University of Alberta

Efficiently Searching Archival Data for Historical Similarities

by

Reza Sherkat ©

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Doctor of Philosophy**.

Department of Computing Science

Edmonton, Alberta
Fall 2007

Your file Votre référence
ISBN: 978-0-494-33064-7
Our file Notre référence
ISBN: 978-0-494-33064-7

Abstract

With recent developments in the areas of data warehousing and data mining, there has been an increasing interest in querying multiple snapshots of data, often stored in temporal databases, semi-structured document collections, and OLAP applications. Similarity queries, in particular, is an important class of queries with growing applications in fields as diverse as data mining, pattern recognition, multimedia databases, and bioinformatics. This thesis addresses the problem of efficiently answering similarity queries on historical market-basket data and multidimensional histories.

The thesis introduces a domain-independent filter-and-refine framework for evaluating order-preserving similarity queries on historical market-basket data. For instance, given a database of customer transactions and a time period, a query is "find customers with similar purchasing behaviors over this period." Our work is different from previous work on time-series, in that we address the general problem where a history cannot be modeled effectively as a time-series, hence the conventional relevant approaches are not applicable. We propose a similarity measure for histories, based on a constrained aggregation of the similarities between their constituent observations. Given the non-metric nature of our measure, some upper bounds are proposed and an algorithm is developed that uses an index to prune histories that are guaranteed not to be in the answer set of a query. Experimental results on real and synthetic data confirm the effectiveness and efficiency of our approach. For instance, when the minimum length of a match is provided, our approach achieves up to an order of magnitude speed up over alternative methods.

The thesis further studies the problem in a slightly more constrained domain where a history is modeled as a d separate time-series. While there are some solutions for special cases where $d \leq 4$, none of these works scale up well to high-dimensional histories. To address the problem, we propose a class of summaries for histories with a few interesting properties. First, for commonly used distance functions, the summaries can be used to efficiently prune histories that are guaranteed not to be in the answer set of a query. 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, before seeing queries. We also investigate adaptive splitting of histories and develop a few splitting schemes and heuristics to enhance the quality of our approximations, and to improve the performance of our similarity queries. Our experimental results confirm that the combination of our feature extraction approaches and the indexability of our summaries can improve upon existing methods when $d \leq 4$ and scales up for larger values of d and database sizes, based on our experiments on real and synthetic data of 17-dimensional histories.

Acknowledgements

During the time that I have been at the University of Alberta, I have been extremely fortunate to have the guidance, support, and friendship of a number of people who have helped me grow both academically and personally. This thesis would have been much different or would not exist at all if it were not for them.

My deepest thanks and appreciations go to my parents for always providing me with unconditional love, support, and encouragement. At the end of my formal education, I can tell that it was their informal education that taught me most.

Special thanks to my supervisor Dr. Davood Rafiei for his guidance and patience during my PhD program. From the very beginning, Davood provided me with immeasurable assistance and unsparing support to settle in Edmonton and pursue my studies and research. Most importantly, he gave me the freedom to pursue my own interests and the opportunity to learn from my own mistakes.

My gratitude and appreciation to my examiners Dr. Lisa M. Given, Dr. Russell Greiner, Dr. Jörg Sander, and Dr. Raymond T. Ng for carefully reading my thesis and providing valuable comments that helped to improve the quality of this document. Also, thanks to Dr. Greg Kondrak, Dr. Paul R. Messinger, and Dr. Dekang Lin for reading my research proposal and providing valuable suggestions and feedbacks in early stages of this research.

Many thanks to all the extremely personable and supportive people who make the Department of Computing Science into an essentially perfect work environment. My gratitude to Edith Drummond, whose dedication in helping graduate students goes well beyond her regular job duties. Also, thanks to Frances Moore and Karen Berg for their administrative assistance and support during my PhD program. Many thanks to Steve Sutpen for answering my technical questions.

Besides research, I had the opportunity to perform as a teaching assistant for undergraduate courses. Many thanks to Dr. Joseph Culberson for providing mentorship and invaluable advice which helped me improve my skills as a teaching assistant. Many thanks to the instructional support group, specially Chris Helmers and Roman Fedoriw. They provided me with invaluable guidance in lab tutoring, dealing with students, preparing and delivering lab materials, and more. Furthermore, Roman introduced me to the wonderful world of Tai Chi to relax after a day at work.

My friends in Edmonton made a big difference in my life and supported me during my PhD studies. There is not enough space and time to list everyone who deserves to be mentioned here. However, I should especially thank Dr. Ali Aghazadeh, Mazyar and Baharak, Reza and Laleh, Stanley and Deise, Dr. Carol Boliek and Dr. Paul Hagler, Alireza and Parastoo, and Mohammad Reza and Nasimeh for their friendship and kind attention.

I have very much enjoyed being a member of the Database Research Group at the University of Alberta. I am thankful to the professors and other fellow students - Stanley Oliveira, Fan Deng, Alex Coman, Luiza Antonie, Pirooz Chubak, Gabriella Moise, Baljeet Malhotra, Amit Satsangi, Vahid Jazayeri, Jianjun Zhou, and Pouria Pirzadeh. I had fruitful discussions with Fan and Stanley on many topics ranging from research projects to future plans.

A special thanks goes to my lovely wife, Leila, for always being with me through the good and bad times. Her unlimited support and unconditional love gave me energy and endurance throughout every step of this unforgettable journey and I cannot thank her enough for that.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Traditional databases store a single, often most recent, snapshot of a modeled real world. With recent developments in the areas of data warehousing and data mining, there has been an increasing interest in querying multiple snapshots of data, often stored in temporal databases, semi-structured document collections, and OLAP applications. Previous work on querying histories has mainly focused on detecting and representing changes in order to provide a better support for selection and projection queries over multiple versions of data.

Similarity queries on historical archives is important since it enables various forms of analysis on time evolving data. An evidence of this is the increasing interest in efficiently retrieving archival data, based on historical similarity, when the history of an object is described as a sequence of real values (time series). For instance in financial applications, we may want to find stocks that behave in similar fashion within a given time interval by providing the stock of another company or we may want to group together all companies with similar sale patterns. However, in many real-life applications, the history of an object is more complex and cannot be modeled as a time series. Efficient retrieval of objects based on historical similarity is important to understand underlying mechanisms, to extract behavioral patterns, and to analyze trends for decision support. Here are some examples of similarity queries on histories:

- In meteorology, measurements such as temperature, precipitation, wind speed, pressure, moisture, and snowfall are regularly collected (e.g. hourly, daily, or weekly) for many earth surfaces by weather stations. Detecting possible 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.

- In retail, customer transactions are often recorded in a data warehouse for further querying and analysis. The purchase history of a customer, in particular, may show changes of the needs and the preferences over time. It might be desired to find customers with a purchase history similar to a given customer, for example, to provide personal recommendations. Clustering customers with similar purchase history can be used to find trends in market segments.

- In a web archive, such as the Internet Archive[1], several versions of each web page are crawled and stored. An interesting query over such an archive can be: *find web pages with change histories similar to the change history of a given page*. It is quite possible to find web pages that are similar in content at one or more points in time but have different change histories. It is also possible to find rather dissimilar web pages that show similar change histories[2].

- In a hospital, routine observations are made about patients. These observations can be made by doctors or nurses and may include general symptoms, such as "high fever," "rash," "high blood cholesterol," "bleeding," the medications used, the medical advice given, and the responses to treatments. Finding patients with histories similar to the history of a given patient can provide assistance to a doctor in making diagnosis or prescribing medications.

- In financial sector, the history of a stock may be tracked using several indicators such as daily opening and closing prices, and trading volume. The similarities between the histories of two stocks may help to predict

[1]www.archive.org

[2]This can happen for two pages, perhaps maintained by the same authority.

or explain short-term and long-term trends in the market.

The problem to be addressed in this thesis is efficiently evaluating similarity queries on histories where a history is a sequence of observations. Each observation is a point in multi-dimensional space. A timestamp may also be assigned to each observation to indicate the time the observation is recorded. Efficient support of similarity queries for histories is a difficult problem. In general, a history can be represented as a vector whose dimensionality is equal to the number of observations in the history times the dimensionality of each observation. Indexing vectors in high-dimensional space has been considered as a challenging problem in the database community [110]. Weber et al. [122] argue, based on a quantitative analysis, that almost all space partitioning and data clustering approaches to indexing exhibit a linear complexity on the dataset size at higher dimensionalities. They state that existing methods are usually outperformed by a simple linear scan when dimensionality exceeds 10. On the other extreme, Beyer et al. [14] argue that nearest neighbor queries are not meaningful for a dataset when the variance of pairwise distances of objects is low. They state that a similarity query becomes unstable when the difference in the distance between the nearest neighbor and other points in the dataset becomes negligible in this case and a linear scan can outperform any algorithm which uses an index to prune search space.

Despite the challenge, there has been a large body of work that addresses the similarity search problem for special types of histories. For instance, when each observation is a real value and only one observation is recorded at each time instance, a history becomes a time series and there are quite a number of approaches to index time series (e.g. [2], [40], and [97]). If an observation is the spatial position of a moving object at a time point, a history, often called a trajectory, may describe the trace of a moving object in a plane or space, and there are also works to index trajectories (e.g. [74], [119], and [64]). In a relatively more complex setting, an observation can be an image frame, which is in fact a point in high-dimensional space, and there has been some work to index video sequences (e.g. [73], [111]). However, we are not aware of any work that scales up to large databases of high-dimensional histories while satisfying the following requirements:

- *Being robust to noise and outliers* -
- Being robust to time shifting and warping,
- Being exact in returning all candidate results (no false drops), and
- Being order preserving and respecting the temporal ordering of observations.

In this thesis, we propose a filter-and-refine approach to efficiently evaluate similarity queries on historical market-basket data and multi-dimensional histories. In particular, we develop similarity measures, lower bounds, and summarization techniques that can be used to support nearest neighbor and range queries while satisfying all the aforementioned requirements.

1.1 Thesis Statement

The central thesis statement of this research is presented as follows:

> *Efficiently processing nearest-neighbor and range queries over large collection of histories, under some realistic and non-restrictive assumptions, is possible.*

1.2 Thesis Organization

Chapter 2 provides the background material on similarity queries over historical archives. It presents the filter-and-refine framework that is the basis of most promising solutions for exact similarity search as well as our approach. A review of related work on indexing time series, multi-dimensional time series, and sequence data is also presented. The chapter ends with an overview of the problem to be addressed in this dissertation and a list of our main contributions.

Chapter 3 presents a domain-independent framework for formulating and efficiently evaluating order-preserving similarity queries over historical market-basket data. A similarity score is derived for histories based on an aggregation of the similarities between observations matched in a conditional alignment. Two upper bounds for similarity measures are proposed that take advantage of the sparsity of observations and the existence of a temporal neighborhood constraint. Our experimental results on both real and synthetic data confirm the effectiveness, efficiency, and scalability of our approach. In particular, when the minimum length of a conditional alignment is provided, our algorithm achieves up to an order of magnitude speed up over alternative methods.

Chapter 4 studies the problem of efficiently evaluating similarity queries on multi-dimensional histories. A novel summarization technique is proposed with an interesting property that for a large class of order-preserving distance functions commonly used to compare sequences, the distance between summaries lower-bounds the distance between histories. To further reduce the number of unnecessary costly distance computations, a fine-level query independent approximation of histories is proposed based on the notion of commonly used Minimum Bounding hyper-Rectangles (MBRs). Also adaptive splitting schemes are investigated to further improve the tightness of the approximation for high-dimensional histories. Experimental results on real and synthetic data confirm that the indexability of extracted summaries combined with the enhanced pruning power of our approximation improves upon sequential scan, which turns out to be the only competitor on high-dimensional histories.

Chapter 5 summarizes the main points of the thesis, highlights our main contributions, and provides some directions for future research.

Chapter 2

Similarity Queries on Histories

This chapter provides background material on similarity queries over historical archives. In particular, section 2.1 presents a formal definition of histories, our notion of similarity, and the set of queries being studied. Section 2.2 discusses some of the challenges in efficiently evaluating similarity queries, a few general solutions, and some of their limitations. A review of related work on indexing time series, multi-dimensional histories, and sequences data is presented in section 2.3, as well as their limitations in indexing more general histories that are considered in this thesis. Section 2.4 provides an overview of the problem to be addressed in this thesis and highlights the main contributions.

2.1 Basic Definitions and Notations

This section introduces a few notations and terminologies that are referred to in the rest of the thesis.

Definition 1 (Observation). *Let $I = \{t_1, \ldots, t_n\}$ be a set of items. An observation is a set of pairs (t_i, w_i) such that $t_i \in I$ is an item and w_i, a real number, is the unique weight of the item t_i in the observation. Given an ordering of the items in I, an observation can be represented as a vector:*

$$[w_1, w_2, \ldots, w_n]^T.$$

We assume that the weight of an item which is not in the observation is zero. An observation x may be associated with a timestamp in which case we use $ts(x)$ to refer to this timestamp. The set of values a timestamp can take (referred to as its domain) depends on the time granularity in which observations are sampled from real world.

Definition 2 (History). *A history is a chronologically-ordered sequence of observations denoted as:*

$$X = \langle x_1, x_2, \ldots, x_m \rangle$$

such that $ts(x_i) \leq ts(x_j)$ *iff* $i \leq j$. *The length of history* X*, denoted by* $|X|$*, is the number of observations in the history, here* $|X| = m$.

An observation may describe the state of an object at a time point. Alternatively, if we are interested in state transitions rather than the state descriptions, an observation may only describe the changes. In a simple case, an observation can be a real value and the corresponding history is a time series. For instance, the history of a stock may be tracked using an indicator such as daily opening price. In this case, the set of items has only one member, the weight of which is equal to the scalar value recorded at each time instance. For a Web page, the set of items contains all possible terms that may appear in a web page. An observation can be either the content or the updates to a previous version and a timestamp can be the time the page is crawled or the changes are observed. The weight of each term in a web page could be assigned using the *tf.idf* model [100]. The history of web pages can be organized in a temporal database [36] or a semi-structured collection [24, 29]. In spatio-temporal databases [92], an observation may correspond to the spatial location of a moving object, such as a car. In this case, the set of items correspond to the geometric coordinates and the weights correspond to the spatial location of the object at the time the observation is recorded. A sequence of these observations, referred to as a *trajectory*, gives the history of a moving object. In a customer database, an observation can be the set of items purchased within a transaction and the weight of each item in a transaction may indicate the importance of the item (e.g. quantity, price, profit) to the transaction.

2.1.1 The Notion of Similarity

Given two histories, we are often interested in measuring their closeness or similarity. For instance, we may want to find the similarity of the medical histories of two patients. In general, the notion of similarity can vary with queries and may be based on either exact or approximate occurrences of patterns in the two histories. A somewhat domain-independent approach to formulate similarity queries has been proposed by Jagadish et al. [62] where object A is considered similar to object B if B can be reduced to A by a sequence of similarity preserving transformations, for instance moving average for time series. In more domain-dependent settings, a similarity measure or a distance function is used to capture the degree of closeness of two histories in terms of the closeness of their observations. In this thesis, we make the assumption that the observations of two histories are comparable, i.e. there exist a function that measures the closeness of any pair of observations; such a function can be provided as a look-up table by a domain expert, or it can be stated in the form of a closed-form function such as the cosine measure [105], the Jaccard coefficient [100], or its extensions [113]. The distance of two histories can be formulated as an aggregation of the pairwise distances of their observations if the histories are of the same length. In general, the comparison of two histories can be regarded as an alignment process and the score of an optimal alignment, as discussed in chapter 3, can be used to measure the similarity of two histories. A similarity measure is considered *order-preserving* when the similarity between two histories depends on the temporal order in which the observations of the two histories are recorded.

2.1.2 Similarity Queries

Consider a collection S of objects (e.g. documents, time series, histories) that are represented in some feature space; for example, an object may be described as a set of terms or a set of points in multi-dimensional space. Let $d(x, y)$ denote the distance of objects x, y in S. A typical similarity query on the collection S can be stated as follows:

- Nearest neighbors query: Given a query object q and integer k, find k objects in S that have the smallest distances to q.

- Range query: Given a query object q and a distance threshold ϵ, find all objects $x \in S$, such that $d(q,x) \leq \epsilon$.

- All-pair query: Given a threshold ϵ, find all object pairs $(x,y) \in S \times S$, such that $d(x,y) \leq \epsilon$.

Similarity queries play an important role in both searching non-traditional data stored in relational databases and as a building block within different application domains. Here are some examples:

- in machine learning [35], nearest neighbor queries can be used to assign a class to an object using a majority vote among the class of its nearest neighbors. This classifier is simple to construct, compared to complex classifiers such as neural networks, and has shown to have high accuracy in many problems with arbitrary number of classes, e.g. [12] and [117]. Cover et al. [32] show that asymptotically the error-rate of a nearest-neighbor classifier is not more than twice that of the optimal Bayes classifier.

- in clustering [37, 9], range queries can be used to reduce the number of pairwise distance computations, which is $O(n^2)$ for a collection of size n. The intuition is that often, computing a fraction of pairwise distances is sufficient to form clusters and some extra computations can be pruned. In particular, the authors of the DBSCAN algorithm [37] show that using an efficient evaluation of range queries reduces the average number of distance computations for clustering to $O(n \log n)$. An all-pair query can be used to pre-compute the ϵ-neighborhood of all data points to improve the performance of clustering algorithms.

- in visualization [115], nearest neighbors queries can be used to construct a non-linear mapping of objects in a high-dimensional space with complex distance functions into a low-dimensional space. A constraint on the mapping is that the pairwise distances between objects, as well as the notion of neighborhood, must be relatively preserved in order to provide an intuitive representation.

Efficiently processing similarity queries on large databases is a major issue in many of these applications.

2.2 Efficient Retrieval of Similar Histories

A brute-force approach to evaluate similarity queries such as range and nearest neighbors queries is to perform a linear scan and to compare the query with all histories in the database. Although this approach is simple to implement, it is costly to scan a large database and perform a usually expensive distance computation for every record. Retrieving long histories can also be very expensive depending on the access methods employed and the size of the dataset (see Salzberg et al. survey of access methods for time evolving data [106]). For instance, for multi-version documents, often a reference-based scheme is used [29, 24] to represent document histories, where unmodified sections are replaced by links to the corresponding sections in previous versions. This representation preserves the logical structure of documents and can support temporal selection and structural projection queries. However, loading the history of a document can be quite challenging and even a linear scan needs to perform several random disk access to resolve links required for materializing each history.

On the other hand, computing the distance between two histories can be quite expensive, in particular for more robust and flexible distance functions such as dynamic time warping and longest common subsequence. Since these distance functions construct the best correspondence between the observations of two histories using a dynamic programming algorithm, distance computation time can dominate disk access time for complex distance functions and long histories. A solution is to prune the search space using an index structure, and to reduce the number of disk accesses and distance computations.

When objects are represented as points in real vector space and the distance measure is L_p-norm, existing spatial index structures such as R-tree [50] and its variants (e.g. R+tree [109]) can be used to improve the performance of a similarity search. In a more general case where the distance measure is a metric, existing index structures proposed for metric space such as M-tree [31], MVP-tree [16], and the OMNI-family of access methods [42] may be applicable; Chavez et al. [23] present a survey of access methods for metric space. However, in many real settings where objects are represented as points in high-dimensional space, the performance of existing spatial index structures degrades due to a phenomenon referred to as the curse of dimensionality[1]. Moreover, L_p-norm distances are not robust to noise and time-shifting and often more flexible distance functions (e.g. dynamic time warping) are not metric, hence existing metric space index structures cannot be applied directly. Several promising techniques have been proposed to address the problem using a filter-and-refine framework.

2.2.1 A General Filter-and-refine Framework

A general domain independent framework to speed up similarity search is the filter-and-refine framework [2] (also known as signature-based or GEMINI). The technique is originally proposed to index points in high dimensional space, but it is also applied to domains with computationally costly distance functions. The main component of this framework is an embedding which performs dimensionality reduction. A property of this mapping, referred to as the *lower-bounding* property, is that for any pair of objects, the distance of the objects in the transformed space must match or underestimate the true distance of the objects in the original space. In an offline pre-processing step, objects in the database are transformed into their corresponding features and the features are organized in an index called F-index (feature index) in [40]. Similarity queries are processed in two steps:

- In a *filtering* step, F-index is consulted to filter ideally a large fraction of the search space and to efficiently retrieve a superset of the answer set.

[1]In the context of indexing, the data structures scale poorly with data dimensionality [122] making linear scan the only viable choice.

- In a *refining* step, a more accurate but computationally expensive distance function is applied to the retrieved superset to further prune false positives[2].

The lower-bounding property guarantees that the filtering step returns a superset of the answer to a query [40]. For exact similarity queries, the result of the refining step is the exact answer to the query and no qualifying object is missing. Seidl et al. [108] propose an algorithm which uses a priority queue to interleave the filter and refine steps to improve the performance of nearest neighbors queries. The filter-and-refine framework has also been used to support efficiently processing similarity queries over time-series for computationally expensive and/or non-metric distance functions such as Euclidean distance [2], arbitrary L_p-norms [125], dynamic time warping [66], and general metric distance functions [55].

2.2.2 Similarity Queries with False Negatives

A variation of the filter-and-refine framework has also been proposed for similarity search in domains where query response time is the main factor and missing a small number of qualifying objects can be tolerated. The idea is that the lower-bounding property of the embedding can be relaxed and there is a chance of having false negatives. Often there is no rigorous analysis on the number of false negatives and the embeddings are evaluated empirically based on their performance and effectiveness, measured in terms of precision and recall[3]. For instance, to index time series where distance is measured by dynamic time warping, Yi et al. [126] use Fastmap [39] for dimensionality reduction and the Euclidean distance in the feature space. To search for near duplicate documents, Broder et al. [18] propose to hash documents into compact sketches and use the Jaccard coefficient to estimate the resemblance of documents in the feature space. To answer nearest neighbors queries, Gionis et

[2]*false positives* are retrieved candidates that are not in the result set of a query; *false negatives* are the answers to the query that are not retrieved.

[3]Given a query, let *rel* and *ret*, respectively, be the set of objects relevant to the query and the set of objects retrieved as the answer of the query. Precision and recall are defined as $\frac{|rel \bigcap ret|}{|ret|}$ and $\frac{|rel \bigcap ret|}{|rel|}$.

al. [47] propose locally sensitive hashing where similar objects are hashed, with high probability, into the same (or close) locations. A number of methods have been proposed to embed arbitrary spaces into vector space for efficient search, visualization, and other data mining tasks (e.g. [15], [39], [54], and [102]). Athitsos [10] surveys embedding methods for similarity search in non-metric spaces; none of these embeddings satisfy the lower-bounding property.

2.3 Related Work

Similarity search has received much attention in the database community and it is now a relatively mature area of research. The focus of this dissertation is efficiently evaluating similarity queries on large collection of histories. There has been a large body of work on indexing three specializations of histories: (1) time series where each observation is a real value, (2) multi-dimensional time series where each observation is a vector, and (3) sequence data where each observation is a character selected from a finite alphabet. In this section, we review related work on indexing these three types of histories and identify why existing solutions cannot be directly applied to general histories considered in this thesis, namely historical market-basket data and multi-dimensional histories, while satisfying the four main requirements mentioned in chapter 1. Similarity search also has been studied for other data types including shapes [61], images [43], sets [1, 46], graph structures [53], and mixed types of strings and numeric attributes [63]; these other works will not be reviewed here.

2.3.1 Time series

We organize related work on querying time series into a few categories based on the distance function[4] used, the support for shifting and noise, and the possibility of having false negatives. We should note that some categories may overlap and some works may relate to more than one category. Except for the methods that are based on symbolic representations of time series, the related work on time series to some extent take advantage of the numerical repre-

[4]Chapter 4 presents a detailed definition of the distance functions studied in this section.

sentation of observations and cannot be generalized effectively for historical market-basket data. Also, the large alphabet size of historical market-basket data precludes the adaptation of the solutions based on string matching, as discussed in section 2.3.3. However, related work on indexing time series can be used to index history summaries that we derive from multi-dimensional histories, as discussed in chapter 4; this explains our review of the related work.

Indexing time series under Euclidean distance - The first indexing method for fast retrieval of time series is proposed by Agrawal et al. [2] where each time series is considered as a vector and the Euclidean distance is used to measure the dissimilarity of two time series. The authors use Discrete Fourier Transform (DFT) to extract features from time series, based on the observations that for most real time series, only the first few DFT coefficients that correspond to the first few frequencies are strong, i.e. have large amplitudes. Because the distance between DFT coefficients of two time series is equal to the Euclidean distance of the two time series due to Parseval's theorem [84], the first few DFT coefficients can be used to derive a lower-bound of the true distance. The authors organize DFT coefficients in an R*-tree index for efficient filtering. There are some other approaches where DFT is replaced with another transformation, for instance Discrete Wavelet Transform [85, 93] and Chebyshev coefficients [19].

Several optimizations and alternatives have been proposed for DFT-based indexing of time series [2]. For instance, Rafiei et al. [96] take advantage of the symmetric property of DFT coefficients to derive a tighter lower-bound of the true distance without increasing the number of coefficients to be kept in the index. In another work, Vlachos et al. [120] argue that keeping the first DFT coefficients is not as effective as keeping the first few coefficients with highest energy for periodic time series.

Several other feature extraction approaches have been proposed to approximate time series and to speed up similarity queries. The main idea is to segment time series and represent each segment using either a constant value (e.g. [38, 125, 67]) or a Minimum Bounding Rectangle (MBR) (e.g. [80, 75, 94]).

In piecewise constant approximations [38], each time series is partitioned into s segments of equal length. Each segment is approximated by its mean; thus each time series is mapped into an s-dimensional vector. Yi et al. [125] use the same feature extraction approach and develop lower-bounds for an arbitrary L_p-norm distance of two time series based on the L_p-norm distance of their piecewise constant approximations. An improvement to the segmentation approach is proposed by Keogh et al. [67]; their idea is to adaptively divide time series into segments of potentially different lengths in order to derive a better approximation of time series.

On MBR approximation of time series, Moon et al. [80] move a sliding window on time series and extract MBR of the windows as features. In another work, Li et al. [75] group similar segments of different time series and represent each group using a single MBR [75]. In both [80] and [75], the extracted MBRs are organized in an R-tree structure. Another organization for MBRs is proposed by Qu et al. [94] where a relational database stores the extracted MBRs and similarity search is formulated as pattern queries that are evaluated using relational operators.

Supporting shift and scale for Euclidean distance - Comparing two time series using their Euclidean distance is sensitive to shift and scale in both amplitude and time. Several transformations have been proposed to normalize time series before a comparison, in order to make Euclidean distance more robust to shift and scale. Goldin et al. [48] propose similarity transformations and normal forms for time series, where a time series is considered *normal* if it has a zero mean and a unit variance. Normalizing time series is performed in two steps; first the mean is subtracted from every value; second, each value of the resulting series is divided by the variance. We should mention that local shift and scale are not removed by such normalization. In another work, Rafiei et al. [97] propose a set of linear transformations that can model moving average and time scaling, in addition to the operations supported in [48].

In another work, Chu et al. [30] show how similar sequences can be efficiently searched irrespective of differences in offset translation and amplitude scaling. The basic idea is to transform time series onto some shift-eliminated plane where vectors are invariant to linear transformations in time and scale. Another scale- and shift-invariant measure of similarity is proposed in [64] where the distance between two time series is the smallest Euclidean distance after scaling and shifting either one of the trajectories to make it as close as possible to the other one. Agrawal et al. [3] propose a model of similarity where two time series are considered similar if they have enough non-overlapping time-ordered pairs of similar subsequences. The model allows amplitude scaling and offset adjustment and unlike [30, 3], non-matching gaps are also considered while matching two subsequences. This makes the similarity model more robust to noise and outliers.

Methods based on symbolic representations - Several past works use techniques from string matching to compare symbolic representations of time series. Unlike L_p-norm distances including the Euclidean distance, where two time series must have the same length to be comparable, most of the related work in this category can measure the distance of time series of different lengths. Agrawal et al. [4] propose a shape definition language to describe and to retrieve time series. They quantize the changes and represent each change by a distinct symbol which is assigned based on the difference of every two consecutive values. A query is stated very much like a regular expression predicate over shape descriptions. To more efficiently support pattern matching, Huang et al. [56] index symbolic representations of time series using a suffix tree. Chen et al. [26] use string edit distance of symbolic representations to compare time series.

There are some works that combine a symbolic representation with the Euclidean distance. Lin et al. [76] apply piecewise constant approximation and represent each segment of time series using a symbol. They define a distance function for symbols that is a lower bound of the Euclidean distance of the time series. Megalooikonomou et al. [79] use vector quantization and define a weight for each symbol which is used in the distance computation. The weight of a symbol in a time series is determined based on its frequency in the time series, normalized by the number of time series in the database that contain the symbol.

In another approach, Perng et al. [91] model time series as a set of landmarks. They consider each time series as a function which maps a time point into a single value. The n-th order landmarks are time points where the n-th derivative of the function is zero. For instance, first-order landmarks are local extrema and second-order landmarks are inflation points. Weights can be assigned to landmarks to show their importance. For instance, a weighting scheme may assign large weights to global extrema and small weights to local ones. An interesting observation is that for a properly chosen landmark, the similarity is invariant to shifting, uniform amplitude and time scaling, and non-uniform amplitude scaling. We should note that the concept of time scaling considered by Perng et al. is different from dynamic time warping in that a transformation is applied as a *fixed* function to all time series in a database whereas in dynamic time warping, no assumption is made about the nature of warping.

Dynamic time warping and its variants - Even though dynamic time warping (DTW) is a well-known technique in speech processing, Berndt et al. [13] seems to be the first in the database community reported using it. The first attempt to speed up DTW queries is proposed by Yi et al. [126]. They use FastMap [39] to embed time-series into the Euclidean space, where classic multi-dimensional index structures can be used. Because the embedding does not satisfy the lower bounding property, this approach introduces false negatives. To speed up computations, Keogh et al. [69] compute DTW over the piecewise aggregate approximations of time series, but this approach can have false negatives as well.

Kim et al. [71] propose the first solution to exact indexing of DTW. The method extracts four features corresponding to the first, the last, the minimum, and the maximum of the time series. They propose a metric lower bound for DTW based on L_∞-norm distance of features. Park et al. [89] organize symbolic representations of time series using a disk-based suffix-tree structure and provide a lower bound to filter dissimilar subsequences. However, the index size can be 1-2 order of magnitudes larger than the database size. Keogh [66] studies constrained DTW and constructs from a given query, an envelope which surrounds all possible matching points of a time series. Each time series in the database is divided into a pre-specified number of segments. An MBR is constructed for each segment and the set of MBRs are organized in an R-tree. A lower bound for DTW is introduced which operates on query envelope and MBR representations of time series. Later, Zhu et al. [128] propose an improvement by using the idea of piecewise aggregate approximation to encode query envelopes. Sakurai et al. [104] suggest a new lower bounding measure to approximate time warping distance with no warping range constraint. The idea is to construct a coarse representation of time series using some MBRs that approximate time series at several resolutions. The resolution of an MBR is determined by its temporal extent, which can vary from one (finest resolution) to the length of time series. Ratanamahatana et al. [99] develop a bit level representation of time series and a lower bound for DTW based on such representation, which can also take advantage of the run-length coding to improve compression ratio.

There are other variations of DTW which are more suitable for indexing. Chen et al. [25] propose a modified version of DTW called Edit distance with Real Penalty (ERP) which is robust to time shifting but not outliers. Because ERP satisfies the triangle inequality, it can be indexed using metric index structures such as M-tree [31]. In ERP, each point in one time series can be matched with either a point in another time series or a gap. The metric property of ERP is independent of the value assigned for the gap. However, the distance of the two time series depends on the value chosen for the gap. The authors also propose a lower bound for ERP which is metric and can be indexed using a B+tree. Bozkaya et al. [17] introduce a modified version of edit distance where two time series are considered similar if the majority of their points match. Fu et al. [44] combine time warping with uniform sampling, a technique that allows global scaling of time series. They argue that such approach is useful in domains where time series are recorded by different sampling rates and performing a global scaling would result into a better alignment. A continuous variant of DTW has been proposed [82] where points in one sequence can match an interpolated point in the other sequence.

Longest common subsequence and its variants - Das et al. [33] use the length of the longest common subsequence (LCSS) of two time series as a measure of their similarity. They propose an algorithm to estimate the length of LCSS, in order to speed up the queries. Vlachos et al. [119] first partition time series into sets according to their lengths and prove a weaker version of the triangle inequality for LCSS for each set. They apply hierarchical clustering to produce a tree structure for each set and the trees are used for pruning. In another work, Vlachos et al. [117] propose an approach with no false negatives when a constrained version of longest common subsequence is used to compare time series. The main idea is to extract bounding envelops that contain potential matches of time series. As a variant of LCSS, Chen et al. [27] propose Edit Distance on Real sequences (EDR) which assigns penalties to the gaps between two matched subsequences.

Similarity search with false negatives - The works in this group relax the lower-bounding property to improve performance. For instance, Korn et al. [72] keep the first few principal components of a time series as features to reduce database size and to speed up distance computations, when similarity is measured by Euclidean distance. However, since a lossy feature extraction approach is used and the lower-bounding property is not established, there might be some false negatives. Keogh et al. [70] take a probabilistic approach to sequence retrieval and extract characteristic features (e.g. peaks, plateaus, or troughs) to represent time series. Global shape information is represented by the prior probability of relative locations of individual features. A probabilistic model integrates local and global shape features to measure similarity.

There are some other works on constructing signatures from time series. For instance, Keogh et al. [68] hash subsequences of time series into bit strings. A distance function is developed such that, given a segment of a query and a bit string, it provides an estimate of the Euclidean distance between the segment of the query and any segment that can be presented by the bit string. In another work, Indyk et al. [59] transform segments of time series into sketches. Although the distance between the sketches of two segments is guaranteed to be bounded by a factor to the true distance between the original segments with high probability, in both [68] and [68], the lower-bounding property does not hold and there can be false negatives.

Streaming time series - There has been several works (e.g. [45], [124], [123], and [103]) on similarity search in data stream settings where new points are observed continuously and the goal is to monitor the stream and to find some predefined patterns.

2.3.2 Multi-dimensional Time series

We organize related work in this section into two broad categories; (1) indexing trajectories where the dimensionality of each observation does not exceed 4 and (2) retrieving video sequences where each observation is an image frame. There has been some work in spatio-temporal databases that do not fall into the aforementioned categories. These works, such as [92], [114], and [87] study topological and navigational queries and not the similarity between trajectories. They more concentrate on finding objects that are close to a query object at a time instant or during a period.

Indexing trajectories - The first reported work on indexing multi-dimensional data sequences is by Lee et al. [74], where they use the Euclidean distance to compare trajectories. The authors generalize a previous work [40] on time series and replace the piecewise constant approximation of segments with an MBR approximation. Each trajectory is represented by a number of MBRs which are organized in an R-tree. The index is probed for matching trajectories based on query MBRs. A heuristic similar to that of Kamel and Faloutsos [65] is used to segment the trajectories and to derive the MBRs. The main intuition behind this heuristic is to reduce the expected number of indexed MBRs that overlap with a query MBR. Reducing the number of overlaps can in turn lower the expected number of disk accesses when the index is used to filter distant trajectories. However, this heuristic does not provide a tight approximation of trajectories, in particular when the change scale in different dimensions is not the same, as discussed in Chapter 4. Experiments are conducted over datasets with the dimensionality of observations varied from 2 to 4; the scalability of the work for higher dimensionalities is not investigated. For d-dimensional histories, the dimensionality of indexed MBRs is equal to $2(d+1)$. The performance of index structures, including R-tree, degrades when dimensionality increases [108]. As another feature extraction approach, Cai et al. [19] decompose each trajectory into a few time series, represent each time series by a few Chebyshev coefficients, put together the Chebyshev coefficients into a single vector, and organize feature vectors in an R-tree. A drawback of this approach, as well as the work by Lee et al. [74], is the use of Euclidean distance, which is sensitive to noise, time shifting, and amplitude scaling.

Kahveci et al. [64] map trajectories into a shift eliminated plane [30] to support uniform shifting and scaling, where all points of a trajectory are shifted or scaled by the same offset or scale factor. This approach is not applicable when shifts are not uniform or a trajectory is recorded with different sampling rates at different time points.

Vlachos et al. [117, 118] construct an index on trajectories that supports Euclidean distance, DTW, and LCSS using the same index structure. The authors organize the MBRs of trajectories in an R-tree and probe the index using the MBRs of query envelop. The heuristic used to partition trajectories is to minimize total MBR volume, which is also used in an earlier work to index spatio-temporal trajectories [51]. The authors claim that the same heuristic is also useful for filtering. Experiments are performed on 2-4 dimensional histories and the scalability for higher-dimensional histories is not investigated.

Retrieving video sequences - Several approaches have been proposed to extract high-level features from video sequences. However, such feature extractions are often lossy and can result in false negatives. For instance, Chang et al. [21] use as features of a video sequence a small number of key frames with the maximum fidelity to the original video sequence. Cheung et al. [28] select a few seed frames at random plus a small collection of closest frames to each seed. In another work, Indyk et al. [58] keep sequences of shot durations to summarize the activities in a video sequence.

There has been some work on representing and indexing video sequences. For instance, Lee et al. [73] propose a graph-based data structure where frames are segmented into regions and the spatial adjacency of regions along with their velocity and moving directions are used to separate object regions from background. The history of an object region is represented as an object graph. The authors cluster background and object regions to impose a hierarchical structure which also supports similarity search. Similarity is measured by an extension of the edit distance. However, this work is specialized to video sequences, since it explicitly decomposes a sequence into background and moving objects, and it is not clear if it can be applied to historical market-basket data and multi-dimensional histories.

In another work, Shen et al. [111] model video streams as sequences of image frames, and measure the video similarity in terms of the number of similar frames between two video sequences, irrespective of the temporal ordering of the frames. This approach is not order preserving.

2.3.3 Sequence Data

Approximate sequence matching is a problem that arises in many applications including text searching, computational biology, and signal processing [83]. The similarity between two strings is usually defined as the cost of applying a transformation that makes the two strings the same. In particular, the edit distance is defined as the minimum number of single character insertions, deletions, and replacements required to make two strings equal. Edit distance has been used for measuring the functional and evolutionary similarity of DNA and protein strings [112]. The weighted version of edit distance assigns costs to operations and/or characters. As an alternative to the edit distance, the block edit distance (e.g. Varre [116]) measures the minimum number of block edit operations.

Because computing the edit distance of two sequences requires constructing an optimal correspondence between the symbols of the two sequences using a dynamic programming algorithm, which becomes computationally expensive for long sequences, several heuristics have been proposed that trade precision for performance. In particular, FASTA [90] and BLAST [7] rely on the so-called *hit-and-extend* heuristic. The main intuition behind these approaches is that two similar sequences are likely to contain short identical substrings. A deterministic finite automaton (DFA) is constructed from one sequence and this automaton is run against the other sequence to find every identical substring of a fixed length. Unfortunately, the size of the DFA grows exponentially with the size of the alphabet. For historical market basket data, for instance, the alphabet is the power set of the set of items I. Even when the number of items is as small as 1000, the alphabet size is $\simeq$ 1.07e+301 which is large compared to 4 and 20 for DNA and protein sequences respectively.

To scale up accurate optimal alignments [112] to large databases, Hunt et al. [57] and Meek et al. [78] propose disk-based suffix trees that employ a dynamic programming A*-search to prune some extra computations. A suffix-tree is a PATRICIA trie [81] that stores every suffix of an input sequence in a tree. The branching factor of this tree is at most equal to the cardinality of the symbol alphabet which, as we mentioned, is very large for historical market-basket data. For integer alphabets, i.e. when each symbol is an integer in $[1, n]$, Farach [41] constructs suffix-trees that scale up to large alphabets in time linear to alphabet size; this is still very large for historical market-basket data.

Wang et. al. [121] study the problem of retrieving similar event sequences where a match between two sequences is defined in terms of the matching events and the weights of the events. While this work does consider temporal aspects of sequences and might be applicable to histories, matches between observations are limited to *exact* matches and therefore the approach is not robust to noise.

2.3.4 Limitations of Existing Solutions

We have grouped the past work on similarity queries on histories into three categories: (1) time series, (2) multi-dimensional time series, and (3) sequence data. For many real-life scenarios, as listed in Chapter 1, the history of an object cannot be effectively modeled as a time series. On the other hand, many of the solutions developed for efficiently retrieving time series take advantage of the numeric representations of observations and do not apply to more general settings where an observation is a set. Existing solutions developed for multi-dimensional time series

1. *are optimized for 2-4 dimensional histories, and*
2. *do not support warping and time shifting for high dimensional histories.*

On the other hand, the solutions proposed for specific domains, such as indexing video sequences and natural language translations, take advantage of specific features that can be extracted from histories. For instance, to index multimedia, features such as color, texture, background, and motion are extracted from histories. However, we are considering the problem in the general framework in which a similarity measure, which is domain independent, is used to capture the closeness of two histories. In this framework, existing solutions developed for sequence data

1. *do not scale up for larger vocabularies, and*
2. *do not consider temporal aspects of histories.*

These issues are the original motivations for the research conducted and reported in this thesis.

2.4 This Dissertation

The focus of this thesis is on efficiently evaluating nearest neighbors and range queries on historical data. In particular, we are interested in similarity search with no false negatives because of the high risk and cost associated with missing potential answers in real life applications, such as disease diagnosis, drug discovery, and market investment. Two types of histories are considered in this thesis: *multi-dimensional histories* where each observation is a vector and *historical market-basket data* where the set of items is large and each observation is represented more effectively as a set items.

2.4.1 Main Contributions

For historical market basket data, we propose a notion of similarity which generalizes the idea of edit distance to histories. Given the length and the score of an optimal alignment for two histories, our enumeration algorithm efficiently finds a set of common patterns that are observed in the same order in two histories, hence generalizing the concept of the longest common subsequence (commonly used for character strings) to histories. We propose a few upper bounds that help in efficiently processing similarity queries. In particular, one of our upper bounds that is non-metric, targets histories with sparse observations and uses an index structure to speed up our similarity queries.

For queries on multi-dimensional histories, we propose techniques for extracting summaries from histories. We show that a large class of distance functions can be more efficiently computed on history summaries, and that the distance between summaries under-estimates the true distance between corresponding histories. Furthermore, we derive a query-independent optimality criterion for MBR approximations of histories. This approximation is significantly faster to derive, compared to a recently proposed alternative method and can improve upon a traditional volume-based splitting.

To further improve the tightness of our MBR approximations of multi-dimensional histories, we propose a few adaptive splitting strategies. These strategies take advantage of the correlations between dimensions and the variance of the changes in different dimensions to improve upon traditional fixed splitting schemes. Our experiments show that the adaptive splitting schemes can improve the lower bounds for expensive distance functions and can be beneficial for pruning.

Chapter 3

Efficient Retrieval of Historical Market-Basket Data

In this chapter we introduce a new domain-independent framework for formulating and efficiently evaluating similarity queries over historical market-basket data, where given a query history as a sequence of timestamped observations and the pair-wise similarity of observations, we want to find similar histories. We derive a similarity measure for histories, based on an aggregation of the similarities between the observations of the two histories, and propose efficient algorithms for finding an optimal alignment between two histories. To process similarity queries efficiently, we develop some upper bounds for our similarity measure and an algorithm that makes use of those bounds to prune histories that are guaranteed not to be in the answer set. Experimental results on real and synthetic data confirm the effectiveness and efficiency of our approach.

The rest of the chapter is organized as follows: the next section provides an example of historical market-basket data and motivates our work in this chapter. Background materials on sequence alignment are presented in section 3.2. Section 3.3 presents conditional alignment and our similarity model for histories followed by Section 3.4 which presents our approach to process queries over large collections of histories. Experimental results are reported in Section 3.5. A review of related research appears in Section 3.6. Section 3.7 is conclusion and the summary of our contributions.

Table 3.1: An example showing three histories over a period of four days

	h1	*h2*	*h3*
Day 1	{a, b}	{b, c, d}	{b, c, d}
Day 2	{c, d}	{f, g}	{a}
Day 3	{f, g}	{g, h, i}	{i}
Day 4	{h, i, j}	{h, k}	{g, h, i}

3.1 Motivating Example

In a hospital, routine observations are made about patients. These observations can be made by doctors or nurses and may include general symptoms such as "high fever," "rash," "high blood cholesterol," "bleeding," the medications used, responses to the medications, and the medical advice given. If each sign, symptom, or medication is assigned a symbol, then an observation simply becomes a set of symbols, and the medical history of a patient can be described as a sequence of sets. The example in Table 3.1 shows this scenario for three histories over a period of 4 days. An interesting query is *"find medical histories similar to h2."* Suppose the query returns the medical histories *h1* and *h3*. We expect to find some common patterns between similar histories so the next interesting query can be *"in what respect are histories h2 and h1 similar?"* For day 1, the symbol *b* is observed in both *h1* and *h2*. There is no common symbol for day 2, but the two histories also share a symbol for days 3 and 4. We can find a larger overlap between the two histories if we compare days 2, 3 and 4 of *h1* respectively with days 1, 2, and 3 of *h2* where the common pattern will be $\langle\{c,d\},\{f,g\},\{h,i\}\rangle$. Similarly, the common pattern with the largest overlap between *h2* and *h3* is $\langle\{b,c,d\},\{g,h,i\}\rangle$. We might be also interested in a common pattern that covers at least three days of *h2* and *h3* (i.e. $\langle\{b,c,d\},\{i\},\{h\}\rangle$).

3.2 Preliminaries: Optimal Alignment of Histories

Since a history is modeled as a sequence of observations, alignment techniques can be used to measure the similarity between two histories. An alignment, or more precisely a local alignment [49], is a way to line up subsequences of two histories where each observation of a history is matched with either an observation in the other history or a gap.

Definition 3 (Alignment). *An alignment of two histories is a sequence of the following edit operations:*

- $(\alpha \rightarrow \epsilon)$ *denotes the deletion of observation* α*,*
- $(\epsilon \rightarrow \beta)$ *denotes the insertion of observation* β*, and*
- $(\alpha \rightarrow \beta)$ *denotes the matching of* α *with* β

where α *is an observation in one history,* β *is an observation in the other history and* ϵ *denotes a null observation.*

An alignment is assigned a score and this score may be used to compare two alignments.

Definition 4 (Alignment Score). *Let* $\sigma(\alpha \rightarrow \beta)$ *denote the score of matching two observations* α *and* β*, with the constraint that at most one observation can be a null observation. The score of an alignment is defined as an aggregate score of the matches in the alignment.*

If the aggregation function is fixed to *sum*, which is commonly used to compare strings [49], the score of an alignment $A = \langle \alpha_1 \rightarrow \beta_1, \ldots, \alpha_{|A|} \rightarrow \beta_{|A|} \rangle$ is defined as:

$$\sum_{i=1}^{|A|} \sigma(\alpha_i \rightarrow \beta_i) \tag{3.1}$$

3.3 Conditional Optimal Alignments

In general, it is possible to find multiple alignments between two histories, but we are often interested in an alignment with some desired properties. For instance, we may want to find an alignment with the highest score or the longest possible alignment[1]. Such properties of an alignment may be specified in a query in the form of some constraints on the length and/or the score of an alignment.

3.3.1 Limitations of the Existing Solutions

A related problem is string alignment which has been extensively studied in bioinformatics [107], approximate string matching, speech processing, etc. The Smith-Waterman (SW) algorithm [112] is commonly used to find an alignment with the highest score. This algorithm, when applied to two histories of lengths m and n, can find an optimal alignment in $O(mn)$ time using a dynamic programming approach, assuming that two observations can be compared in constant time. However, there are two problems when the SW algorithm is applied to histories. First, it may not be realistic to assume that two observations can be compared in constant time, in particular when the observations are long or the similarity function σ is not trivial. The number of possible observations is also typically huge, and it is not an option to pre-compute the pair-wise similarity between all observations, as the number of observations increase exponentially with the size of alphabet. Second, we may not be interested in an alignment with the highest score. Instead, we might be interested in an alignment of a specific length with the highest score or the longest alignment(s) with a score greater than a threshold. In both cases, the desired alignment is not necessarily an extension of an alignment found by the SW algorithm. Consider, for instance, the histories *h2* and *h3* in Table 3.1. Given σ as the fraction of items common to two observations, an alignment that maximizes the score in Eq. 3.1 and can be found by the SW algorithm is $\langle \{b,c,d\} \rightarrow \{b,c,d\}, \{g,h,i\} \rightarrow \{g,h,i\} \rangle$.

[1]excluding those matches that contain null observations

However, if we are interested in an optimal alignment of length three, i.e. $\langle\{b,c,d\} \rightarrow \{b,c,d\}, \{g,h,i\} \rightarrow \{i\}, \{h,k\} \rightarrow \{g,h,i\}\rangle$, the result of the SW algorithm cannot be extended to find this alignment. Next, we discuss our algorithm for finding the score of an optimal alignment of length l, here referred to as l-alignment.

3.3.2 Finding the Score of an Optimal l-Alignment

Given two histories, we want to find the score of an optimal alignment of a given length l. We can relate the problem of finding the score of an optimal l-alignment to the problem of finding the score of a shorter optimal alignment if the alignment scoring function satisfies the principle of optimality [11]. Let h_1 and h_2 be two histories and denote an optimal alignment of the two histories with A^*. An alignment scoring function $f(\cdot)$ satisfies the principle of optimality if for any pair of non-overlapping prefixes and suffixes of h_1 and h_2:

$$f(A^*) \geq f(A_p^* \oplus A_s^*) \tag{3.2}$$

where A_p^* is an optimal alignment between the two prefixes, A_s^* is an optimal alignment between the two suffixes, and $\oplus$ is the concatenation operator. A large class of functions, including Eq. 3.1, satisfy the principle of optimality; a detailed discussion of these functions can be found in [11]. For the sake of presentation clarity, from now on we will use Eq. 3.1 as our scoring function, but the algorithm discussed here should be applicable to any function that satisfies the principle of optimality. Next, we propose a divide-and-conquer approach to find the score of an optimal l-alignment.

Lemma 1. *For two histories X and Y, let $G_{i,j}^l$ be the score of an optimal l-alignment of two suffixes $\langle x_i, \ldots, x_m\rangle$ and $\langle y_j, \ldots, y_n\rangle$. Then*

$$G_{i,j}^l = max \begin{cases} \sigma(x_i \rightarrow \epsilon) + G_{i+1,j}^l \\ \sigma(\epsilon \rightarrow y_j) + G_{i,j+1}^l \\ \sigma(x_i \rightarrow y_j) + G_{i+1,j+1}^{l-1} \end{cases} \tag{3.3}$$

where $1 \leq l \leq min(m,n)$, $i \leq m-l+1$ and $j \leq n-l+1$. $G_{i,j}^l$ is zero for $i > m-l+1$ or $j > n-l+1$.

Proof. When each history has only one observation, the score of optimal 1-alignment is equal to $\sigma(x_1 \to y_1)$ and Eq.3.3 holds. Let the score of optimal $l-1$-alignment of $\langle x_i, \ldots, x_m\rangle$ and $\langle y_j, \ldots, y_n\rangle$ be computed as $G^{l-1}_{i,j}$. One of the following constructions gives an l-alignment of $\langle x_i, \ldots, x_m\rangle$ and $\langle y_j, \ldots, y_n\rangle$:

- Leave x_i unmatched and find the score of an optimal l-alignment of $\langle x_{i+1}, \ldots, x_m\rangle$ and $\langle y_j, \ldots, y_n\rangle$. The score of this alignment is $G^l_{i+1,j}$ plus the penalty of leaving x_i unmatched, i.e. $\sigma(x_i \to \epsilon)$. Similar argument applies when y_j is left unmatched.

- Match x_i with y_j and find the score of an optimal $(l-1)$-alignment of $\langle x_{i+1}, \ldots, x_m\rangle$ and $\langle y_{j+1}, \ldots, y_n\rangle$. The score of this alignment is $G^{l-1}_{i+1,j+1}$ plus the score of matching x_i with y_j.

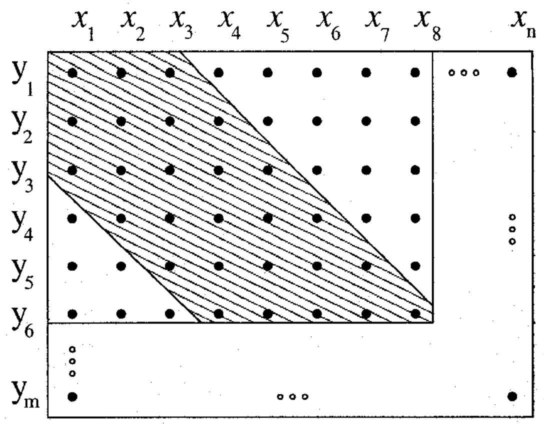
The score of an optimal l-alignment is the maximum of the score of possible l-alignments, thus Eq.3.3 holds. □

According to the definition of $G^l_{i,j}$, the score of an optimal l-alignment of two histories is equal to $G^l_{1,1}$, which can be found using a dynamic programming algorithm which requires $O(mnl)$ calls to $\sigma(\cdot)$ and $O(mn)$ space. This approach, however, will become expensive for long histories. Therefore, we identify some special cases in which these time and space complexities can be reduced.

There are often scenarios in which the two observations cannot be matched if they are recorded far apart. For instance, when aligning the histories of two customers, it may not be reasonable to match purchase transactions that are recorded more than a month apart. Therefore, to preserve a temporal locality among matched observations, we may enforce a constraint similar to the Sakoe-Chiba band[2] which was used in [13] to define the allowed range of warping in dynamic time warping.

Definition 5 (r-neighborhood constraint). *An alignment satisfies the r-neighborhood constraint if for all matches $(x_i \to y_j)$ in the alignment, obser-*

[2]Other bands, e.g. Itakura Parallelogram [95], could also be considered.



x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_n
y_1 y_2 y_3 y_4 y_5 y_6 y_m

Figure 3.1: r-neighborhood constraint for $r = 2$

vation y_j is recorded in the r-neighborhood temporal extent of observation x_i, i.e. $|ts(x_i) - ts(y_j)| \leq r$.

When $r = 0$, only observations that are recorded at the same time could be considered for a match. Increasing r adds some flexibility in matching observations that are recorded within a time frame. Figure 3.1 illustrates the concept for two histories X and Y, when $r = 2$ and the index of each observation corresponds to its timestamp. Each match of the observations must be in the shaded area. For instance, observation x_2 can match with observations y_1, y_2, y_3 and y_4.

Our first improvement takes advantage of an r-neighborhood constraint, when it is present. Let X and Y be two histories of lengths m and n respectively and let $m \geq n$ (the role of X and Y can be interchanged otherwise). For $r > 0$, let m_{2r} denote the maximum number of observations in X that are recorded in a time frame of length $2r$. Since each observation of Y can be matched to one of at most m_{2r} observations in X, the number of calls of $\sigma(\cdot)$ and the space requirement for computing $G^l_{1,1}$ reduce to $O(m_{2r}nl)$ and $O(m_{2r}n)$, respectively. The improvement is significant when X is long and $m_{2r} \ll m$.

Our next improvement is useful if a minimum threshold is specified for the score of matches of an alignment. The idea is to remove observations that cannot participate in an alignment before running a dynamic programming algorithm. More specifically, given two histories X and Y, X is transformed into possibly a shorter history by removing all observations x_i in X such that $\sigma(x_i \to y_j)$ is less than the given threshold for all observations y_j in Y; a similar transformation can be applied to Y. The transformations can be applied in two steps. First, for each observation of x_i in X, $\sigma(x_i \to y_j)$ is computed for each observation y_j in Y, and x_i is removed from X if for all observation y_j in Y, $\sigma(x_i \to y_j)$ is less than the threshold. Removing observation x_i requires $O(n)$ calls to $\sigma(\cdot)$. Removing all observations like x_i from X requires $O(mn)$ calls to $\sigma(\cdot)$. The same argument applies for removing observations from Y. Overall, to apply the transformation to X and Y requires $O(mn)$ calls to $\sigma(\cdot)$.

3.3.3 Finding Common Patterns of Two Histories

Further to finding a degree of similarity between two histories, it is interesting to find out in what respect two histories are similar. More specifically, we want to identify the common patterns that arise in two histories and may give rise to a similarity. Finding such patterns for histories is related to the problem of finding the longest common subsequence (LCS) for strings. However, the idea of only matching identical observations in LCS is too restrictive; we can hardly find any identical observation in two histories. Therefore, we generalize LCS by relaxing the condition that the matched observations must be identical. We do this in two phases: first, we find an alignment of a desired length and score; this is discussed in the next subsection. Then, we can identify the common items in the matched observations and construct a common pattern, referred to here as a *common signature.*

In our setting, an observation is a set. Therefore, given an alignment, a common signature can be a sequence of sets, each being the intersection of two observations matched in the alignment. The common signature for histories is a generalization of the LCS for strings. However, unlike the LCS, we are interested in finding optimal alignments that contain a desired number of matches since the alignment score depends on the number of matches in the alignment.

3.3.4 Enumerating Optimal Conditional Alignments

Given two histories we want to enumerate all optimal l-alignments whose score is greater than a threshold s.

Lemma 2. *Let X and Y be two histories with m and n observations each. For any $1 \leq p \leq l$, if $\langle x_{i_1} \to y_{j_1}, \ldots, x_{i_p} \to y_{j_p}, \ldots, x_{i_l} \to y_{j_l} \rangle$ is an optimal l-alignment of X and Y, then it is necessary that*

- *$\langle x_{i_1} \to y_{j_1}, \ldots, x_{i_{p-1}} \to y_{j_{p-1}} \rangle$ is an optimal $(p-1)$-alignment of two prefixes of X and Y, i.e. $\langle x_1, \ldots, x_{i_p-1} \rangle$ and $\langle y_1, \ldots, y_{j_p-1} \rangle$.*
- *$\langle x_{i_{p+1}} \to y_{j_{p+1}}, \ldots, x_{i_l} \to y_{j_l} \rangle$ is an optimal $(l-p+1)$-alignment of two suffixes of X and Y, i.e. $\langle x_{i_p+1}, \ldots, x_m \rangle$ and $\langle y_{j_p+1}, \ldots, y_n \rangle$.*

This lemma is a direct result of the principle of optimality. To find desired alignments of two histories, we can first locate a match that is guaranteed to be in a desired alignment; we refer to this match as a *pivot.* A desired alignment then can be formed by concatenating the optimal $(p-1)$-alignment of the prefixes, the pivot, and the optimal $(l-p+1)$-alignment of the suffixes. By construction, a match $(x_{i_p} \to y_{j_p})$ is a pivot only if the score of the optimal $(p-1)$-alignment of prefixes $\langle x_1, \ldots, x_{i_p-1} \rangle$ and $\langle y_1, \ldots, y_{j_p-1} \rangle$ plus $G^{l-p}_{i_p,j_p}$ is not less than the desired alignment score s. Note that for $p = 1$, the $(p-1)$-alignment of prefixes will be empty and the pivot will be the first match of the alignment. Algorithm 1 conducts a branch-and-bound search to enumerate all desired alignments of two histories. In each step, the algorithm identifies pivots provided that $G^{l-p}_{i_p,j_p}$ is computed ahead using Eq. 3.3 and the $(p-1)$-alignment of prefixes is available in A. All $(l-p)$-alignments of suffixes $\langle x_{i_p}, \ldots, x_m \rangle$ and $\langle y_{j_p}, \ldots, y_n \rangle$ that cannot contribute to form a desired alignment are pruned effectively. Algorithm 1 can be parametrized to find the following alignments:

1. All alignments of length l: call $Enum(X, Y, l, 0, \langle\rangle)$.
2. k alignments of length at least l_1 with the highest scores: call $Enum(\cdot)$ with the top k scores of $G^l_{i,j}$ such that $l \geq l_1$, $i \leq m-l+1$ and $j \leq n-l+1$.

Algorithm 1: Enumerate

```
Input  : $X = \langle x_1, \dots, x_m \rangle$, $Y = \langle y_1, \dots, y_n \rangle$, $l$, $s$, $A$
/* $l$ and $s$ are the length and the minimum score of desired alignment
respectively. $A$ is the alignment constructed so far:
$\langle x_{i_1} \rightarrow y_{j_1}, \dots, x_{i_{p-1}} \rightarrow y_{j_{p-1}} \rangle$, initially empty */
Output: Enumerates desired alignments.

Procedure Enum( $X$, $Y$, $l$, $s$, $A$ )
begin
    if $l = 0$ then
        if $s \leq 0$ then return $A$
        return
    if $A = \langle \rangle$ then
        $R_1 \leftarrow \{1, \dots, m - l + 1\}$
        $R_2 \leftarrow \{1, \dots, n - l + 1\}$
    else
        $R_1 \leftarrow \{i_{p-1} + 1, \dots, m - l + 1\}$
        $R_2 \leftarrow \{j_{p-1} + 1, \dots, n - l + 1\}$
    foreach $(i_p, j_p) \in R_1 \times R_2$ do
        if $(x_{i_p} \rightarrow y_{j_p})$ is a pivot then
            $A' = A \oplus (x_{i_p} \rightarrow y_{j_p})$; /*concat $A$ & pivot*/
            Enum( $X$, $Y$, $l - 1$, $s - \sigma(x_{i_p} \rightarrow y_{j_p})$ , $A'$)
end
```

3. k longest alignments with a score more than a threshold: call $Enum(\cdot)$ for k pairs of length and score, where the length varies from $min(m, n)$ to 1 and the score is greater than the given threshold.

Modifying Algorithm 1 to accommodate a gap constraint is straightforward; basically a match that violates the r-neighborhood constraint cannot be considered as a pivot. In the next section, we show how to process similarity queries efficiently over a large database of histories.

3.4 Queries over Large Collection of Histories

We now consider the problem of efficiently evaluating similarity queries over a large collection of histories where the query is a history with two parameters r and l, which respectively specify the r-neighborhood constraint and the minimum length for desired alignments. A history in the database is a candidate if it can form an alignment with the query history such that the alignment contains at least l matches and satisfies the r-neighborhood constraint. Histories may be regarded as points in high-dimensional space, where the dimensionality depends on the cardinality of the itemset I and the length of histories. Since the performance of the structures proposed to index high-dimensional spaces degrades when the number of dimensions increases [108], ideally, we want to construct an index on histories using metric space structures (e.g. VP-tree [127] and M-tree [31]). However, this requires a metric distance[3] function. In an attempt to form a distance function from the alignment score in Eq. 3.1, we first define the similarity of two histories as a normalized score of their optimal alignments. Let A denote an optimal l-alignment of two histories X and Y,

$$sim_l(X, Y) = \frac{f(A)}{min(|X|, |Y|)}. \tag{3.4}$$

The similarity of two histories is guaranteed to be in $[0, 1]$ provided that the score of matching two observations always lies in $[0, 1]$. Under this condition, the distance between two histories can be defined as $d_l(X, Y) = 1 - sim_l(X, Y)$. This function in general is not metric, except under very specific settings. For instance when $1 - \sigma(\cdot)$ is a metric and there is no constraint on the length of a match and also there is no r-neighborhood constraint, the distance function becomes the edit distance and is a metric. We are not sure if the distance function can be modified into a metric while still keeping its generality. In particular, the function is not a metric when $1 - \sigma(\cdot)$ is not a metric. This can be proved using a counterexample; consider the case where each history has only one observation. Even if $1 - \sigma(\cdot)$ is a metric, there are other cases where the function $d_l(X, Y)$ is not a metric. For instance, let the score of matching two sets R and S be $\sigma(R \rightarrow S) = \frac{2|R \cap S|}{|R|+|S|}$; here $1 - \sigma(\cdot)$ is a metric [107]. Consider $h_1 = \langle\{a\}, \{b, c\}\rangle$, $h_2 = \langle\{a\}, \{b, c\}, \{d\}\rangle$, and $h_3 = \langle\{b, c\}, \{d, e\}\rangle$.

[3]A distance function d is metric if for any three objects o_1, o_2, and o_3

1. it is symmetric, i.e. $d(o_1, o_2) = d(o_2, o_1)$,
2. it is non-negative and $d(o_1, o_2) = 0$ iff $o_1 = o_2$, and
3. it satisfies the triangle inequality, i.e. $d(o_1, o_3) < d(o_1, o_2) + d(o_2, o_3)$.

Because h_1 and h_2 have two equal observations in common, $d_2(h_1, h_2) = 0$. Also, $d_2(h_2, h_3) = \frac{1}{6}$, and $d_2(h_1, h_3) = 1$, the triangle inequality does not hold. Therefore, the distance between histories, as discussed above, cannot be indexed, for instance using a spatial access method, in general. A straightforward alternative is to do a sequential scanning; but this is not efficient due to the large number of disk access required to read all histories and the complexity of comparing two histories. A B+tree index on the length of histories can filter those histories of length less than l which cannot be candidates. This will save a fraction of disk accesses and computations; however, the similarity is still computed for non-qualifying histories. To prune some of those similarity computations, we propose two upper bounds for $sim_l(X, Y)$ in the next subsection. The first upper bound is quick to compute but requires reading the histories. The second upper bound can be evaluated efficiently for a subset of the database using an index structure. In what follows, we assume that X is the query history and Y is a data history in the database, with m and n observations each.

3.4.1 A General Upper Bound

For each observation y_i of Y, let x_{i^*} be an observation with the highest similarity to y_i among all observations of X that are recorded in the r-neighborhood of y_i. In case no such observation exists, we assume that x_{i^*} is the null observation (i.e. ϵ) and have $\sigma(x_{i^*} \to y_i) = 0$. Let S_l be a set that contains l observations of Y, such that for every pair of observations y_i and y_j, if $y_i \in S_l$ and $y_j \notin S_l$:

$$\sigma(x_{i^*} \to y_i) \geq \sigma(x_{j^*} \to y_j)$$

Example 1. For two histories $X = \langle x_1, x_2, x_3 \rangle$ and $Y = \langle y_1, y_2, y_3 \rangle$, let $\Sigma_{i,j}$ be the score of matching x_i with y_j, and

$$\Sigma = \begin{bmatrix} 0.2 & 0.5 & 0.3 \\ 0.4 & 0.1 & 0.1 \\ 0.8 & 0.2 & 0.3 \end{bmatrix}.$$

The two observations of Y having highest matching scores with any observation of X are y_1 and y_2, therefore $S_2 = \{y_1, y_2\}$. For y_1, $x_{1^*} = x_3$ and for y_1,

$x_{2^*} = x_1$.

Lemma 3. *For two histories X and Y, $usim_l(X,Y)$ defined as*

$$usim_l(X,Y) = \frac{\sum_{y_i \in S_l} \sigma(x_{i^*} \rightarrow y_i)}{min(m,n)} \tag{3.5}$$

provides an upper bound for $sim_l(X,Y)$.

Informally, $usim_l(X,Y)$ can be seen as the score of an optimal *relaxed l-alignment*. Each observation in X could be potentially matched with more than one observation of Y. The order of observations in individual histories may not be preserved completely in the alignment, i.e. there could be two observations x_{i_1} and x_{i_2} in X that are respectively matched with y_{j_2} and y_{j_1} of Y, such that $i_1 < i_2$ and $j_1 < j_2$.

Compared to $sim_l(X,Y)$, $usim_l(X,Y)$ can be computed in less time. Let m_{2r} be the maximum number of observations recorded for X in a time interval of length $2r$. The upper bound can be computed in $O(n(m_{2r} + \log l) + l)$ time, compared to $O(m_{2r}nl)$ which is required to compute the actual similarity. The upper bound can be used to prune non-qualifying histories before a similarity computation.

In the case of a k nearest neighbor (k-NN) query X, a data history Y can be filtered out safely when the upper bound $usim_l(X,Y)$ is less than the score of the k-th best candidate found so far. Otherwise, the similarity of the history and the query is computed and the list of k best candidates is updated if the similarity is more than the score of the k-th best candidate. After processing all histories, the result of the query is the list of k best candidates.

In the case of a range query X, a data history Y can be filtered out safely if its upper bound $usim_l(X,Y)$ is less than the threshold of the range query. Otherwise, the similarity of the history and the query must be computed and the data history is included in the result of the query if the similarity is greater than the threshold.

For both types of queries, every history must be read before we can decide if a history can be pruned or not. In fact, the upper bound is computed even for histories that have no observation similar to any observation of the query. An interesting question is if it is possible to filter some histories prior to computing the upper bound. We believe that an exact answer to this question depends on functions $\sigma(\cdot)$ and $f(\cdot)$. Next, we provide an affirmative answer to this question under conditions defined in the next section when observations are normalized to have a unit norm.

3.4.2 An Index-based Upper Bound for Sparse Observations

Our proposed upper bound in this section is aimed at sparse observations that are common, for instance, in market basket data where a transaction typically consists of a few items (out of the set of all possible items). Let $\vec{x}[t]$ denotes the weight of item $t \in I$ in observation $\vec{x}$. We propose an upper bound for $sim_l(\cdot)$ that takes advantage of the sparsity of the observations to reduce the number of histories that need to be scanned or compared to the query. Unlike $usim_l(\cdot)$, this new upper bound can be efficiently computed using an inverted index on observations. In many real-life applications where only a small subset of the histories in the database are similar to a query, our approach turns out to be more efficient than a sequential scan (as shown in our experiments).

Lemma 4. *Let $\vec{x}_j[t]$ denote the weight of item $t \in I$ in observation $\vec{x}_j$ of history X. For history X, let $I_X \subseteq I$ denote the set of all items that appear in at least one observation of X with a non-zero weight. $\sigma^*(\vec{x}_{i^*} \to \vec{y}_i)$, defined as*

$$\sum_{t \in I_X} \vec{y}_i[t] \cdot \begin{cases} max_j \left\{ \vec{x}_j[t] \;\middle|\; |ts(\vec{y}_i) - ts(\vec{x}_j)| \leq r \right\} & if\ \vec{y}_i[t] > 0 \\ 0 & \\ min_j \left\{ \vec{x}_j[t] \;\middle|\; |ts(\vec{y}_i) - ts(\vec{x}_j)| \leq r \right\} & if\ \vec{y}_i[t] < 0 \end{cases}$$

overestimates $\sigma(\vec{x}_{i^*} \to \vec{y}_i)$, when $\sigma(\cdot)$ is the cosine or the extended Jaccard coefficient.

Proof. The cosine measure and the extended Jaccard coefficient of two vectors

$\vec{x}$ and $\vec{y}$ are defined as

$$\begin{aligned} s_{cosine}(\vec{x}, \vec{y}) &= \frac{\vec{x}^T \vec{y}}{\|\vec{x}\|_2 \|\vec{y}\|_2} \\ s_{Jaccard}(\vec{x}, \vec{y}) &= \frac{\vec{x}^T \vec{y}}{\vec{x}^T \vec{x} + \vec{y}^T \vec{y} - \vec{x}^T \vec{y}} \end{aligned}$$

where $\|\vec{x}\|_2$ denotes the L_2-norm of vector $\vec{x}$. For two vectors $\vec{y}_i$ and $\vec{x}_{i^*}$ of unit norm:

$$\begin{aligned} s_{Jaccard}(\vec{x}_{i^*}, \vec{y}_i) &\leq s_{cosine}(\vec{x}_{i^*}, \vec{y}_i) \\ &= \sum_{t \in I} \vec{x}_{i^*}[t] \cdot \vec{y}_i[t] \end{aligned} \tag{3.6}$$

For any observation $\vec{x}_{i^*}$:

$$\vec{x}_{i^*}[t] \leq max_j \left\{ \vec{x}_j[t] \;\middle|\; |ts(\vec{y}_i) - ts(\vec{x}_j)| \leq r \right\} \tag{3.7}$$

and similarly,

$$\vec{x}_{i^*}[t] \geq min_j \left\{ \vec{x}_j[t] \;\middle|\; |ts(\vec{y}_i) - ts(\vec{x}_j)| \leq r \right\} \tag{3.8}$$

Replacing $\vec{x}_{i^*}[t]$ in Eq.3.6 with the right hand side of Eq.3.7(Eq.3.8) when $\vec{y}_i[t]$ is positive(negative) and zero otherwise results into an upper-bound for Eq.3.6 and establishes the proof. □

Lemma 5. *For two histories X and Y, if $|Y| \geq l$,*

$$Usim(X, Y) = \frac{\sum_{i=1}^{|Y|} \sigma^*(\vec{x}_{i^*} \rightarrow \vec{y}_i)}{min(m, n)} \tag{3.9}$$

provides an upper bound for $sim_l(X, Y)$.

Proof. While $usim_l(X, Y)$ considers the score of l best matches for the optimal relaxed l-alignment, $Usim(X, Y)$ considers $|Y| \geq l$ best matches. Furthermore, for each match, an upper bound of the score is considered. Therefore, $Usim(X, Y) \geq usim_l(X, Y) \geq sim_l(X, Y)$. □

Intuitively, this upper bound is the score of an optimal *relaxed* alignment that matches each observation $\vec{y}_i$ with the best observation that can be constructed from all observations of X in the r-neighborhood of $\vec{y}_i$. Indeed,

$Usim(X, Y)$ can be computed efficiently using an inverted index that maps each item $t \in I$ to a list of (h_{id}, ts, w) triplets. Each such triplet indicates that item t has a non-zero weight of w in the observation recorded at timestamp ts for history h_{id}. For each query X, first the set of items I_X is extracted from the query and $Usim(X, Y)$ is set to zero for all histories Y in the database. Next, for each item $t \in I_X$, the list associated with t is scanned from the inverted index. For each triplet $(Y, ts(\vec{y}_i), \vec{y}_i[t])$ in this list, $\vec{x}_{i^*}[t]$, the maximum value of $\vec{x}_j[t]$ for all observations of X in the r-neighborhood of $\vec{y}_i$, is identified and $Usim(X, Y)$ is updated accordingly. To use this upper bound for a range query, it is necessary to retrieve a history Y only if $Usim(X, Y)$ is greater than the threshold of the range query where X is a query history. Similarly, for a k-NN query, it is necessary to retrieve a history Y only if $Usim(X, Y)$ is greater than the similarity of the query and the k-th best candidate found so far. In both cases, $Usim(X, Y)$ can be computed using an inverted index that maps items to observations. Both queries can also use $usim_l$ to further prune some histories not already filtered by $Usim$, since it overestimates $usim_l$ and there can be still false positives. The correctness of this approach is guaranteed by the fact that both $Usim(X, Y)$ and $usim_l(X, Y)$ overestimate $sim_l(X, Y)$.

3.5 Experimental Evaluation

We present the result of an experimental study of our approach on both real and synthetic data sets. We also examine some of the solutions developed for time series data and show why they are not applicable to the problem discussed in this chapter. We ran experiments to investigate both the effectiveness of our scheme and the efficiency of our approach of processing queries. The results show that our similarity measure is effective to retrieve histories with similar patterns and that our algorithms are efficient and scalable with the number of histories and the number of items. Experiments were performed on a machine with a single AMD/XP2600 CPU running Red Hat Linux, and all algorithms were implemented in C.

3.5.1 Datasets

We used three data sets in our experiments: a real dataset (the DBLP collection) and two synthetic datasets. One synthetic dataset was generated based on a simple model for changes between consecutive observations of a history and the other synthetic dataset was generated using a modified version of the data generator for sequential market basket data [6].

The DBLP collection [34] contains bibliography of publications in computer science since 1936. We considered each journal or conference as a sequence of observations, each containing the set of terms in the table of contents of the corresponding journal issue or a conference proceeding (excluding author names). Terms were assigned weights using the *tf.idf* scheme [105]. The timestamp of each observation was the year that the journal was published or the conference was held. The history for the VLDB conference, for instance, had 29 observations(as of March 20, 2004). From this dataset, we could extract extract 2,784 histories; we used this dataset to provide anecdotal examples of the naturalness of our similarity measure in finding conferences/journals related to a query. Each query is stated as the history of a conference or journal.

The Synth1 collection contains histories of synthetic documents. We modeled each document as a set of terms, which in turn, was represented using a bit string of length n, with a *one* in position i indicating the presence of term i and with a *zero* indicating the absence of the corresponding term in the document. We further assumed that the number of changes between two consecutive versions of a document (i.e. the insertion of new terms or the removal of some existing terms) follows a Poisson distribution [88], and that the numbers of changes in non-overlapping intervals were independent for all intervals. To make the next version of a document predictable from the current version, we assumed that changes follows the Gray code order, although other orders could also be considered. In other words, if the number of changes between version v_i and v_{i+1} is k, the bit string representation for v_{i+1} corresponds to the k-th bit string that follows v_i in the gray code order. This dataset contained $20,000$ histories, $n = 8$, and the first observation of each history was selected uniformly at random from the first 16 gray codes. For each history, the parameter for the Poisson distribution that generated the number of changes was selected uniformly from $\{1, \ldots, 10\}$. The number of observations in each history was uniformly distributed in the range $[32, 64]$. We used this dataset to evaluate the effectiveness of our similarity measure in retrieving histories that were generated using nearly the same parameters.

The Synth2 collection simulates a database of customer purchase histories. We used the synthetic data generator introduced in [6], but also assigned a hypothetical timestamp to each transaction and a weight to each item in a transaction. This dataset contained $8,000$ histories. The number of distinct items was $1,000$. The average number of observations in each history and the average number of items in each observation was 10, which is the default setting used in data mining experiments, e.g. [5] and [6]. For each history, the timestamp for the first transaction, as well as the difference in timestamps for two consecutive transactions, was a natural number, chosen uniformly at random from $\{1, 2, 3, 4, 5\}$. The weight for each item in a transaction was a uniformly distributed random number in $[0, 1]$. Each observation was normalized to have a unit norm, so that we could use the upper bound we proposed in Section 4.2. We used this dataset to measure the performance and scalability of our algorithms.

3.5.2 Effectiveness of sim_l

We conducted some experiments to examine the effectiveness of our similarity measure on the DBLP collection. We posed publications, either conferences or journals, as queries and retrieved a ranked list of similar publications as reported by our similarity measure. Similar publications share one or several topics of interest that change in time. Likewise, new approaches and ideas are mostly introduced and developed in similar publications. Therefore it is likely that similar change trends are observed in publications that belong to the same or related communities. Table 3.2 lists the result of 10-NN query for the VLDB, the KDD, and the AAAI conferences. As it can be seen, the publications are focused on topics related to *databases* for the VLDB, topics

Table 3.2: 10-NN query for the VLDB, the KDD, and the AAAI conferences

VLDB	KDD	AAAI
VLDB	KDD	AAAI
ICDE	PKDD	IJCAI
SIGMOD	DaWak	ECAI
DEXA	ICML	AAAI/IAAI
IDEAS	CIKM	FLAIRS
IEEE TKDE	FLAIRS	PRICAI
DASFAA	IEEE TKDE	Artificial Intelligence
CIKM	ICTAI	ICTAI
EDBT	ICDE	IEA/AIE
DEXA Workshop	ISMIS	GECCO

related to *data mining* for the KDD, and topics related to *artificial intelligence* for the AAAI. However, this result cannot be used as a strong evidence of the effectiveness of sim_l; two similar publications may have a larger overlap in their sets of terms, compared to two publications which are not similar. Thus a question that still remains is whether the result reported in Table 3.2 can be obtained from the *static* similarity of two publications, which is solely based on the overlap between their term sets.

To further investigate this issue and to objectively compare sim_l with other possible alternatives in retrieving similar histories, we designed another experiment where the set of terms was the same for all histories but the change pattern of histories were different. We generated $2,000$ queries using the same mechanism used to generate Synth1. We retrieved 3 ranked lists using a k-NN query with $k = 10$, based on sim_l (Eq.3.4) and two other measures; (1) LAST, which measures the similarity between the last observations of two histories as bag of words, and (2) UNIONALL which measures the similarity between two observations each formed by performing a union of all observations in the corresponding history. We used the Jaccard coefficient to measure the similarity between two observations and selected l, the desired length of an alignment, uniformly at random from $\{32, \ldots, 64\}$. Let $\lambda_q \in \{32, \ldots, 64\}$ be the parameter used to generate the query and λ_i, $i = 1, \ldots, k$, be the parameter used to generate the history that is ranked i-th in the answer set. Since λ_q and λ_i are responsible for the change pattern of the corresponding histories, the

Table 3.3: Mean and Stand. Dev. of $MD(\lambda_q, k)$ for 1-NN and 10-NN queries

Similarity measure	$MD(\lambda_q, 1)$		$MD(\lambda_q, 10)$	
	Mean	Stand. Dev.	Mean	Stand. Dev.
sim_l	0.30	0.49	0.30	0.29
UNIONALL	1.81	1.69	1.90	1.14
LAST	3.05	2.36	3.12	1.03

difference between λ_q and λ_i must be small for two histories having similar change patterns. Therefore, we evaluated the mean deviation of λ_i from λ_q, defined as

$$MD(\lambda_q, k) = \frac{\sum_{i=1}^{k} |\lambda_i - \lambda_q|}{k}$$

to assess the effectiveness of the similarity measures. Table 3.3 shows the average and standard deviation of $MD(\lambda_q, k)$ for the best and top 10 results. According to the results, λ_i is expected to be closer to λ_q for sim_l, which in turn confirms the effectiveness of sim_l. Since LAST only considers the last observation of each history, there is a high chance that two histories with different initial observations and generating parameters have exactly the same last observations. The major drawback of UNIONALL is that the union of observations may include all the terms, which makes a history pretty much similar to any other history independent of the generating parameter. Moreover, LAST is not sensitive to the order of observations, i.e. all the permutations of a given history will be treated as if they are identical.

3.5.3 Pruning Power and Efficiency

We conducted experiments to evaluate the performance of processing similarity queries on Synth2 dataset. Each query was a history selected randomly from the dataset. The parameters for each query (either k-NN or range query) are r and l, which specify the r-neighborhood constraint and the minimum number of desired matches in an alignment between a query and a data history, respectively. We compared a naive scan, which reads and computes sim_l for all histories in the database, with three pruning schemes; (1) LP uses a B$^+$tree and evaluates similarity for histories that have more than l observations, (2) LUBP uses $usim_l$ (Eq.3.5) in addition to the number of observations to prune unnec-

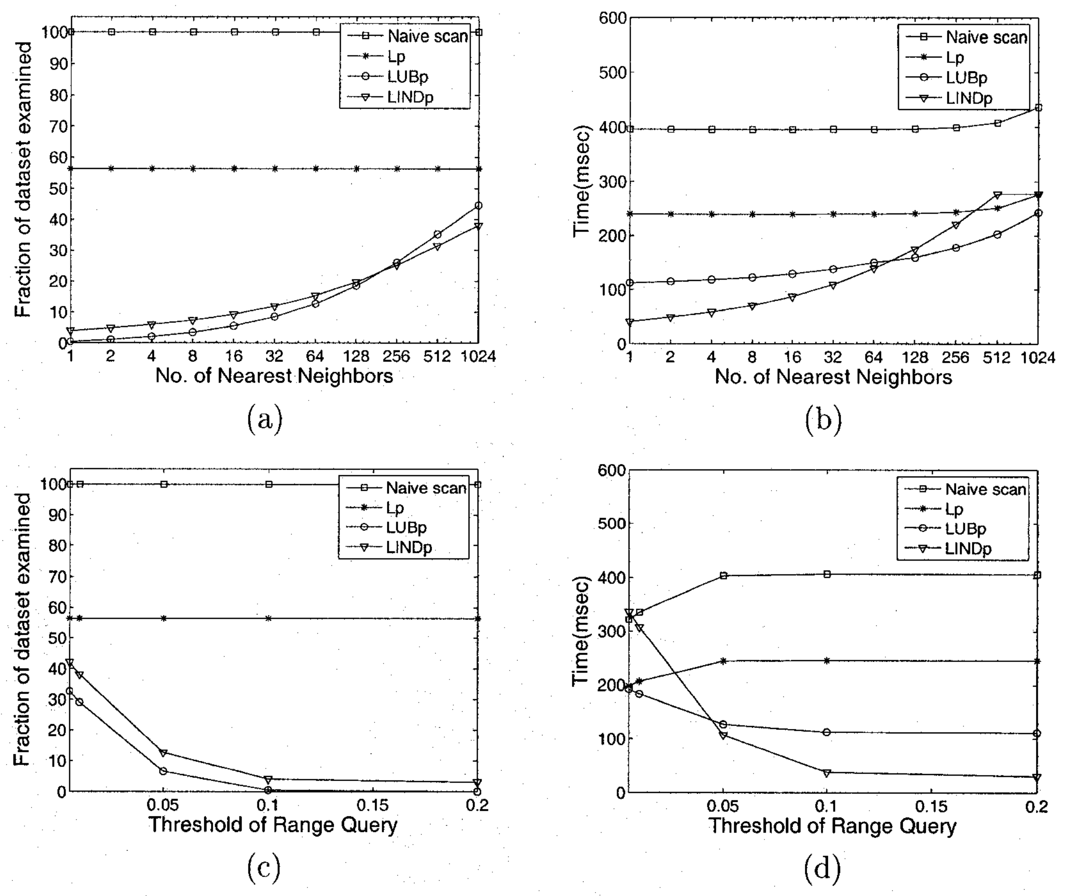


Figure 3.2: Pruning power and average query processing time for k-NN (top) and range (bottom) queries

essary computations, and (3) LINDP uses $Usim$ (Eq.3.9), an inverted index on items, and the length of histories to read and evaluate the similarity only for a fraction of the histories of the dataset. Since our proposed approaches guarantied the returning of all qualifying histories (our upper bounds overestimate sim_l), we measure only pruning power (the fraction of dataset for which actual similarity is evaluated) and query response time. In each case, the reported result is the average of preforming each experiment for 200 queries with the cosine measure used to quantify the similarity of observations; we obtained very similar results when the extended Jaccard coefficient was used, hence the results are not reported.

In the first experiment, we selected r and l randomly from $\{1, \ldots, 4\}$ and $\{1, \ldots, 20\}$, respectively. Figure 3.2(a) compares the pruning power for k-

NN queries when k varies from 1 (i.e. the nearest neighbor) to $1,024$ (which returns approximately 12% of the dataset). Using the length of the history results in pruning about 45% of the histories safely. Using the proposed upper bounds in addition to length results in a remarkable pruning. The pruning decreases as k increases since the similarity needs to be evaluated for a larger fraction of the database. The pruning reduces the response time for k-NN queries (Figure 3.2(b)). Note that although we observe a better pruning for LUBP compared to LINDP when $k \leq 192$[4], LINDP has a better response time showing up to an order of magnitude speed-up over the naive scan for nearest-neighbor queries. This speedup occurs because LINDP avoids reading some of the non-qualifying histories. However, the speedup comes with the extra cost of performing random disk accesses, which dominates the cost of sequential scan in LUBP when $k > 192$ and query selectivity[5] is high.

Figure 3.2(c) compares the pruning power for a range query when the threshold of the query is increased from 0.005 to 0.2. Note that LUBP always shows a better pruning power compared to LINDP since LINDP uses an over-estimation of the upper bound used by LUBP. However, LINDP reads a smaller fraction of the database, and shows a better response time as the number of histories to be scanned decreases (Figure 3.2(d)), making it more efficient than LUBP when the threshold is greater than 0.05, i.e. smaller selectivity.

We next investigate how the parameters r and l could affect the performance of our proposed methods. We report only the results for nearest-neighbor queries; we obtained very similar results for k-NN and ranges queries. First we varied r from 0 to 16 and for each r, we selected l randomly from $[1, 20]$ and picked (randomly) a history that had more than l observations as a query. Figure 3.3(a) shows that increasing r increases the response times for both LUBP and LINDP. However, LINDP slows down more quickly. Note that both methods overestimate sim_l using the score of a relaxed alignment. The chance of matching an observation of a data history with more than one

[4]This is expected since $Usim$ overestimates $usim_l$

[5]Defined as the fraction of records referenced by a query (i.e. satisfying a condition)

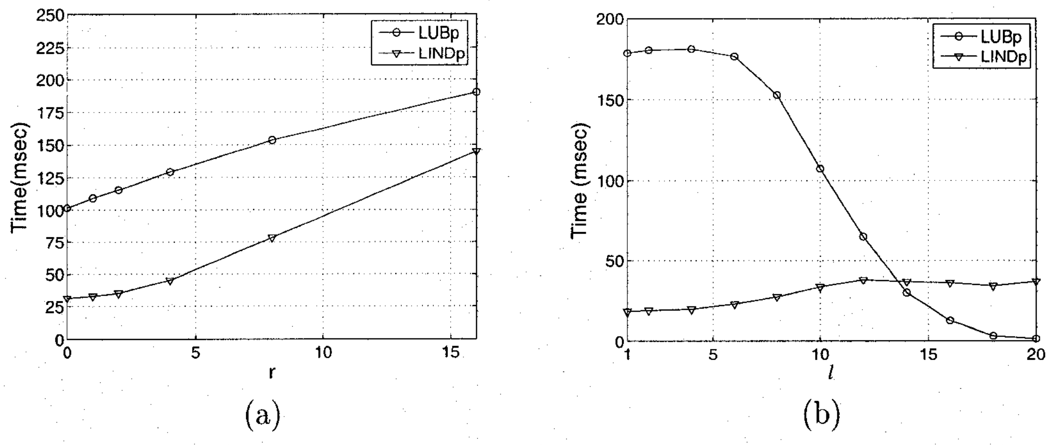


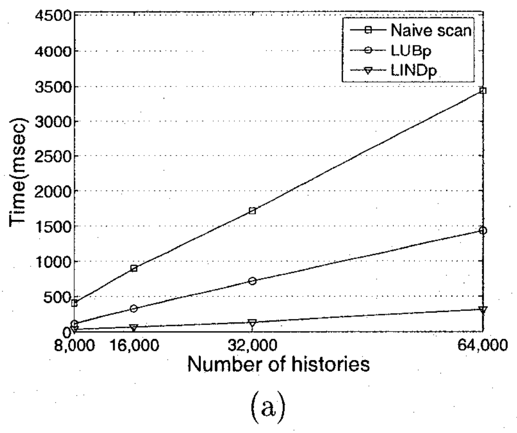
Figure 3.3: Average time for processing nearest neighbor queries varying (a) the size of the r-neighborhood (b) the desired minimum number of matches l

observation of a query increases with r, making the upper bound less tight and consequently increasing the response time for both approaches. However, LINDP is more influenced since $Usim(X, Y) \geq usim_l(X, Y)$.

Next we changed l from 1 to 20 and selected r randomly from $[0, 4]$. For each l, we randomly selected a history (from the database) with at least l observations as a query. Figure 3.3(b) compares the running time of a nearest neighbor query using LUBP and LINDP for pruning. The upper bounds employed in both methods are close to actual similarity when l is small. However, the number of histories with more than l observations decreases as l increases, which helps LUBP to reduce the number of sequential disk accesses and redundant computations. For LINDP, the number of random disk accesses (due to using an index) does not change, but the time required for computing similarity for histories not pruned increases, which justifies the observed trend.

3.5.4 Scalability Test

To compare the scalability of LUBP and LINDP to a naive scan, we increased the number of histories in the collection from 8,000 to 64,000 and measured the average query processing time for a nearest neighbor query. Both r and l were selected randomly from ranges $[1, 4]$ and $[1, 20]$ respectively. As shown in Figure 3.4(c), both LUBP and LINDP scale linearly, but the performance



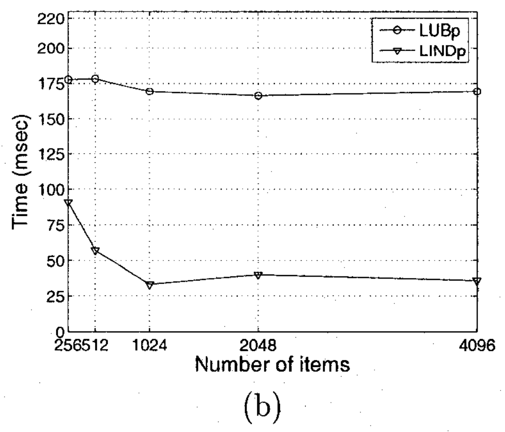


Figure 3.4: Average time for processing nearest neighbor queries varying (a) the size of the collection (b) the number of distinct items

gap between these methods and the naive scan increases with the number of histories in the database. In another experiment, we kept the number of histories fixed at $8,000$ and varied the number of items from 256 to $4,096$. Figure 3.4(d) depicts the average response time of a nearest neighbor query. Although LUBP is not significantly affected by this increase, the response time for LINDP decreases when the number of items increases from 256 to $1,024$ and remains unaffected after that. This is mainly because the dataset becomes sparse as the number of items increases. For instance, the probability that two observations, each with 10 random items, have a non-zero similarity is $1 - \prod_{i=1}^{10} \frac{247-i}{256} \approx 0.44$ for 256 items and 0.13 for $1,024$ items; LINDP takes advantage of this sparsity to reduce the query response time.

3.6 Other Related Work

Related research includes the work on detecting, representing, and querying changes. Chawathe et al. [24] propose a framework to represent changes by annotating the changed data using tags. A tag contains the type of change, a timestamp, and a reference to the modified values. Both data and annotations are modeled as nodes edges of a graph. The queries supported in this framework have the familiar *select-from-where* syntax over the annotated-graph. Chien et al. [29] represent the history of an evolving XML document using another XML document. Temporal and content-based queries are supported on the versions or the changes of XML documents. Our work differs from the aforementioned work in that we focus on similarity queries on historical data.

3.7 Conclusions

We have introduced a new domain-independent framework to both formulate and efficiently evaluate similarity queries over historical data. Our work generalizes a few concepts including the edit distance and the longest common subsequence to histories. This generalization is helpful; for instance, it enables us to find a common signature between histories based on their optimal alignments. We have developed some upper bounds for our similarity queries and one of our upper bounds has this interesting property that it makes use of an index even though it is not metric. Finding similar histories over order preserving data has many potential applications, of which we have considered historical market basket data and multi-version documents in our experiments. Our experiments on real and synthetic data confirm the effectiveness of our proposed scheme and the efficiency of our algorithms. For instance, when the minimum length of a match is provided, our algorithm achieves up to an order of magnitude speed-up over linear scan. Our contributions may be summarized as follows:

- A measure of similarity which generalizes the idea of an edit distance to histories and is useful in many practical settings. We propose the notion of an optimal alignment between two histories and an efficient dynamic programming algorithm that finds the score of an optimal alignment of any given length.

- An enumeration algorithm which finds a set of common signatures, given the length and the score of an optimal alignment for two histories. The common signature shows the common patterns that are observed in the same order in two histories, hence generalizing the concept of the longest common subsequence from character strings to histories.

- Two upper bounds that help in effective pruning of non-qualifying histories. In particular, one of the upper bounds targets the cases of sparse observations where only a small fraction of items from the set of all possible items appear in each observation; such cases are common in many real-life applications and datasets that we have been experimenting with.

Chapter 4

Similarity Search over Multi-dimensional Archival Data

In this chapter we study efficient retrieval of similar histories where each history is modeled as a multi-dimensional time series. Our framework is based on the traditional filter-and-refine paradigm. We first propose a class of history summaries for filtering and then give a finer representation of histories for refinement. Our summaries have two important properties. First, for any distance function which is formulated as an aggregation of the distances between the observations of two histories, the summaries can be used to efficiently prune 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 an approximation of histories which is finer than a summary. This approximation satisfies some notion of optimality for pruning. Our experiments show that the combination of our feature extraction approaches and the indexability of our summaries can improve upon existing methods for 2-4 dimensional histories and scales up for large databases.

This chapter is organized as follows: the next section motivates the work by discussing examples of similarity search over high-dimensional histories. Section 4.2 presents a generalization of few distance functions, commonly used for time series, that can be used to compare histories. Section 4.3 and 4.4, respectively, present our techniques to extract summaries and to derive fine-level approximations of histories. Our adaptive splitting techniques, which improve history approximations, are presented in section 4.5. We develop our algorithm to process nearest-neighbors queries in section 4.6. Performance evaluation and experimental results are reported in section 4.7. Related research is reviewed in section 4.8. Section 4.9 concludes the chapter and summarizes our contributions.

4.1 Motivating Examples

In the financial sector, the history of a stock may be tracked using indicators such as daily opening and closing prices, trading volume, etc. In health and medicine, changes to body temperature, blood pressure, heart beat rate and blood sugar may be recorded to monitor the recovery history of a patient. 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. Similarities between histories can be 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 possible 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.

4.2 Preliminaries: Distance between Histories

Let $D_{base}(\vec{x}, \vec{y})$ denote a function that measures the distance between two points $\vec{x}$ and $\vec{y}$; for instance the weighted L_p-norm distance of two points $\vec{x}$ and $\vec{y}$, defined as

$$\left(\sum_{i=1}^{d} (w_i \cdot |x_i - y_i|)^p \right)^{\frac{1}{p}} \tag{4.1}$$

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 equal and therefore numerically comparable, or to emphasize the significance of some dimensions over others. The distance of two histories $A = \langle \vec{a}_1 \ldots \vec{a}_n \rangle$ and $B = \langle \vec{b}_1 \ldots \vec{b}_m \rangle$ can be formulated as a combination of the pairwise distances of their points. A simple way to measure the distance of two histories A and B of the same length n is to aggregate the distances between their corresponding points as

$$D_p(A, B) = \left(\sum_{i=1}^{n} D_{base}(\vec{a}_i, \vec{b}_i)^p \right)^{\frac{1}{p}}. \tag{4.2}$$

Two histories may be considered similar, even if they are out-of-phase (i.e. one is shifted in time) or are of different lengths, for example, when histories are collected at different rates. The dynamic time warping [13] 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 [126]):

$$D_{dtw}(A, B) = \begin{cases} 0 & \text{if both } A \text{ and } B \text{ are empty} \\ \infty & \text{if exactly one of } A \text{ or } B \text{ is empty} \\ & \\ \multicolumn{2}{l}{D_{base}(head(A), head(B)) +} \\ \multicolumn{2}{l}{\quad min \begin{cases} D_{dtw}(A, rest(B)), \\ D_{dtw}(rest(A), B), \\ D_{dtw}(rest(A), rest(B)) \end{cases}} \\ \multicolumn{2}{l}{\quad \text{otherwise.}} \end{cases}$$

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 [13].

An alternative measure to compare two histories is their Longest Common Subsequence Score (LCSS) [119]. 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

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

Similar to DTW, a constrained variant of LCSS can be introduced. A distance function can be defined for histories based on LCSS:

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

4.3 History Summaries

Let f be a function from the domain of d-dimensional points to real values. For instance, f may transform a point to its norm. We refer to f as a *kernel function*, and use it to construct summaries from histories.

Definition 6 (HSum). *The summary of history $A = \langle \vec{a}_1, \ldots, \vec{a}_n \rangle$, with respect to a kernel function f, is a time series denoted by $hs_f(A)$ and defined as*

$$hs_f(A) = [f(\vec{a}_i)] \quad , \quad i = 1, \ldots, n. \tag{4.3}$$

The idea of HSum is illustrated in Fig.4.1 for 2-dimensional histories A and B, when the kernel function $f(\vec{x})$ returns the average value of the coordinates of vector $\vec{x}$. An HSum is actually a feature extracted from a history, in the form of a time series. Although any function can replace the kernel to extract an HSum from a history, we are interested in kernels that generate history summaries that can be used for filtering purpose without introducing false negatives. This requires that the distance of two HSums be bounded from above by the distance between their respective histories. It can be proved

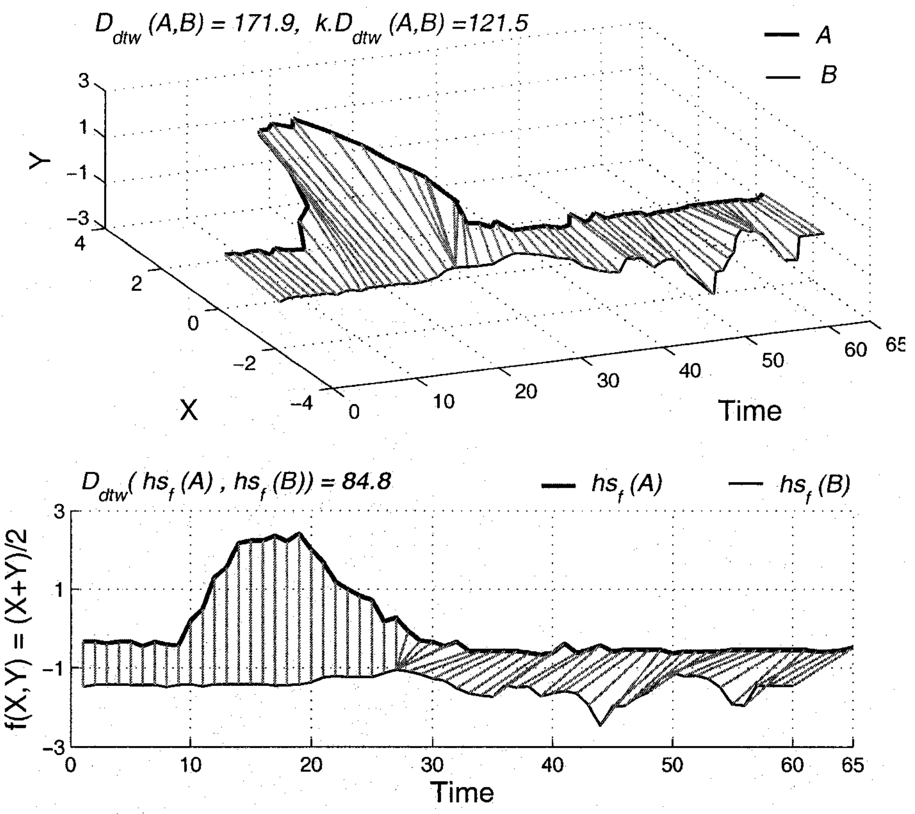


Figure 4.1: 2-d histories A and B and their HSums

by contradiction that only the class of functions that perform non-expansive mapping can be used as kernel functions.

Definition 7 (Non-Expansive Mapping). *Let ϕ_d and ϕ_1 be distance functions defined respectively in $\mathbb{R}^d$ and $\mathbb{R}$. A function f from $\mathbb{R}^d$ to $\mathbb{R}$ is a non-expansive mapping if for all $\vec{x}$ and $\vec{y}$ in $\mathbb{R}^d$*

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

where $0 < k \leq 1$ is a constant.

If the above condition is satisfied for $0 < k < 1$, then the mapping is said to be *contractive* and the constant k shows the maximum size of a contraction. For instance, in Fig.4.1, when the distance between two points is measured using L_2-norm, the average function described earlier is contractive and $k = 1/\sqrt{2}$. The choice of a kernel function and a distance function for points in $\mathbb{R}$ depends on the distance function used for points in $\mathbb{R}^d$ and its properties. For the brevity of our presentation in the rest of this chapter, we assume that the distance between two points (i.e. ϕ_d) is measured using the weighted L_p-norm, which is a metric distance. We outline two possible choices for kernel function f and corresponding distance function ϕ_1.

Weighted sum. Given a weight vector $\vec{w}$, the weighted sum of a point $\vec{x}$ is the scalar $\vec{w}^T\vec{x}$. If we assume that each weight w_i gives the importance of a variable x_i, then the weighted sum would give a collective assessment of the variables. For instance, in the case of patients, the weighted sum may give the overall condition of a patient. When $w_i = 1/d$ for $i = 1, \ldots, d$, the weighted sum becomes the average and measures the central tendency of a point. In this case, the weighted sum can be used to construct an approximation of the point in d-dimensional space and such approximation is guaranteed to be optimal, in terms of the sum of squared error. The weighted sum can also be used to classify points by a classifier such as Perceptron [101]. For our purpose, the weighted sum of a point is a kernel function that can reflect the changes along the dimensions of a history.

Lemma 6. *The weighted sum performs a non-expansive mapping from $\mathbb{R}^d$ to $\mathbb{R}$ if ϕ_1 is set to the weighted L_1-norm.*

Proof. To show that weighted sum is a non-expansive mapping, we use one of the properties of convex functions. Let g_c be a convex function defined on real numbers, i.e. for real numbers $\lambda_1, \ldots, \lambda_d$ such that $\sum_{i=1}^{d} \lambda_i = 1$, the following inequality holds

$$g_c(\sum_{i=1}^{d} \lambda_i a_i) \leq \sum_{i=1}^{d} \lambda_i g_c(a_i) \tag{4.4}$$

for any set of real numbers $a_1, \ldots, a_d$. Let $\vec{x}$ and $\vec{y}$ be arbitrary d-dimensional points. Replacing $g_c(x)$ with $|x|^p$, a convex function ($p \geq 1$), λ_i with $1/d$ for all i, and a_i with $w_i(x_i - y_i)$ in Eq.4.4 yields

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

The left hand side is equal to $\left|\frac{1}{d}\left(\vec{w}^T\vec{x} - \vec{w}^T\vec{y}\right)\right|^p$; therefore the above inequality can be written as

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

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

Metric space embeddings. Since ϕ_d is a metric, we can use a large class of metric space embeddings as kernel functions. In particular, given a reference point $\vec{r}$, the kernel can be defined as $f(\vec{x}) = \phi_d(\vec{x}, \vec{r})$ [55]. In the case of patients, the reference can be the conditions of a normal healthy person. Given two reference points, the kernel function can be defined as the projection of a point on the line that connects the two references [39]. 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 of reference points R and define the kernel function as

$$f(\vec{x}) = min_{\vec{y}\in R}\{\phi_d(\vec{x}, \vec{y})\}$$

which is a special case of Lipschitz embedding [15].

Proposition 1. *All the above functions perform a non-expansive mapping from $\mathbb{R}^d$ to $\mathbb{R}$ if ϕ_1 is set to L_1-norm.*

4.3.1 Properties of HSums

The next theorem states an interesting property of HSums, when the kernel function is a non-expansive mapping.

Theorem 1. *Let f be a non-expansive kernel function. The distance between two HSums with respect to f provides a lower-bound for the distance between their respective histories formulated as any aggregation of the distance between points of the corresponding histories.*

Proof. (Sketch) Because a non-expansive kernel reduces the distance between any two points of two histories, any aggregation of distances is also reduced. □

The property is useful for filtering histories efficiently as the distance between two HSums can be computed more efficiently, compared to the distance between two histories. For instance, in D_p, for two histories A and B of the same length n,

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

which is obtained by replacing D_{base} in Eq.4.2 with ϕ_1. Since f is non-expansive, we can rewrite the right hand side as

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

Therefore, the lower-bounding property holds for D_p since $0 < k \leq 1$. For two d-dimensional histories of the same length n, D_p can be computed in $O(dn)$ time, whereas D_p for can be computed in $O(n)$ time for their HSums. Likewise, because D_{dtw} aggregates the distances between matched points of two histories, the lower-bounding property also holds. The case for D_{lcss} is slightly different, because a threshold ϵ is used to decide if two points can be matched. We show that the *similarity* of two HSums *upper-bounds* the similarity between histories, which implies the lower-bounding property for D_{lcss}. When points $\vec{a}_i$ and $\vec{b}_j$ are matched, from the definition of S_{lcss} and the non-expansive property of f,

$$\phi_d\left(\vec{a}_i, \vec{b}_j\right) \leq \epsilon \Rightarrow \phi_1\left(f(\vec{a}_i), f(\vec{b}_j)\right) \leq k \cdot \epsilon$$

the reverse is sometimes not true. $S_{lcss}(A, B, \epsilon) \leq S_{lcss}(hs_f(A), hs_f(B), k \cdot \epsilon)$, and since the length of HSums is the same as the length of the respective histories, the lower-bounding property is established. Since HSums are time series, D_{dtw} and D_{lcss} are computed in $O(mn)$ time for HSums and in $O(dmn)$ time for the respective histories. The property holds for other distance functions such as ERP [25] and EDR [27].

4.3.2 Pruning Histories by 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. Therefore, 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. Fig.4.1 illustrates the lower bounding property for D_{dtw} and 2-d histories.

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. The amount of this distortion directly depends on the kernel function used. The next section presents a finer representation of histories; we will use this finer representation to further prune false positives that cannot be pruned based on their HSums.

4.4 A Finer Approximation of Histories

We consider approximating a history using a set of MBRs which encloses all points of the history. This representation, also commonly used for organizing spatial and spatio-temporal objects ([86, 40, 74, 51]), provides a concise abstraction for histories. The set of MBRs of a history, in general, preserves trends in individual dimensions of a 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. [74, 66, 117]). In this section, we propose a criterion to derive a finer approximation of histories optimized for pruning.

4.4.1 MBR-based Approximation of Histories

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 point t of the history at dimension r. Segment i of the history is approximated by $A_i = (s_i, e_i, \vec{l_i}, \vec{h_i})$ where

$$\begin{aligned} l_i[r] &= min\{a_t[r]\} \qquad s_i \le t \le e_i \\ h_i[r] &= max\{a_t[r]\} \qquad s_i \le t \le e_i \end{aligned}$$

for $1 \le r \le d$. A_i is a hyper-rectangle which tightly encloses all points 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 A as a set of k MBRs. Because the distance between the MBRs of two histories is a lower-bound of the distance between the two histories, finding the optimal approximation A^k can be stated as an optimization problem: *find A^k which minimizes distance approximation error.*

When the query history is provided, a straightforward approach to find A^k can be developed as a dynamic programming algorithm. However, often the splitting is performed in a pre-processing step; therefore the algorithm to derive A^k must be independent of the query. Another approach, which has been used extensively for indexing spatial and spatio-temporal objects (e.g. [51, 117, 74]), 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. Anagnostopoulos et al. [8] give a more effective solution for this problem under the assumption that queries are selected from the set of histories to be indexed. They propose a global distance-based segmentation algorithm to approximate histories aiming at preserving all pairwise distances. However, 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.

4.4.2 Our Optimality Criterion (uDAE)

Suppose the distance between two histories is measured using $D(\cdot)$. Given a query history Q, an MBR approximation A^k of history A is optimized for pruning if it minimizes the distance approximation error defined as

$$DAE(A, A^k, Q) = D(A, Q) - D(A^k, Q) \tag{4.5}$$

where $D(A^k, Q)$ is the minimum distance between Q and any history that can be approximated using A^k; therefore $D(A^k, Q)$ is a lower-bound of the true distance of A and Q. Because finding an A^k that optimizes Eq.4.5 requires knowledge of query Q, we propose an upper-bound for $DAE(\cdot)$ which can be minimized independent from Q and therefore can provide a near-optimal approximation of history A. For the brevity of our discussion, we assume that the distance between two histories is evaluated using their L_1-norm, however, our approach should be generalized to L_p-norm distances as well. Replacing $D(\cdot)$ in Eq.4.5 with L_1-norm gives

$$\sum_{j=1}^{k}\sum_{i=s_j}^{e_j} D_{base}(\vec{a_i}, \vec{q_i}) - D_{base}(A_j, \vec{q_i}) \tag{4.6}$$

where $D_{base}(A_j, \vec{q_i})$ denotes the distance of point $\vec{q_i}$ to MBR A_j. Let $N_c(\vec{x}, A_j)$ and $N_f(\vec{x}, A_j)$ be two points inside MBR A_j with respectively the closest and the farthest distances to $\vec{x}$, in terms of D_{base} (as depicted in Fig.4.2 for two points $\vec{q_i}$ and $\vec{a_i}$ and MBR A_j). For any query point $\vec{q_i}$ and any MBR A_j, the

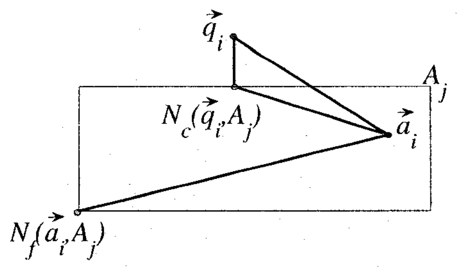


Figure 4.2: $N_c(\vec{q_i}, A_j)$ and $N_f(\vec{a_i}, A_j)$ for two points $\vec{q_i}$ and $\vec{a_i}$ and MBR A_j

metric property of D_{base} implies that

$$\begin{aligned} D_{base}(A_j, \vec{q_i}) &= D_{base}\left(N_c(\vec{q_i}, A_j), \vec{q_i}\right) \\ &\geq D_{base}(\vec{a_i}, \vec{q_i}) - D_{base}\left(N_c(\vec{q_i}, A_j), \vec{a_i}\right) \\ &\geq D_{base}(\vec{a_i}, \vec{q_i}) - D_{base}\left(N_f(\vec{a_i}, A_j), \vec{a_i}\right). \qquad (4.7) \end{aligned}$$

Note that $D_{base}\left(N_c(\vec{q_i}, A_j), \vec{q_i}\right) \geq \left|D_{base}(\vec{a_i}, \vec{q_i}) - D_{base}\left(N_c(\vec{q_i}, A_j), \vec{a_i}\right)\right|$, but because $D_{base}(\vec{a_i}, \vec{q_i})$ is not less than $D_{base}\left(N_c(\vec{q_i}, A_j), \vec{a_i}\right)$, it is safe to remove $|\cdot|$. Replacing $D_{base}(A_j, \vec{q_i})$ in Eq.4.6 with its lower-bound derived in Eq.4.7 provides an upper-bound for distance approximation error, denoted by uDAE[1] for short

$$uDAE(A, A^k) = \sum_{j=1}^{k} \sum_{i=s_j}^{e_j} D_{base}\left(N_f(\vec{a_i}, A_j), \vec{a_i}\right).$$

The same criterion can be used to derive a near-optimal splitting when the distance function is D_{dtw} and D_{lcss}; we consider D_{dtw} here and omit the argument for D_{lcss} for brevity. From Eq.4.7, the error of matching point $\vec{a_i}$ in history A with any point of history Q is at most $D_{base}\left(N_f(\vec{a_i}, A_j), \vec{a_i}\right)$. Let $n(\vec{a_i}, Q) \geq 1$ denote the number of points in history Q which are matched with point $\vec{a_i}$ in history A. An upper-bound for distance approximation error can be formulated as a weighted sum of the errors of individual matches, i.e.

$$\sum_{j=1}^{k} \sum_{i=s_j}^{e_j} n(\vec{a_i}, Q) \cdot D_{base}\left(N_f(\vec{a_i}, A_j), \vec{a_i}\right) \qquad (4.8)$$

Often a warping constraint is employed to restrict $n(\vec{a_i}, Q)$ from above to a warping range ω. This is because a full length warping is not often desired

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

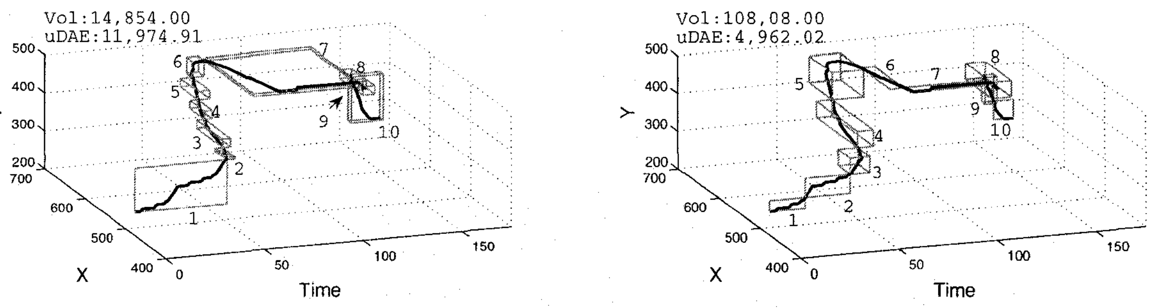


Figure 4.3: MBRs of a 2-d history; one with minimum volume (left) and one with minimum uDAE (right)

and might result into unrealistic matches [95]. Thus, $n(\vec{a}_i, Q)$ could be modeled as a discrete random number that takes its value uniformly at random from $\{1, \ldots, \omega\}$. We replace $n(\vec{a}_i, Q)$ in Eq.4.8 by $E[n(\vec{a}_i, Q)]$ to derive the expected value of Eq.4.8 as $(\frac{1+\omega}{2}) \cdot uDAE(A, A^k)$, which can be optimized for A^k independent from ω.

An interesting property of uDAE is that a near-optimal MBR approximation of a history can be derived for D_p, D_{dtw}, and D_{lcss} distance functions, independent from a query. Similar to volume, uDAE can be computed locally for each history and therefore, it is straightforward to develop a dynamic programming algorithm similar to DPSplit [51] to find optimum A^k in $O(kn^2)$ time where n is the length of history A. On the other hand, uDAE has some interesting properties, when compared to volume, as discussed next.

4.4.3 uDAE Compared to MBR Volume

Minimizing total volume, in general, does not lead to an optimal representation of a history in terms of approximation error. Fig.4.3 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, an optimal strategy with respect to total MBR volume is to assign a single MBR for the whole interval. This is because the total volume for any possible splitting of the history in that interval is the same (i.e. zero); this is shown for MBR 1 in Fig.4.3(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 more tight MBRs when uDAE is minimized in Fig.4.3(right). The same problem is observed for histories when changes happen in all dimensions but there is a big variance in the degree of the changes; this is shown for MBRs 7 and 10 in Fig.4.3(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 of histories is expected to improve both the tightness of the lower-bounds and the effectiveness of pruning, as shown in our experiments (Sec.4.7).

4.5 Adaptive Splitting of Histories

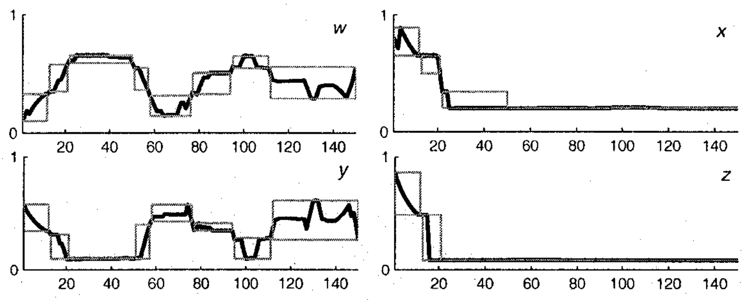
The MBR approximation presented in the previous section adopts a *fixed splitting* policy where the same splitting intervals are considered for all dimensions of a history. With a fixed splitting policy, we need to maintain for each segment, the minimum and the maximum values along each dimension, and 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 data compression techniques [129] to reduce the space requirement of A^k, hence increasing k indirectly. However, if we could find a *better* splitting of a history, without increasing k, we would also benefit from compression algorithms.

A fixed splitting policy would be a desirable property 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 [108], and there is no justification for a fixed splitting. Fixed splitting may even be ineffective, for instance, when changes are observed on a subset of the dimensions or the absolute values of the changes on several dimensions are not correlated. For instance, in our patient example, the body temperature may remain constant within a time interval while heart beat rate and blood pressure change similarly. In such cases, a fixed splitting is not a clever strategy. This motivates us to look for more adaptive and data-aware splitting strategies.

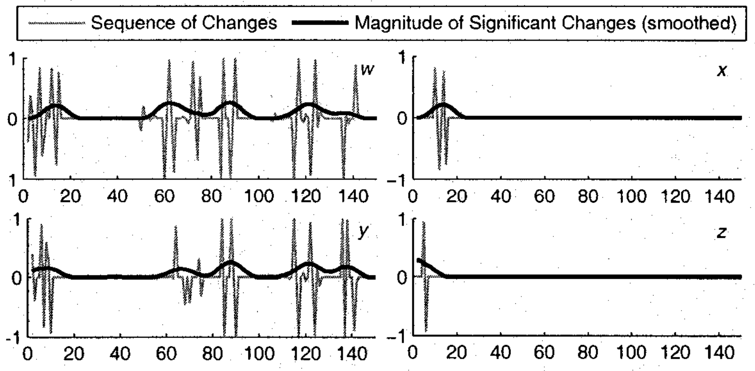
Let $\mathbb{D} = \{1, \ldots, d\}$ denote the set of dimensions and M be the space available to store the MBRs of each history. The adaptive splitting can be stated as an optimization problem: *find an approximation $\tilde{A}$ of history A, such that $\tilde{A}$ could be stored using M features and $uDAE(A, \tilde{A})$ is minimized.* An approach closely related to our adaptive splitting is Adaptive Piecewise Constant Approximation (APCA) [20] where the split points are adjusted to derive an optimal approximation of time series. However, 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, thus we can improve upon APCA, if extended to histories. Our heuristics consider the correlation between dimensions and the similarity of their change trends to improve upon an optimal fixed splitting scheme.

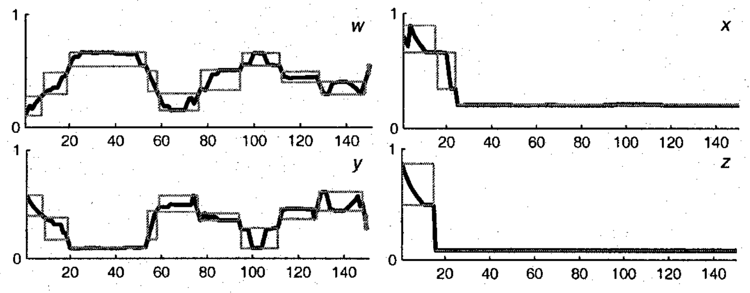
4.5.1 Variable Splitting

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 considers a different number of splitting points for each partition. Moreover, split points are placed in each partition independent from



$P_{fixed} = \{\langle \pi(A, \{w, x, y, z\}), 8 \rangle\}, uDAE(P_{fixed}) = 28.3$





$P_{vs} = \{\langle \pi(A, \{w, y\}), 10 \rangle, \langle \pi(A, \{x, z\}), 4 \rangle\}, uDAE(P_{vs}) = 24.7$

Figure 4.4: Optimal fixed splitting, change trend, and variable splitting

other partitions. We illustrate the main idea of this heuristic in Fig.4.4 where an optimal splitting of a 4-d history using 8 MBRs is shown. The change trends, or more precisely the magnitude of significant changes, depicted in Fig.4.4 (explained later in this section) indicate that dimensions $\{w, y\}$ exhibit similar change patterns. Likewise, dimensions $\{x, z\}$ have similar change trends. Our first heuristic is to examine the projections of a history on disjoint subsets of dimensions, and split each projection independently. For instance, the history shown in Fig.4.4 can be projected on two subsets $\{w, y\}$ and $\{x, z\}$, because of the similarities in change trends of the respective dimensions. We use the following definitions to formalize our variable splitting heuristic.

Definition 8 (Induced History). *Let D_i be a non-empty subset of $\mathbb{D}$. The induced history $\pi(A, D_i)$ is a history derived from history A by removing all dimensions in $\mathbb{D} - D_i$.*

Definition 9 (Variable Splitting). *Let $\{D_1, \ldots, D_p\}$ be a partitioning of the set $\mathbb{D}$, i.e. $D_i \cap D_j = \varnothing$ 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.*

With this definition, a fixed splitting becomes a special case of variable splitting where $P = \{\langle \mathbb{D}, k\rangle\}$. Each pair $\langle D_i, k_i\rangle$ specifies the number of MBRs to be allocated for induced history $\pi(A, D_i)$. The split points for each induced history have to be determined independently 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 $A(P)$, a unique approximation of history A, in $O(|P|\bar{k})$ time where $\bar{k}$ is the average number of MBRs of induced histories. The optimal variable splitting policy must minimize $uDAE(A, A(P))$, subject to space constraint, i.e.

$$\sum_{\langle D_i, k_i\rangle \in P} k_i(2|D_i| + 1) = M. \tag{4.9}$$

It should be noted that extra space must be allocated to store $\langle D_i, k_i\rangle$ information; we consider this extra space in more details in section 4.7.4.

Lemma 7. *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 claim follows. $\square$

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 dedicated for each partition must be determined. The number of possible partitioning of $\mathbb{D}$ into non-empty subsets is B_d, the bell number[2], which is equal

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

to $115,975$ for $d = 10$. Because performing an exhaustive search becomes inefficient when $d \geq 10$ for large databases, we exploit a simple heuristic which considers only one partitioning of $\mathbb{D}$. To form this partitioning, we use a clustering that groups the dimensions that are likely to benefit from the same splitting points into the same group. As the criterion of clustering, we use the similarity between the change trends of dimensions.

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. Fig.4.4 illustrates this process for a 4-d history. We used a sliding window of length 7 (resembling a week) and set $\alpha = 1.5$, as set by Vlachos et al.[120] in a similar experiment.

Finding an Optimal Assignment

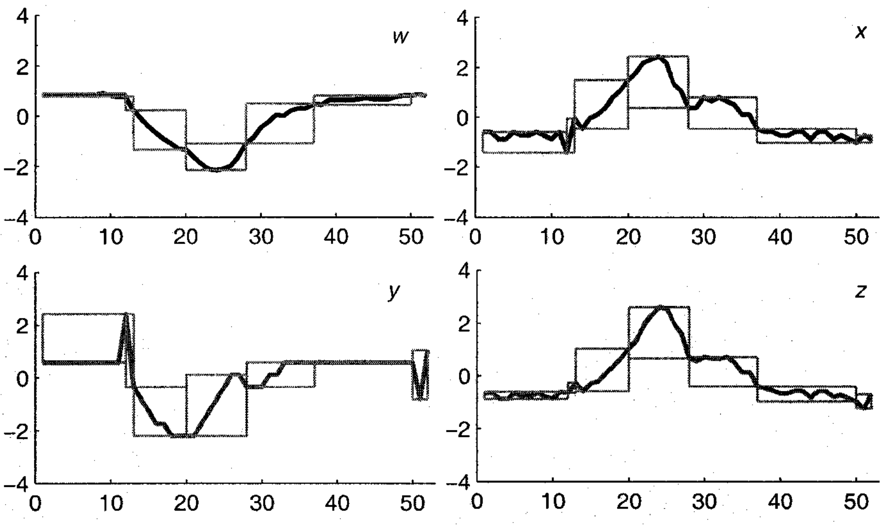
Given a partitioning $\{D_1, \ldots, D_p\}$ of $\mathbb{D}$, we want to assign the number (and the position) of splitting points for each induced history. The brute-force approach would consider all possible assignments of k_i, $i = 1, \ldots, p$, which satisfy Eq.4.9. For each assignment, a separate dynamic programming algorithm, which we refer to as DPSplit(uDAE), must be performed to find the positions of the splits. After the split points for each induced history is determined, the corresponding $A(P)$ must be constructed and $uDAE(A, A(P))$ must be evaluated to identify and keep an optimal variable splitting. Since each call to DPSPlit(uDAE) algorithm requires $O(k_i n^2)$ time, the brute-force approach is not efficient as it calls DPSplit(uDAE) once for every assignment. 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 call is required for each induced history $\pi(A, D_i)$, for k_i set to

$$\left\lfloor \frac{M - \sum_{j=1}^{p} (2|D_j| + 1)}{2|D_i| + 1} \right\rfloor \quad , \quad j \neq i \tag{4.10}$$

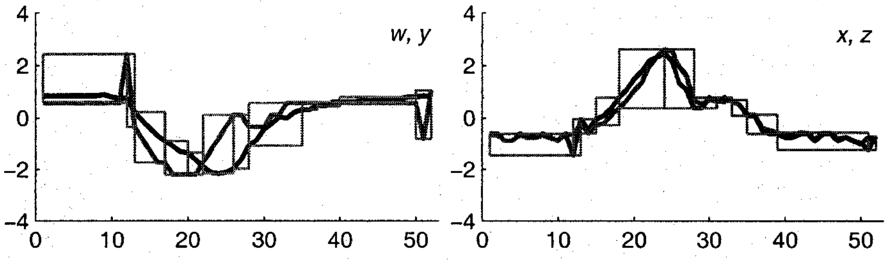
which is the maximum number of MBRs that could be allocated to $\pi(A, D_i)$, when only one MBR is assigned for each induced history $\pi(A, D_j)$, $j \neq i$. A branch-and-bound algorithm could be developed to find an optimal assignment of MBRs for a given partitioning. This, combined with our heuristic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

4.5.2 Superimposed Encoding

With a variable splitting strategy, each induced history $\pi(A, D_i)$ is approximated by a sequence of MBRs corresponding to $|D_i|$-dimensional segments. In some cases, however, there is a high similarity among the dimensions of an induced history and this similarity could be used to significantly reduce the size of encoding. Note that such similarity may have a low support over the whole dataset, therefore traditional dimensionality reduction techniques might not consider it significant. However, an adaptive splitting scheme can take advantage of such, rather local, similarity to reduce the space required to encode the MBRs of induced histories. In Fig.4.5, for instance, dimensions $\{w, y\}$ show a great degree of similarity. In order to reduce the number of required features, $\pi(A, \{w, y\})$ can be represented by a set of MBRs, where each MBR has one minimum, one maximum, and one temporal extent, as depicted in Fig.4.5(bottom and left). As a comparison, a fixed splitting of $\pi(A, \{w, y\})$ using k_i MBRs requires $5k_i$ features whereas in our superimposed encoding, only $3k_i$ features need to be kept. The saving is the result of imposing similar dimensions into one representative which is decided adaptively for different MBRs. This saving would allow us to virtually increase the number



$P_{fixed} = \{\langle\pi(A,\{w,x,y,z\}),7\rangle\}, uDAE(P_{fixed}) = 70.2$



$P_{se} = \{\langle\pi(A,\{w,y\}),13\rangle,\langle\pi(A,\{x,z\}),12\rangle\}, uDAE(P_{se}) = 55.3$

Figure 4.5: Optimal fixed splitting and superimposed encoding

of splits while keeping the space requirement the same. Increasing the number of splits is expected to produce a better approximation of the induced history. For instance, in Fig.4.5, an improvement is observed in the representation of dimension $x(z)$ in interval $[30 - 52]$.

The approach discussed for variable splitting can be easily modified to implement our superimposed encoding; since similar dimensions are represented by one dimensional MBRs, $|D_i|$ in Eq.4.9 is replaced by one to derive space constraint. The setting of maximum value for k_i in Eq.4.10 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[3].

Our idea of superimposed encoding has similarities with Skyline Bounding Regions (SBR) of Li et al. [75], but the two are different. While SBRs are built

[3]Note that two time series with similar change trends might not show a strong correlation, in general.

on multiple time series that are similar but may have no other relationships, a superimposed encoding is done on a partition, which includes similar dimensions, of a single history. Also Li et al. use the area of SBRs to find their best approximations while we use uDAE because of its advantages for pruning as discussed earlier and confirmed by our experiments.

4.6 Similarity Search for 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 HSum and uDAE-MBR approximation of data histories, whereas internal nodes are built based on HSums. Fig.4.6 gives an example of a multi-step nearest-neighbors search algorithm [108] which performs filtering using the index (line 1-5) and pruning based on uDAE-MBRs (line 8). The algorithm first retrieves k histories that have 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, for all histories H in the answer set of the original query, it must hold that $D(hs_f(H), hs_f(Q)) \leq r$; hence, the algorithm performs a range query on the index on HSums to retrieve a superset of the qualifying histories. Some false positives are pruned by comparing $\tilde{H}$ with $\tilde{Q}$ using D_{lb}, where $\tilde{H}(\tilde{Q})$ is the uDAE-MBR approximation of history $H(Q)$ and $D_{lb}(\tilde{H}, \tilde{Q})$ is a lower bound of the distance between H and Q[4]. True distance are computed to prune false positives which are not pruned using HSum index and uDAE-MBR approximations.

4.7 Experimental Evaluations

This section presents the result of an experimental evaluation of our algorithms on both real and synthetic data. First we compare the efficiency of our splittings and the quality of our generated MBRs with a related and recently proposed splitting algorithm [8]. Second, we compare our splitting algorithm

[4]$D_{lb}(\cdot)$ can be any function that lower-bounds $D(\cdot)$ e.g. [117].

```
Algorithm 2: K-NN Search
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 $f$ on $Q$ to extract $hs_f(Q)$.
   Apply adaptive splitting to obtain $\tilde{Q}$.

Search:
 (1)  Find $k$-NN of $hs_f(Q)$ using the index.
 (2)  Let $\mathcal{H}$ be all records in the result set.
 (3)  Let $r$ be the largest value of $D(H,Q)$, $H \in \mathcal{H}$.
 (4)  Perform a range search for $hs_f(Q)$ and range $r$.
 (5)  Read $\tilde{H}$ for all records in the result set.
 (6)  Initialize Topk list to the first $k$ records.
 (7)  For each $\tilde{H}$ in the result set
 (8)    If $D_{lb}(\tilde{H},\tilde{Q}) \geq$ Topk.dist
 (9)      Prune $H$
(10)    Else
(11)      Read $H$ the full history corresponding to $\tilde{H}$.
(12)      Compute $D(H,Q)$; update Topk list if required
(13)    EndIf
(14)  EndFor
```

Figure 4.6: Algorithm for k-NN search

with the traditional volume based splitting in terms of the tightness of the 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 its pruning power, running time, and scalability with database size. Experiments are performed on a machine with a single AMD/XP2600 CPU, 512MB RAM, running Red Hat Linux.

4.7.1 Datasets

Four dataset collections were used in our experimental study:

The Real1 collection contains a number of real datasets from UCR time

Table 4.1: 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

series archive[5], including those used in [8], spanning over a wide range of areas including computer networks, medicine, robotics, and random walk. Each dataset consists of 50 time series of length 512 each.

The Real2 collection consists of a few datasets of 2-4 dimensional histories. Table 4.1 provides a summary of the datasets in this group. These datasets have been used in the related work on indexing multi-dimensional time series (e.g. [117]).

The Web collection contains the history of a sample of the Web as a collection of 17-dimensional histories. We used Google Directory[6] 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 [60]. 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

[5]http://www.cs.ucr.edu/~eamonn/TSDMA/
[6]http://directory.google.com/

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.

The Synthetic dataset simulates a large archive of 17-dimensional histories, which was constructed to investigate the scalability of our approach on a large and realistic 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.

In a pre-processing step, we normalized all histories so the mean for each dimension was zero.

4.7.2 uDAE-based vs. Distance-based Splitting

We investigated how a splitting algorithm which minimizes uDAE could be compared with a related algorithm [8] which performs a global distance-based segmentation of histories, aiming at maximizing pairwise distance preservation. Since the exact solution of the proposed approach in [8] could not be applied to even a database of moderate size, due to its intractable complexity, Anagnostopoulos et al. propose a greedy solution that preserves closely the sum of pairwise distances; we refer to this approach as AVHKY and compare it with a dynamic programming algorithm based on DPSplit [51] which minimizes total uDAE for histories (our approach). To be in-line with the ex-

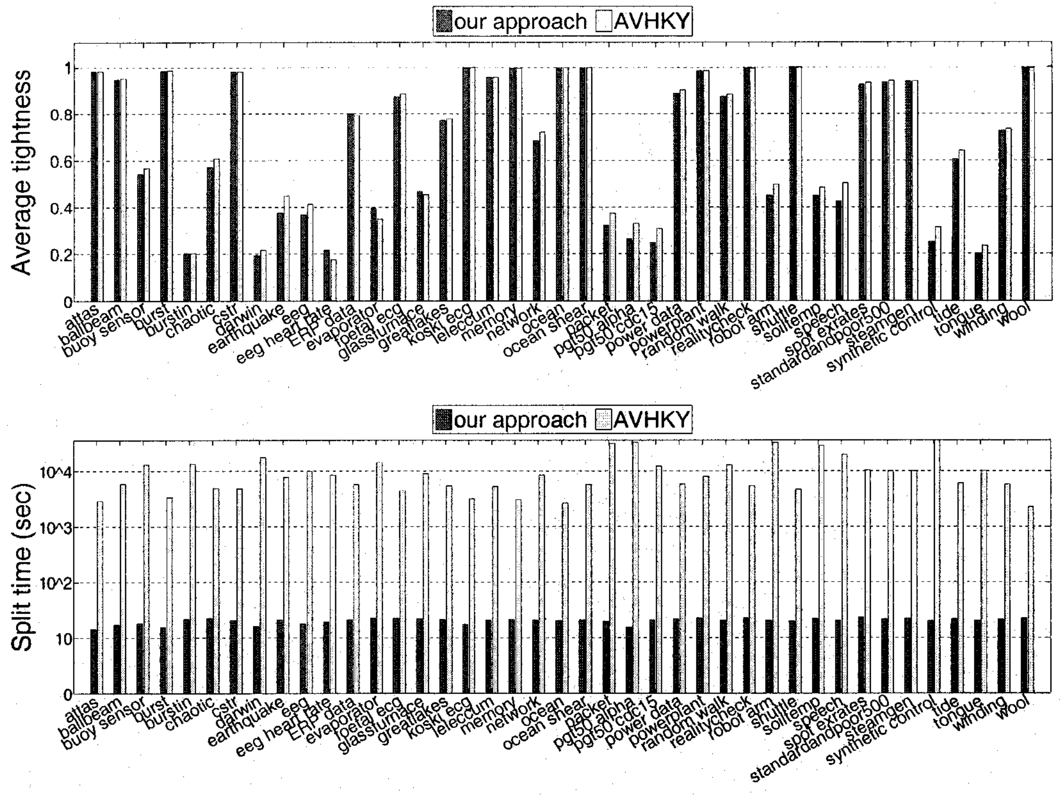


Figure 4.7: Average tightness and split time

periments performed in [8], we used the same datasets (Real1) and Euclidean distance to compare histories.

To imitate a real similarity search in which queries are not selected directly from the indexed data, we partitioned each dataset and used one-fourth of data as queries and the rest as data to be indexed. For each query, we set the number of MBRs to 10% of the length of the query. Similar to AVHKY, we set the number of MBRs for the collection to be split equal to 10% of the total sum number of observations in the collection. We wished to split each query using the same approach used to split the corresponding dataset. However, 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. For each query and data history pair, we measured tightness as the ratio of estimated distance over true distance; a tightness closer to one indicates a more distance preserving splitting.

Fig.4.7 shows average tightness and split time for Real1 collection. In general, our approach performed very close to AVHKY in terms of average tightness. However, as we expected, uDAE-based splitting had a significantly better running time. This is because uDAE is a local measure which, for each history, can be computed independently from other histories, whereas AVHKY computes the distances for the corresponding segments of all pairs of histories in the dataset in order to make a decision on merging consecutive segments of each history.

4.7.3 uDAE-based vs. Traditional Splitting

Tightness of Lower-bounds

We investigate how a uDAE-based splitting scheme could be compared with a volume-based approach [51], w.r.t. the tightness of the lower-bounds proposed in [117]. The only similar comparison, which we are aware of, is reported by Anagnostopoulos et al. [8] for time series and Euclidean distance; we considered high dimensional histories in Real2 and Web datasets with more flexible distance functions. We split each history individually; the number of splitting points for each history was set to 10% of its length. To measure Euclidean distance between histories of different lengths, we truncated the longer history at the end. 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 are suggested in [117]. Table 4.2 reports average tightness computed for fifty histories selected uniformly at random from each dataset. Even though uDAE minimizes an upper bound (instead of the exact value) of distance approximation error, we observed that for most datasets and distance functions, using uDAE made estimated distances closer to true distances, thus it is 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 for finding this number is proposed by Hadjieleftheriou et al. [52]. The main idea is to increase the number of partitions of a history and monitor the reduction in the total volume of MBR approxi-

Table 4.2: Average 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

mation and fix the number of MBRs to a point where volume reduction is no longer strong. 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 the history approximation improved with k, the number of MBRs, for Web and Word (Real2) datasets. Let v_1 and u_1 be, respectively, the total volume and 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, we derived an optimal approximation of the histories using k MBRs, where optimality was measured using total volume and total uDAE.

Fig.4.8 shows the total volume and uDAE, normalized respectively by v_1 and 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 and became zero earlier. For instance, in Web dataset, total volume was zero when as few as 30 MBRs were assigned to all histories, which means that increasing the number of MBRs beyond this point does not increase the accuracy of the approximation. In contrary, uDAE was much higher which indicates that increasing the number of MBRs beyond 30 resulted in a better approximation and more pruning. While increasing the numbers of MBRs beyond 30 did not reduce total volume in Web dataset, it reduced total uDAE and resulted into more distance preserving MBRs.

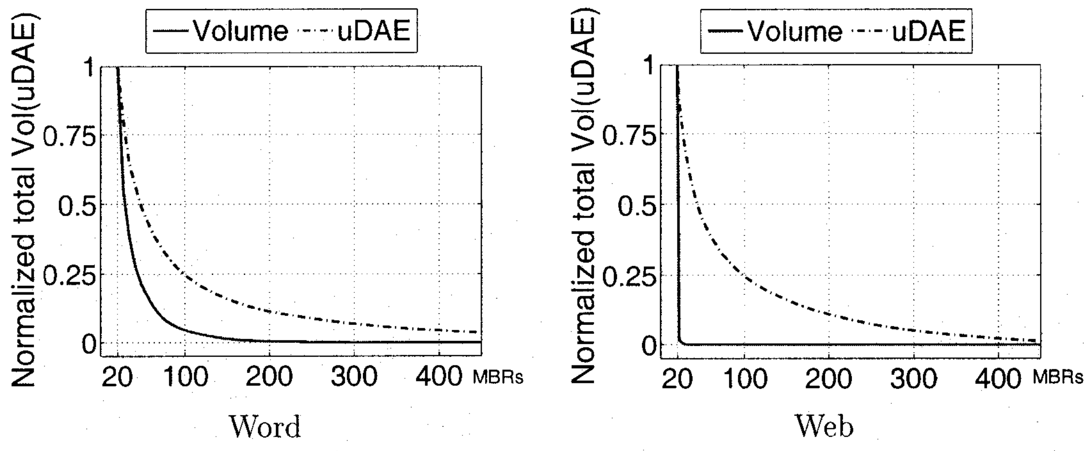


Figure 4.8: Sensitivity of volume and uDAE to the Number of MBRs

4.7.4 Effectiveness of Adaptive Splitting

We measured average uDAE reduction for high-dimensional histories when our heuristics are employed. For each d-dimensional 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. We implemented an adaptive splitting scheme given the same amount of space as M and measured uDAE reduction compared to optimal fixed splitting. Since finding an optimal adaptive splitting is computationally expensive (section 4.5.1), an approximate adaptive splitting is used to divide the dimensions of each history into n_p partitions. To store k_i MBRs of induced history $\pi(A, D_i)$, where A is a history and D_i is a partitioning of its dimensions, we allocated $k_i(2|D_i|n_m + n_t) + n_d$ bytes in Variable Splitting (VS) and $k_i(2n_m + n_t) + n_d$ bytes in Superimposed Encoding (SE); here n_d is the extra space required to store (D_i, k_i). We set $n_m = 4$, $n_t = 1$, and $n_d = 2$ bytes.

Table 4.3 reports average uDAE reduction over optimal fixed splitting for Web dataset when VS and SE were used and n_p varied from 2 to 5. Although approximate adaptive splitting was used, our heuristics were still effective and improved upon optimal fixed splitting. In particular, more reduction was observed for SE; after examining this dataset we found that for affected histories,

Table 4.3: Average uDAE reduction for adaptive splitting over optimal fixed splitting for Web dataset

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 4.4: Average tightness of lower bounds for Web dataset

	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

there were two or more similar dimensions which were grouped together in SE, but not in VS.

We also measured the effect of adaptive splitting on the tightness of lower bounds for the Web dataset; Table 4.4 reports the results. As we expected, not only uDAE-based fixed splitting improved the lower bounds over the volume based scheme, taking advantage of the redundancy and the sparseness present in Web dataset, our adaptive splitting could improve up to 25%(33%) upon uDAE-(volume-) based fixed splitting, which confirms the effectiveness of adaptive splitting.

4.7.5 Performance Evaluation

We compared our algorithm with the framework proposed in [?], 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. Fig.4.9 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 to answer each query.

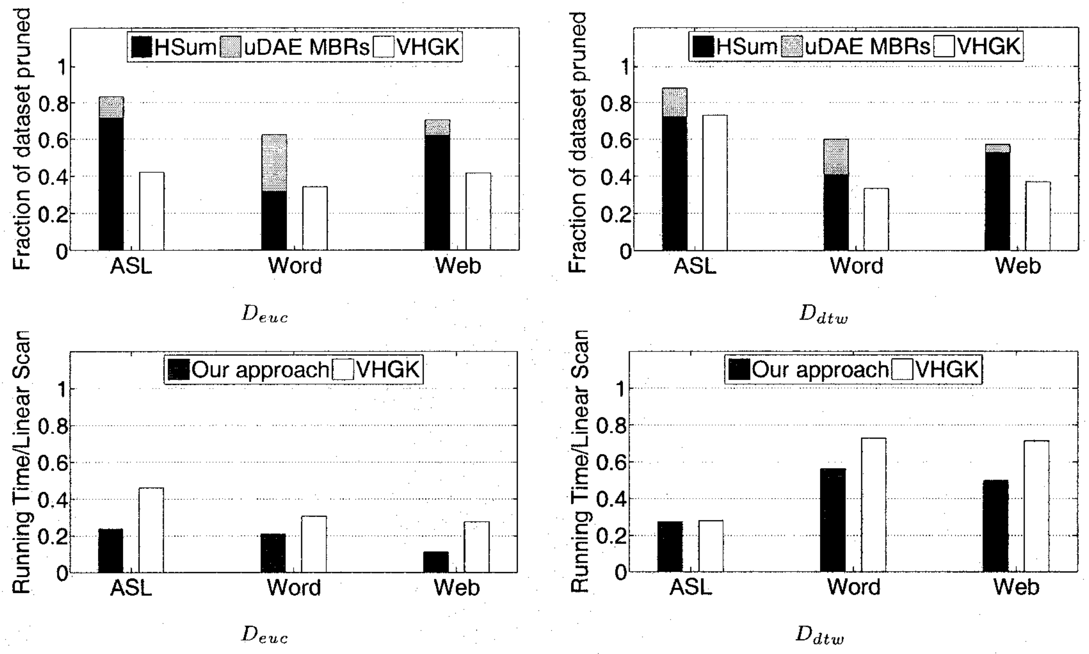
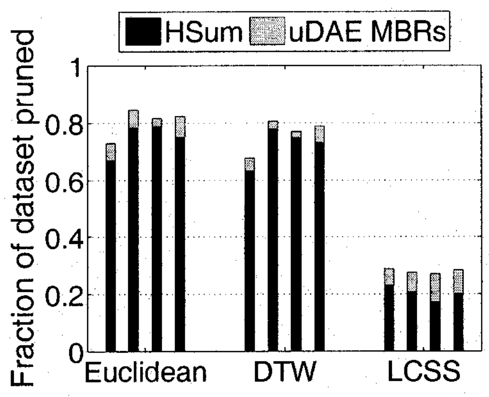


Figure 4.9: Pruning and relative query processing time averaged for fifty 10-NN queries

4.7.6 Scalability Test

We measured the performance of our algorithm over 17-dimensional histories. To the best of our knowledge, no experiment has been reported on efficiently retrieving histories with more than 4 dimensions, using Euclidean distance, DTW, and LCSS as distance functions. Therefore, we compared our approach to the only candidate, i.e. naive scan. We used synthetic datasets with $1k$, $2k$, $4k$, and $8k$ histories constructed from our Web dataset as discussed in section 4.7.1. Fig.4.10 reports pruning and relative query processing time over linear scan averaged over fifty 10-NN queries. Our synthetic data included pairs of histories where one was formed after a random permutation of the dimensions of the other. Pruning these histories using HSum was a challenge because average is not sensitive to the order of dimensions; thus random permutation generated dissimilar histories with equal HSums. However, still our approach had a strong pruning power for a wide range of database sizes, which made it superior to linear scan and scalable with database size.



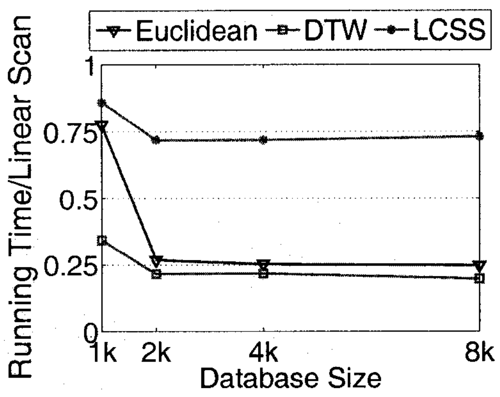


Figure 4.10: Average pruning and relative query processing time; bars for each distance from left to right are for synthetic data of 1k, 2k, 4k and, 8k histories.

4.8 Related Work

Several approaches have been proposed to extract features from time series, including DFT [2], DWT [93], and APCA [20] for Euclidean distance, PAA [125] for arbitrary L_p-norm, and edit distance with real penalty [25]. In each approach, the features could be used to estimate one type of distance function, unlike our HSum, as it was shown in Theorem 1. Compared to PAA and the work in [25], the length of an HSum is the same as the length of the history it represents, hence, it can better preserve trends. Compared to the work in [20], the APCA representation of a query needs to be computed for every history that is compared with the query while in our approach, one representation of the query is required.

4.9 Conclusion

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 historical datasets of high dimensionalities for fixed and adaptive splitting policies. Our contributions may be summarized as follows:

- Techniques for extracting compact HSums from histories with some interesting properties: (1) for a large class of distance functions, 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 histories that are far from a query; (3) HSums can be indexed, hence the pruning and retrievals can be done efficiently.

- 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 distance function, and (2) it resolves some of the limitations of previous MBR-based techniques for approximating histories in higher dimensions.

- Two adaptive splitting strategies to further improve the tightness of uDAE-based MBR approximations while keeping the same space requirements. To adjust the splitting policy, our heuristics targets cases where there is a large difference, in terms of the degree of changes, between dimensions or where two or more dimensions are correlated and it might be possible to reduce the dimensionality of uDAE-based MBRs without increasing the maximum distance estimation error.

Chapter 5

Conclusions

In this chapter the main points of the thesis are summarized, our contributions are highlighted, and possible directions of future work are proposed.

5.1 Summary

In general, our work addresses the problem of efficiently retrieving similar histories which are in the form of either historical market-basket data or multi-dimensional time series. We have introduced a new domain-independent framework to both formulate and efficiently evaluate similarity queries over historical market-basket data. Our work generalizes a few concepts including the edit distance and the longest common subsequence to histories. This generalization is helpful; for instance, it enables us to find a common signature between histories based on their optimal alignments. Our experiments on real and synthetic data confirm the effectiveness of our proposed scheme and the efficiency of our algorithms.

We have also addressed the problem of efficiently evaluating similarity queries on high-dimensional histories, by developing techniques for finding summaries of histories at different levels of detail. We have investigated the use of these summaries for indexing and pruning. Our techniques for deriving summaries make use of a kernel function that maps a d-dimensional point to a real number. We have identified a class of functions that can be used as kernels; hence a spectrum of summaries can be obtained.

We have developed uDAE as a measure of the tightness of history approximations and proposed adaptive splitting strategies to further improve the tightness of uDAE-based MBR approximations and to resolve some of the limitations of previous MBR-based techniques for approximating histories in higher dimensions. Our experiments show that (1) uDAE-based MBR approximations give a more accurate estimate of the true distances, compared to a traditionally used volume-based scheme, (2) uDAE-based MBR approximations are comparable, in terms of distance preservation, to a related approach [8] but are at least two orders of magnitude faster to derive. This makes our approach applicable to large databases.

5.2 Contributions

Our contributions can be listed as follows:

- A domain-independent measure of similarity, using optimal conditional alignment, which generalizes the idea of an edit distance to histories.

- An efficient enumeration algorithm to find a set of common patterns that are observed in the same order in two histories, given the length and the score of an optimal conditional alignment. This is a generalization of the concept of the longest common subsequence to histories.

- Techniques for extracting compact summaries from histories and the observation that the summarization is independent of the distance function that may be used. Therefore, for a large class of distance functions, the same distance function that operates on histories can be computed more efficiently using summaries, under-estimating the true distance of the histories.

- Proposing the uDAE-based MBR approximations of histories and investigating whether such approximations are more effective for pruning than the traditionally proposed volume based MBRs. An interesting observation is that uDAE-based MBR approximations can be derived independently from queries and that the same representation can provide a near optimal approximation of histories, in terms of distance estimation error, for three commonly used distance functions: L_p-norm, dynamic time warping, and longest common subsequence.

- An adaptive splitting scheme to improve the tightness of history approximations; we observe that variable splitting and superimposed encoding can improve MBR-approximations of high-dimensional histories. This is the case in real-life applications where often there is a correlation between the dimensions of histories or there is a large difference, in terms of the degree of changes, between the dimensions.

- Integration within a filter-and-refine framework of our similarity measures, lower-bounds, summarization and feature extraction techniques, and algorithms to support exact similarity queries on historical market-basket data and multi-dimensional histories. Our extensive experiments confirm the naturalness of our similarity measure, the advantages of our feature extraction approaches, and the efficiency and scalability of our algorithms on both real and synthetic datasets.

5.3 Future Work

- We have considered nearest-neighbors and range queries in this thesis. All-pair queries is another interesting problem with many potential applications in clustering and data cleaning. Related work includes the work of Ramasamy et al. [98] on set containment joins, that of Mamoulis [77] on set equality, containment, and overlap joins, and the work of Chaudhuri et al. [22] on string similarity join for several similarity measures including the Jaccard coefficient, the Edit distance, and the hamming distance. One straightforward approach to process all-pair queries on histories is to perform a range query for all histories in the database. Future work may investigate improvements over this base, for instance, using the summaries and the indexes we have proposed for nearest-neighbor and range queries.

- 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.

- We have observed that history summaries and uDAE-based MBR approximations are useful to derive compact features for multi-dimensional histories. These techniques may have applications in summarizing historical market-basket data as well. For instance, instead of presenting the history of changes of a web page, we could divide the history into some meaningful segments and present these episodes. Generalizing HSum and uDAE MBRs to historical market-basket data is an open future work.

- For the problem of efficiently retrieving histories, there are very few works that can scale up to 3- or 4- dimensional histories; our work opens up the door for further research in this area.

Bibliography

[1] C. C. Aggarwal, J. L. Wolf, and P. S. Yu. A new method for similarity indexing of market basket data. In *ACM SIGMOD Conference*, 1999.

[2] R. Agrawal, C. Faloutsos, and A. N. Swami. Efficient similarity search in sequence databases. In *International Conference of Foundations of Data Organization and Algorithms (FODO)*, 1993.

[3] 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 *Very Large Data Bases (VLDB) Conference*, pages 490–501, 1995.

[4] R. Agrawal, G. Psaila, E. L. Wimmers, and M. Zaıt. Querying shapes of histories. In *Very Large Data Bases (VLDB) Conference*, 1995.

[5] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *Very Large Data Bases (VLDB) Conference*, 1994.

[6] R. Agrawal and R. Srikant. Mining sequential patterns. In *International Conference on Data Engineering (ICDE)*, pages 3–14, 1995.

[7] S. Altschul, W. Gish, W. Miller, E.Myers, and D. Lippman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, 1990.

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

[9] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander. Optics: ordering points to identify the clustering structure. In *ACM SIGMOD Conference*, 1999.

[10] V. Athitsos. Learning embeddings for indexing, retrieval, and classification with applications to object and shape recognition in image databases. PhD thesis, University of Boston, 2006.

[11] J. Bather. *Decision Theory: An Introduction to Dynamic Programming and Sequential Decisions.* John Wiley & Sons, 2000.

[12] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):509–522, 2002.

[13] D. J. Berndt and J. Clifford. Using dynamic time warping to find patterns in time series. In *KDD Workshop*, pages 359–370, 1994.

[14] K. S. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is "nearest neighbor" meaningful? In *International Conference on Database Theory (ICDT)*, 1999.

[15] J. Bourgain. On lipschitz embedding of finite metric spaces in hilbert space. *Israel Journal of Mathematics*, 52(1-2):46–52, 1985.

[16] T. Bozkaya and Z. Meral Özsoyoglu. Indexing large metric spaces for similarity search queries. *ACM Transaction on Database Systems (TODS)*, 24(3):361–404, 1999.

[17] T. Bozkaya, N. Yazdani, and Z. M. Özsoyoglu. Matching and indexing sequences of different lengths. In *International Conference on Information and Knowledge Management (CIKM)*, 1997.

[18] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher. Min-wise independent permutations (extended abstract). In *Symposium on the Theory of Computing (STOC)*, pages 327–336, 1998.

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

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

[21] H. S. Chang, S. Sull, and S. U. Lee. Efficient video indexing scheme for content-based retrieval. *IEEE Transactions on Circuits and Systems for Video Technology*, 9(8):1269–1279, 1999.

[22] S. Chaudhuri, V. Ganti, and R. Kaushik. A primitive operator for similarity joins in data cleaning. In *International Conference on Data Engineering (ICDE)*, 2006.

[23] E. Chavez, G. Navarro, R. Baeza-Yates, and J. L. Marroquin. Searching in metric spaces. *ACM Computing Surveys*, 33(3), 2001.

[24] S. S. Chawathe, S. Abiteboul, and J. Widom. Representing and querying changes in semistructured data. In *International Conference on Data Engineering (ICDE)*, pages 4–13, 1998.

[25] L. Chen and R. T. Ng. On the marriage of lp-norms and edit distance. In *Very Large Data Bases (VLDB) Conference*, 2004.

[26] L. Chen, M. T. Özsu, and V. Oria. Symbolic representation and retrieval of moving object trajectories. In *ACM SIGMM Workshop on Multimedia information retrieval*, 2004.

[27] L. Chen, M. Tamer Özsu, and V. Oria. Robust and fast similarity search for moving object trajectories. In *ACM SIGMOD Conference*, 2005.

[28] S. S. Cheung and A. Zakhor. Efficient video similarity measurement with video signature. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(1):59–74, 2003.

[29] S. Y. Chien, V. J. Tsotras, and C. Zaniolo. Efficient management of multiversion documents by object referencing. In *Very Large Data Bases (VLDB) Conference*, pages 291–300, 2001.

[30] K. K. W. Chu and M. H. Wong. Fast time-series searching with scaling and shifting. In *Symposium on Principles of Database Systems (PODS)*, 1999.

[31] P. Ciaccia, M. Patella, and P. Zezula. M-tree: An efficient access method for similarity search in metric spaces. In *Very Large Data Bases (VLDB) Conference*, pages 426–435, 1997.

[32] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transaction on Information Theory*, IT-11:21–27, 1967.

[33] G. Das, D. Gunopulos, and H. Mannila. Finding similar time series. In *European Conference on Principles of Data Mining and Knowledge Discovery (PKDD)*, 1997.

[34] DBLP. Digital Bibliography and Library Project. http://www.informatik.uni-trier.de/ ley/db/.

[35] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern classification*. Wiley-Interscience, second(sub) edition, 2000.

[36] C. E. Dyreson, H. l. Lin, and Y. Wang. Managing versions of web documents in a transaction-time web server. In *World Wide Web (WWW) Conference*, pages 422–432, 2004.

[37] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *ACM SIGKDD Conference*, 1996.

[38] C. Faloutsos, H. Jagadish, A. Mendelzon, and T. Milo. A signature technique for similarity-based queries. In *Proceedings of the Compression and Complexity of Sequences*, page 2, 1997.

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

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

[41] M. Farach. Optimal suffix tree construction with large alphabets. In *Annual Symposium on Foundations of Computer Science (FOCS)*, 1997.

[42] R. F. Santos Filho, A. J. M. Traina, C. Traina Jr., and C. Faloutsos. Similarity search without tears: the omni family of all-purpose access methods. In *International Conference on Data Engineering (ICDE)*, 2001.

[43] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. Query by image and video content: the qbic system. *IEEE Computer*, 28(9), 1995.

[44] A. W.-C. Fu, E. J. Keogh, L. Y. H. Lau, and C. Ratanamahatana. Scaling and time warping in time series querying. In *Very Large Data Bases (VLDB) Conference*, 2005.

[45] L. Gao and X. S. Wang. Continually evaluating similarity-based pattern queries on a streaming time series. In *ACM SIGMOD Conference*, 2002.

[46] A. Gionis, D. Gunopulos, and N. Koudas. Efficient and tunable similar set retrieval. In *ACM SIGMOD Conference*, 2001.

[47] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *Very Large Data Bases (VLDB) Conference*, pages 518–529, 1999.

[48] 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 (CP)*, 1995.

[49] D. Gusfield. *Algorithms on strings, trees and sequences: computer science and computational biology.* Cambridge University Press, first edition, 1997.

[50] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *ACM SIGMOD Conference*, pages 47–57, 1984.

[51] M. Hadjieleftheriou, G. Kollios, V. J. Tsotras, and D. Gunopulos. Efficient indexing of spatiotemporal objects. In *International Conference on Extending Database Technology (EDBT)*, 2002.

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

[53] J. Han, X. Yan, and P. S. Yu. Mining, indexing, and similarity search in graphs and complex structures. In *International Conference on Data Engineering (ICDE)*, 2006.

[54] G. Hirstescu and M. Farach-Colton. Cluster-preserving embeding of proteins. Technical Rep. 99-50, CS Department, Rutgers University, 1999.

[55] G. R. Hjaltason and H. Samet. Properties of embedding methods for similarity searching in metric spaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5):530–549, 2003.

[56] Y.-W. Huang and P. S. Yu. Adaptive query processing for time-series data. In *ACM SIGKDD Conference*, 1999.

[57] E. Hunt, M. P. Atkinson, and R. W. Irving. Database indexing for large DNA and protein sequence collections. 11(3):256–271, 2002.

[58] P. Indyk, G. Iyengar, and N. Shivakumar. Finding pirated video sequences on the internet. Technical Report, Stanford Inforlab, 1999.

[59] P. Indyk, N. Koudas, and S. Muthukrishnan. Identifying representative trends in massive time series data sets using sketches. In *Very Large Data Bases (VLDB) Conference*, 2000.

[60] Internet Archive. http://www.archive.org. As of 2004.

[61] H. V. Jagadish. A retrieval technique for similar shapes. In *ACM SIGMOD Conference*, 1991.

[62] H. V. Jagadish, A. O. Mendelzon, and T. Milo. Similarity-based queries. In *Symposium on Principles of Database Systems (PODS)*, 1995.

[63] L. Jin, N. Koudas, C. Li, and A. K. H. Tung. Indexing mixed types for approximate retrieval. In *Very Large Data Bases (VLDB) Conference*, 2005.

[64] T. Kahveci, A. K. Singh, and A. Gürel. Similarity searching for multi-attribute sequences. In *International Conference on Statistical and Scientific Database Management (SSDBM)*, 2002.

[65] I. Kamel and C. Faloutsos. On packing r-trees. In *International Conference on Information and Knowledge Management (CIKM)*, 1993.

[66] E. J. Keogh. Exact indexing of dynamic time warping. In *Very Large Data Bases (VLDB) Conference*, 2002.

[67] E. J. Keogh, K. Chakrabarti, S. Mehrotra, and M. J. Pazzani. Locally adaptive dimensionality reduction for indexing large time series databases. In *ACM SIGMOD Conference*, 2001.

[68] E. J. Keogh and M. J. Pazzani. An indexing scheme for fast similarity search in large time series databases. In *International Conference on Statistical and Scientific Database Management (SSDBM)*, 1999.

[69] E. J. Keogh and M. J. Pazzani. Scaling up dynamic time warping for datamining applications. In *ACM SIGKDD Conference*, 2000.

[70] E. J. Keogh and P. Smyth. A probabilistic approach to fast pattern matching in time series databases. In *ACM SIGKDD Conference*, 1997.

[71] S.-W. Kim, S. Park, and W. W. Chu. An index-based approach for similarity search supporting time warping in large sequence databases. In *International Conference on Data Engineering (ICDE)*, 2001.

[72] F. Korn, H. V. Jagadish, and C. Faloutsos. Efficiently supporting ad hoc queries in large datasets of time sequences. In *ACM SIGMOD Conference*, 1997.

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

[74] S.-L. Lee, S.-J. Chun, D.-H. Kim, J.-H. Lee, and C.-W. Chung. Similarity search for multidimensional data sequences. In *International Conference on Data Engineering (ICDE)*, 2000.

[75] Q. Li, I. F. V. López, and B. Moon. Skyline index for time series data. *IEEE Transactions on Knowledge and Data Engineering*, 16(6):669–684, 2004.

[76] Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Pranav Patel. Finding motifs in time series. In *ACM SIGKDD Workshop on Temporal Data Mining*, 2002.

[77] N. Mamoulis. Efficient processing of joins on set-valued attributes. In *ACM SIGMOD Conference*, 2003.

[78] C. Meek, J. M. Patel, and S. Kasetty. Oasis: An online and accurate technique for local-alignment searches on biological sequences. In *Very Large Data Bases (VLDB) Conference*, 2003.

[79] V. Megalooikonomou, Q. Wang, G. Li, and C. Faloutsos. A multiresolution symbolic representation of time series. In *International Conference on Data Engineering (ICDE)*, 2005.

[80] Y.-S. Moon, K.-Y. Whang, and W.-K. Loh. Duality-based subsequence matching in time-series databases. In *International Conference on Data Engineering (ICDE)*, 2001.

[81] D. R. Morrison. PATRICIA - practical algorithm to retrieve information coded in alphanumeric. *Journal of the ACM*, 15(4):514–534, 1968.

[82] M. E. Munich and P. Perona. Continuous dynamic time warping for translation-invariant curve alignment with applications to signature verification. In *IEEE International Conference on Computer Vision (ICCV)*, 1999.

[83] G. Navarro. A guided tour to approximate string matching. *ACM Computing Surveys*, 33(1):31–88, 2001.

[84] A.V. Oppenheim and R.W. Schafer. *Digital Signal Processing.* Prentice-Hall, first edition, 1975.

[85] K. p. Chan and A. W.-C. Fu. Efficient time series matching by wavelets. In *International Conference on Data Engineering (ICDE)*, 1999.

[86] B.-U. Pagel, H.-W. Six, H. Toben, and P. Widmayer. Towards an analysis of range query performance in spatial data structures. In *Symposium on Principles of Database Systems (PODS)*, 1993.

[87] D. Papadias, Y. Tao, P. Kalnis, and J. Zhang. Indexing spatio-temporal data warehouses. In *International Conference on Data Engineering (ICDE)*, 2002.

[88] A. Papoulis. *Probability, random variables, and stochastic processes.* McGraw-Hill, third edition, 1991.

[89] S. Park, W. W. Chu, J. Yoon, and C. Hsu. Efficient searches for similar subsequences of different lengths in sequence databases. In *International Conference on Data Engineering (ICDE)*, 2000.

[90] W.R. Pearson. Rapid and sensitive sequence comparison with fastp and fasta. *Methods in Enzymology*, 183:63–98, 1990.

[91] C.-S. Perng, H. Wang, S. R. Zhang, and D. S. Parker Jr. Landmarks: a new model for similarity-based pattern querying in time series databases. In *International Conference on Data Engineering (ICDE)*, 2000.

[92] D. Pfoser, C. S. Jensen, and Y. Theodoridis. Novel approaches in query processing for moving object trajectories. In *Very Large Data Bases (VLDB) Conference*, 2000.

[93] I. Popivanov and R. J. Miller. Similarity search over time-series data using wavelets. In *International Conference on Data Engineering (ICDE)*, 2002.

[94] Y. Qu, C. Wang, and X. S. Wang. Supporting fast search in time series for movement patterns in multiple scales. In *International Conference on Information and Knowledge Management (CIKM)*, 1998.

[95] L. Rabiner and B.-H. Juang. *Fundamentals of speech recognition.* Prentice Hall, 1993.

[96] D. Rafiei and A. O. Mendelzon. Efficient retrieval of similar time sequences using dft. In *International Conference of Foundations of Data Organization and Algorithms (FODO)*, 1998.

[97] D. Rafiei and A.O. Mendelzon. Similarity-based queries for time series data. In *ACM SIGMOD Conference*, pages 13–25, 1997.

[98] K. Ramasamy, J. M. Patel, J. F. Naughton, and R. Kaushik. Set containment joins: the good, the bad and the ugly. In *Very Large Data Bases (VLDB) Conference*, 2000.

[99] C. Ratanamahatana, E. J. Keogh, A. J. Bagnall, and S. Lonardi. A novel bit level time series representation with implication of similarity search and clustering. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, 2005.

[100] C. J. V. Rijsbergen. *Information Retrieval.* Butterworths, second edition, 1979.

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

[102] S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5000):2323–2326, 2000.

[103] Y. Sakurai, S. Papadimitriou, and C. Faloutsos. Braid: stream mining through group lag correlations. In *ACM SIGMOD Conference*, 2005.

[104] Y. Sakurai, M. Yoshikawa, and C. Faloutsos. Ftw: fast similarity search under the time warping distance. In *Symposium on Principles of Database Systems (PODS)*, 2005.

[105] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[106] B. Salzberg and V. J. Tsotras. Comparison of access methods for time-evolving data. *ACM Computing Surveys*, 31(2):158–221, 1999.

[107] D. Sankoff and J. Kruskal. *Time warps, string edits, and macromolecules, the theory and practice of sequence comparison.* Addison-Wesley, 1983.

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

[109] T. K. Sellis, N. Roussopoulos, and C. Faloutsos. The r+-tree: A dynamic index for multi-dimensional objects. In *The VLDB Journal*, pages 507–518, 1987.

[110] U. Shaft and R. Ramakrishnan. Theory of nearest neighbors indexability. *ACM Transaction on Database Systems (TODS)*, 31(3):814–838, 2006.

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

[112] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197, 1981.

[113] A. Strehl and J. Ghosh. Value-based customer grouping from large retail data-sets. In *SPIE Conference on Data Mining and Knowledge Discovery*, volume 4057, 2000.

[114] Y. Tao and D. Papadias. Mv3r-tree: A spatio-temporal access method for timestamp and interval queries. In *Very Large Data Bases (VLDB) Conference*, 2001.

[115] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5000):2319–2323, 2000.

[116] J. S. Varre, J. P. Delahaye, and E. Rivals. Transformation distances: a family of dissimilarity measures based on movements of segments. *Bioinformatics*, 15(3):194–202, 1999.

[117] M. Vlachos, M. Hadjieleftheriou, D. Gunopulos, and E. Keogh. Indexing multi-dimensional time-series with support for multiple distance measures. In *ACM SIGKDD Conference*, pages 216–225, 2003.

[118] M. Vlachos, M. Hadjieleftheriou, D. Gunopulos, and E.J. Keogh. Indexing multidimensional time-series. *The VLDB Journal*, 15(1):1–20, 2006.

[119] M. Vlachos, G. Kollios, and D. Gunopulos. Discovering similar multidimensional trajectories. In *International Conference on Data Engineering (ICDE)*, pages 673–684, 2002.

[120] M. Vlachos, C. Meek, Z. Vagena, and D. Gunopulos. Identifying similarities, periodicities and bursts for online search queries. In *ACM SIGMOD Conference*, 2004.

[121] H. Wang, C. S.Perng, W. Fan, S. Park, and P. S. Yu. Indexing weighted-sequences in large databases. In *International Conference on Data Engineering (ICDE)*, pages 25–36, 2003.

[122] R. Weber, H.-J. Schek, and S. Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *Very Large Data Bases (VLDB) Conference*, 1998.

[123] L. Wei, E. J. Keogh, H. V. Herle, and A. Mafra-Neto. Atomic wedgie: efficient query filtering for streaming times series. In *IEEE International Conference on Data Mining (ICDM)*, 2005.

[124] Huanmei Wu, Betty Salzberg, and Donghui Zhang. Online event-driven subsequence matching over financial data streams. In *ACM SIGMOD Conference*, 2004.

[125] B.-K. Yi and C. Faloutsos. Fast time sequence indexing for arbitrary lp norms. In *Very Large Data Bases (VLDB) Conference*, 2000.

[126] B.-K. Yi, H. V. Jagadish, and C. Faloutsos. Efficient retrieval of similar time sequences under time warping. In *International Conference on Data Engineering (ICDE)*, 1998.

[127] P. N. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *Symposium on Discrete Algorithms (SODA)*, pages 311–321, 1993.

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

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