on average 1.5 MBR to each history). This is because each dimension in this dataset represents the similarity of a page to one of 17 categories; for most web sites, the pages in the history are similar to only a subset of categories and show zero (or very small) similarity to other categories. For instance, the history of `http://www.mealtime.org/` shows high similarity to {Home, Health, Science, Shopping} and a relatively small similarity to {Arts, Business, Games, Computers}. When

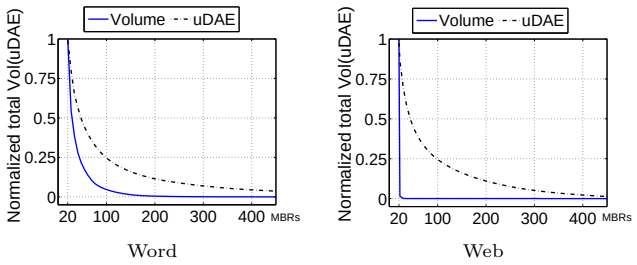


Figure 8: Normalized vol.(uDAE) vs. No. of MBRs

the similarity of a web page to a set of categories remains relatively low over time, a single MBR would have a total volume close to zero and is therefore optimal. However, uDAE is not affected by this phenomenon and increasing the numbers of MBRs beyond 30 resulted into more distance preserving approximation of histories. We expect to observe a similar pattern for high-dimensional histories if only a sub-space of dimensions changes.

7.4 Effectiveness of Adaptive Splitting

We measured average uDAE reduction for high-dimensional histories when our heuristics were employed. For each history, we derived an optimal fixed splitting using k MBRs; k was set to 10% of the length of the history. Such an optimal splitting requires $M = k(2dn_m + n_t)$ bytes to store where n_m and n_t are, respectively, the space required to store the extents of each dimension (min. and max.) and the temporal length of each MBR. Because for each 17-dimensional history, finding optimal adaptive splitting required considering $B_{17} \simeq 8.2 * 10^{10}$ partitioning of the set of dimensions, we implemented our heuristics and considered only n_p partitions, where n_p ranged from 2 to 4. We measured uDAE reduction compared to optimal uDAE-based fixed splitting, given M bytes of space. To store k_i MBRs of induced history $\pi(D_i, A)$, we allocated $k_i(2|D_i|n_m + n_t) + n_d$ bytes in VS and $k_i(2n_m + n_t) + n_d$ bytes in SE; here n_d is the space required to store (D_i, k_i). We set $n_m = 4$, $n_t = 1$, and $n_d = 2$ bytes.

Table 4 reports average uDAE reduction for Web dataset for VS and SE. Although we derived suboptimal solutions for adaptive splitting, our heuristics were still effective and improved upon optimal fixed splitting. This is significant as it demonstrates the possibility to improve upon the *optimal* fixed splitting scheme. We observed more reduction for SE than the marginal uDAE reduction in VS, which is to some extent natural for this dataset because it has several correlated dimensions and SE benefits from supper-imposing similar dimensions. Because similar dimensions are represented using the same extrema in SE; SE increases the number of splits for each induced history.

To find how much the reduction in uDAE is significant, we measured its effect on the tightness of lower bounds for the Web dataset in Table 5. As we expected, not only uDAE-based fixed splitting improved the lower bounds over the volume based scheme, taking advantage of the redundancy in similar dimensions and the sparseness present in Web dataset for dimensions that do not change, our adaptive splitting could improve up to 25%(33%) upon uDAE-(volume-) based fixed splitting. For other datasets, we didn't observe much improvement when adaptive splitting is used.

Table 4: Avg. uDAE reduction (Web)

VS,2	VS,3	VS,4	SE,2	SE,3	SE,4	SE,5
0.30%	0.82%	1.33%	3.8%	5.0%	6.0%	7.2%

Table 5: Avg. tightness of lower bounds(Web)

	volume	uDAE	SE,2	SE,3	SE,4	SE,5
D_{euc}	0.45	0.53	0.70	0.76	0.78	0.78
D_{dtw}	0.44	0.48	0.53	0.57	0.57	0.61
D_{lcss}	0.53	0.56	0.64	0.69	0.69	0.75

7.5 Performance Evaluation

We compared our algorithm with the framework proposed in [31], henceforth VHGK. Two indices were constructed, one for organizing the MBRs of d-dimensional histories (for VHGK) and one for the MBRs of HSums. For each history, we set s_i, the number of splits, to 10% of its length for the first index and to $\frac{2(d+1)}{4}s_i$ for the second index, to make the index sizes equal for a fair comparison. To derive MBRs in VHGK, we optimized total volume using DPSplit [16]. Fig.9 reports results averaged over fifty 10-NN queries. Marine, VT1, and VT2 were relatively small and a linear scan could outperform an index. Since our approach uses two pruning steps, once by HSum and once by uDAE-MBRs, it shows a better overall pruning and performance compared to VHGK, even though it uses the index twice (line 1 and 4 of our algorithm in Fig.6) to answer each query.

We measured the scalability of our approach for 17-d histories. To the best of our knowledge, no experiment has been reported on histories with more than 4 dimensions, considering Euclidean distance, DTW, and LCSS. We used synthetic dataset with 1k, 2k, 4k, and 8k histories constructed from our Web dataset. Fig.10 reports the fraction of dataset pruned and relative query processing time over linear scan. In each case, the average is reported for fifty 10-NN queries. Our approach shows a strong pruning power for a wide range of database sizes, which makes it superior to linear scan and scalable with database size.

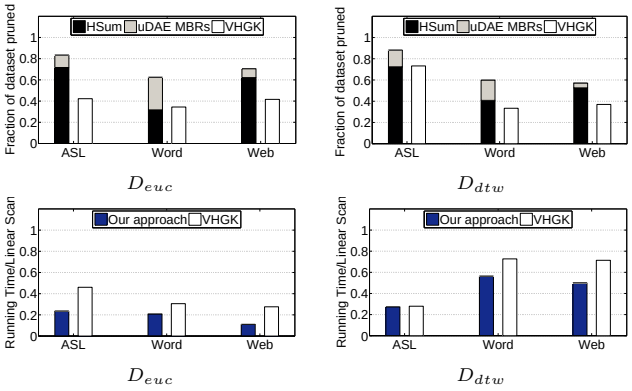


Figure 9: Pruning and relative query processing time averaged for fifty 10-NN queries

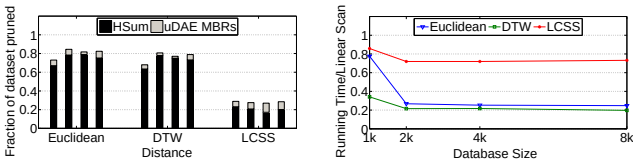


Figure 10: Avg. pruning and relative query time; bars for each distance (left to right) are for datasets with 1k, 2k, 4k and, 8k histories.

8. RELATED WORK

Time-series Several approaches have been proposed to index time series for Euclidean distance [1, 7, 26]), L_p-norm [33], edit distance with real penalty [8], dynamic time warping [20, 35], and longest common subsequence [15, 31]. As a result of Theorem 2, our approach can take advantage of index structures proposed for time series to index history summaries for efficient filtering.

Multi-dimensional time-series For Euclidean distance, Lee et al [22] partition data sequences into subsequences, each subsequence represented by an MBRs. The MBRs are organized in an R*-tree index structure, which is probed by query MBRs to prune irrelevant data sequences. Kahveci et al. [19] propose an index structure for shift- and scale- invariant comparison of time sequences. Cai et al. [6] use the coefficients of chebyshef polynomials as features to approximate data sequences. Frentzos et al. [13] consider the temporal aspect of trajectories (i.e. histories of moving objects) and define, as distance measure, the integral of the function of time of Euclidean distance between the two trajectories. A pair-wise comparison of points cannot be used when histories are of different lengths or a non-uniform time-shifting is present. Vlachos et al. [31] propose an index structure that supports multiple distance measures. They organize MBRs extracted from data sequences in an R-tree and based on the intersection of the query MBRs with those in the index, to prune irrelevant data sequences. The dimensionality of the MBRs increases with d, facing multi-dimensional index structures with performance degradation [28].

Video-sequences Shen et al. [29] measure the similarity by the fraction of similar frames that are common between two sequences, without considering the order of the frames in the sequence. Lee et al. [21] propose a graph based data structure to capture spatial and temporal features of video data. They use a model-based expectation maximization approach to group similar object graphs. This method is specific to video data and we are not sure if it can be applied to more general histories. Also the assumption that data follows some basic model (Gaussian in [21]) can be violated, which would result in missing qualifying histories. Other domain specific approaches have been proposed. For instance, given the motion sequence of a person, Assa et al. [3] extract a sequence of joint aspects and compute dissimilarity between motion sequences using affinity matrices which are extracted from joint data.

9. CONCLUSIONS AND FUTURE WORK

We have addressed the problem of efficiently evaluating similarity queries on histories, proposed techniques for finding summaries of histories at different levels of detail, and investigated the use of these summaries for indexing and pruning. We have developed uDAE as a measure of the tightness of history approximations and empirically evaluated its effectiveness on real and synthetic datasets of higher

dimensionality for fixed and adaptive splitting policies. Our work leads to a few interesting directions. First, many data mining tasks such as clustering and pattern recognition require a large number of distance computations. We believe that HSum and uDAE-based approximations can improve the efficiency of such data mining tasks without much affecting their accuracies; further work may examine the relationships between these tasks and our approximations in more details. Second, there are applications where a history can be represented more effectively as a sequence of time-stamped market-basket data [30]. Generalizing HSum and uDAE to such datasets is future work. Finally, as the dimensionality of points increases, it might be more effective to consider a subspace of points to measure the distance. Extending our work to support history summaries extracted from a subspace of points is a direction for future research.

Acknowledgments

This work is supported by Natural Sciences and Engineering Research Council of Canada. Thanks to Marios Hadjieleftheriou for providing data and code of [2] and to Eamonn Keogh for providing Real2 data.

10. REFERENCES

[1] R. Agrawal, K.-I. Lin, H. S. Sawhney, and K. Shim. Fast similarity search in the presence of noise, scaling, and translation in time-series databases. In *VLDB*, 1995.

[2] A. Anagnostopoulos, M. Vlachos, M. Hadjieleftheriou, E. J. Keogh, and P. S. Yu. Global distance-based segmentation of trajectories. In *SIGKDD*, 2006.

[3] J. Assa, Y. Caspi, and D. Cohen-Or. Action synopsis: Pose selection and illustration. In *ACM SIGGRAPH*, 2005.

[4] D. J. Berndt and J. Clifford. Using dynamic time warping to find patterns in time series. In *KDD Workshop*, 1994.

[5] J. Bourgain. On lipschitz embedding of finite metric spaces in hilbert space. *Israel J. Math.*, 52(1):46–52, 1985.

[6] Y. Cai and R. T. Ng. Indexing spatio-temporal trajectories with chebyshev polynomials. In *SIGMOD*, 2004.

[7] K. Chakrabarti, E. J. Keogh, S. Mehrotra, and M. J. Pazzani. Locally adaptive dimensionality reduction for indexing large time series databases. *ACM TODS*, 27(2):188–228, 2002.

[8] L. Chen and R. T. Ng. On the marriage of lp-norms and edit distance. In *VLDB*, 2004.

[9] L. Chen, M. T. Özsu, and V. Oria. Robust and fast similarity search for moving object trajectories. In *SIGMOD*, 2005.

[10] C. Faloutsos, H. Jagadish, A. Mendelzon, and T. Milo. A signature technique for similarity-based queries. In *Compression and Complexity of Sequences*, 1997.

[11] C. Faloutsos and K.-I. Lin. FastMap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In *SIGMOD*, 1995.

[12] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series databases. In *SIGMOD*, 1994.

[13] E. Frentzos, K. Gratsias, and Y. Theodoridis. Index-based most similar trajectory search. In *ICDE*, 2007.

[14] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Elsevier, 1990.

[15] D. Q. Goldin and P. C. Kanellakis. On similarity queries for time-series data: Constraint specification and implementation. In *Principles and Practice of Constraint Programming*, 1995.

[16] M. Hadjieleftheriou, G. Kollios, V. J. Tsotras, and D. Gunopulos. Efficient indexing of spatiotemporal objects. In *EDBT*, 2002.

[17] M. Hadjieleftheriou, G. Kollios, V. J. Tsotras, and D. Gunopulos. Indexing spatiotemporal archives. *VLDB Journal*, 15(2):143–164, 2006.

[18] G. R. Hjaltason and H. Samet. Properties of embedding methods for similarity searching in metric spaces. *IEEE Trans. on PAMI*, 25(5):530–549, 2003.

[19] T. Kahveci, A. K. Singh, and A. Gürel. Similarity searching for multi-attribute sequences. In *SSDBM*, 2002.

[20] E. J. Keogh. Exact indexing of dynamic time warping. In *VLDB*, 2002.

[21] J. Lee, J.-H. Oh, and S. Hwang. Strg-index: Spatio-temporal region graph indexing for large video databases. In *SIGMOD*, 2005.

[22] S.-L. Lee, S.-J. Chun, D.-H. Kim, J.-H. Lee, and C.-W. Chung. Similarity search for multidimensional data sequences. In *ICDE*, 2000.

[23] Q. Li, I. F. V. López, and B. Moon. Skyline index for time series data. *IEEE TKDE*, 16(6):669–684, 2004.

[24] B.-U. Pagel, H.-W. Six, H. Toben, and P. Widmayer. Towards an analysis of range query performance in spatial data structures. In *PODS*, 1993.

[25] L. Rabiner and B.-H. Juang. *Fundamentals of Speech Recognition*. Prentice Hall, 1993.

[26] D. Rafiei and A. Mendelzon. Similarity-based queries for time series data. In *SIGMOD*, 1997.

[27] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

[28] T. Seidl and H.-P. Kriegel. Optimal multi-step k-nearest neighbor search. In *SIGMOD*, 1998.

[29] H. T. Shen, B. C. Ooi, and X. Zhou. Towards effective indexing for very large video sequence database. In *SIGMOD*, 2005.

[30] R. Sherkat and D. Rafiei. Efficiently evaluating order preserving similarity queries over historical market-basket data. In *ICDE*, 2006.

[31] M. Vlachos, M. Hadjieleftheriou, D. Gunopulos, and E. Keogh. Indexing multi-dimensional time-series with support for multiple distance measures. In *SIGKDD*, 2003.

[32] M. Vlachos, G. Kollios, and D. Gunopulos. Discovering similar multidimensional trajectories. In *ICDE*, 2002.

[33] B.-K. Yi and C. Faloutsos. Fast time sequence indexing for arbitrary lp norms. In *VLDB*, 2000.

[34] B.-K. Yi, H. V. Jagadish, and C. Faloutsos. Efficient retrieval of similar time sequences under time warping. In *ICDE*, 1998.

[35] Y. Zhu and D. Shasha. Warping indexes with envelope transforms for query by humming. In *SIGMOD*, 2003.

[36] J. Ziv and A. Lempel. A universal algorithm for sequential data compression. *IEEE Trans. on Information Theory*, 23(3):337–343, 1977.

APPENDIX

Proof of Lemma 1. To show that weighted sum is a non-expansive mapping, we use one of the properties of convex functions. Let g be a convex function defined on real numbers, i.e. for real numbers $c_1, \ldots, c_d$ such that $\sum_{i=1}^{d} c_i = 1$, the following inequality holds

$$g(\sum_{i=1}^{d} c_i a_i) \leq \sum_{i=1}^{d} c_i g(a_i) \tag{10}$$

for any set of real numbers $a_1, \ldots, a_d$. Let $\vec{x}$ and $\vec{y}$ be arbitrary d-dimensional points. Replacing $g(x)$ with $|x|^p$ (which is a convex function), c_i with $1/d$ for all i, and a_i with $w_i(x_i - y_i)$ in Eq.10 yields

$$\left|\sum_{i=1}^{d} \frac{1}{d} w_i(x_i - y_i)\right|^p \leq \sum_{i=1}^{d} \frac{1}{d} \left|w_i(x_i - y_i)\right|^p.$$

The above inequality can be written as

$$\left| d^{\frac{1-p}{p}} (\vec{w}^T \vec{x}) - d^{\frac{1-p}{p}} (\vec{w}^T \vec{y}) \right| \leq \left(\sum_{i=1}^{d} \left(w_i \left| x_i - y_i \right| \right)^p \right)^{\frac{1}{p}}$$

The left hand side is the weighted L_1-norm of $\vec{w}^T \vec{x}$ and $\vec{w}^T \vec{y}$, with a normalizing coefficient of $d^{\frac{1-p}{p}}$. □

Proof of Theorem 1. Let $A = \langle \vec{a}_1, \ldots, \vec{a}_n \rangle$ and $B = \langle \vec{b}_1, \ldots, \vec{b}_m \rangle$ be two histories. For D_p, when $n = m$, the distance between two summaries is derived as

$$D_p(hs_\kappa(A), hs_\kappa(B)) = \left(\sum_{i=1}^{n} \phi_1 \left(\kappa(\vec{a}_i), \kappa(\vec{b}_i) \right)^p \right)^{\frac{1}{p}}$$

which is obtained by replacing ϕ_d in Eq.2 with ϕ_1, because a summary is a time series and each point in the summary is a real value. Since κ is a non-expansive mapping, $\phi_1 \left(\kappa(\vec{a}_i), \kappa(\vec{b}_i) \right)$ is bounded from above by $\lambda \cdot \phi_d(\vec{a}_i, \vec{b}_i)$, thus

$$\begin{aligned} D_p(hs_\kappa(A), hs_\kappa(B)) &\leq \left(\sum_{i=1}^{n} \left(\lambda \cdot \phi_d \left(\vec{a}_i, \vec{b}_i \right) \right)^p \right)^{\frac{1}{p}} \\ &= \lambda \cdot D_p(A, B) \end{aligned}$$

and, the lower-bounding property holds for D_p, because $0 < \lambda \leq 1$. Likewise, we can show that the lower-bounding property holds for D_{dtw}, i.e.

$$D_{dtw}(hs_\kappa(A), hs_\kappa(B)) \leq \lambda \cdot D_{dtw}(A, B).$$

where $0 < \lambda \leq 1$; the proof is by induction on the length of a match[11]. The case for D_{lcss} is slightly different, because a threshold ϵ is used to decide if two points can be matched. To show the lower-bounding property for D_{lcss}, we show that the *similarity* of two HSums *upper-bounds* the similarity between histories. When points $\vec{a}_i$ and $\vec{b}_j$ are matched, from the definition of S_{lcss}, it must hold that $\phi_d(\vec{a}_i, \vec{b}_j) \leq \epsilon$. Because κ performs a non-expansive mapping, $\phi_1(\kappa(\vec{a}_i), \kappa(\vec{b}_j)) \leq \lambda \cdot \epsilon$ for $0 < \lambda \leq 1$. Therefore,

$$S_{lcss}(A, B, \epsilon) \leq S_{lcss}(hs_\kappa(A), hs_\kappa(B), \lambda \cdot \epsilon).$$

[11]Since $\frac{1}{\lambda} \geq 1$, one can get a tighter lower bound for D_p and D_{dtw} by scaling the distance between summaries by $\frac{1}{\lambda}$.

The lengths of HSums are the same as the lengths of their respective histories, thus the lower-bounding property holds. □