Physiological Time Series Retrieval and Prediction with Locality-Sensitive Hashing

by

Yongwook Bryce Kim

Sc.B., Brown University (2005)
S.M., Massachusetts Institute of Technology (2008)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2017

Author ....
Department of Electrical Engineering and Computer Science
May 19, 2017

Certified by ....
Una-May O'Reilly
Principal Research Scientist
Thesis Supervisor

Accepted by ....
Leslie A. Kolodziejski
Chair, Department Committee on Graduate Students

Physiological Time Series Retrieval and Prediction with Locality-Sensitive Hashing

by

Yongwook Bryce Kim

Submitted to the Department of Electrical Engineering and Computer Science
on May 19, 2017, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

The amount of time series data collected in the medical community has recently been exploding due to widespread affordable sensors and storage devices. However, while the massive repositories of such physiological time series data provide enormous opportunities for machine learning to make significant impacts, they are largely under-utilized due to their granular detail, overwhelming size, and lack of proper tools. Besides scale, fast yet accurate processing of physiological waveform data is desired in medical practice, especially in time-critical settings such as the intensive care unit (ICU). Efficiently leveraging these massive datasets is a key challenge that, when resolved, will support a new paradigm of scientific discovery and operational innovation in medicine.

In this thesis, we develop highly efficient similarity-based methods that make it practical to search massive physiological time series repositories to rapidly identify waveforms similar to those from a given individual. We call this concept "*patients with trajectories like mine*." Our goal is to exploit rapid similar waveform retrieval to enable critical event prediction in the ICU setting. In order to achieve this goal, we propose to apply locality-sensitive hashing (LSH), which supports a very fast approximate nearest neighbor search in high dimensions. We empirically demonstrate that LSH based retrieval and prediction methods vastly speed up querying time while sacrificing only a trivial amount of accuracy as a cost.

Despite being fast and accurate, the generic LSH has two shortcomings. First, it is capable of utilizing only one similarity measure at a time. To overcome this limit, we introduce Stratified LSH (SLSH) which finds similarity among the data from a more integrated perspective by employing multiple distance metrics in one framework. SLSH is essentially a dual-level hierarchical LSH where each LSH layer is associated with a distinct distance metric capturing a unique facet of similarity. The second shortcoming and the main bottleneck of the generic LSH is that it involves exhaustive distance calculations as a subroutine when short-listing the candidate set to find the final nearest neighbors. To surmount this, we propose Collision Frequency LSH (CFLSH) which short-lists the candidate set by simply counting the frequency of collision based on the key idea that the more frequently an element and a query collide across multiple LSH hash tables, the more similar they are. We show that with SLSH and CFLSH, we improve the efficiency of LSH in terms of both prediction accuracy and querying speed.

We demonstrate our proposed methods on a mean arterial blood pressure dataset extracted from the MIMIC II database in the context of predicting acute hypotensive episodes in ICU. To examine the generality of our methods with respect to scaling, we validate our methods on datasets with various dimensions and item counts.

Thesis Supervisor: Una-May O'Reilly
Title: Principal Research Scientist

Acknowledgments

I am greatly indebted to many people around me at MIT. I can never thank them enough. First of all, I would like to express my sincere gratitude to my research advisor Dr. Una-May O'Reilly whose guidance and support were unwavering these past five years. She was always positive in the face of my numerous failures along the way and encouraged me to pursue a broad vision and freely explore my own research interests. Without her passion, generosity, and patience, completion of this thesis would have not been possible. I am much obliged to my thesis committee members Professor Peter Szolovits and Professor Piotr Indyk. I thank Pete for introducing me to the field of biomedical computing and for his mentorship and thorough comments which greatly improved this thesis. Piotr's support has been invaluable for me to pursue research on LSH. His feedbacks on my research were always very insightful, encouraging, and judicious. I would also like to acknowledge my academic advisor Professor Berthold Horn for his guidance during my graduate study.

I am grateful for the help and friendship of Erik Hemberg. Erik has been an excellent collaborator and a co-author of mine for several papers. I would like to thank the past and present members of the ALFA group including Miguel Paredes, Franck Dernoncourt, Stjepan Picek, Sarah Alhumoud, Nicole Hoffman, Ignacio Arnaldo, Kalyan Veeramachaneni, Jacob Rosen, and Chidube Ezeozue for creating a great research environment. I also appreciate my officemates Shaiyan Keshvari, Wenzhen Yuan, and Ben Wolfe for all interesting conversations and distractions. I owe huge thanks to my friends for their continual support during my ups and downs: Eunsuk Kang, Sangwoo Jun, Donghyun Jin, Jaekyung Ha, Sangtae Kim, Sangwon Byun, Hijung Shin, Sejoon Lim, Joseph Lim, Eunhee Sohn, Joohyun Seo, Heesang Lee, Yisoo Pyon, Adam Kuang, Mic Byrne, Charlie Maher, Laura Kim, and Jina Yoo. My graduate study was supported by a generous fellowship from Kwanjeong Educational Foundation.

Finally, I would like to express my deepest gratitude and dedicate this thesis to my amazing father, mother, brother, and grandmother for their unconditional love and unfailing trust in me.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

This thesis consists of methodological studies using locality-sensitive hashing (LSH) on two clinically important tasks: efficient retrieval of similar physiological waveforms given an individual and subsequent medically critical event prediction. We seek to bring computer science theory into practice in the field of clinical medicine. We demonstrate our LSH-based retrieval and prediction methods on an arterial blood pressure dataset extracted from the MIMIC II database. In this chapter, we introduce motivations and background behind our work, technical challenges we face, our proposed approach to overcome challenges, and our contributions to the field.

1.1 Motivation

The amount of data collected in the medical community has recently been exploding and is becoming more overwhelming due to widespread use of affordable sensors and storage devices. Contexts range from EEG (electroencephalography), ECG (electrocardiography), and blood pressure sensing in hospitals, to mobile phones or lightweight wearable health tracking devices in homes and ambulatory care settings.

The ever increasing volume and detail of information captured from hospitals and personal healthcare devices show promising potential to direct the medical practice toward more data-driven, evidence-based, and personalized medicine. However, the massive repositories of such physiological time series are largely under-utilized due

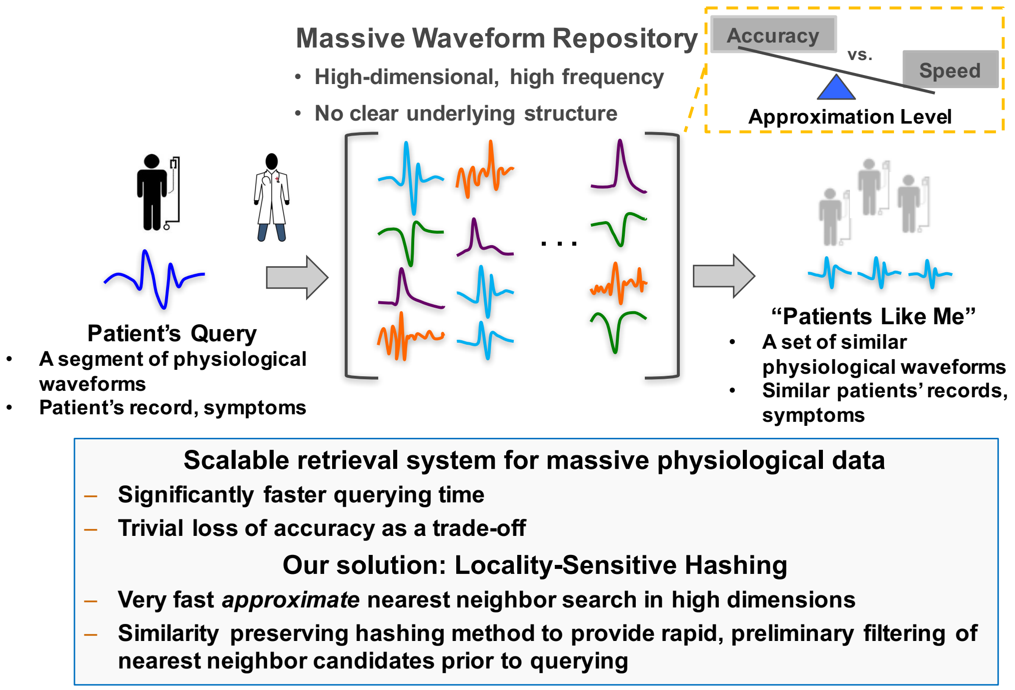

Massive Waveform Repository
• High-dimensional, high frequency
• No clear underlying structure
Accuracy
vs.
Speed
Approximation Level
Patient's Query
• A segment of physiological waveforms
• Patient's record, symptoms
"Patients Like Me"
• A set of similar physiological waveforms
• Similar patients' records, symptoms
Scalable retrieval system for massive physiological data
– Significantly faster querying time
– Trivial loss of accuracy as a trade-off
Our solution: Locality-Sensitive Hashing
– Very fast approximate nearest neighbor search in high dimensions
– Similarity preserving hashing method to provide rapid, preliminary filtering of nearest neighbor candidates prior to querying

Figure 1-1: Overview of efficient retrieval of similar physiological waveforms given an individual ("*patients with trajectories like mine*").

to their granular detail, overwhelming size, and lack of proper tools. *Efficiently leveraging these massive datasets* is a key challenge that, when resolved, will support a new paradigm of scientific discovery and operational innovation in medicine. Access to unprecedented amounts of data opens up an opportunity for deeper insight, earlier intervention, and engagement.

In this thesis, we develop highly efficient methods that make it practical to search massive physiological time series repositories to rapidly identify waveforms similar to those from a given individual. We call this concept "*patients with trajectories like mine*" (Figure 1-1). The thesis also develops methods to exploit rapid similar waveform retrieval to enable critical event prediction in the intensive care unit (ICU) setting. In the long term, our work will potentially contribute toward to replacing population-based models of disease with specific models based on groups of highly similar individuals, thereby allowing more precise diagnoses, critical event detection, prediction of illness trajectories, and more individualized medical interventions.

We provide one example where our work can potentially be valuable. In-hospital cardiac arrest (IHCA) is a crucial event that affects 200,000 adults and 6000 pediatric patients each year in the US [36]. Studies suggest that most cardiac arrests are predictable. In one study, 75% of IHCA cases were preceded by premonitory deteriorations in blood pressure, respiration, heart rate rhythm, or oximetry [11, 13] and over one third were determined preventable [36]. However, premonitory trends are frequently missed due to poor recognition of trends, which can be subtle. Thus, the ability to efficiently compare a given patient's physiological monitoring data with large databases to identify prior patients with similar trajectories and known outcomes will allow more precise predictions and support smarter alarms.

1.2 Background and Technical Challenge

The main axes of challenge in mining meaningful information from massive repositories are *time, accuracy, and scale.* Fast yet accurate processing of physiological waveform data is becoming essential in medical practice, especially in urgent care and ICU settings, as everything there is time-critical and also requires a high degree of correctness. Plus, the size of medical record corpora that we extract information from is vast and keeps increasing. Therefore, there is a strong need for large-scale, yet accurate data processing in almost real time. Many efforts in the past have sought to meet such need with various parametric and non-parametric methods. We briefly review the approaches tried in the past and the technical challenges we face when using these approaches in the context of physiological time series analysis.

1.2.1 Parametric Time Series Analysis Methods

Pattern mining plays a crucial role permitting medical practitioners and researchers to acquire high-quality relevant information from a massive repository of medical records. Traditional statistical methods for estimation of such patterns rely heavily on the use of parametric models [17, 48] in order to provide measures of effect and statistical significance. These are models that typically assume the entire data-generating distribution (i.e., the underlying mechanism that created the data) can be defined by relatively few parameters. However, these assumptions become highly unrealistic in healthcare settings because, due to the very complex and noisy nature of medical and physiological data, there is often no clear sense of its underlying structure which may vary by different patients and symptoms. Accordingly, these methods may be unreliable due to bias introduced by misspecified parametric models and are generally not flexible or scalable enough to handle a large number of variables and massive quantity of data.

For example, many of recent time series analysis methods are based on extensions of a parametric method called Dynamic Bayesian Network (DBN) [94]. However, the approach using variants of DBN has several limitations due to the requirement of a good hidden state model and the complicated, expensive learning and inference. In particular, the complexity of clinical data makes it difficult to form good prior knowledge which many conventional parametric and Bayesian models depend on. On the other hand, albeit less popular, there has been a series of works which showed the effectiveness of simple non-parametric similarity-based methods over many popular parametric models for time series classification [52, 133].

1.2.2 Similarity-based Search: *k*-Nearest Neighbor Method

A competitive alternative is to use a non-parametric approach which "lets the data speak for itself" without much restriction of parametric models. One of the core problems in such an approach is similarity-based nearest neighbor (NN) search [23, 131]. For a given query (such as a patient's record, a list of symptoms, or a piece of the physiological waveforms of interest herein), NN retrieval returns a set of records that are similar to the query. The NN set also offers extrapolative information. It may also reveal a complex pattern. When records extend forward in time past that of the query, they reveal outcomes of patients, chosen protocols, and diagnostics. Questions such as, for people with the same symptoms, what critical events subsequently occurred, what treatments produced the best recovery, and/or what side-effects were observed, can be answered.

The k-nearest neighbor (KNN) method is a simple non-parametric similarity-based learning algorithm. In essence, it memorizes the entire reference dataset and finds a group of k samples that are closest to a query by exhaustively going through every point in the reference dataset and computing its distance to the query. Typically, one uses the NN set to extrapolate a class label or response variable for the query based on predominance.

Unlike parametric models like DBN, KNN can “let the data speak for itself" since it does not summarize the input data by a number of parameter values of a certain model in the training phase or make any assumption on the underlying state and distribution of input data. Instead, the algorithm simply stores the entire training data without any summarization or generalization. Thus, the method is particularly useful when we do not have any prior knowledge about the data. Since it is very difficult to assume the underlying mechanism of human physiological signals, NN methods can be advantageous. Since KNN based classifiers can handle highly non-linear decision boundaries, it can be more advantageous for complex physiological state classification than many linear models.

The simplicity and practicality of KNN comes with several limitations.

- First, it is heavily influenced by the choice of distance metric and neighbor weighting rule. These choices are difficult for patient waveform data.

- Second, a distance metric only expresses a single perspective but waveforms may be similar based on various criteria, such as shape and amplitude, where each is expressed by a different distance metric.

- Third, KNN becomes highly impractical when the dimensionality of data is high and/or when the quantity of data is massive due to the curse of dimensionality. It is known that either the search time or space requirement is exponential in the number of dimensions and is linear to the dataset size [32].

In particular, high dimensionality is a property of physiological waveforms. Distance measures break down in high dimensions [1] and research has shown how the most efficient repository indexing strategies (such as C4.5 and tree-based methods [131], intended to support sub-linear time retrieval) become inefficient and exhibit linear behavior as dimensionality increases [127]. The traditional tree-based indexing methods (such as R-trees [46], k-d trees [14], and sr-trees [66]) degenerate into a linear scan in sufficiently high dimensions (larger than 10 dimensions [40]) both in theory and in practice [127].

1.3 Proposed Approach

To overcome the challenges we face with previously attempted approaches, our proposal is to utilize locality-sensitive hashing which is a fast approximate nearest neighbor search that is effective in large, high dimensional data. We briefly explain the basics of LSH and the research questions we aim to answer in this thesis.

1.3.1 Locality-Sensitive Hashing

Our goal is to build a scalable retrieval and prediction system for high dimensional massive physiological data, with a significantly faster querying time, while maintaining accuracy in a reasonable range in comparison to the linear KNN or tree-based methods. In order to achieve this goal, in this thesis, we propose retrieval and prediction methods based on a computer science theoretical foundation called locality-sensitive hashing (LSH) [54], which allows a very fast, approximate nearest neighbor search in very high dimensions.

Whereas the linear, exhaustive KNN method searches for the exact NNs, LSH aims to speed-up the search process by looking for *approximate* NNs instead. LSH is an approximate search method enabling a quick retrieval of a small approximate nearest neighbor set with provable sub-linear query time and sub-quadratic space complexity. It uses a specialized similarity preserving hashing method to provide preliminary filtering of NN candidates to reduce the time cost of a follow-up linear search among them. Locality-sensitive hash functions have the unique property that similar elements are statistically likely to be hashed to the same value (i.e. *collision*).

Given a particular distance metric and its corresponding hash function family, LSH maintains a number of hash tables containing the dataset points. The approximate nearest neighbors of a query can be obtained by hashing the query and scanning the hash buckets which the query collides with across the tables. By being approximate, LSH intrinsically introduces the trade-off between accuracy and speed depending on the level of approximation. Users can decide whether to wait for the exact answer by spending more time or to be satisfied with a much quicker approximation [40]. It is important to note that approximation of NNs is justifiable for most practical purposes because even in exact search, a distance measure is only an approximation to the ground truth. A detailed explanation of LSH is presented in Section 4.2.

In this thesis, we demonstrate the effectiveness of LSH on the waveform retrieval and event prediction tasks on an arterial blood pressure dataset extracted from the MIMIC II database [108]. To examine LSH on patient waveform retrieval, we reference a repository of tens of thousands of highly complex blood pressure waveform segments. The critical event of interest for us to predict is an acute hypotensive episode (AHE). An AHE is a sudden dropping of arterial blood pressure to below a critical level for some duration of a time window in ICU that demands immediate attentions and interventions. It is crucial to detect AHE accurately and fast, because if left untreated, such episodes may lead to irreversible organ damage and eventually death.

1.3.2 Research Questions

Throughout the thesis, we aim to bring answers to the following research questions.

- Given a repository of highly complex physiological waveforms and a query, is there an efficient way of retrieving similar physiological time series that is *fast*, *accurate*, and *scalable*?
  - How effective is LSH to meet the above requirements (in terms of retrieval accuracy and querying time) in comparison to the linear KNN method?
  - Given we use a similarity-based method, what are the appropriate bases (i.e. distance metrics)?
  - How does retrieval performance (accuracy and time) scale as dimension or quantity of data changes?

  - How sensitive or robust is the retrieval performance with respect to the parameters of LSH?

- Can a similarity-based retrieval set of arterial blood pressure waveforms effectively be leveraged for prediction of a critical event (acute hypotension) in ICU?
  - How effective is the prediction of AHE based on extrapolating the information of the retrieved nearest neighbors obtained by LSH, in terms of prediction accuracy and querying time?
  - What is the cause of the large difference in the querying speeds among LSH based on different distance metrics?
  - How does prediction performance scale as the lag duration[1]or quantity of data changes?

- Given the limit of LSH that it can use only one distance metric at a time, is there an effective way of utilizing multiple distance metrics in one LSH framework?
  - Is the multi-metric strategy more effective than using a single metric in terms of prediction accuracy and querying time?

- Given the bottleneck of LSH which is caused by entailing exhaustive distance calculations between a query and its nearest neighbor candidates, is there any effective way of circumventing the bottleneck?

1.4 Contributions

The main contributions of this thesis are as follows. To the best of our knowledge, our work to date is the first extensive application of LSH to physiological time series retrieval and event prediction.

- We address the question of how we can achieve fast, yet accurate and scalable retrieval of similar physiological waveform time series for a given query.

[1] The length of historical data prior to the event prediction window (explained in detail in Section 3.3).

We are the first to apply locality-sensitive hashing to approach this problem. When compared to the exhaustive KNN, our method based on LSH largely speeds up the retrieval time of similar physiological waveforms without sacrificing significant accuracy when demonstrated on an arterial blood pressure dataset extracted from the MIMIC II database. This work was published in [73] and is presented in Chapter 4.

- We answer the question of whether a high precision similarity-based retrieval set of arterial blood pressure waveforms can effectively be exploited to predict acute hypotensive episodes in ICU. In doing so, we extend the LSH-based retrieval to the prediction task by extrapolating the information of similar waveforms via majority vote. Similar to the retrieval case, compared to using the linear exhaustive KNN, our proposed method based on LSH vastly speeds up the prediction time up to two orders of magnitude while sacrificing only 1% of prediction accuracy. This work was published in [74] and is presented in Chapter 5.

- We propose a new similarity based prediction technique called stratified locality-sensitive hashing (SLSH). It finds similarity among the data from a more integrated perspective by employing multiple distance metrics in one framework, which previously was not feasible with the standard LSH. Comparing SLSH to the standard LSH, we demonstrate that SLSH yields a higher prediction accuracy and further shortens the sub-linear querying time of the standard LSH while adding only trivial storage overhead. A part of this work was published in [70, 71] and the more extended version has been submitted to a conference for review. This contribution is presented in Chapter 6.

- We address the question of whether the short-listing by calculating the distances between the query and every candidate set element (the main bottleneck of LSH) is optimal and whether there exists an effective way that avoids the bottleneck. To answer this, we propose a new variant of LSH, namely collision frequency locality-sensitive hashing (CFLSH). Unlike the standard LSH which only utilizes a distance metric, in CFLSH, the short-listing step from a pool of pre-selected candidates filtered by locality-sensitive hash functions to the final nearest neighbor set relies upon the frequency of collision along with distance information. We show that CFLSH with the L1 distance has a higher prediction accuracy and further accelerates the sub-linear querying time obtained by the standard LSH. This work will be published in [72]. This contribution is presented in Chapter 7.

1.5 Organization

The rest of this thesis is organized as follows. In Chapter 2, we describe related work. Then, we explain datasets used in this thesis in Chapter 3. Chapters 4 and 5 present the result of applying LSH on the problems of physiological time series retrieval and of critical event prediction, respectively. Chapter 6 introduces stratified LSH. In Chapter 7, we present collision frequency LSH. Finally, conclusions and future directions are in Chapter 8.

Chapter 2

Related Work

Although there is a vast literature on time series analysis in general [37,48], searching through high frequency, high dimensional physiological time series data is a relatively unexplored topic. Developing robust algorithms for correctly finding predictive patterns in long non-stationary time series data is challenging. This chapter is organized as follows: we provide a survey and discuss works on similarity-based time series search, prediction methods applied in critical care, theories and applications of locality-sensitive hashing (LSH), and acute hypotension prediction.

2.1 Similarity-based Time Series Methods

Clinical decision making based on extrapolating information from similar patients of a query patient has a long history. In the 1970s, the similarity-based "patients like me" approach was applied on electronic health records (patient charts) to make prognoses of ischemic [90] and coronary heart disease [105]. More recently, Google Correlate was released that finds web search terms whose popularity over time best match a user-provided time series [117]. Based on asymmetric hashing for Pearson correlation, one of its highlighted applications is predicting flu trends. Likewise, Lehman *et al.* [84] used the Gaussian mixture model and the k-nearest neighbor method to learn dynamical patterns of temporal data and find similarity among them. They used their method on applications such as search-by-example based data retrieval, event classification, and forecasting hypotensive episodes. Saeed *et al.* [107] proposed a wavelet-based symbolic transformation that allows the use of existing efficient document information retrieval algorithms to assess similar patterns in multi-parameter physiologic time series. They applied their method to predict hemodynamic deterioration. For more extensive review of how machine learning in used in decision support in critical care, one should refer to [62]. For in-depth overview of the general time series methods based on data mining and on econometrics based time series analysis (e.g. AR, ARMA, ARIMA models), [37] and [48] provide thorough reviews, respectively.

Recent literature in data analytics suggests applying simple nonparametric methods with a large quantity of data in order to let the "Big Data" truly speak for itself instead of using sophisticated parametric models with a small amount of data [47]. Numerous studies show the effectiveness of non-parametric nearest neighbor methods over many popular parametric models for time series classification. For instance, [133] showed that the one-nearest-neighbor classifier with the dynamic time warping (DTW) distance measure has a superior performance over a multi-layer perceptron neural network, hidden Markov model, and decision tree. In [52], the authors showed that their nearest neighbor based classifier performs better than support vector machine, naive Bayes, and C4.5 decision tree for classifying patients with abnormal hearts from a small electrocardiography (ECG) dataset. [75] showed that the nearest neighbor classifier for predicting acute hypotensive episodes continuously improves as the dataset size gets larger while the dynamic Bayesian network does not scale well with increasing data size. However, the nearest neighbor search is in general expensive for large-scale datasets due to high dimensionality and large quantity of data. Several works seek to overcome such difficulties by finding either effective *data representations* or efficient *distance measures*.

Representation There has been a series of work on finding the best representation for time series data such as Discrete Fourier Transformation [35], Single Value Decomposition [35], Discrete Cosine Transformation [78], Discrete Wavelet Transformation [19], Piecewise Aggregate Approximation [67], and Symbolic Aggregate

Approximation [86]. Most of the representations focus on dimensionality reduction and finding intrinsic dimensionality of data. In [31], it was shown that over many data sets, there is no single representation that performs better than others and the optimal choice of representation method is data-specific. In our LSH methods presented in this thesis, we use a raw representation of data in its vector form without any dimensionality reduction and still remain efficient since LSH is known to be effective in high dimension [49].

Distance Measures Similarly, in conjunction with a representation, many efforts have been made to speed up the computation of the following distance measures: Euclidean distance [35, 93], Manhattan distance [134], Lp-norm [134], DTW [16, 68, 103], Edit Distance based on Longest Common Subsequence [119], and Edit Distance [21, 22]. Among these, the UCR suite for DTW [103] and Euclidean distance [93] is known to be the fastest in each distance category. DTW performs better than others for data sets of small size and short time series [31], but the performance converges to that of Euclidean distance for large datasets. In general, branch-and-bound techniques [103] do not scale well with long time series, although they are known to be able to process massive datasets efficiently. In our work, we emphasize robustness to the length of time series and scalable performance that improves with greater item count.

2.2 Locality-Sensitive Hashing

Since its first introduction by Indyk and Motwani [54], locality-sensitive hashing has made a significant impact on the problem of large-scale nearest neighbor search in high-dimensional data. In large, the focus of research on LSH can be divided into three aspects: developing different LSH hash families for various distance metrics, exploring the theoretical boundaries of LSH, and improving the performance of the LSH methods [122]. We provide a survey on each aspect in the following subsections. We additionally review applications of LSH and two specific variants of LSH (multilevel and frequency based LSH) whose works have similar principles to our works.

2.2.1 Hash Function Families

The primary focus of LSH research is to develop locality-sensitive hash function families for various distance metrics. Since the original LSH was proposed for the Hamming distance [40, 54], several variations of the original version have been proposed for locality-sensitive hash function families over a wide range of distance metrics. We categorize the distance metrics into four groups: the angle-based (cosine) distance, the Lp distance, the Jaccard coefficient, and the rest.

The LSH for the angle-based distance includes the random projection LSH [4, 20], super-bit LSH [59], kernelized LSH [80], LSH with learnt metric [81], concomitant LSH [34], hyperplane hashing [56], and cross-polytope LSH [6]. Many efforts have also been made to design hash families for the Lp distance, especially for the Euclidean distance. They include LSH for the L1 distance [3, 40], the Euclidean distance [7, 26], the p-stable distributions [26], leech lattice LSH [4], and spherical LSH [8, 115]. An extensive comparison of different LSH methods for the Euclidean distance is found in [100]. Developments for the Jaccard coefficient (used extensively in information retrieval) include min-hash LSH [18], min-max hash [58], and B-bit minwise hashing [85]. Additionally, variants of LSH have been developed for the Hamming distance [40, 54] and the χ^2 distance (for data represented as a histogram) [43]. For more extensive review of the above LSH methods, refer to [5, 122, 125].

2.2.2 Theoretical Bounds

Another line of research, especially popular in the theoretical computer science community is exploring theoretical bounds of LSH. As this research area is continuously being updated, we briefly cover only the most up-to-date noteworthy results.

For the Euclidean distance on the unit sphere (the special case which is equivalent to the angular distance or cosine similarity used in many applications), spherical LSH [115] is known to have the best known provable guarantees, but has a very limited practical use because it is based on complex hash functions that are time consuming to evaluate. On the other hand, the seminal hyperplane LSH [20] has worse theoretical guarantees, but works very well in practice. Andoni *et al.* [6] closes the gap between theory and practice. With their cross-polytope and multi-probing based LSH for the angular distance, they meet the optimal guarantee of [115], but also improves over the hyperplane LSH. They also provide a practical algorithm released as the FALCONN software package [104].

For the Hamming distance, [54,95] prove the optimal lower bound. Also, [4] proves the tight bounds for the Euclidean distance. "Beyond LSH" [7] presents a new data-dependent data structure based on multilevel hashing for the approximate nearest neighbor problem in the Euclidean space (and in the Hamming space with a simple reduction) that is the first improvement over [4,54] and the first data structure that bypasses the classic LSH lower bound by [95]. [8] makes further improvement over "Beyond LSH" with a better theoretical guarantee.

2.2.3 Performance Improvement

Several strategies have been proposed to improve the performance of the original LSH in various aspects.

Space requirement One of the main drawbacks of LSH is that in practice, it requires a large number of hash tables to achieve good search quality. Panigrahi *et al.* [99] proposed entropy-based LSH which attempts to reduce the storage requirement for LSH. It does so by using both the original query point and its randomly perturbed nearby points as additional queries to combine the candidate sets.

Another effort to reduce the storage requirement of LSH is the multi-probe LSH [64,88]. It probes the matching hash bucket of a query as well as several other buckets in the same hash tables. The additionally probed buckets are the ones with hash keys not too distant from that of the colliding hash bucket. It was shown that the multi-probe LSH effectively reduced the space requirement by 90% in practice. However, these space reducing LSH methods are known to have longer query times.

Parameter tuning Another significant drawback of LSH is that it is sensitive to several model parameters which need to be chosen empirically. [33] provides automatic tuning scheme for parameters needed to run multi-probe LSH with performance guarantees. Similar yet more specific analysis of LSH parameter selection scheme is presented in [112].

LSH Forest [12] was proposed to overcome the sensitivity of LSH against various parameters and data distributions. Each hash table is represented as a tree whose leaves correspond to each data point (the set of hash tables represented as a set of trees, hence the name "forest"). Any sub-trees that do not contain any data points are pruned. By having a flexible tree structure, LSH Forest can adapt to different data distribution and also to the situation where additional points are added or deleted. [9] provides an improved version of LSH Forest. Their simple modification is that for each node, they store a constant number of points close to the mean of the corresponding subset of the dataset, which are compared to any query point reaching that node. Not only being effective in practice, this modification of LSH Forest is also provably better than the best LSH algorithm for the Hamming space [54].

Query and data adaptive LSH Performance of LSH on a query point depends not only on the distribution of the data, but also on the local geometry in the vicinity of the particular query [33]. Thus, several works have developed query and data specific LSH. For example, [33] provides adaptive multi-probing scheme which determines the appropriate number of multi-probing buckets just enough to achieve the required search quality. Query-adaptive LSH was proposed in [57] where, for each query, the method picks the hash functions that are most likely to return the nearest neighbors from a large pool of random hash functions. [53] proposes a similar idea with their query-aware LSH. In the context of computer vision, Korman and Avidan proposed coherency-sensitive hashing where immediate spatial neighbors of points in the matching bucket of a query are also included in the approximate nearest neighbor search [76].

In reality, datasets are typically not distributed uniformly over the space, and as a result, the buckets of LSH are unbalanced, which causes the performance of LSH to degrade. Several works have sought to solve this problem. For example, [39] proposed data sensitive hashing which designs data-adaptive hash functions based on adaptive boosting and spectral techniques, treating the hash function family as a strong classifier while each hash function in the family serves as a weak classifier. [61] introduced distribution density aware hashing which extends the random projection based LSH by first sub-grouping the data with the k-means algorithm, generating random projections that best separate each pair of groups, and then using maximum entropy principle to select the final set of random projections. [45] analyzed the non-uniform problem of the Euclidean LSH and proposed a pivot-based algorithm to accelerate the query process of the Euclidean LSH by using triangle inequality to prune the search process.

Parallelism Several works have sought to efficiently parallelize LSH. In [113], Parallel LSH was introduced which is designed to be extremely efficient, capable of scaling out on multiple nodes and multiple cores supporting high-throughput streaming of Twitter data. They utilized several novel ideas such as cache-conscious hash table layout, using a two-level merge algorithm for hash table construction, duplicate elimination during hash-table querying, an insert-optimized hash table structure, and efficient data expiration algorithm for streaming data.

In the distributed setting, each query requiring a network call per hash bucket look-up leads to a large network load. [10] proposed an efficiently distributed scheme for the entropy based LSH [99]. It used a layered hashing based on distributed entropy LSH using MapReduce and active distributed hash table to minimize the network cost while maintaining good load balance between different machines.

2.2.4 Applications

In the early days of LSH, it was successfully used in duplicate detection [18, 51, 89], link-based similarity search [24, 27], and image retrieval [44, 80, 111]. Recently, with the advent of “Big Data”, the demand for LSH has increased. LSH has been used extensively in a wider variety of application areas to deal with scaling issues of new massive and high-dimensional data. Recent application areas in the past few years include: speaker identification [110], music search [106], similarity join size estimation in databases [83], genome sequencing [15, 121], social network analysis [113, 132], motion planning in robotics [98], patch finding in images [77], in evolutionary algorithms for multi-solution optimization [138], lattice based cryptography [82], video anomaly detection [137], image forgery detection [2], signal processing [45], audio source separation [69], malware clustering [96], entity resolution [118], privacy preservation in cloud computing [136], geography analysis [135], and speech recognition [116].

2.2.5 Locality-Sensitive Hashing on Physiological Data

In this thesis, we extensively apply LSH on physiological time series data. Only a few studies have explored LSH in the healthcare and medical domain. LSH was applied in [65] where the focus was on introducing a kernel based method to adapt various types of similarity measures, demonstrated on pediatric ICU and surgical data. LSH was also used in indexing ECG time series using salient segmentation of data, but the data, containing only very short segments, was not large enough to show significant advantages [130]. Syed *et al.* [114] applied LSH to automatically discover patterns that distinguish between sequences belonging to different labeled groups. On symbolized ECG time series from patients with coronary syndromes, their LSH-based approach identified approximately conserved sequences of morphology variations that are predicative of future death. In [60], LSH was used to design a sensor fusion scheme to intelligently process wearable sensors with context awareness for the elderly. In contrast to these works, the datasets used in our study are orders of magnitude larger, and we examine the scaling properties of LSH.

2.2.6 Multilevel Locality-Sensitive Hashing

In Chapter 6, we introduce a multilevel LSH which hybridizes multiple distance measures in one framework. Two other notable works using multilevel LSH have been conducted in the field. Andoni *et al.* [7] introduced a two-level hashing method which results in the best lower bound complexity beyond the general LSH techniques. Data independent ball-carving LSH was used at the outer level and the data dependent spherical LSH was used at the inner level. However, this work only provides theoretical analysis of its method and lacks experimental evaluations or a practical implementation.

Pan *et al.* [97] also presented a two-level LSH method. In the first level, they use a random projection tree to partition the dataset into subgroups. Then in the second level, a single LSH hash table for each subgroup along with a hierarchical structure based on space-filling curves is computed. They demonstrated efficiency of their method over the standard LSH for image retrieval task. In contrast to our multi-layer framework which integrates multiple distance metrics, in [97], the first level operation is rather a data pre-processing step instead of being an actual layer of LSH, and only a single distance measure based on the Lp distance is used at the second level.

2.2.7 Frequency-based Locality-Sensitive Hashing

In Chapter 7, we propose LSH based on collision frequency counting. In [87], the authors proposed a frequency based LSH. It utilizes a single function based on the p-stable distribution as the hash function of a hash table and uses a frequency threshold to select only those points which collide with the query more than the threshold times as the candidate approximate nearest neighbors. On the other hand, our method can be used for any distance measures with a valid locality-sensitive hash function family.

In the database community, LSH based on dynamic collision counting was introduced, where the method uses a base of m single LSH functions to construct dynamic compound hash functions [38]. If the number of LSH hash functions under which a data point collides with the query is greater than a pre-specified threshold, the point is selected to be a candidate for the approximate nearest neighbors of the query. Both of the above methods are very sensitive to the value of the threshold, while our proposed method is threshold-free.

2.3 Acute Hypotensive Episode Prediction

The primary task of interest in this thesis is predicting acute hypotensive episodes with LSH. The 10th PhysioNet/Computers in Cardiology Challenge in 2009 first addressed the problem of predicting acute hypotensive episodes [91]. A small subset of the MIMIC II data [108] was made available to use, which included ECG and arterial blood pressure (ABP) signals, as well as the time series of vital signs sampled once per minute. The best performing model used the generalized regression neural network multi-models on the ABP data [50]. This approach requires extensive training of a neural network for each training sample, thus is ill suited for large scale problems. Moreover, despite the successful performance of many proposed solutions in the challenge, the size of the challenge dataset was very small (10 hours of lag data for each of 60 patients), and many proposed approaches would not scale well for realistic massive data. In contrast, the datasets used in our experiments are at least two orders of magnitude larger in terms of the number of patients.

With the advent of "big data" and time and space efficient cloud computing, machine learning is more readily applied to large repositories, see e.g. [120] which predicted acute event prediction with hidden state Markov modeling and [30]'s distributed feature selection for acute hypotensive episode prediction using wavelets optimized with Gaussian processes. In [29], a large scale machine learning and analytics framework, named *beatDB*, for mining knowledge from high resolution physiological waveforms was introduced where users can flexibly configure various set-ups of data, hypothesis defining, and algorithmic parameters. The utility of this framework was demonstrated for the acute hypotension prediction problem, but the choice of algorithm was limited only to logistic regression. Our work examines the scaling issue with respect to dimension and size of data, which was not a part of the above works.

Chapter 3

Data

In this chapter, we describe the datasets used to demonstrate our methodology. We define a critical event (acute hypotension) of our interest herein, provide an overview of the MIMIC II database, and discuss preprocessing steps and properties of our datasets.

3.1 MIMIC II Database

Our data comes from the MIMIC II (Multiparameter Intelligent Monitoring in Intensive Care) Database version 3 which contains physiologic signals and vital signs time series captured from patient monitors, and comprehensive clinical data obtained from hospital medical information systems, for tens of thousands of intensive care unit (ICU) patients [41, 92, 108]. It is one of largest clinical medical databases that are currently publicly available. Data were collected between 2001 and 2008 from a variety of ICUs (including medical, surgical, coronary care, and neonatal) at the Beth Israel Deaconess Medical Center in Boston, MA. The overview of the MIMIC II database is illustrated in Figure 3-1.

The MIMIC II database is composed of two distinct components.

- The Waveform Database contains records of continuous high-resolution physiologic waveforms and minute-by-minute numeric time series (trends) of physiologic measurements.

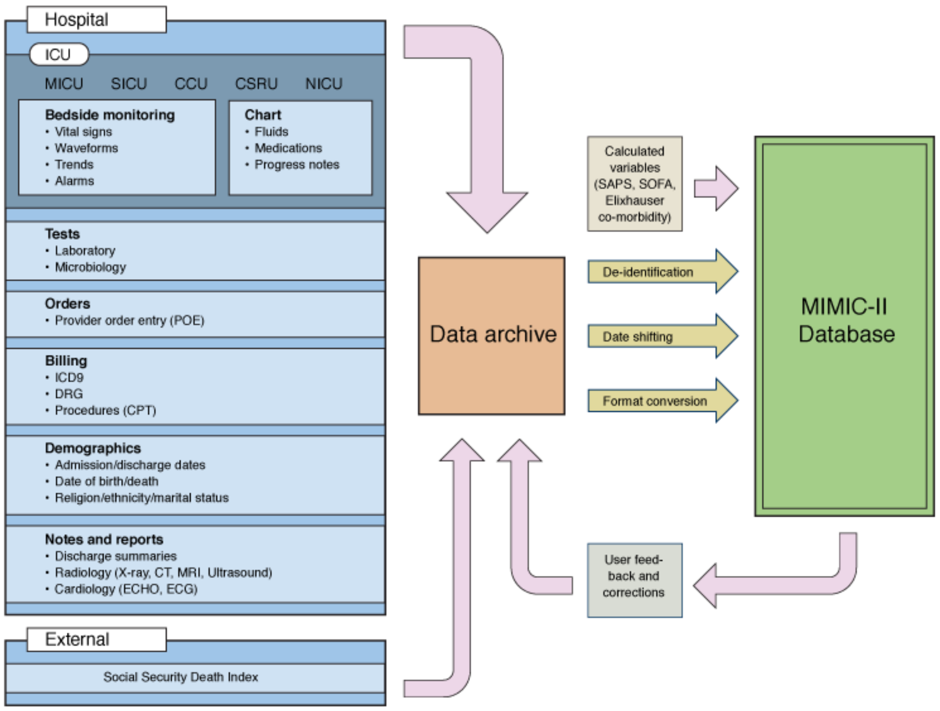


Figure 3-1: MIMIC II database overview. Image source: MIT Critical Data [25].

- The waveform measurements include a variety of blood pressure waveforms (e.g. arterial blood pressure (ABP), a signal of our interest herein), electrocardiogram waveforms (AVF, AVL, AVR, I, II, III, MCL, MCL1, V, V1, and V2), PLETH (uncalibrated raw output of fingertip plethysmograph), and RESP (uncalibrated respiration waveform, estimated from thoracic impedance).
- The numeric/trend measurements include non-invasive blood pressure, cardiac output, carbon dioxide output, heart rate, respiration rate, oxygen saturation, and temperature.

• The Clinical Database includes the general information (e.g. patient demographics, hospital admissions and discharge dates, room tracking, death dates, ICD-9 codes), medications, lab tests, fluid balance, clinical notes (e.g. discharge summary, nursing progress notes), and reports. These records have been de-identified.

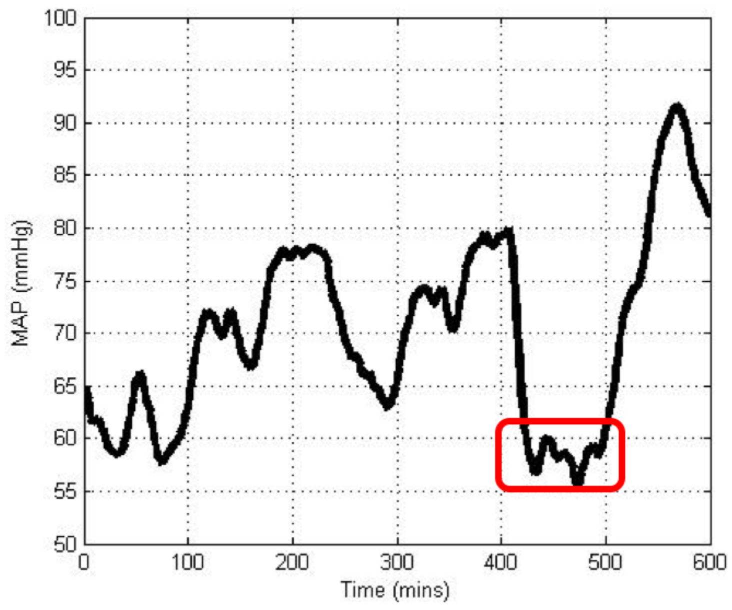


Figure 3-2: An example of an acute hypotensive episode (red box).

Many, but not all, of the Waveform Database records are matched to corresponding Clinical Database records.

The database contains 25,328 distinct ICU stays from 22,870 hospital admissions. There were patients who were admitted to the ICU multiple times. For more extensive statistics of the MIMIC II database, refer to [108].

At the time of this thesis writing, the new MIMIC III database has been published [63]. The biggest change is the larger dataset acquired over a longer time span (2001-2012). MIMIC III is a superset of MIMIC II. However, since the physiological waveform repository of the MIMIC III was unreleased to the public at the time of conducting research for this thesis, we adhere to the MIMIC II data.

3.2 Acute Hypotensive Episode

In our work, we are particularly interested in predicting the acute hypotensive episode (AHE) event (figure 3-2), which is a sudden dropping of blood pressure that demands immediate attention and intervention. If left untreated, such episodes may lead to irreversible organ damage and eventually death. Determining which intervention is proper largely depends on the diagnosis of the cause of the episode, which includes "sepsis, myocardial infarction, cardiac arrhythmia, pulmonary embolism, hemorrhage, dehydration, anaphylaxis, effects of medication, insufficient cardiac output, or vasodilatory shock" [91]. Typically, the best intervention is rather a suboptimal, yet relatively safe one which buys enough additional time to select more effective care plan without exposing the patient to additional risks.

It is known that about one third of patients in ICUs experience AHE, and the mortality rate of patients with AHE is more than twice that of patients without AHE [91]. Thus, AHE is a critical event in ICUs, requiring immediate medical intervention from hospital staff. Developing a good event predictor that can trigger timely and appropriate proactive intervention would make a significant contribution.

The definition of AHE differs from physician to physician. In this thesis, we follow the one used in the 2009 Physionet Challenge [91].

Definition 1. *AHE is defined as an interval in which at least 90% of the non-overlapping one-minute means of the arterial blood pressure waveform (MAP) were in the acute hypotensive range during any 30-minute window within the interval. The acute hypotensive range is defined to be under 60 mmHG (millimeters of mercury).*

The definition of AHE can actually be parametrized according to the time window we consider (e.g. 30 minutes), the threshold for the acute hypotensive range (e.g. 60 mmHG), and the percentage of beats whose MAP is too low (e.g. 90%). For the impact of these parameters on the prediction accuracy and associated sensitivity analysis, one should refer to the previous study conducted in [28].

3.3 Preprocessing

In this thesis, we focus on arterial blood pressure (ABP) waveforms since AHE is defined solely in terms of ABP. ABP waveforms (Figure 3-3) are recorded using arterial catheterization and a pressure sensor probe sampling at 125 Hz from a single

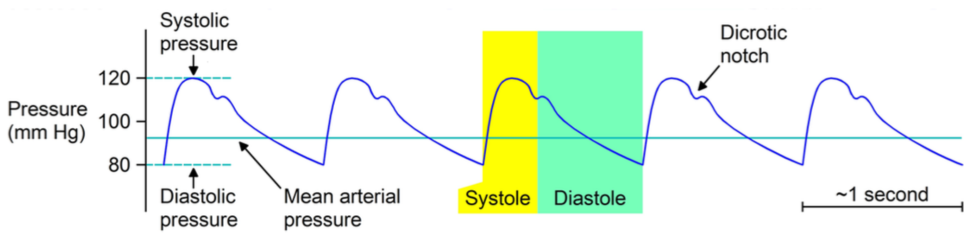


Figure 3-3: Arterial blood pressure (ABP) beats and morphological properties. Image source: PhysiologyWeb [101].

channel. The arterial line is connected to a tube filled with a saline solution, which is connected to a pressure bag. A pressure transducer is placed in the tube and converts pressure into an analog electrical signal measured at 125 Hz. The measurement is subject to noise from various sources including transducer placement, clotting in the arterial catheter, and device failures [28]. The Waveform Database of MIMIC II contains 6,232 patient ABP records. Given a sampling rate of 125 Hz and 240,000 hours of ABP data, there are 108 billion sample points.

For each patient record, we preprocess and validate this data by applying the onset detection algorithm on the ABP waveform to find the beginning and end of each beat and by examining its validity as a beat [120]. Beat detection reveals approximately 1.2 million beats of which approximately 0.9 million are valid [28, 120].

Then, for each beat, we calculate its corresponding mean arterial pressure (MAP) value since AHE is defined in terms of MAP. MAP is defined as a time-weighted average of systolic and diastolic pressure. The systolic pressure (the contraction phase of the heart) is the peak of blood pressure in a single beat whereas the diastolic pressure (the relaxation phase of the heart) corresponds to the minimum blood pressure in a beat (Figure 3-3). The ventricles approximately spend 1/3 of their time in systole and 2/3 of their time in diastole. MAP is defined as follows.

Definition 2. *mean arterial pressure = (2/3 * diastolic pressure) + (1/3 * systolic pressure).*

Then, we transform the time series of per-beat MAP values to the time series of per-minute average of MAP values for each patient (recall that AHE is defined in

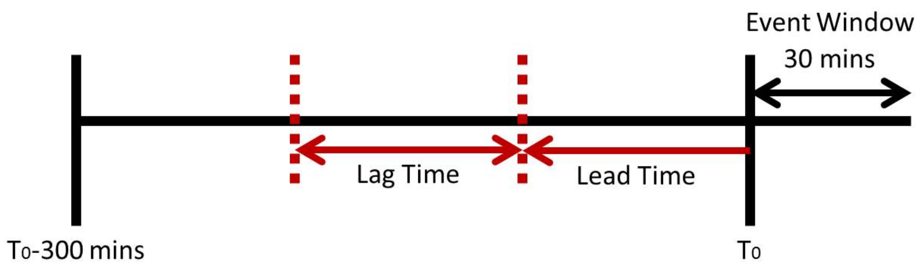


Figure 3-4: Problem definition of AHE prediction

terms of *one minute* averages of MAP).

Since patients' recordings can include jumps in time due to various source of signal artifacts, we treat a signal with an interruption as two individual segments if the gap in the original signal is longer than 5 seconds. Thus, there can be multiple segments of one-minute average MAP values for each patient.

Lag is the duration of the historical (training) data that the prediction model is based on, prior to the event window (Figure 3-4)[1]. We further divide the dataset into positive and negative sets. From the segments that include the presence of the AHE event in their time span, each sample in the positive set consists of *lag* minutes of time series prior to the beginning of the AHE event. In case where the available amount of data is less than *lag* minutes prior to the event, we discard such data. Thus, the shorter the lag is, the more data we are able to extract. On the other hand, the negative set is composed of samples that are randomly sampled *lag* minutes of time series from the segments that do not embrace any AHE event. We assign each of the positive and negative segments a label of 1 and 0, respectively. Like many critical event data, our data is naturally highly skewed toward AHE negatives.

3.4 Three Datasets

We use the following three datasets in our experiments.

[1] *Lead* time (Figure 3-4) defines how much *in advance* we make prediction prior to the event window. In this study, we fix the lead time as zero. For the impact of lead time on prediction performance, refer to [28, 75].

$\text{Data}_{\textbf{Lag 300, 1x}}$ We select patients who have segments of $lag = 300$ minutes of contiguous signal ($d = 300$, 300 dimensional)[2], which results in 6,467 segments. The data has 476 AHE positive (7.36%) and 5,991 AHE negative (92.64%) segments. We set 6,467 as the unit dataset size and refer to it as 1x data for convenience. We use this dataset as the main representative one to demonstrate our methods in the later chapters.

$\text{Data}_{\textbf{Lag 30, 10x}}$ If we lower the minimum duration of a valid segment, we can extract even bigger datasets. This time, we select patients who have segments of $lag = 30$ minutes of contiguous signal ($d = 30$, 30 dimensional), which results in 53,857 segments. The data has 3,694 AHE positive (6.86%) and 50,163 AHE negative (93.14%) segments. Since the size of this dataset is an order of magnitude larger than that of $\text{Data}_{\text{Lag 300, 1x}}$, we simply refer this dataset as 10x sized data.

$\text{Data}_{\textbf{Lag 30, 1x}}$ We sub-sample (without replacement) from $\text{Data}_{\text{Lag 30, 10x}}$ to construct a data set with lag of 30 mins and size equal to 6,467. It has the same class balance as $\text{Data}_{\text{Lag 30, 10x}}$. Since it has the same dataset size as $\text{Data}_{\text{Lag 300, 1x}}$, we refer to this as 1x sized data.

The last two data sets are used to experiment with LSH with respect to scale, duration of lag, and dimensionality. In the chapters to follow, we investigate the scaling impact of the data quantity (1x versus 10x) on the retrieval/prediction accuracy and querying speed of LSH by comparing results based on $\text{Data}_{\text{Lag 30, 1x}}$ and $\text{Data}_{\text{Lag 30, 10x}}$. Likewise, we measure the impact of the lag duration (300 versus 30) on the retrieval/prediction accuracy and querying speed by comparing experimental results based on $\text{Data}_{\text{Lag 300, 1x}}$ and $\text{Data}_{\text{Lag 30, 1x}}$.

3.5 Data Properties

In this section, we explore and describe the properties of our datasets introduced in the previous section. In particular, we focus on our main dataset, $\text{Data}_{\text{Lag 300, 1x}}$.

[2] We define the dimensionality of data as the length of time series.

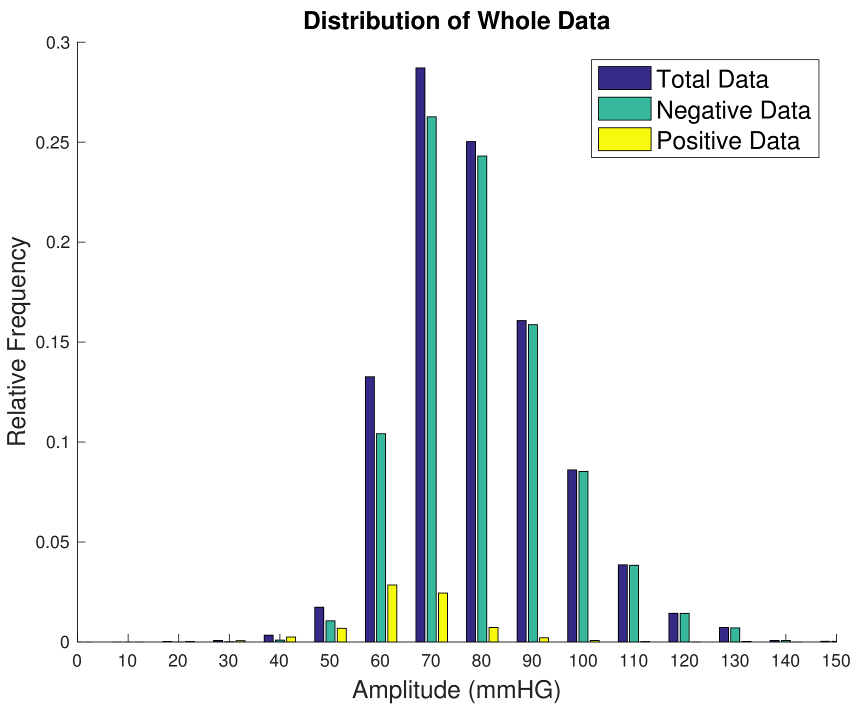


Figure 3-5: The distribution of the whole data for $\text{Data}_{\text{Lag 300, 1x}}$. Frequencies of each kind of data are relative to the size of the total data set (6,467 segments). AHE positive data has a lower mean than the AHE negative data.

3.5.1 Descriptive Statistics

We first take a look at the distribution of the whole data (regardless of individual segments). Summarized in Table 3.1, the whole data has a mean MAP of 79.27 with a standard deviation of 15.09 (in mmHG). The AHE negative data has a very similar profile of having a mean of 80.41 with a standard deviation of 14.73. In contrast, the AHE positive data has a significantly lower mean of 64.94 with a standard deviation of 11.89. Although our datasets contain only the lag minutes of data *prior* to the event window (the sample points in the event window are not part of the dataset and they are only used to define the class label), we observe that the AHE positive dataset in general has a lower mean. However, we also note that the margins of error of AHE positive and negative datasets overlap with each other, which makes the prediction problem of AHE challenging. Figure 3-5 illustrates the distribution of the whole data.

Next, we examine per segment data statistics. In particular, we look at the mean and standard deviation within individual segments. For each time series segment, we

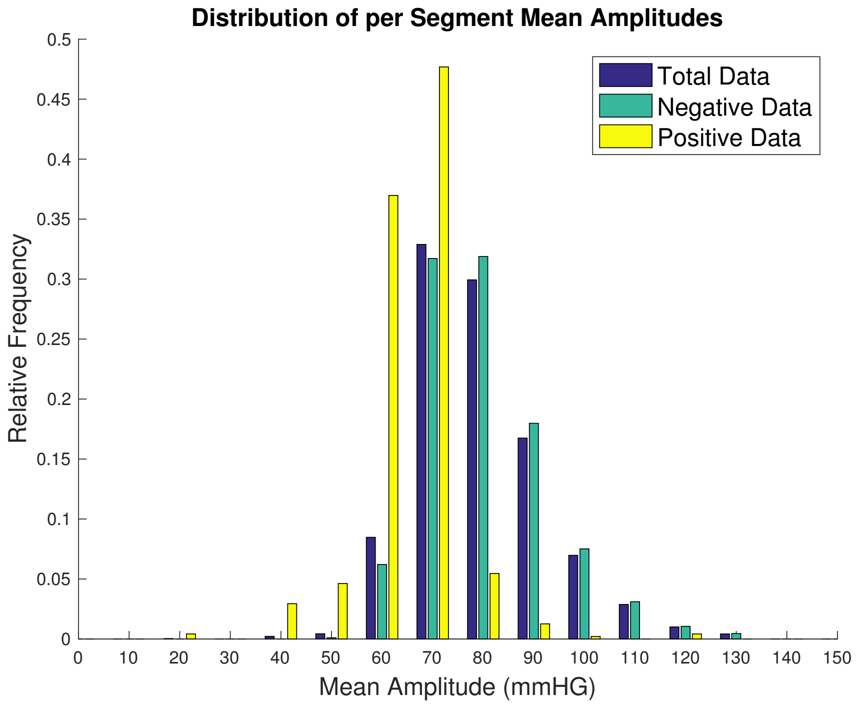


Figure 3-6: Distribution of per segment means for $\text{Data}_{\text{Lag 300, 1x}}$. Frequencies of each kind of data are relative to its own data size. AHE positive data has a lower mean than the AHE negative data.

Table 3.1: Data statistics for $\text{Data}_{\text{Lag 300, 1x}}$ (unit: mmHG).

	Total	AHE Positive	AHE Negative
Whole Data	79.27 ± 15.09	64.93 ± 11.89	80.41 ± 14.73
Per Segment Mean	79.27 ± 12.77	64.93 ± 8.98	80.41 ± 12.32
Per Segment Standard Deviation	7.31 ± 3.36	7.11 ± 3.23	7.33 ± 3.37

calculate the mean and standard deviation. It gives us the average value and the degree of variability within a single segment. Figure 3-6 shows the distribution of the per segment means. Summarized in Table 3.1, it presents the same trend as the previous case that the AHE positive data has a lower mean while the AHE negative and the total data has almost identical profiles. It shows the same means as the means for the whole data (without per segment differentiation) because the mean of means of individual segments is equal to the mean of the whole data. Only the standard deviation of means slightly differs from the standard deviation of the whole data.

As a reminder, AHE is a *sudden dropping* of blood pressure. One could question

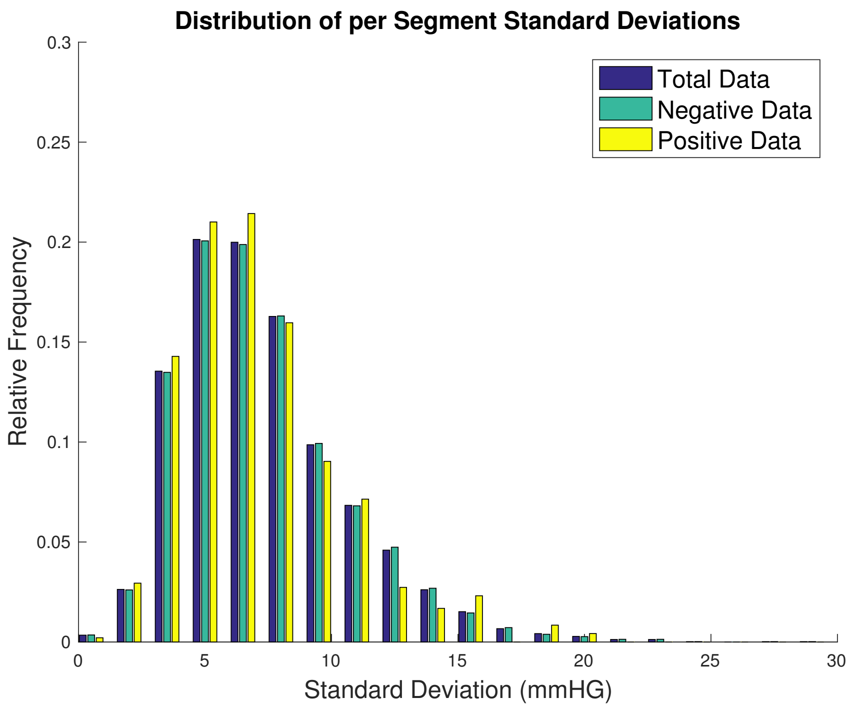


Figure 3-7: Distribution of per segment standard deviations for $\text{Data}_{\text{Lag 300, 1x}}$. Frequencies of each kind of data are relative to its own data size. There is no significant difference between the AHE positive and negative data.

if there is a larger variability in the AHE positive data. So, to check whether the AHE positive data has a larger variability within the segments (precursor to the event window), we compare the per segment standard deviations (Figure 3-7). From the figure and Table 3.1, we observe that there is no significant difference between the standard deviations for the AHE positive and negative data. This makes AHE prediction a challenging problem.

3.5.2 Interpoint Distance

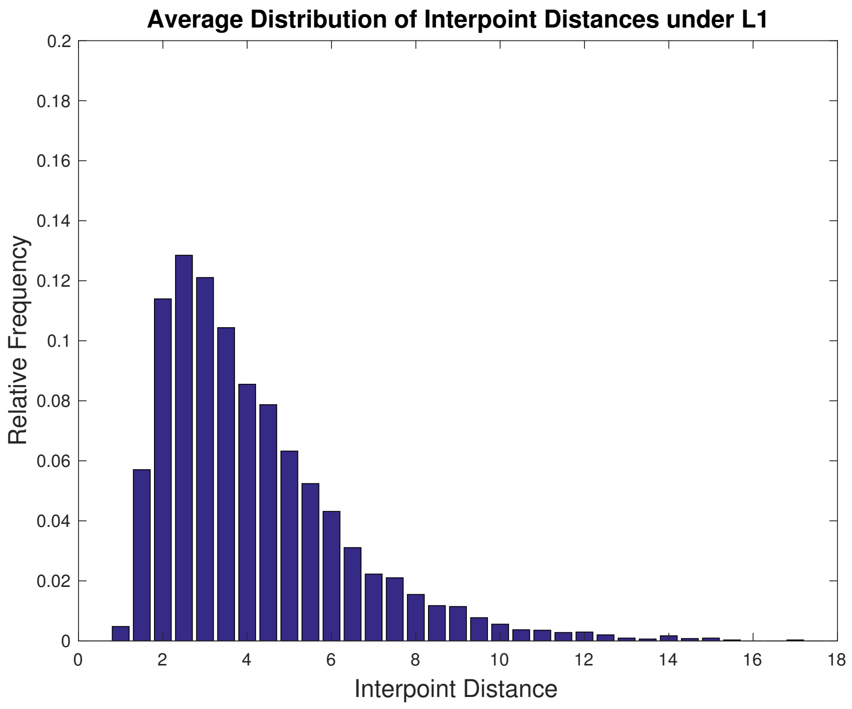
Besides the descriptive statistics of our data, we provide another angle on looking at the data by examining the distribution of interpoint distances in the data. The interpoint distance between two points is defined as follows.

Definition 3. *For a point q, its interpoint distance (IPD) to another point p under a distance measure D is equal to the distance between q and p divided by the distance* *between q and the nearest neighbor (NN) of q.*

$$IPD(q,p)_D = \frac{||q-p||_D}{||q-NN_q||_D}$$

It effectively measures how other points besides the nearest neighbor are distributed in the unit of the distance between the query point and its closest point. For each point q, we get a distribution of interpoint distances. To obtain the distribution for the whole data, we average the individual distributions over all q.

Figure 3-8 (Top) shows the average interpoint distance distribution under L1 distance. It has a long tail distribution with a mean of 4.15 and a standard deviation of 2.21. On average, data points are located 4.15 times farther away from the distance between a query and its nearest neighbor. It agrees with the previous study [126] that the distribution of the L1 distance between two arbitrary points follows the log-normal distribution. Likewise, Figure 3-8 (Bottom) shows the average interpoint distance distribution under the cosine distance. It follows the Gaussian distribution and has a mean of 3.50 with a standard deviation of 0.94. Thus, in the metric space of the cosine distance, the points in the dataset are more tightly grouped together compared to the points in the metric space of L1. In general, as the distribution moves to the left (toward IPD of 1), finding the correct nearest neighbor becomes harder as there is more chance for "any" points to be the nearest neighbor (i.e. higher false positives). Overall, the main implication is that even on the same dataset, the distributions of neighbors differ when measured with different distance metrics. Thus, we have to be careful how we apply each distance metric on the data in our LSH methods and subsequent analysis.



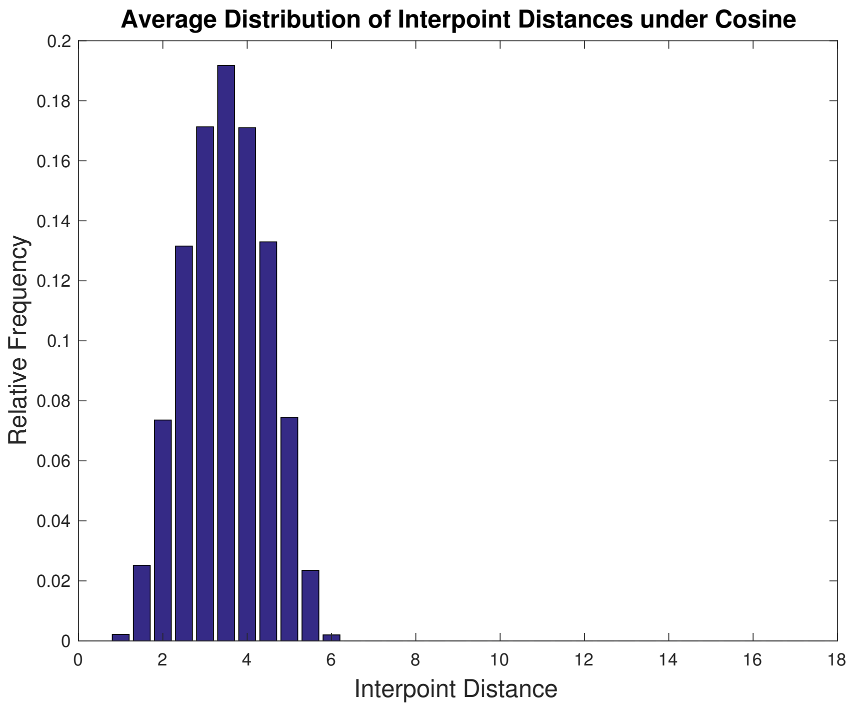


Figure 3-8: Average distribution of interpoint distances under (*Top*) L1 and (*Bottom*) cosine for $\text{Data}_{\text{Lag 300, 1x}}$.

Chapter 4

Locality-Sensitive Hashing for Waveform Retrieval

This chapter addresses the question of how we can achieve fast, yet accurate retrieval of similar physiological waveform time series for a given query. To answer this, we propose to apply locality-sensitive hashing (LSH). We explain the procedures of LSH in detail. When compared to the linear k-nearest neighbor search, we show that the LSH method largely speeds up the retrieval time of similar physiological waveforms without sacrificing significant accuracy, but LSH is highly sensitive to its hyper-parameter values. We also investigate the question of how dimension and quantity of data impact retrieval performance and observe that using data with a lower dimension and a larger quantity each improves retrieval accuracy and speed. Our specific demonstrations and evaluations use arterial blood pressure waveforms extracted from the MIMIC II database. The main content of this chapter was published in [73].

4.1 Motivation

Although the naive nearest neighbor (NN) search method [23] for retrieval works very well in practice with moderately sized data, its performance deteriorates rapidly for large, high-dimensional data. Our goal is to build a scalable, efficient retrieval system for high-dimensional massive physiological data, which has a significantly faster querying time, while maintaining the retrieval quality in a reasonable range in comparison to the linear search. In order to achieve this goal, we propose a retrieval method based on LSH [54], which allows a very fast approximate NN search in high dimensions. Whereas the naive NN method searches for the exact NNs, LSH aims to speed-up the search process by looking for approximate NNs instead. Approximate neighbors are valuable because even in the exact search, the distance measure D is also only an approximation to the ground truth.

In this chapter, the main research question is whether LSH is advantageous in terms of querying speed and retrieval accuracy, and by how much, compared to k-nearest neighbor (KNN) method. Our evaluations include sensitivity analysis of LSH performance with respect to its hyper-parameters, as well as dimension and quantity of data. To the best of our knowledge, this work is the first application of LSH on the retrieval task of physiological time series referencing a repository with tens of thousands of patients.

4.2 Method

In this section, we define the properties of locality-sensitive hash functions, explain three examples of locality-sensitive hash function families, lay out the procedures of LSH, and discuss the advantages of using LSH on physiological time series. We emphasize that this section will serve as a foundation for other event prediction methods to be presented in Chapters 5, 6, and 7.

4.2.1 Locality-Sensitive Hash Function Family

The central idea of LSH is to hash data points by multiple locality-sensitive hash functions with a special property that, for each hash function, the probability of hashing to the same hash value (i.e. *collision*) is much higher for points close in high-dimensional space than those that are far away from each other. This similarity preserving property is what distinguishes LSH from the conventional hashing as the goal of the latter is to avoid collisions even for close points. To preserve this similarity-based locality, a hash function h is chosen from a hash function family H that is (R, cR, P_1, P_2)-*sensitive*, i.e., for any points $p, q \in \mathbb{R}^d$,

- if $\|p - q\| \leq R$, then $\Pr_H[h(p) = h(q)] \geq P_1$
- if $\|p - q\| \geq cR$, then $\Pr_H[h(p) = h(q)] \leq P_2$

for constants $c, R > 0$, and $0 \leq P_2 < P_1 \leq 1$. The parameter $\rho = \frac{\log 1/P_1}{\log 1/P_2}$, which can usually be expressed in terms of the distance gap c, determines the search performance. The smaller ρ is, the faster the search performance is. The difference between P_1 and P_2 should ideally be large to increase the probability of collision. This is done by applying multiple hash functions.

There exists a direct correspondence between a distance metric and its associated family of locality-sensitive hash functions. However, it is important to note that for many distance metrics, the corresponding hash families have not been developed. Here, we experiment with three hash families that map the points in the original space to the binary Hamming space.

Bit Sampling based LSH for the L1 Distance: L1LSH

In our study, we utilize the hash function family for the L1 distance, proposed in [40, 54], $H_{L1} = \{h : \mathbb{X}^d \to \{0, 1\}\}$ such that

$$h(p) = \begin{cases} 0 & \text{if} \quad p_i < t_i \\ 1 & \text{if} \quad p_i \geq t_i \end{cases}$$

where p_i is the value on the i^{th} coordinate of $p \in \mathbb{X}^d$. i is a single dimension of the data chosen uniformly at random from $\{1, \ldots, d\}$ and t_i is a threshold chosen uniformly from the range of the data in that particular dimension. We choose this family of hash functions because it is parameter-free and requires no tuning, unlike other more sophisticated hash families for L1 [4]. Plus, this family is equivalent to the thoroughly studied bit sampling based hash function family for the Hamming distance. d-dimensional P (the set of N points) can be embedded into the Hamming cube of dimension $d' = wd$ by applying a unary function on each coordinate in P, where w is the largest coordinate of all points in P. It is a known fact that the Hamming distance on this embedded space preserves the L1 distance in the original space. For the detailed implementation procedures, analysis, and theoretical justification, one should refer to [40, 54].

Random Projection based LSH for the Cosine Distance: COSLSH

The second family we investigate is the random projection based locality-sensitive hash function family for the cosine distance. For $p, q \in \mathbb{X}^d$, the angle between them is $\theta(p,q) = \arccos(\frac{p \cdot q}{\|p\|\|q\|})$. Charikar *et al.* [20] defines the locality-sensitive hash function family for the cosine distance, $H_{cos} = \{h : \mathbb{X}^d \rightarrow \{0,1\}\}$ such that

$$h_r(p) = \begin{cases} 0 & \text{if} \quad p \cdot r < 0 \\ 1 & \text{if} \quad p \cdot r \geq 0 \end{cases}$$

where $r \in \mathbb{R}^d$ is constructed by picking each coordinate of r from the isotropic Gaussian distribution $N(0,1)$. This hash function is equivalent to dividing the original d-dimensional space into two subspaces by a randomly chosen hyperplane r, and the hash value is determined by on which side of the hyperplane p lands. Unlike L1LSH, which randomly and independently selects only a single dimension to generate a hash value at a time, COSLSH has a property that the random projection simultaneously considers all dimensions together.

The cosine distance in LSH has a strict requirement that the data must lie on a unit sphere. Data normalization is a safe preprocessing step in many application areas, but using COSLSH alone is not desirable for physiological time series since the meaningful amplitude information gets lost due to normalization.

Random Projection based LSH for the Euclidean Distance: E2LSH

The third hash function family we investigate is the random projection based locality-sensitive hash function family for the Euclidean distance (E2LSH) [26], $H_{E2} = \{h :$

$\mathbb{X}^d \to \{0, 1, \ldots, w\}\}$ such that

$$h_{r,b}(p) = \lfloor \frac{p \cdot r + b}{w} \rfloor$$

where $w \in \mathbb{Z}^+$ is the discrete quantization step chosen according to the data and the offset b is randomly drawn from the uniform distribution from 0 to w. The inner product $p \cdot r$ is the projected value of $p \in \mathbb{R}^d$ onto the direction r, where the projection vector $r \in \mathbb{R}^d$ is constructed by picking each coordinate of r from the isotropic Gaussian distribution $N(0, 1)$. The quantization step w influences the performance of E2LSH as it controls the resolution (fine or coarse grained) of the space of valid hash values. It needs to be chosen empirically. Unlike L1LSH which randomly and independently selects only a single dimension to generate a hash value at a time, E2LSH has a property that the random projection simultaneously considers all dimensions together at a cost of involving an extra parameter w. To compare E2LSH to L1LSH and COSLSH in the binary hash value space, we experiment with $w = 2$ in this work.

4.2.2 Locality-Sensitive Hashing Construction (Indexing)

LSH has a fixed cost construction (i.e. data indexing) phase that supports subsequent retrievals. The overall procedure is illustrated by the red arrow in Figure 4-1. We construct L hash tables each using m independently selected hash functions over the reference set. For *indexing* (details described in Algorithm 1), we select m functions such that $g_l = (h_{1,l}, h_{2,l}, \ldots, h_{m,l})$ for each $l = [1, 2, \ldots, L]$. The hash functions h's are randomly chosen from a LSH family H (lines 3-4). Then, we construct L independent hash tables where each hash table T_l contains the dataset points hashed using the function g_l (line 5). The value of g_l for each data point defines its hash key for its corresponding hash bucket.

When hashed by g, in order for two points p and q to belong to the same hash bucket, their hash values have to match for every one of m distinct hash functions h. Therefore, the larger/smaller the quantity m of hash functions, the stricter/more-

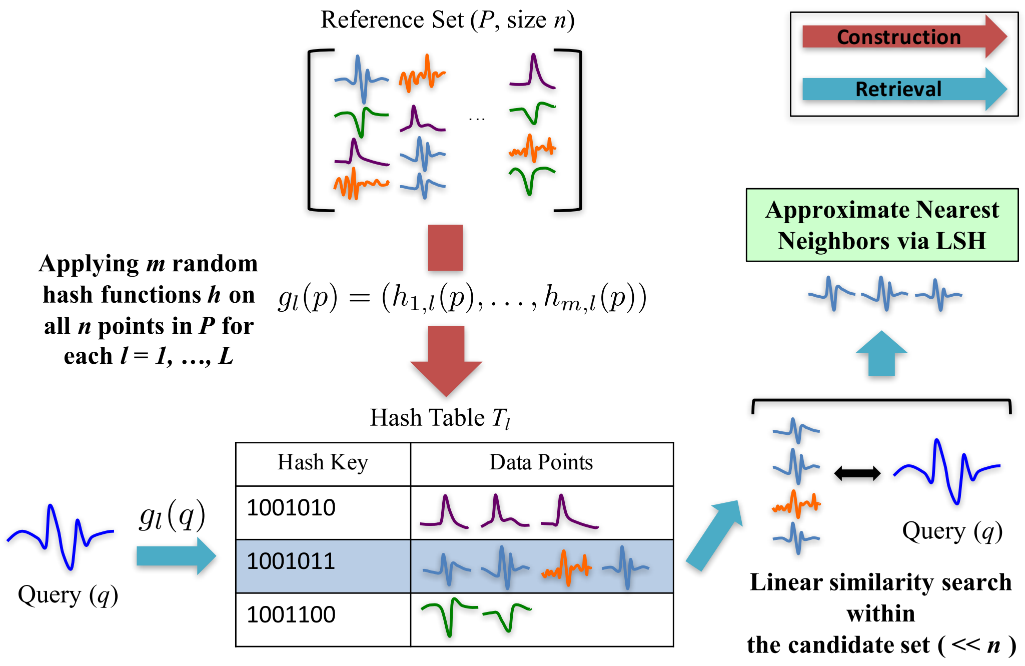

Reference Set (P, size n)
Construction
Retrieval
Approximate Nearest Neighbors via LSH
Applying m random hash functions h on all n points in P for each $l = 1, \ldots, L$
$g_l(p) = (h_{1,l}(p), \ldots, h_{m,l}(p))$
Hash Table T_l
Hash Key
Data Points
1001010
1001011
1001100
$g_l(q)$
Query (q)
Query (q)
Linear similarity search within the candidate set (<< n)

Figure 4-1: The overview of LSH construction (red arrow) and retrieval (blue arrow) procedures for a single hash table. For construction, similar waveforms are indexed into the same hash buckets. For retrieval, a query is first hashed to find the points included in its matching bucket in the hash table. Then, we linearly compute the distance between the query and such set of points (the candidate set with a size significantly smaller than that of the reference set) to retrieve the approximate nearest neighbors of the query. For multiple tables, the final candidate set for the linear similarity search consists of the union of the points included in all matching buckets from all L hash tables.

approximate the match. This parameter m is what we will explore experimentally to derive an optimal trade-off in accuracy and speed up. When setting up the standard LSH, the desirable number of hash functions used per table is the number that would partition the original data as uniformly as possible over *all possible* hash buckets so that each bucket contains only a small amount of the data.

A hash table, organized by hash keys, is to be used for inverse-lookup. It can return all items corresponding to a certain hash key in constant time independent of the data size. This is the key to achieving speed-up in LSH. It is important to note that LSH tables only need to store the pointers to the data instead of the original data itself. By applying standard hashing to store only non-empty buckets, it creates only a trivial additional memory cost of only $O(nL)$.

Algorithm 1 Standard Locality-Sensitive Hashing: Indexing

```
Require: $m$, number of hash functions per table; $L$, number of hash tables; $\mathbf{X}$,
    reference data; $H$, hash function family
1: procedure LSH-INDEXING($m, L, \mathbf{X}, H$)
2:     for $l = [1, 2, \ldots, L]$ do
3:         Sample $m$ hash functions $h_{j,l}$ uniformly random from $H$
4:         $g_l \leftarrow (h_{1,l}, h_{2,l}, \ldots, h_{m,l})$
5:         Hash table $T_l \leftarrow g_l(\mathbf{X})$                ▷ all data hashed into tables
6:     end for
7:     Return all $T_l$'s ($\mathbf{T}$) and $g_l$'s ($\mathbf{g}$)
8: end procedure
```

4.2.3 Locality-Sensitive Hashing Retrieval (Querying)

The goal of *querying* is to retrieve the nearest database items to a given query. The overview is presented by the blue arrow in Figure 4-1. The algorithmic procedures are laid out in Algorithm 2. For retrieval by the standard LSH, we

1. hash a query by the same set of hash functions g_l's used for indexing (line 4),

2. compose *the candidate set*, which is defined as the union of all points contained in the colliding hash buckets of the query from each hash table (line 5), and

3. retrieve a group of k items that are most similar to the query by the *linear search* within the candidate set (lines 7-8). We call this step *short-listing.*

The short-listing by the linear search (line 7-8) is the major bottleneck of the LSH querying procedure, which takes 95% or more of the overall running time.

There are two LSH parameters, the number of tables constructed (L) and the number of hash functions used per table (m), which need to be chosen empirically. The optimal pair of (m, L) provides the most efficient data structure to index the data for the fastest and the most accurate retrieval and prediction. To increase the *precision* of collision, m should be large enough to reduce false positives. On the other hand, in order to increase *recall*, L should be large enough. As the quantity of tables goes up, more approximate matches, each from different tables, are possible which increases the chance of finding the exact nearest neighbors of the query. The choice of the LSH parameters, m and L, depends on the specific application and

Algorithm 2 Standard Locality-Sensitive Hashing: Querying

```
Require: $q$, the query; $\mathbf{T}$, hash tables; $\mathbf{g}$, hash functions; $k$, number of NNs to retrieve
1: procedure LSH-QUERYING($q, \mathbf{T}, \mathbf{g}, k$)
2:     $\mathcal{C} = \emptyset$                                   ▷ candidate set
3:     for $l = [1, 2, \ldots, L]$ do                ▷ $L$ is the number of hash tables in $\mathbf{T}$
4:         $\mathcal{S} = \{x \in \mathbf{X} | g_l(x) = g_l(q)\}$          ▷ colliding bucket from table $T_l$
5:         $\mathcal{C} \leftarrow \mathcal{C} \cup \mathcal{S}$
6:     end for
7:     Compute distances from $q$ to each element in $\mathcal{C}$    ▷ with the distance metric that defines $\mathbf{g}$
8:     Return $k$ elements with the smallest distances
9: end procedure
```

the underlying distribution of the dataset. In some cases, the size of the candidate set is smaller than k and the query fails to retrieve k NNs. Such case where the k nearest neighbors of the query can not be retrieved is called a *miss*. It is important to note that the search time of LSH is guaranteed to be sub-linear to the size of the data [40], compared to the linear time cost of the naive KNN method. This can make a significant difference in search time in a massive data repository.

4.2.4 Advantages of Locality-Sensitive Hashing

There are several advantages of using LSH on physiological time series data. First, by transforming the high dimensional data into the space of short hash values generated by m hash functions, we effectively achieve dimensionality reduction to a lower order embedding space where the notion of similarity is well preserved. Moreover, while KNN suffers from the selection bias of the distance metric, LSH inherently offers a broader, less-biased coverage over the non-linear characteristics of waveform signals because each hash function h serves as a different basis of comparing points with a low resolution, "weak" similarity. Furthermore, both LSH construction and retrieval processes can easily be distributed since each table is generated and queried independently. Also, it is scalable because for new data, only the small step of applying the stored hash functions g is needed, which does not require another pass over the entire data.

4.3 Experiment

Baseline Given a pair of LSH parameters (m, L), for each query, we find k approximate NNs via LSH and compare them to those from the linear KNN search. It is complicated to evaluate nearest neighbor retrieval accuracy because there is no ground truth, i.e., no single notion of similarity is perfect, each being dependent on some distance metric. Practically, we proceed by using the linear KNN method as our baseline. It returns the result of an exhaustive linear search, and LSH will be compared to this "most accurate" (though most costly) method in terms of retrieval accuracy and querying time. KNNs with L1, cosine, and Euclidean as a distance metric will be compared to L1LSH, COSLSH, and E2LSH, respectively. Using the same distance metric as KNN in the final linear search step (short-listing) of LSH unifies the comparison basis.

Performance Measures To compare LSH to KNN, we formalize two performance measures: *retrieval accuracy* and *speed-up factor*. We define the *retrieval accuracy* as the recall, i.e. the fraction of the exact k-nearest neighbors that are also retrieved by LSH. We let $K(q)$ denote the approximate k-NNs of the query q retrieved by LSH and $I(q)$ denote the exact k-NNs obtained by the linear KNN. So, mathematically, the retrieval accuracy (recall) is $|K(q) \cap I(q)|/|I(q)|$. We do not consider the order within the k-NNs, but care only about the intersection between two sets. Accordingly, the overall retrieval accuracy is the average of the individual accuracies over all queries.

For cost, we are only interested in the querying time because the space requirement and the offline construction cost of hash tables are both linear in the number of hash tables and the dataset size [33]. Thus, we investigate the retrieval time, where the speed-up factor of LSH is the retrieval time of a query by LSH relative to that by the linear method. The major part of query processing in LSH is scanning through the candidate set to find the k-NNs of a query (short-listing). Empirically, we find that this step accounts for more than 95% of the total querying time. The time to apply the hash functions on the query and to locate its matching hash buckets to scan account for the other 5%. The querying time is mostly spent on computing distances between each point in the candidate set and the query, and is thus proportional to the size of the candidate set. We let $C(q)$ denote the candidate set of a query q. We define *selectivity* as $|C(q)|/N$, where N is the size of the whole dataset. Selectivity is a good indicator of the querying time that is independent of specific hardware configurations or the choice of programming languages. Thus, we define the *speed-up factor* of LSH as the inverse of selectivity.

Additional Information We define the optimal values of (m, L) as the parameters that result in the fastest retrieval time among the ones with retrieval accuracies higher than 95%. We perform a grid search by varying m from 5 to 100 with an increment of 5 and L from 10 to 100 with an increment of 10. We apply the above procedure to retrieve 1-NN, 5-NNs, and 10-NNs. We demonstrate LSH mainly on $\text{Data}_{\text{Lag 300, 1x}}$. For the purpose of the retrieval task, this dataset both serves as the reference set from which neighbors are retrieved and as the set of queries. We verify the correctness of our method by checking whether it always retrieves the query itself.

4.4 Results and Discussion

4.4.1 Nearest Neighbor Retrieval

L1LSH Figure 4-2 shows the trade-off between the retrieval quality and time for L1LSH. Each point represents the average retrieval accuracy and the speed-up factor of LSH over 10 trials for each pair of (m, L). The square boxes indicate the optimal parameters for each neighborhood size, k. For 1-NN, with (m, L) = (30, 40), we achieve the optimal nearest neighbor retrieval 12 times faster with LSH than the linear search, while sacrificing less than 5% accuracy. For 5-NNs and 10-NNs, we find the optimal LSH parameters are (30, 50) and (30, 60), respectively, and we still get reasonable retrieval times 8.5 and 7 times faster with LSH than the linear search.

For fixed m and L, only the retrieval accuracies degrade with an increasing k while the retrieval times remain almost the same. This is an expected behavior since

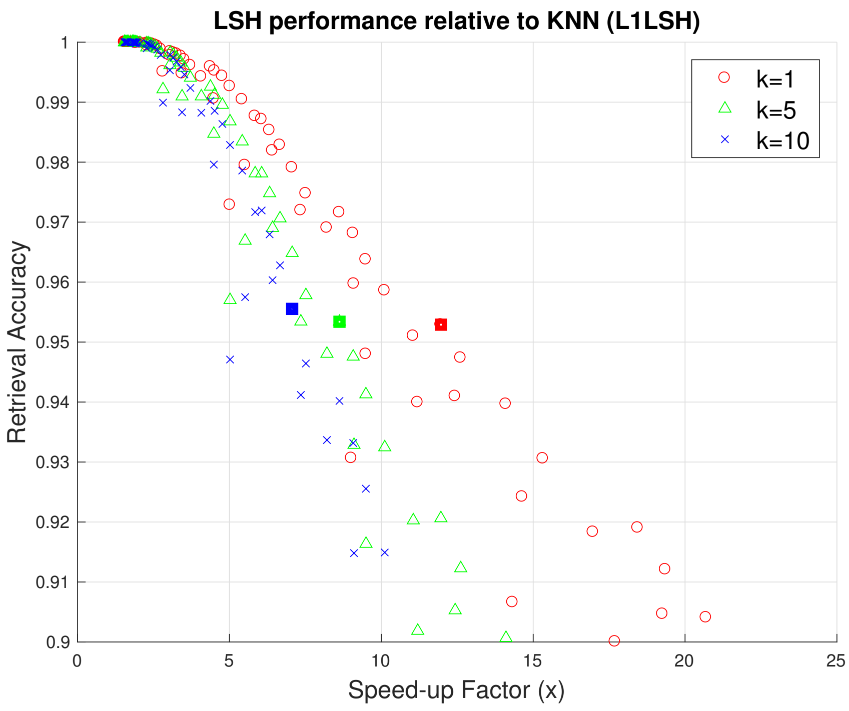


Figure 4-2: Trade-off between retrieval accuracy and speed-up factor of L1LSH performance relative to the linear KNN. Each point corresponds to a parameter configuration of LSH. Squared points indicate the optimal parameters for different values of k.

it is much harder for the farthest neighbors among the k-NNs from both methods to match for a large k, while retrieving more neighbors adds only a negligible time cost.

While all queries successfully return their 1-NNs, the fraction of queries that are not able to retrieve the requested number of NNs (i.e. the *miss* cases where the size of the candidate set of a query was smaller than k) was merely 0.11% and 0.19% of the entire data for 5-NNs and 10-NNs, respectively.

While LSH is faster and close to the accuracy of KNN, it does incur two preliminary costs: time to hash the data over L tables and storing hash tables. We investigated the break-even point of this cost with respect to query time saved relative to the linear search time. Given the target accuracy and speed-up factor, the time to construct the optimal LSH data structure was approximately equivalent to the retrieval time of 15% of the total queries (by the wall clock time). So after processing 15% of the queries, we would have saved enough time to make up for the cost

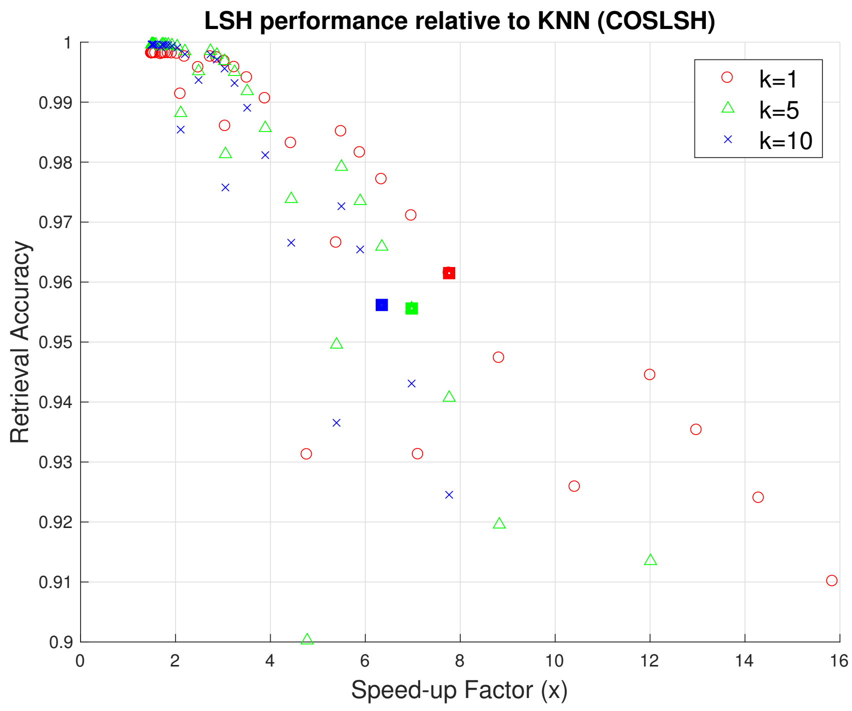


Figure 4-3: Trade-off between retrieval accuracy and speed-up factor of COSLSH performance relative to the linear KNN. Each point corresponds to a parameter configuration of LSH. Squared points indicate the optimal parameters for different values of k.

of building the optimal LSH data structure. The extra storage cost of hash tables was less than 1% of that of the original data since the hash tables only contain pointers to the original data.

COSLSH Figure 4-3 shows the trade-off between the retrieval accuracy and speed-up factor for COSLSH. The square boxes indicate the optimal parameters for each k. For 1-NN, with $(m, L) = (10, 60)$, we achieve the optimal nearest neighbor retrieval 7.9 times faster with LSH than the linear search, while sacrificing less than 5% accuracy. For 5-NNs and 10-NNs, we find the optimal LSH parameters are (10, 70) and (10, 80), respectively, and we get retrieval times 7 and 6.3 times faster with COSLSH than the linear KNN search.

E2LSH Figure 4-4 shows the trade-off between the retrieval accuracy and speed-up factor for E2LSH. The square boxes indicate the optimal parameters for each k. For 1-NN, with $(m, L) = (100, 40)$, we achieve the optimal nearest neighbor retrieval

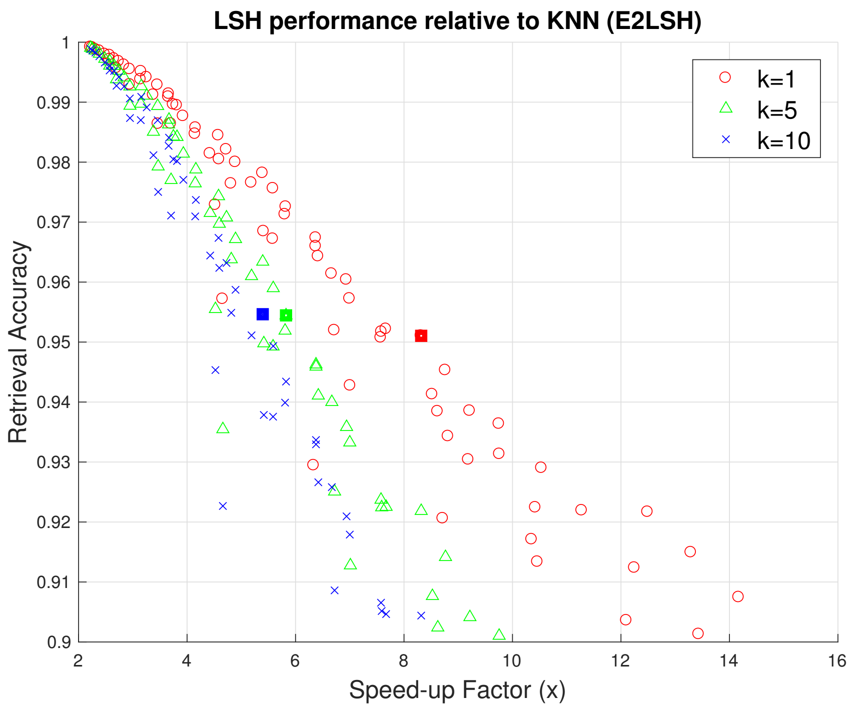


Figure 4-4: Trade-off between retrieval accuracy and speed-up factor of E2LSH performance relative to the linear KNN. Each point corresponds to a parameter configuration of LSH. Squared points indicate the optimal parameters for different values of k.

8.2 times faster with E2LSH than the linear search, while sacrificing less than 5% accuracy. For 5-NNs and 10-NNs, we find the optimal LSH parameters are (100, 60) and (100, 70), respectively, and we get retrieval times 5.8 and 5.6 times faster with E2LSH than the linear KNN.

From the above observations, it may look like there exists a heuristic that only the optimal L increases for increasing k while the optimal m remains unchanged. However, when we examine this heuristic on other datasets besides $\text{Data}_{\text{Lag 300, 1x}}$, the heuristic does not hold and the pattern in our observations is likely to be a coincidence.

Comparison In Figure 4-5, we compare L1LSH, COSLSH, and E2LSH on a wider range of retrieval accuracy (y-axis) from 0.5 to 1. While there is not much difference between them in the region of high retrieval accuracy and low speed-up, the general trend shows that COSLSH has a higher speed-up than L1LSH and that L1LSH has

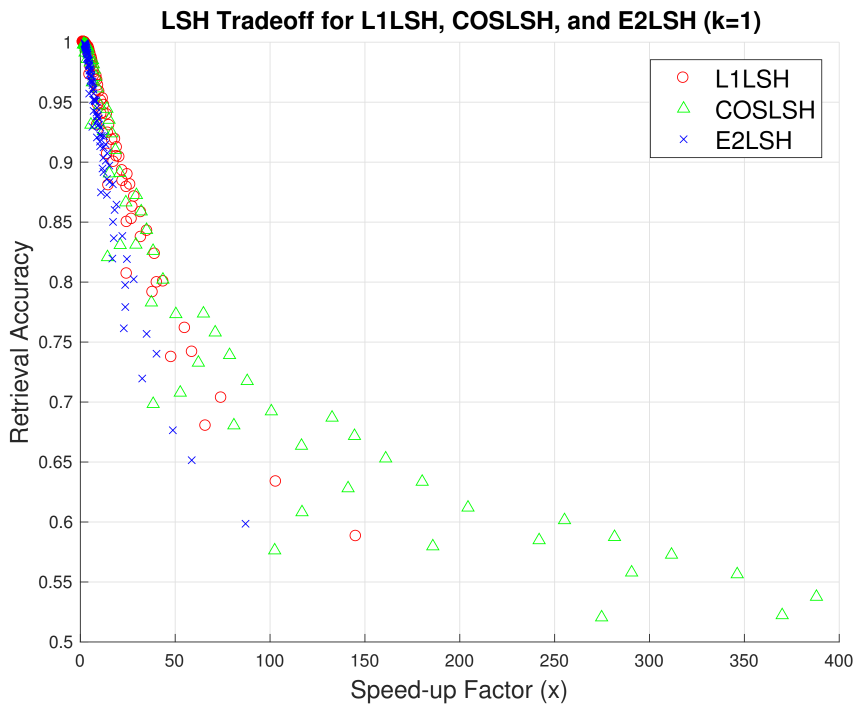


Figure 4-5: Comparison of the retrieval accuracy and speed-up trade-off profiles of L1LSH, COSLSH, and E2LSH. Each point corresponds to a parameter configuration of LSH. For a given accuracy, speed-ups are in the order of COSLSH, L1LSH, and E2LSH from the fastest to the slowest.

a higher speed-up than E2LSH for a given retrieval accuracy. We explain that the difference comes from the difference in the distribution of hash bucket sizes among L1LSH, COSLSH, and E2LSH. When constructing hash tables, points are distributed and indexed more uniformly with COSLSH than L1LSH and E2LSH, and thus result in a smaller average selectivity leading to a higher speed-up. This phenomenon will be explained in more depth in Section 5.4.3.

4.4.2 Sensitivity Analysis

The performance of LSH is known to be sensitive to several parameters. In this section, we present the sensitivity analysis of retrieval accuracy and speed-up factor with respect to the number of hash functions and the number of hash tables. We use L1LSH as a representative to demonstrate. Figure 4-6 shows such analysis for retrieval accuracy. From Figure 4-6 (Top), we observe that the retrieval accuracy decreases with increasing m. As hash buckets become more fine-grained with increasing m, there is a larger chance for two similar points to belong to different buckets if hash buckets are too fine-grained. Figure 4-6 (Bottom) shows that the retrieval accuracy increases with increasing L. The improvement diminishes for large L as there are more overlaps among the elements from colliding hash buckets for large L.

Sensitivity analysis of the speed-up factor is presented in Figure 4-7. Figure 4-7 (Top) shows that the speed-up factor increases exponentially as m becomes larger. As more hash functions are applied to hash, the number of possible hash buckets become exponentially large (e.g. 2^m for binary hash functions). Thus, the average number of points belonging to a single hash bucket will decrease accordingly, resulting in a smaller selectivity and larger speed-up. In Figure 4-7 (Bottom), we observe that the speed-up factor exponentially decreases as L increases. This is expected because as we add more hash tables to increase recall, the size of the candidate set becomes larger. This leads to a linear search (short-listing) on a larger search space, leading to a decreased speed-up.

4.4.3 Impact of Dimension and Quantity of Data

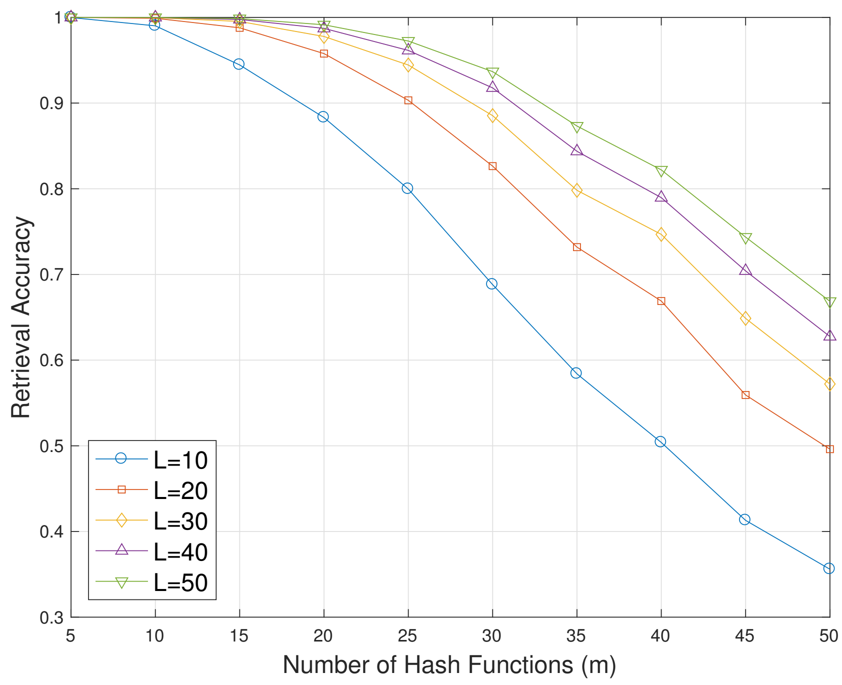
For retrieval, the lag duration corresponds to the dimensionality of data. As discussed in Chapter 3, by reducing the lag duration, we are able to extract a much larger dataset, allowing us to examine scaling. Figure 4-8 (Top) shows the comparison of L1LSH on three different datasets ($\text{Data}_{\text{Lag 300, 1x}}$, $\text{Data}_{\text{Lag 30, 1x}}$, and $\text{Data}_{\text{Lag 30, 10x}}$) for retrieval based on 1-NN. From this, we can deduce two effects: the impact of the data dimensionality and of the data quantity on the retrieval accuracy and querying speed.

First, we can observe the impact of the dimensionality by comparing the accuracy-speed trade-off profiles between $\text{Data}_{\text{Lag 300, 1x}}$ (red) and $\text{Data}_{\text{Lag 30, 1x}}$ (green). The accuracy-speed trade-off profile of $\text{Data}_{\text{Lag 30, 1x}}$ lies above that of $\text{Data}_{\text{Lag 300, 1x}}$. For a given speed-up, we observe that the retrieval is more accurate on the data with a lower dimensionality. It agrees with the general perception that comparing time series with a lower dimensionality is more accurate than that with a higher dimensionality as the distance measures become less effective as the dimensionality becomes higher.

Second, the impact of the data quantity can be deduced by comparing the accuracy-speed trade-off profiles between $\text{Data}_{\text{Lag 30, 10x}}$ (blue) and $\text{Data}_{\text{Lag 30, 1x}}$ (green). We observe that the accuracy-speed trade-off profile of $\text{Data}_{\text{Lag 30, 10x}}$ sits well above that of $\text{Data}_{\text{Lag 30, 1x}}$. It implies that for a given accuracy, the speed-up effect is much higher on a larger dataset. This agrees with the property of LSH that it has a sub-linear time complexity to the size of the dataset. Thus, as the size of data becomes larger, the efficiency of LSH will be more apparent. We observe the identical patterns with COSLSH and E2LSH, as presented in Figure 4-8 (Middle) and Figure 4-8 (Bottom), respectively.

4.5 Chapter Conclusion

We proposed a fast, yet accurate and scalable locality-sensitive hashing method for the retrieval problem of finding similar physiological waveform time series. We demonstrated effectiveness of this method on our arterial blood pressure time series repository extracted from the MIMIC II database. With LSH for various distance metrics, we achieved an order of magnitude speed-up with the cost of decreasing the retrieval accuracy by 5% compared to KNN. With this efficient retrieval method, we extend our work to the problem of predicting acute, critical events in intensive care units in the next chapter.



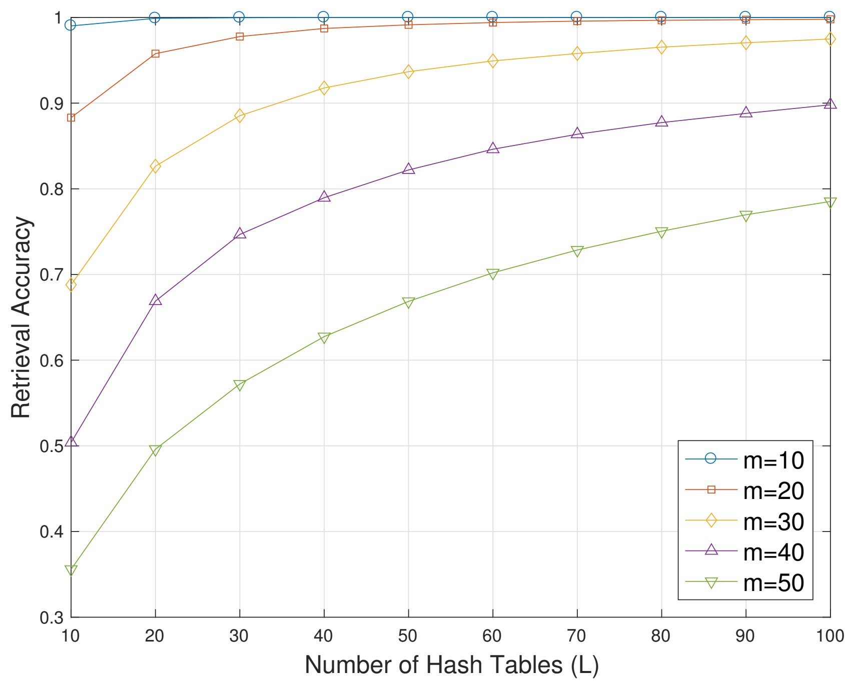
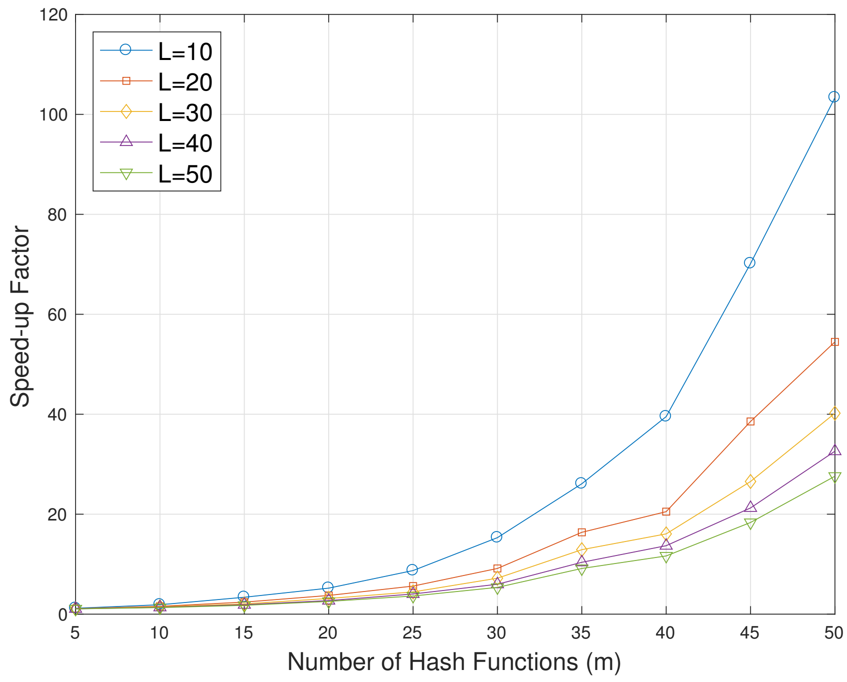


Figure 4-6: Sensitivity analysis of the retrieval accuracy with respect to (*Top*) the number of hash functions (m) per table and (*Bottom*) the number of hash tables (L) for L1LSH.



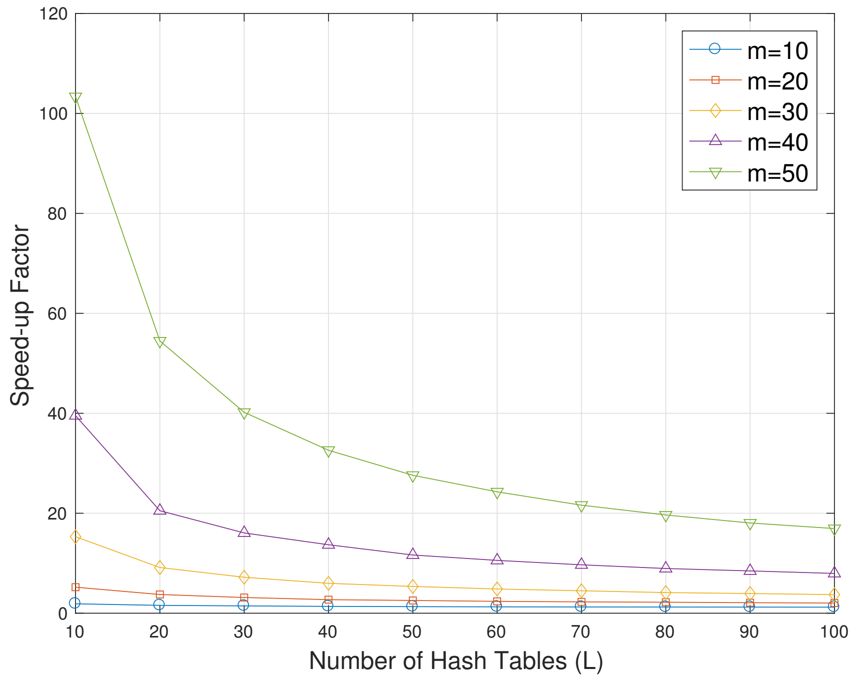


Figure 4-7: Sensitivity analysis of the speed-up factor with respect to (*Top*) the number of hash functions (m) per table and (*Bottom*) the number of hash tables (L) for L1LSH.

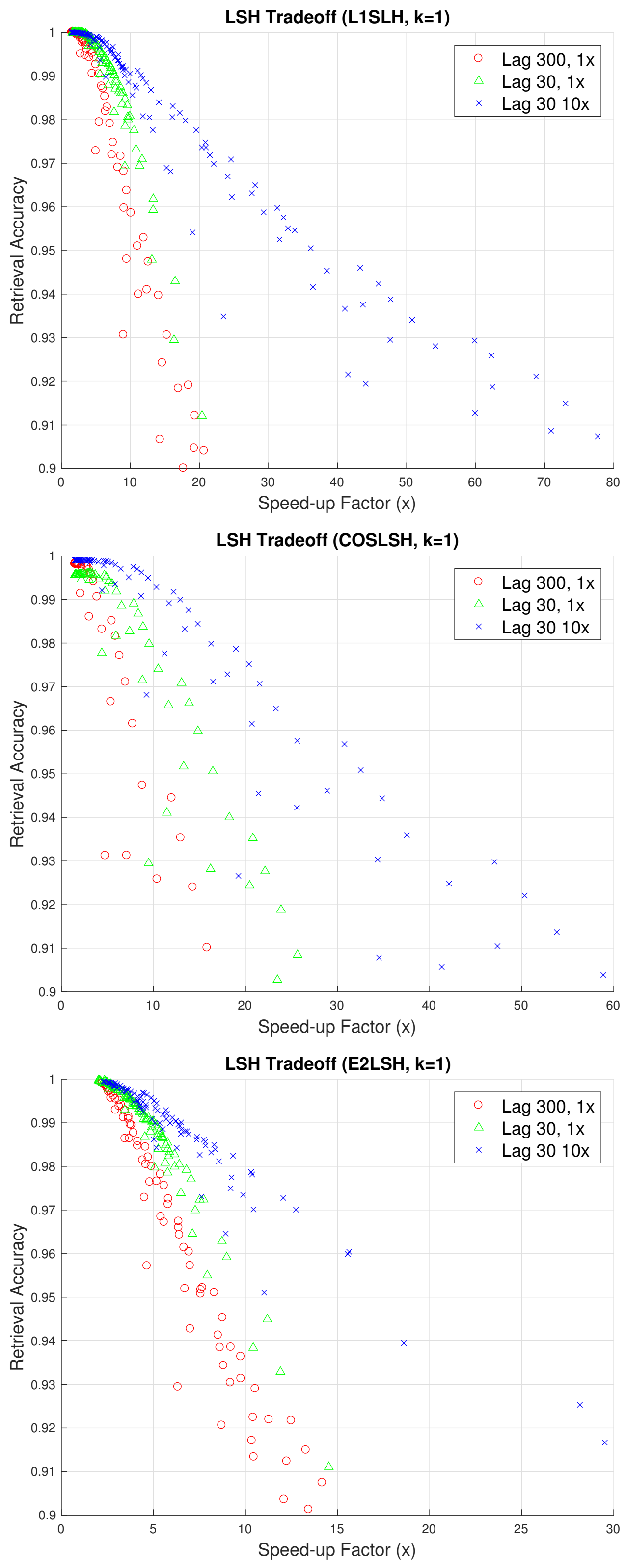


Figure 4-8: Trade-off between retrieval accuracy and speed-up factor of LSH relative to KNN on $\text{Data}_{\text{Lag 300, 1x}}$, $\text{Data}_{\text{Lag 30, 1x}}$, and $\text{Data}_{\text{Lag 30, 10x}}$ under (*Top*) L1 (*Middle*) cosine, and (*Bottom*) Euclidean distances. Using a smaller dimensional data and a larger quantity of data each improves accuracy and speed-up.

Chapter 5

Locality-Sensitive Hashing for Critical Event Prediction

This chapter addresses the question of whether a high quality similarity-based retrieval set of arterial blood pressure waveforms can effectively be leveraged for prediction of a critical event in the intensive care unit (ICU). To answer this, we extend the locality-sensitive hashing (LSH) based waveform retrieval method to the task of predicting acute hypotension by majority discrimination. The prediction results are thoroughly investigated with performance measures such as accuracy, correlation, and the ability to detect false negatives. We also investigate the question of how lag duration and quantity of data impact the prediction performance and what it means in the medical context. From our experiments, we find that LSH largely speeds up the querying time with a negligible decrease in prediction accuracy as a cost, while the large difference in the querying times of LSH based on different distance metrics originates from the difference in their hash bucket distributions. We also observe that using a longer lag and a larger quantity of data each improves accuracy and speed-up. The main content of this chapter was published in [74].

5.1 Motivation

In data-driven medicine, fast yet accurate prediction of acute and critical events based on time series signal data from patient monitors is crucial especially in intensive care units. In such settings, everything is very time critical, so if a task can be completed dramatically faster, it is often acceptable to tolerate a modest amount of approximation. In the previous chapter, we introduced the LSH-based scalable retrieval system for high-dimensional massive physiological time series data, which has a significantly faster querying time while still maintaining the retrieval quality in a competitive range in comparison to the linear k-nearest neighbor (KNN) method. The eventual purpose of retrieving patients with similar time series segments is to make an inference about certain states or conditions of the query based on information that can be leveraged from such neighbors. We are particularly interested in the problem of making a prediction on the future medical condition of a query patient based on the result of a quick retrieval of waveforms similar to that patient's.

Thus, in this chapter, we extend the LSH-based retrieval system to just-in-time critical event prediction. Our ultimate goal is to obtain an accurate *and* fast prediction for the query about an acute event that lies ahead in time with the similarity based methods. The similarity based prediction via LSH or KNN is essentially a two-step process of first quickly retrieving *patients with trajectories like mine*, the nearest neighbors (NNs) of our query of interest by LSH or KNN, and second, extrapolating the information of nearest neighbors (such as their labels) for prediction via majority vote.

Similar to the retrieval problem, we demonstrate LSH-based prediction with L1LSH, E2LSH, and COSLSH on the dataset of mean arterial blood pressure extracted from the MIMIC II database. Our event of interest for prediction is an acute hypotensive episode, explained in Chapter 3.

In this chapter, we ask the following research questions:

- how the querying time saved by LSH is related to a decrease in prediction accuracy,

- whether one LSH method (L1LSH, COSLSH, or E2LSH) has better performance than the others in terms of prediction accuracy and querying speed,
- whether the rank among them changes when compared with alternative performance measures in terms of correlation and detecting false negatives,
- why there exists a large variability in the querying time among different LSH methods,
- how diminishing retrieval accuracy due to approximation affects prediction accuracy, and
- how the performance of prediction based on LSH depends on the lag time and the quantity of data.

5.2 Method

The prediction by LSH is built on top of the standard LSH for retrieval. We then use majority vote only among the small approximate NN set as a means of extrapolation for prediction. The prediction by LSH consists of the following three steps:

1. constructing an efficient data structure (hash table) to *index* (hash) the data for fast retrieval (Section 4.2.2, Algorithm 1),
2. quickly *retrieving* the approximate NNs of the query of interest (Section 4.2.3, Algorithm 2), and
3. *predicting* the label of the query based on predominance of labels of its NNs (Algorithm 3).

A weighting rule is the rule that defines w_i, the weight for each neighbor (i.e. how to combine the labels of the nearest neighbors). Typically, the majority rule with equal weights ($w_i = 1/k$) is applied (as in our work), but more sophisticated approaches can also be used with the cost of extra computation, for example, where neighbors are given weights according to the inverse of their distances to the query.

Algorithm 3 k-Nearest Neighbor Classification

```
Require: $q$, the query; $k$, the number of nearest neighbors retrieved; $K(q) = \{x_i, y_i\}_{i=1}^{k}$, $k$-nearest neighbors of $q$.
1: procedure KNN-Classification($q, k, K(q)$)
2:     $\hat{y}_q \leftarrow \arg\max_v \sum_{(x_i,y_i)\in K(q)} w_i \cdot I(v = y_i)$    ▷ $v$ denotes each valid class label. $w$ denotes sample weights. $y_i$ is the label of the point $x_i$.
3:     Return $\hat{y}_q$, the predicted label for the query $q$
4: end procedure
```

5.3 Experiment

We present the experimental set-up for our investigations regarding the predictive performance and querying speed of LSH on the acute hypotensive episode (AHE) dataset described in Chapter 3.

AHE Prediction We build the prediction models based on LSH and KNN for the occurrence of AHE within an event window. We build the models with a lag time amount of historical data prior to the window, where each data point x_i has a label y_i indicating the occurrence of AHE. We formulate this prediction task as a supervised binary classification problem, where predicted labels $\hat{y}_i$ describe the patients as AHE-positive or negative.

Performance Measures We evaluate performance of predictors mainly with:

- Prediction Accuracy $= (TP + TN)/(TP + FP + TN + FN)$[1], and
- Speed-up factor, which is the average time taken by LSH relative to that by KNN, measured by the inverse of *selectivity* (defined in Section 4.3).

However, accuracy is known to be heavily influenced by the class imbalance in the data (as in our dataset). Thus, additionally, we measure performance with an alternative measure, namely Matthew's correlation coefficient (MCC).

- MCC $= \dfrac{(TP \times TN - FP \times FN)}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$

MCC measures the correlation between observed and predicted binary classification labels. It is regarded as the measure of the quality of binary classification that is

[1] TP = True Positive, FP = False Positive, FN = False Negative, TN = True Negative.

less sensitive and more robust to the class imbalance and data size [102]. We choose to use MCC instead of F-score since the latter sensitively depends on a weighting parameter β between precision and recall. MCC has the range from -1 to 1.

One of the most critical mistakes that can happen in the ICU setting is predicting condition positive patients as condition negative and consequently missing a chance for appropriate intervention. The false negative weighted accuracy (FNWA) weighs these false negative cases more heavily than other cases in the confusion matrix.

- FNWA $= (TP + TN)/(TP + FP + TN + \alpha \times FN)$

Here, we use the factor of $\alpha = 5$ arbitrarily. Like the prediction accuracy, it has a minimum value of 0 and a maximum value of 1.

Number of Nearest Neighbors In this study, we make predictions mainly based on one nearest neighbor (1-NN) for two reasons. First, as we will see in Section 5.4.1, we show that the prediction accuracy based on 1-NN is higher than that of any larger number of nearest neighbors. Second, 1-NN test allows better comparisons among methods because it eliminates the need for hyper-parameter tuning on the appropriate number of nearest neighbors and which neighbor weighting rule to use.

Experimental Procedure We apply L1LSH, COSLSH, and E2LSH for prediction. For any LSH, we vary m from 5 to 100 with an increment of 5 and L from 10 to 100 with an increment of 10. We apply the above parameter configurations to retrieve and predict based on 1-NN, 5-NNs, and 10-NNs. We demonstrate the prediction of AHE by LSH mainly on $\text{Data}_{\text{Lag 300, 1x}}$ as a representative. We additionally apply LSH on $\text{Data}_{\text{Lag 30, 1x}}$ and $\text{Data}_{\text{Lag 30, 10x}}$ with 1-NN to measure the impact of lag duration and of scaling.

5.4 Results and Discussion

We present the results of AHE prediction with various performance measures for prediction accuracy and discuss the source of the large difference in speed-ups among different LSH methods. Also, we discuss the impact of lag and the quantity of data on the prediction accuracy and speed-up factors.

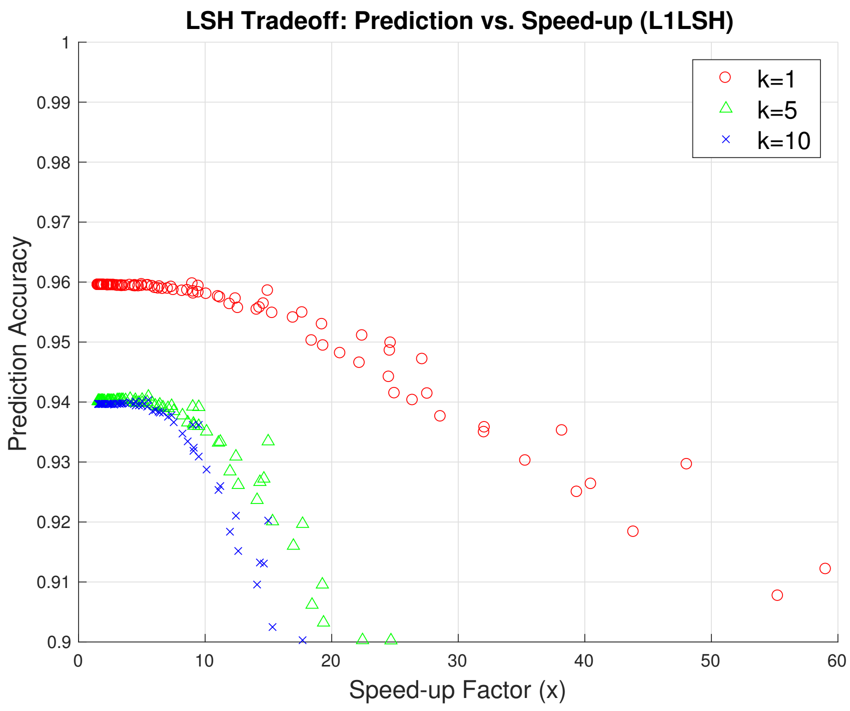


Figure 5-1: Trade-off between prediction accuracy and speed-up factor of L1LSH relative to the linear KNN. Each point corresponds to a parameter configuration of LSH.

5.4.1 Acute Hypotensive Episode Prediction

Figure 5-1 shows the prediction accuracy and speed-up trade-off profile of L1LSH for $k = 1, 5, 10$. Each point corresponds to the average result from a pair of the number of hash functions (m) used per table and the number of hash tables (L) over 10 runs. For fixed m and L, we observe that only the prediction accuracies degrade with an increasing k while the prediction times remain almost the same. On the other hand, for the same loss of accuracy, we observe that the speed-up factor is much lower for a larger k. Thus, making predictions based on a larger number of similar waveforms does not improve the prediction quality and rather introduces noise.

This general pattern of the prediction accuracy being the highest with $k = 1$ is also observed with E2LSH (Figure 5-2) and COSLSH (Figure 5-3). Thus, along with the reasons explained in Section 5.3, for the remaining experiments, we demonstrate our results only with $k = 1$.

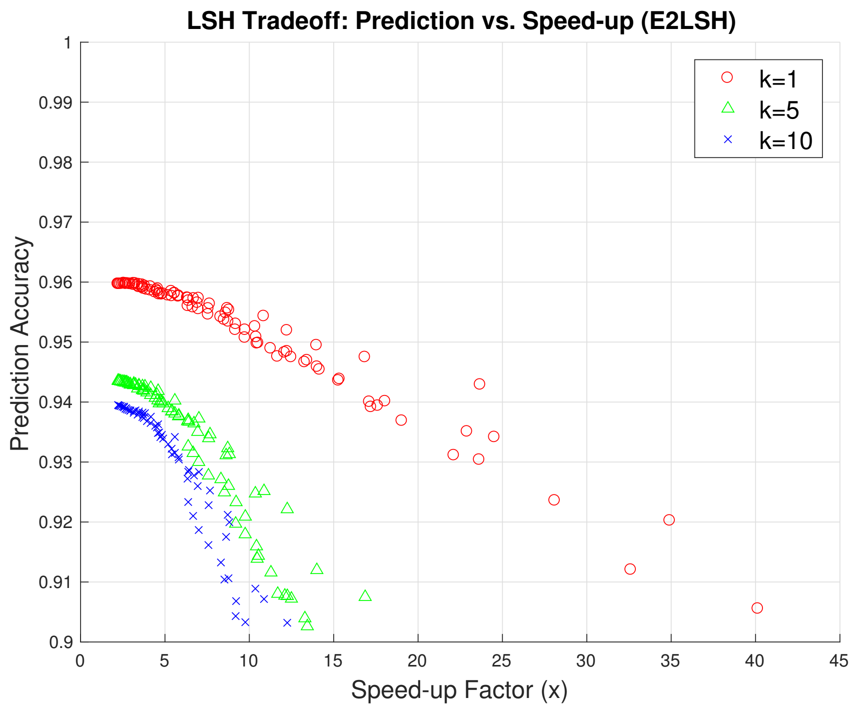


Figure 5-2: Trade-off between prediction accuracy and speed-up factor of E2LSH relative to the linear KNN. Each point corresponds to a parameter configuration of LSH.

We measure how much speed-up gain we achieve for each LSH method when we allow a 1% decrease in prediction accuracy as a cost. From Figure 5-1, we observe that L1LSH becomes 25 times faster than the linear KNN search (95.95% accuracy) when we sacrifice 1% accuracy (94.95%). With E2LSH, we achieve a smaller speed-up of 14x with 1% decrease in prediction accuracy (from 95.95% to 94.95%) (Figure 5-2). On the other hand, as shown in Figure 5-3, while the maximum prediction accuracy of COSLSH (93.74%) is lower than that of L1LSH and E2LSH, it becomes 249 times faster than KNN for just a 1% drop in accuracy (92.74%).

Figure 5-4 presents the prediction accuracies of L1LSH, COSLSH, and E2LSH and the associated speed-up factors with $k = 1$ all together. As another basis for comparing prediction accuracy, we use the performance of the "all-negative" predictor. It is a dummy predictor which classifies condition negative for all queries since our data is highly skewed toward AHE negatives. It has a prediction accuracy of 92.64%. Our predictors based on LSH should do better than the all-negative predic-

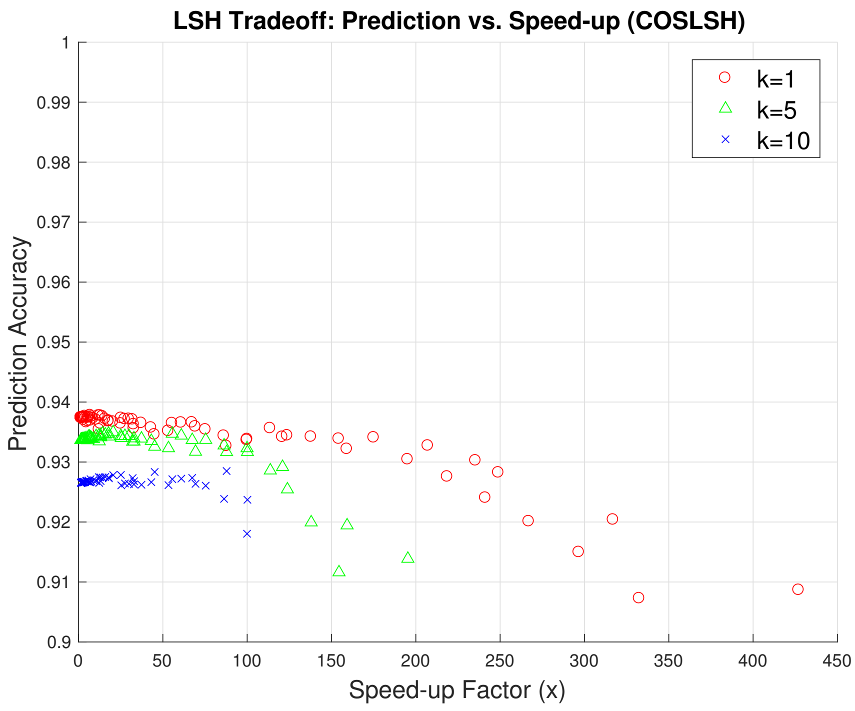


Figure 5-3: Trade-off between prediction accuracy and speed-up factor of COSLSH relative to the linear KNN. Each point corresponds to a parameter configuration of LSH.

tor in order to be meaningful. From the figure, we observe that L1LSH and E2LSH have comparable accuracies to each other while L1LSH is faster than E2LSH as the accuracy-speed trade-off profile of L1LSH sits above that of E2LSH. Their maximum speed-ups at a point where their prediction accuracies cross with that of the dummy predictor are 49 and 24 for L1LSH and E2LSH, respectively. On the other hand, COSLSH has a maximum accuracy of 93.74%, which is lower than that of L1LSH and E2LSH (95.95%). However, it shows a significantly better ability to speed up the querying time (249x). There are two explanations for this trend. First of all, the lower prediction accuracy comes from the information loss during the normalization of data to be used for COSLSH. For a much higher speed-up, it originates from the fact that the average selectivity of COSLSH is much smaller than that of L1LSH and E2LSH. This is the topic of discussion in detail in Section 5.4.3.

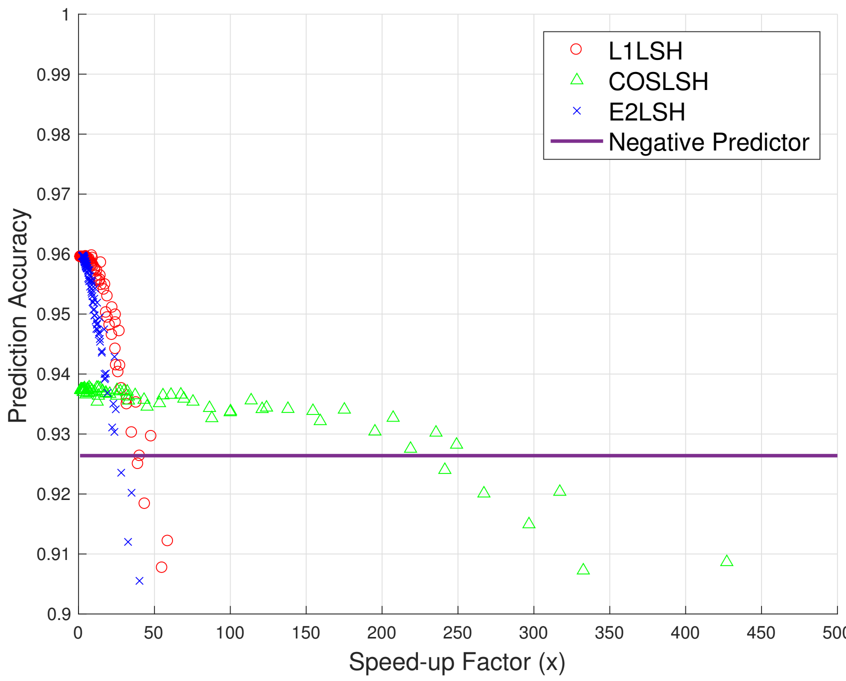


Figure 5-4: Comparison of the prediction accuracy and speed-up trade-off profiles of L1LSH, COSLSH, and E2LSH for prediction based on 1-NN ($k = 1$). Each point corresponds to a parameter configuration of LSH.

5.4.2 Performance under Alternative Measures

As the prediction accuracy is known to be heavily influenced by the class imbalance, we provide analysis with alternative performance measures for prediction. We measure the performances of L1LSH, E2LSH, and COSLSH using Matthew's correlation coefficient (MCC) and false negative weighted accuracy (FNWA). We first look at the performance with MCC. In Figure 5-5, we observe that the MCC values of L1LSH, E2LSH, and COSLSH all lie well above that of the dummy all-negative baseline while having significant speed-ups. The MCC of the baseline all-negative predictor is zero, confirming that there is no correlation between the observed and predicted labels when predicting AHE negative for all queries. Within the range of LSH parameters we tried, we achieve a maximum speed up of 147x, 81x, and 456x with L1LSH, E2LSH, and COSLSH, while their MCC values are larger (by a large margin of 0.4) than that of the dummy predictor.

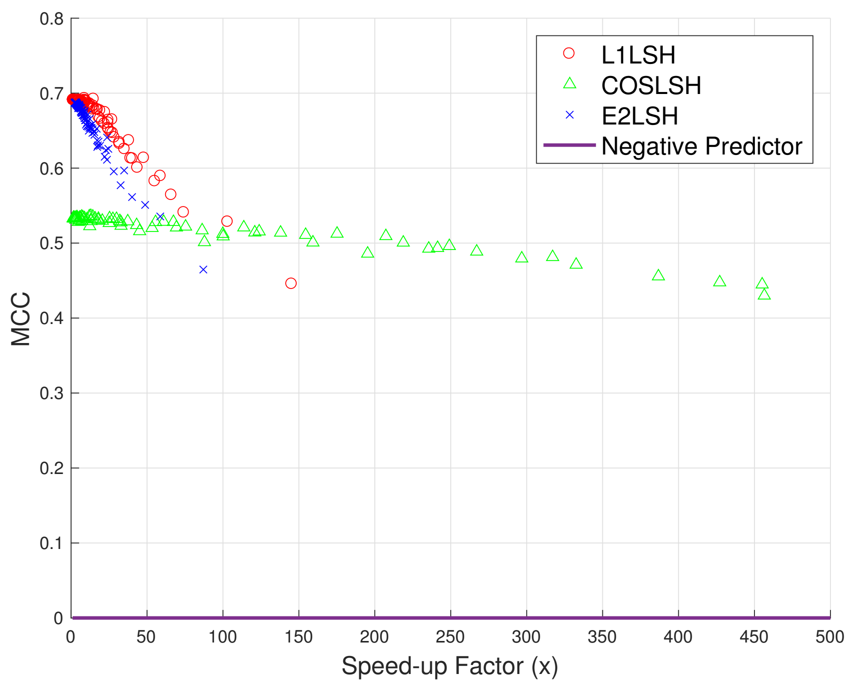


Figure 5-5: Comparison of MCC and speed-up trade-off profiles of L1LSH, COSLSH, and E2LSH for prediction based on 1-NN ($k = 1$). Each point corresponds to a parameter configuration of LSH.

We also look at the performance in terms of FNWA, which assigns a heavier weight on the false negative cases (i.e. AHE patients being classified as healthy) in the confusion matrix. In Figure 5-6, we observe that FNWAs of L1LSH, E2LSH, and COSLSH are all well above that of the dummy all-negative predictor by a margin of approximately 0.1. Within the range of LSH parameters we experimented with, we achieve a maximum speed up of 147x, 81x, and 456x with L1LSH, E2LSH, and COSLSH, while their FNWA values are larger than that of the dummy predictor. This implies that the prediction by LSH is actually effective in predicting the most detrimental cases of AHE in the ICU.

5.4.3 Hash Table Bucket Distribution

We address the question of why there exists a large difference in the speed-up factors among L1LSH, COSLSH, and E2LSH. We hypothesize that it originates from the

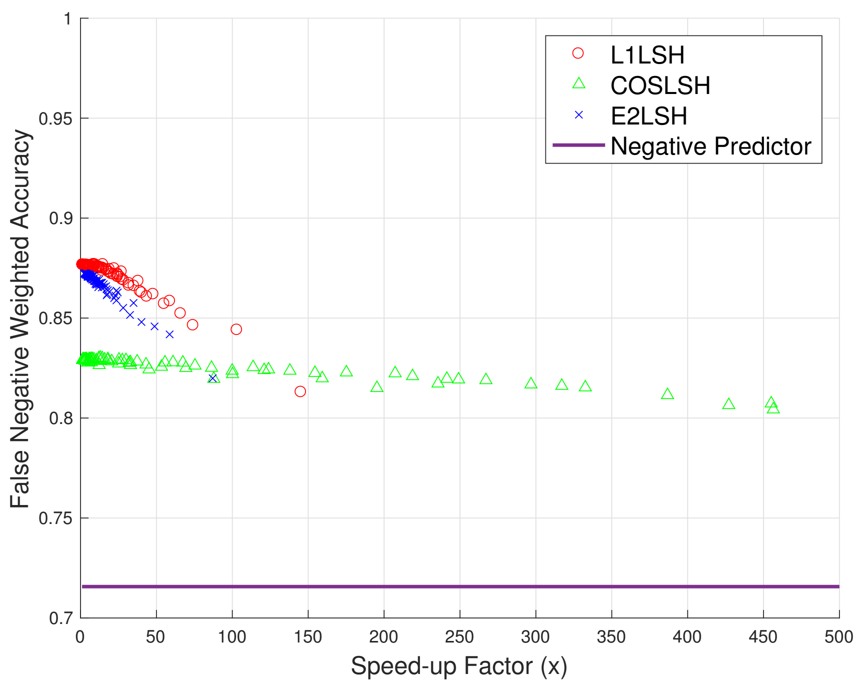


Figure 5-6: Comparison of FNWA and speed-up trade-off profiles of L1LSH, COSLSH, and E2LSH for prediction based on 1-NN ($k = 1$). Each point corresponds to a parameter configuration of LSH.

difference in the distribution of hash bucket sizes. In order for LSH to have a good speed-up, we desire that data are hashed uniformly over all valid hash buckets with a small number of data points in each bucket. If so, during the querying step, the selectivity of querying (e.g. size of the candidate set subject to the linear short-listing) is minimal, leading to a large speed-up. But in reality, this might not be possible assuming data always has some structure in it. We empirically validate our hypothesis for two cases where the number of hash functions used (m) is small and large on $\text{Data}_{\text{Lag 300, 1x}}$.

Figure 5-7 (Top) shows the average bucket size distribution under L1LSH, COSLSH, and E2LSH with $m = 5$. Hash buckets are sorted in terms of their size in descending order on the horizontal axis. Accumulated (normalized) data size is represented on the vertical axis. Cumulated data of 1 denotes the size of the entire data. For example, the fifth largest hash bucket having a cumulated data size of 0.9 means that the top 5 largest buckets contain 90% of the total data. The closer the plot is to the diagonal line, the more uniformly data is indexed across all valid hash buckets. We observe that the data are indexed more evenly over hash buckets with COSLSH, whereas the majority of data belong in the few largest buckets with L1LSH and especially E2LSH. Out of 32 possible hash buckets, the largest bucket of E2LSH and L1LSH contains 78.90% and 43.59% of the total data each, whereas for COSLSH, the largest bucket contains only 8.43% of the data. When examining the top 5 largest buckets, E2LSH and L1LSH hold 99.73% and 89.09% of the total data in those buckets, while COSLSH has only 32.11%. This observation explains why COSLSH is significantly faster than L1LSH and E2LSH.

Figure 5-7 (Bottom) shows the average bucket size distribution under L1LSH, COSLSH, and E2LSH when data is indexed with a larger number of hash functions, $m = 50$. Like the previous case with a small m, COSLSH hashes the data more evenly across all hash buckets (closer to the diagonal line), whereas L1LSH and E2LSH are still highly skewed toward large buckets. Out of approximately 6,000 valid buckets, the largest top one percentile of buckets (60 buckets) under COSLSH hold only 4% of the total data while that number corresponds to 18.37% for L1LSH and 40.85% for E2LSH. Again, this explains the observation that the querying speed of COSLSH is the fastest, followed by L1LSH and E2LSH.

5.4.4 Impact of Retrieval on Prediction

We investigate the question of how the quality of retrieval impacts that of prediction. Figure 5-8 demonstrates the relationship between the relative loss of accuracy in retrieval and that in prediction for L1LSH with $k = 1$. We define the relative retrieval accuracy for each (m, L) as the fraction of the k-NNs that are retrieved by both L1LSH and KNN methods, and the overall relative retrieval accuracy is the average of the individual accuracies over all queries (defined in Section 4.3). Similarly, we define the *relative* prediction accuracy (c.f. absolute prediction accuracy) as the average ratio (LSH prediction accuracy)/(KNN prediction accuracy). One would expect that as the retrieval accuracy of finding correct "patients like me" declines, it would make a significant negative impact on the accuracy of prediction which extrapolates the information from such patients. However, we observe that the loss of accuracy due to approximation in retrieval, in exchange for vast gains in speed-up, has a non-linear diminishing impact on the loss of prediction accuracy. For a given unit of decrease in relative retrieval accuracy, the corresponding amount of decrease in relative prediction accuracy is much smaller (all points sitting well above the linear diagonal line in Figure 5-8). Although the rapidly found “approximate patients like me” by LSH were not as well matched to the query as the set obtained by the linear search, the prediction of AHE by LSH maintained over 90% relative prediction accuracy even when the speed-up gain was large.

5.4.5 Impact of Lag Duration and Data Quantity

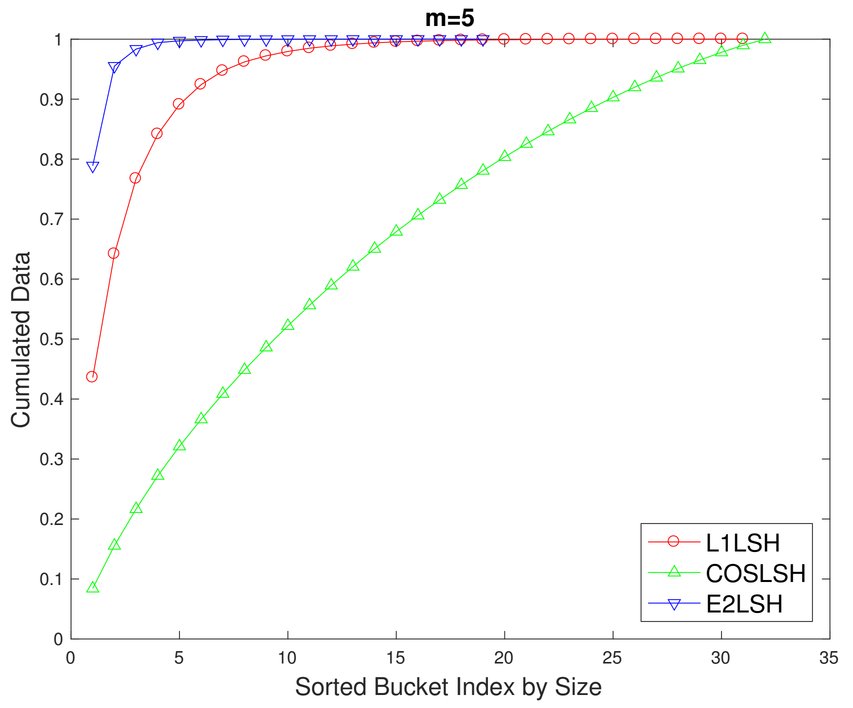
Investigating lag duration is driven by problem domain in the clinical setting as lag data may be limited or the value of effective lag duration may be informative to clinicians. In this case, the lag also impacts data size, allowing us to examine scaling (explained in detail in Section 3.4). Figure 5-9 (Top) shows the comparison of L1LSH on three different datasets ($\text{Data}_{\text{Lag 300, 1x}}$, $\text{Data}_{\text{Lag 30, 1x}}$, and $\text{Data}_{\text{Lag 30, 10x}}$) for prediction of AHE based on 1-NN. From this, we can deduce two effects: the impact of the lag duration and of the data quantity on the prediction accuracy and speed-up factor. First, we observe the impact of the lag duration by comparing the accuracy-speed trade-off profiles between $\text{Data}_{\text{Lag 300, 1x}}$ (red) and $\text{Data}_{\text{Lag 30, 1x}}$ (green). The leftmost points of each profile show the accuracy obtained by KNN with L1 on each dataset. We observe that, with a shorter lag time of 30 minutes, the maximum prediction accuracy (0.9581) is slightly lower than that with 300 minutes of lag time (0.9595). Plus, the trade-off profile of $\text{Data}_{\text{Lag 300, 1x}}$ lies well above that of $\text{Data}_{\text{Lag 30, 1x}}$. These observations all together imply that, for the purpose of predicting AHE, it is more beneficial to use 300 minutes of lag as it generates more accurate prediction outcomes and larger speed-ups. We observe that same pattern with COSLSH and E2LSH, as presented in Figure 5-9 (Middle) and Figure 5-9 (Bottom), respectively.

Second, the impact of the quantity of training data is highlighted by comparing the accuracy-speed trade-off profiles between $\text{Data}_{\text{Lag 30, 1x}}$ (green) and $\text{Data}_{\text{Lag 30, 10x}}$

(blue) in Figure 5-9 (Top) for L1LSH. Here, the maximum prediction accuracy when trained with 10x data (0.9748) is much higher than that when trained with 1x data (0.9581). The trade-off profile of $\text{Data}_{\text{Lag 30, 10x}}$ is located well above that of $\text{Data}_{\text{Lag 30, 1x}}$. Therefore, more data results in higher accuracy and larger speed-up. LSH scales well with accumulation of data and can improve prediction when there is more physiological data available. Figure 5-9 (Middle) and Figure 5-9 (Bottom) confirm that this pattern with respect to the quantity of dataset is also valid with COSLSH and E2LSH.

5.5 Chapter Conclusion

In this chapter, we applied the sublinear time, scalable locality-sensitive hashing and majority discrimination to the problem of predicting AHE based on physiological waveform time series. Compared to using the linear k-nearest neighbor search, our proposed method vastly speeds up prediction time up to an order (with L1LSH and E2LSH) and two orders (with COSLSH) of magnitude faster while sacrificing less than 1% of prediction accuracy as a cost. We found that the large difference in the querying times of LSH based on different distance metrics originates from the difference in their hash bucket distributions and that using a longer lag and a larger quantity of data each improves accuracy and speed-up. We also observed the non-diminishing impact of retrieval quality on the prediction accuracy. In the next two chapters, we propose two new variants of LSH that lead to better performance on the AHE prediction problem.



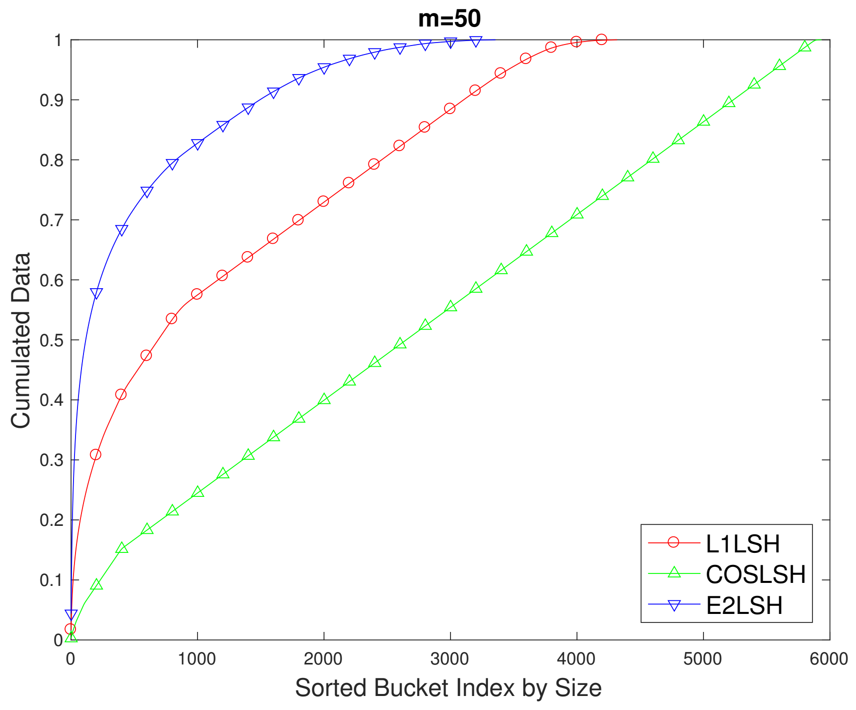


Figure 5-7: Average bucket size distribution under L1LSH, COSLSH, and E2LSH with (*Top*) $m = 5$ and (*Bottom*) $m = 50$ on $\text{Data}_{\text{Lag 300, 1x}}$. Hash buckets are sorted in terms of their size in descending order on the horizontal axis. Accumulated normalized data size is represented on the vertical axis (1 denotes the size of the entire data). The closer the plot is to the diagonal line, the more uniformly the data is indexed across all valid hash buckets.

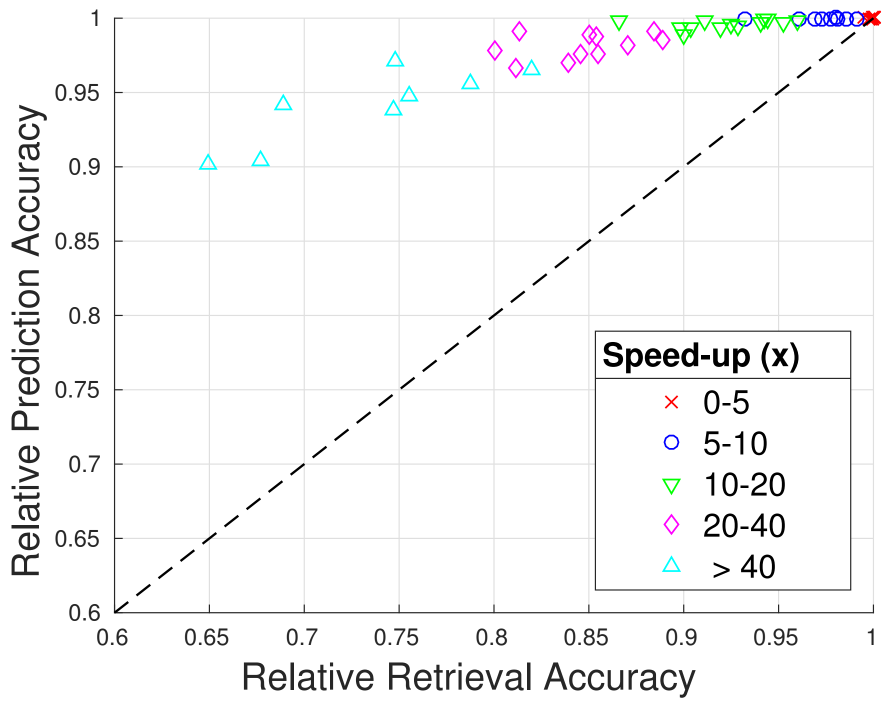


Figure 5-8: Relative retrieval versus prediction accuracy of L1LSH to the linear KNN search for $k = 1$. Each point corresponds to a parameter configuration (m, L) of LSH.

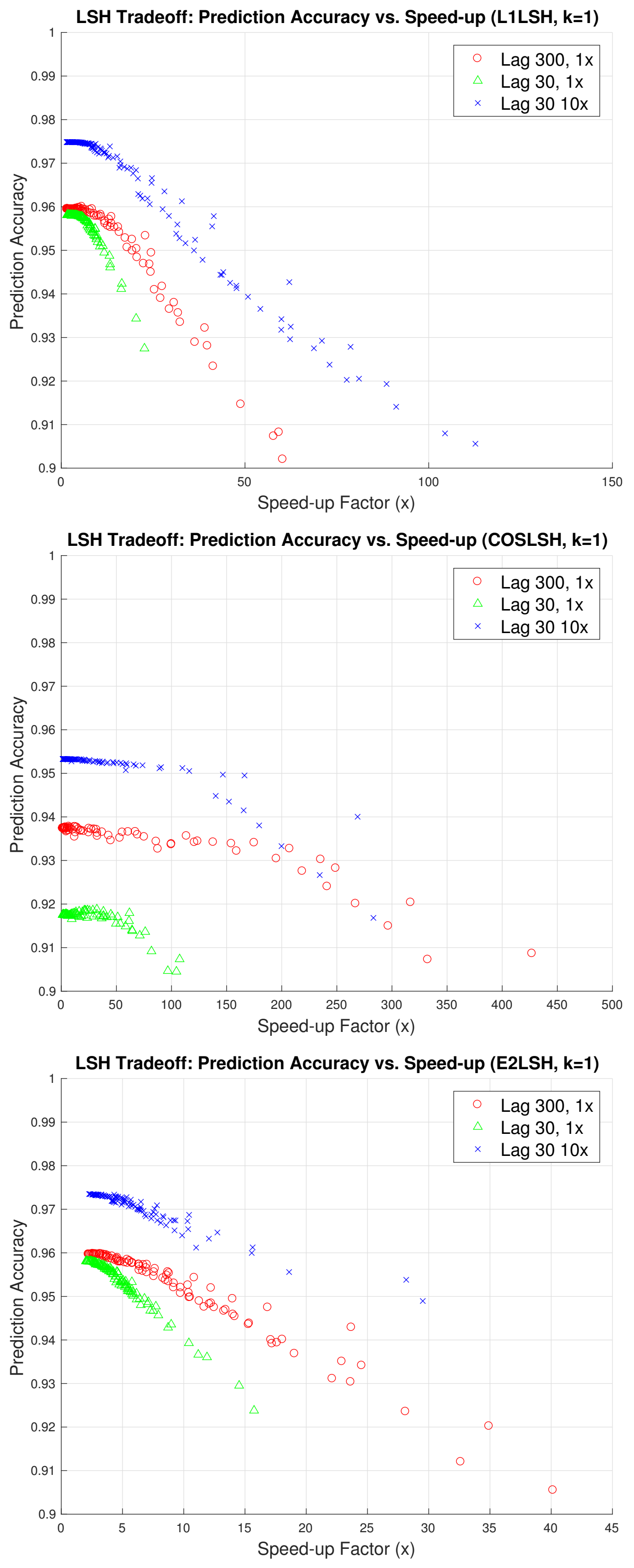


Figure 5-9: Trade-off between prediction accuracy and speed-up factor of the LSH relative to KNN on $\text{Data}_{\text{Lag 300, 1x}}$, $\text{Data}_{\text{Lag 30, 1x}}$, and $\text{Data}_{\text{Lag 30, 10x}}$ with (*Top*) L1LSH (*Middle*) COSLSH, and (*Bottom*) E2LSH. Using a longer lag and a larger quantity of data each improves prediction accuracy and speed-up.

Chapter 6

Stratified Locality-Sensitive Hashing

When sensor stream data such as blood pressure come from a highly complex source like the human body, a single metric is not sufficient to capture its coalesced enigmatic underlying properties. Finding trajectories similar to a given query from such data requires an integrated multi-metric strategy to accurately express underlying semantic similarity. Herein we propose a new similarity based prediction technique called stratified locality-sensitive hashing (SLSH), which finds similarity among the data from a more integrated perspective by employing multiple distance metrics in one framework. Demonstrated on the problem of predicting acute hypotensive episodes, we show that SLSH not only achieves higher prediction accuracy, but also faster querying speed in comparison to the standard locality-sensitive hashing based prediction. The main content of this chapter was published in [70, 71] and was also submitted to a conference for review at the time of thesis writing.

6.1 Motivation

When utilizing locality-sensitive hashing (LSH), the appropriate choice of distance metric for measuring similarity is critical because of the one-to-one relationship between a distance metric and its unique corresponding family of locality-sensitive hash functions. Typically, a single locality-sensitive hash function family offers only one perspective on the data with its associated distance metric. For example, as illus-

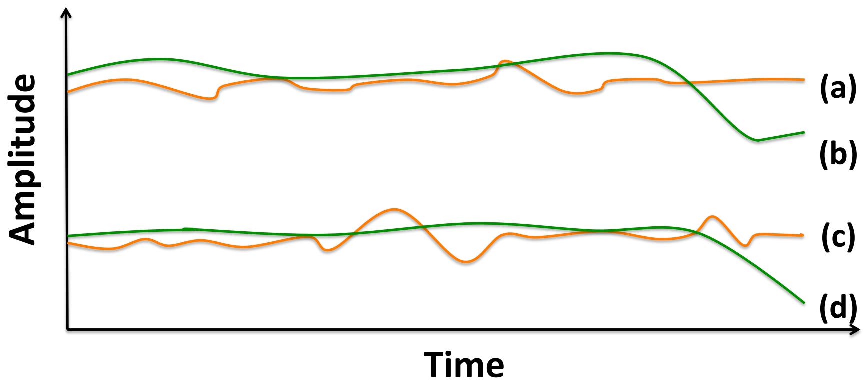


Figure 6-1: An illustration of hypothetical time series with different amplitudes and shapes. By the L1 distance, (a, b) and (c, d) are grouped as similar to each other whereas by the cosine distance, which requires normalization, (a, c) and (b, d) are grouped as similar.

trated in Figure 6-1, when either the L1 or the Euclidean distance is used as the distance metric, the similarity between waveform time series is mainly determined by the *amplitude* of the waveforms. On the other hand, when using the cosine distance, the notion of similarity is based on the *shape* or the *angle* between waveform vectors as it requires data to lie on a unit sphere.

The current limit of LSH is that it can hash the data with only one similarity measure at a time with its associated family of locality-sensitive hash functions. However, being limited to use only one distance metric to measure similarity introduces a semantic gap (the discrepancy between true similarity and what can be captured with a distance metric) and a loss of information because interpreting clinical physiological waveforms requires diverse perspectives on the data. Both the amplitude (e.g. mean blood pressure) and the shape of waveforms (e.g. trend and cycle frequency) contain important clinical information. For example, an acute hypotensive episode is defined as a sudden dropping (shape) of blood pressure to below 60 *mmHg* (amplitude) for a prolonged period of time. There are multiple facets of similarity (e.g., matching based on amplitude or shape of time series) and what constitutes similarity is not entirely quantifiable by a single distance metric. In practice, practitioners often do not precisely know which facets of similarity they are interested in. Or, they just ask for “something similar in general.” Therefore, due to its complex nature, in order to capture the true underlying semantic similarity in physiological signals, the data needs to be examined from multiple, more integrated perspectives. We achieve this with our proposed SLSH which provides a fast means of measuring similarity capable of integrating multiple distance metrics.

SLSH generates a hierarchy of hash tables. Each level of the hierarchy uses a different distance metric for hashing. Specifically, it is a multilevel LSH where,

1. the outer level LSH first stratifies the data by amplitude using the L1 distance, and then

2. hierarchically, within each stratum, the inner level LSH with the cosine distance hashes the data according to angle and shape of time series.

For a query, a set of candidate neighbors is retrieved from the colliding buckets at each level. Finally, a number of candidates are chosen as the final nearest neighbors, and we use majority vote among them to finalize the prediction.

In this chapter, we ask the following research questions, investigating:

- whether SLSH is advantageous and by how much, compared to the standard LSH and the linear k-nearest neighbor (KNN) method for the task of predicting acute hypotensive episodes (AHE),

- whether using multiple distance measures (via SLSH or an ensemble of multiple predictors each based on a different distance metric) improves the prediction performance in terms of accuracy and querying speed,

- whether the order of distance metrics used at the outer and inner level influences the performance of SLSH, and

- how the above performances scale as the lag duration and quantity of data change.

To the best of our knowledge, SLSH is the first scalable practical algorithm that integratively hybridizes multiple distance metrics in the LSH framework, demonstrated on a prediction task based on physiological time series.

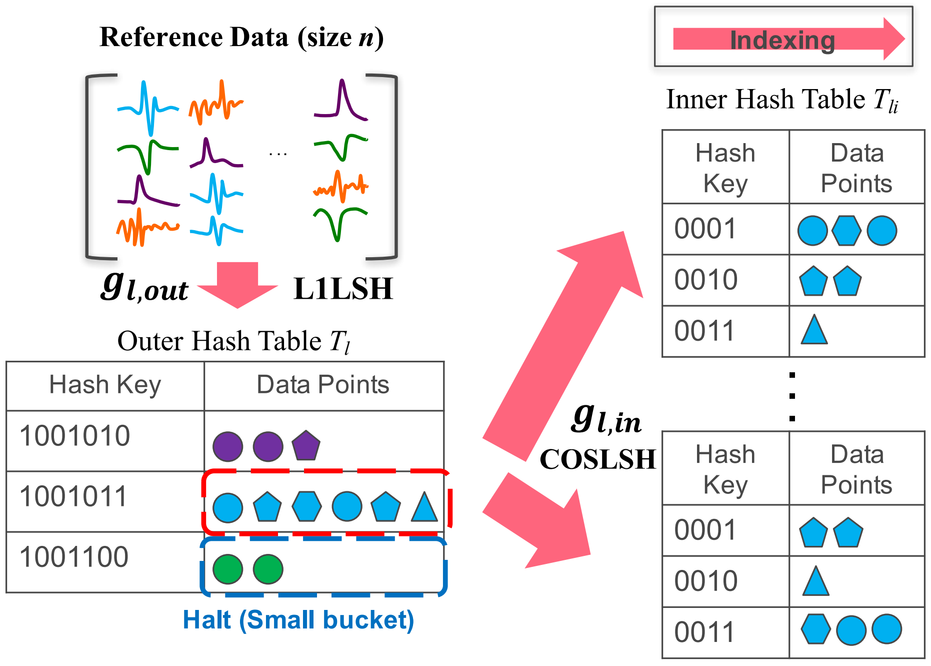


Figure 6-2: SLSH Indexing. We first stratify the data according to L1LSH at the outer level. Then, on each bucket with a significant size, we apply another layer of LSH with COSLSH at the inner level. The figure illustrates the simple case when one hash table is built at the outer level.

6.2 Method

Similar to the prediction based on the standard LSH or KNN, the prediction based on SLSH is essentially a two-step process of first quickly retrieving *patients with trajectories like mine*, the nearest neighbors (NNs) of our query of interest by SLSH, and second, extrapolating the information of NNs for prediction via majority vote. Prior to the retrieval and prediction, we index the data with the two-level SLSH. The procedures of our two-level SLSH (Figure 6-2 and Figure 6-3) are composed of the following tasks built on top of the standard LSH:

1. (*Indexing*) Stratify the data according to L1LSH at the outer level with (m_{out}, L_{out}).

2. (*Indexing*) Only on each bucket/stratum with a significant size ("populous bucket", whose size is larger than $\alpha\%$ of the original data size), we hash one level deeper with COSLSH at the inner level with (m_{in}, L_{in})[1].

[1] Somewhat analogous to top-down hierarchical clustering.

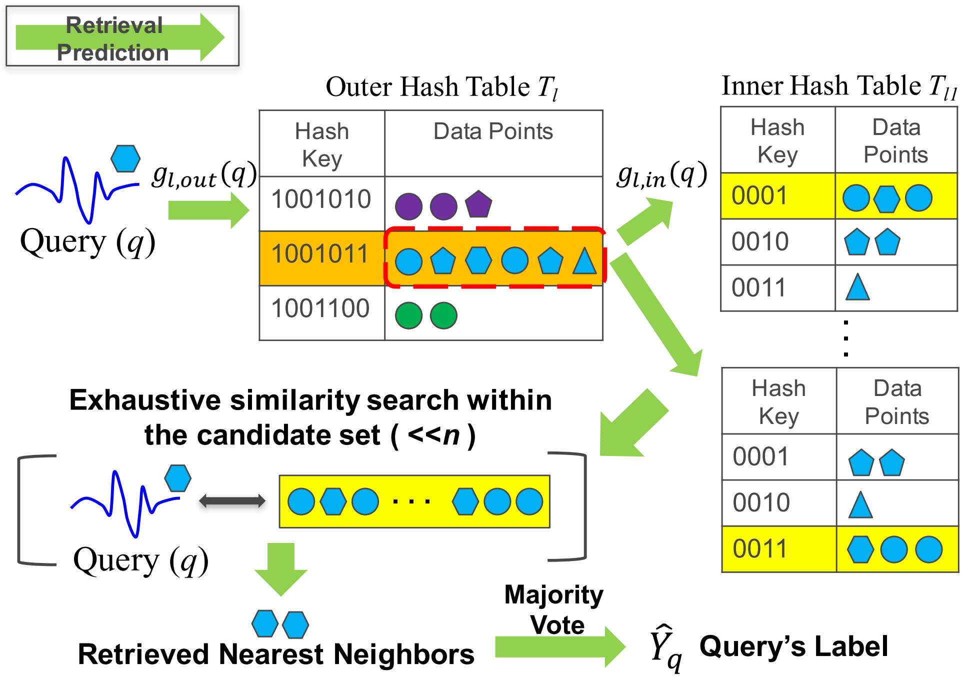


Figure 6-3: SLSH Retrieval and Prediction. We retrieve the approximate nearest neighbors of a query of interest by applying the same outer and inner hash functions used for construction and perform the linear search within the candidate set. Prediction is done by majority vote. The figure illustrates the simple case when one hash table is built at the outer level.

3. (*Retrieval*) Retrieve the approximate NNs of a query of interest by applying the same outer and inner (only when needed) hash functions used for construction, and by performing the linear search within the candidate set.

4. (*Prediction*) Finally, prediction in SLSH is done by taking the majority vote among the retrieved k approximate NNs, identical to the prediction step of the standard LSH (Algorithm 3 in Section 5.2).

The details of SLSH are presented in Algorithm 4 (indexing) and Algorithm 5 (retrieval and prediction).

Compared to the standard LSH, SLSH offers two benefits. First, it retrieves more integrated NNs according to a mixture of two distance measures. Second, it shortens the retrieval time because the candidate set size (selectivity) of SLSH is orders of magnitude smaller than that of LSH due to stratification. When setting up the standard LSH, the desirable number of hash functions used per table is the number

Algorithm 4 Stratified Locality-Sensitive Hashing: Indexing

```
Require: $m_{out}$, number of hash functions per outer table; $L_{out}$, number of outer hash tables; $m_{in}$, number of hash functions per inner table; $L_{in}$, number of inner hash tables; $\mathbf{X}$, reference data; $H_{out}$, hash function family for outer LSH; $H_{in}$, hash function family for inner LSH; $\alpha$, inner LSH threshold
 1: procedure SLSH-INDEXING($m_{out}, L_{out}, m_{in}, L_{in}, \mathbf{X}, H_{out}, H_{in}, \alpha$)
 2:     $\mathbf{T_{out}}, \mathbf{g_{out}} \leftarrow$ LSH-INDEXING($m_{out}, L_{out}, \mathbf{X}, H_{out}$)    ▷ Outer LSH with Algorithm 1. Data is divided into buckets which work as strata in each hash table.
 3:     for $l = [1, 2, \ldots, L_{out}]$ do
 4:         Let $n_l$ be the number of hash buckets in $T_l$ of $\mathbf{T_{out}}$
 5:         for $i = [1, 2, \ldots, n_l]$ do
 6:             Let $B_{li}$ be a hash buckets in $T_l$
 7:             if $|B_{li}| \geq \alpha|\mathbf{X}|$ then
 8:                 $p_{li} \leftarrow 1$                    ▷ populous bucket indicator
 9:                 $\mathbf{T}_{li}, \mathbf{g}_{li} \leftarrow$ LSH-INDEXING($m_{in}, L_{in}, B_{li}, H_{in}$)    ▷ inner LSH with Algorithm 1
10:             else
11:                 $p_{li} \leftarrow 0$
12:             end if
13:         end for
14:     end for
15:     Return All (outer and inner) hash tables $\mathbf{T}$ and functions $\mathbf{g}$
16: end procedure
```

that would partition the original data as uniformly as possible over all possible hash buckets so that each bucket contains only a small amount of the data. In such a way, when finding a hash bucket that collides with a query, the space subject to the linear search is enormously reduced, resulting in a large speed-up gain. In practice, however, tables typically contain a few populous buckets which impose a large bottleneck as shown in Section 5.4.3. This highlights another benefit of SLSH, besides allowing multiple perspectives on the data, that adding another layer of LSH yields more finely partitioned, evenly populated hash buckets effectively avoiding the bottleneck.

We use two-level hashing in this work, but without loss of generality, SLSH can be extended to multiple layers. While we use L1LSH and COSLSH for the outer and inner level LSH, respectively, our SLSH framework can embrace any distance function which has a valid locality-sensitive hash function family. For example, E2LSH can be used in place of L1LSH or COSLSH. It is also possible to reverse the order of L1LSH

Algorithm 5 Stratified Locality-Sensitive Hashing: Querying

```
Require: $q$, a query; $\mathbf{T_{out}}$, outer hash tables; $\mathbf{g_{out}}$, outer hash functions; $\{\mathbf{T_{li}}\}$, inner hash tables; $\{\mathbf{g_{li}}\}$, inner hash functions; $k$, number of NNs to retrieve
1: procedure SLSH-Querying($q, \mathbf{T_{out}}, \mathbf{g_{out}}, \{\mathbf{T_{li}}\}, \{\mathbf{g_{li}}\}, k$)
2:     $\mathcal{C} = \emptyset$                                        ▷ candidate set
3:     Apply lines from 2 to 6 of Algorithm 2 with $\mathbf{T_{out}}$ and $\mathbf{g_{out}}$    ▷ outer level querying
4:     for $l = [1, 2, \ldots, L_{out}]$ do        ▷ $L_{out}$ is the number of hash tables in $\mathbf{T_{out}}$
5:         Let $\tilde{B}_{li}$ be the colliding hash bucket of $q$ in $T_l$ of $\mathbf{T_{out}}$ and $i$ be its bucket index
6:         if $\tilde{p}_{li} = 0$ then                  ▷ collision with a non-populous bucket
7:             $\mathcal{S} \leftarrow$ members of bucket $\tilde{B}_{li}$
8:             $\mathcal{C} \leftarrow \mathcal{C} \cup \mathcal{S}$
9:         else if $\tilde{p}_{li} = 1$ then                 ▷ collision with a populous bucket
10:            Apply lines from 2 to 6 of Algorithm 2 with $\mathbf{T_{li}}, \mathbf{g_{li}}$ on $\tilde{B}_{li}$ and obtain the inner candidate set $C_{li}$          ▷ inner level querying
11:            $\mathcal{C} \leftarrow \mathcal{C} \cup \mathcal{C}_{li}$
12:        end if
13:    end for
14:    Compute distances from $q$ to each element in $\mathcal{C}$    ▷ with the distance metric that defines $\mathbf{g_{out}}$
15:    Return $k$ elements with the smallest distances
16: end procedure
```

and COSLSH. However, when COSLSH is used first at the outer level, it requires the entire data to be normalized prior to hashing and still needs to keep the original unnormalized data to perform L1LSH at the inner level. For both time and memory costs, it is more inefficient than normalizing only subsets of data that belong to the populous buckets in our proposed procedure. In addition, first stratifying the data according to shape has a lower interpretability to characterize each stratum when compared to first stratifying the data with respect to amplitude.

6.3 Experiment

We present the experimental set-up for our investigations regarding the predictive performance and speed of SLSH on the AHE datasets extracted from the MIMIC II database described in Section 3.4.

Table 6.1: Prediction models based on SLSH, KNN, the standard LSH, and ensembles of KNN and LSH. KNN and the standard LSH serve as a baseline for comparison for SLSH.

Prediction Model	Remarks
KNN-L1	KNN predictor with the L1 distance.
KNN-COS	KNN predictor with the cosine distance.
KNN-L1 *AND* KNN-COS	Ensemble predicts 1 (positive) if and only if both methods predict 1. Otherwise, 0.
KNN-L1 *OR* KNN-COS	Ensemble predicts 1 (positive) if either of the two methods predicts 1. Otherwise, 0.
L1LSH	Standard LSH predictor with the L1 distance.
COSLSH	Standard LSH predictor with the cosine distance.
L1LSH *AND* COSLSH	Ensemble predicts 1 (positive) if and only if both methods predict 1. Otherwise, 0.
L1LSH *OR* COSLSH	Ensemble predicts 1 (positive) if either of the two methods predict 1. Otherwise, 0.
SLSH (L1-COS)	Stratified LSH with the outer LSH with L1 and the inner LSH with cosine distances.
SLSH (COS-L1)	The reverse. Stratified LSH with the outer LSH with cosine and the inner LSH with L1 distances.

AHE Prediction Identical to the prediction procedure presented in Chapter 5, we build the prediction models based on SLSH, the standard LSH, and KNN for the occurrence of AHE. We build the models with a lag time amount of historical data prior to the event window, where each data point x_i has a label y_i indicating the occurrence of AHE. We formulate this prediction task as a supervised binary classification problem, where predicted labels $\hat{y}_i$ describe the query patients as AHE-positive or negative.

Prediction Models We build the predictors with KNN, standard LSH, and SLSH, as well as their various ensembles. All prediction models used are explained in Table 6.1. KNN and the standard LSH serve as the baselines for comparison for SLSH. Solo predictors (KNN, LSH) with a single distance metric are compared to the duo ensembles combining the L1 and the cosine distances.

Performance Measures We evaluate performance of predictors (the standard LSH, SLSH, and KNN) mainly with:

- Prediction Accuracy $= (TP + TN)/(TP + FP + TN + FN)$[2], and
- Speed-up factor, the average time taken by the predictor relative to that by KNN, measured by the inverse of *selectivity* (defined in Section 4.3).

For reasons described in Section 5.3 (mainly due to prediction accuracy being the highest with $k = 1$ than any larger number of k and the advantage of eliminating the need to tune the extra hyper-parameter k), we make predictions based on one nearest neighbor (1-NN).

6.4 Results and Discussion

In this section, we start by providing a qualitative analysis of the retrieved nearest neighbors via SLSH compared to those from a single level LSH. We assess prediction accuracy and querying speed of SLSH compared to those of KNN and the standard LSH. Then, we evaluate every prediction model presented in Table 6.1 and compare ensembles to single metric models. We finish by discussing the impact of changing the order of distance measures used at each layer of SLSH. For the following subsections, we present analysis of our experimental results mainly on the dataset $\text{Data}_{\text{Lag 300, 1x}}$.

6.4.1 Retrieved Nearest Neighbor Set

This section addresses the question of whether SLSH is capable of retrieving NNs according to both amplitude and shape. We demonstrate with a visual example. Figure 6-4 presents the retrieved nearest neighbor sets of a time series query using the standard single-level L1LSH, the standard single-level COSLSH, and SLSH. For the given query (red line), in Figure 6-4 (Top), we observe that the set of 5-NNs (colored dashed lines) retrieved by L1LSH is tightly located close to the query in terms of the mean amplitude but oblivious to their various shapes. Likewise, in Figure 6-4 (Middle), the NNs retrieved by COSLSH all resemble the shape of the query, but their amplitudes are well-spread across various levels. Figure 6-4 (Bottom) shows a

[2] TP = True Positive, FP = False Positive, FN = False Negative, TN = True Negative.

qualitative evaluation of SLSH. The NNs obtained via SLSH not only have shapes that are similar to that of the query, but also have their mean amplitudes much closer the query compared to the set retrieved by COSLSH. Satisfying the notion of similarity in terms of both amplitude *and* shape, this qualitatively verifies that SLSH is able to address the data from multiple and more integrated perspectives.

6.4.2 Acute Hypotensive Episode Prediction

We present a quantitative evaluation by empirically demonstrating on $\text{Data}_{\text{Lag 300, 1x}}$ that the prediction by SLSH is faster and more accurate than L1LSH. Figure 6-5 shows the accuracy and the associated speed-up factor of L1LSH (red, green) and SLSH (blue) for the AHE prediction based on 1-NN. First, for each combination (m, L) of LSH parameters $m \in [5, 10, \ldots, 50]$ and $L \in [10, 20, \ldots, 100]$, we make a prediction via the single-level L1LSH. The prediction results with their corresponding speed-ups are shown as the red crosses where each point corresponds to a single parameter instance of (m, L). The leftmost point of the red plot with no speed-up is the accuracy of KNN.

Then, for SLSH, by setting a particular instance of L1LSH as the outer layer of SLSH and $\alpha = 1\%$, we perform the inner SLSH (with COSLSH) with the parameters $m_{in} \in [1, 4, \ldots, 19]$ and $L_{in} \in [1, 4, \ldots, 10]$. We choose $(m_{out}, L_{out}) = (35, 20)$ as the outer level SLSH, which corresponds the instance of L1LSH parameters which outputs 1% loss of accuracy from KNN with a speed-up gain of 25x (reflected as the green point on Figure 6-5). We choose this parameter set instance because it is the fastest one among the ones having a loss of accuracy less than 1%, assuming that our maximum accuracy loss tolerance (as a cost of gaining speed-up) is 1%. For the entire range of SLSH (blue points), we observe that it is *more accurate and faster* than its baseline, the single-level L1LSH (the green point).

There are two additional computational costs for running SLSH compared to the standard LSH: extra querying cost with the inner hash functions and storage cost for the second level inner hash tables. It turns out the both of them are trivial. The additional time taken to apply the inner level hash functions and to locate matching inner

Table 6.2: Prediction performance on $\text{Data}_{\text{Lag 300, 1x}}$, $\text{Data}_{\text{Lag 30, 1x}}$, and $\text{Data}_{\text{Lag 30, 10x}}$. Across all datasets, SLSH outperforms L1LSH with respect to prediction accuracy (%) and speed-up factor (x).

	Lag 300, 1x		Lag 30, 1x		Lag 30, 10x	
Model	Accuracy	Speed-up	Accuracy	Speed-up	Accuracy	Speed-up
KNN-L1	95.95	1	95.81	1	97.48	1
L1LSH (1% loss)	94.95	24.57	94.81	13.11	96.48	24.69
SLSH (L1-COS)	**95.30**	**74.21**	**94.97**	**213.21**	**97.01**	**138.78**

hash buckets in SLSH querying is on average two orders of magnitude smaller than the time required to conduct the linear search in populous buckets in the standard LSH. Thus, measuring speed-up by selectivity is valid since it is minimally affected by the introduction of trivial extra querying cost. For storage, the number of inner hash tables (L_{in}) needed for the optimal SLSH instance across all datasets is only 1 table ($L_{in} = 1$) for the inner layer. Although LSH typically requires a large number of hash tables to produce good approximation, the extra space requirement for SLSH is negligible.

Table 6.2 shows the summary of model performance for KNN-L1, L1LSH, and SLSH (L1-COS) on all three datasets ($\text{Data}_{\text{Lag 300, 1x}}$, $\text{Data}_{\text{Lag 30, 1x}}$, and $\text{Data}_{\text{Lag 30, 10x}}$). We note that the accuracy and speed-up of SLSH is determined by choosing the output of the inner SLSH parameter instance which generates the optimal ("knee") point on the Pareto front of the accuracy-speed trade-off profile of SLSH (Figure 6-6). Across our datasets that either scale in item quantity ($\text{Data}_{\text{Lag 30, 10x}}$) or decrease in the lag duration ($\text{Data}_{\text{Lag 30, 1x}}$), we see an increase in both accuracy and speedup with SLSH over L1LSH. Figure 6-6 illustrates the same result. Our conclusion that SLSH performs better than the single-level standard LSH still holds when the data scales in its size or the lag duration of data changes.

All in all, the experiment shows that SLSH has a higher accuracy and a much faster querying speed (Pareto dominance) than the standard LSH with L1LSH alone. For accuracy, we observe that obtaining a set of NNs from an integrated perspective (matching based on both amplitude and shape) indeed leads to a better prediction. We achieve a large speed up gain with SLSH by avoiding the bottleneck of L1LSH

Table 6.3: Predictor performance on $\text{Data}_{\text{Lag }300,\ 1\text{x}}$. Exploiting the data with multiple distance metrics is more advantageous as shown by the higher accuracies obtained by the ensembles with the *AND* operator and SLSH in comparison to the individual predictors with a single distance metric. SLSH is a better strategy to combine distance metrics than using the ensemble as it generates more accurate and faster results. The order of outer and inner operations of SLSH impacts the prediction performance as SLSH (L1-COS) and SLSH (COS-L1) generate different outcomes.

Prediction Model	Accuracy (%)	Speed-up (x)
KNN-L1	95.95	1
KNN-COS	93.74	1
KNN-L1 *AND* KNN-COS	**96.27**	0.5
KNN-L1 *OR* KNN-COS	93.41	0.5
L1LSH (1% loss)	94.95	24.57
COSLSH (1% loss)	92.74	213.75
L1LSH *AND* COSLSH	**95.10**	24.20
L1LSH *OR* COSLSH	91.73	24.20
SLSH (L1-COS)	**95.30**	**74.21**
SLSH (COS-L1)	93.21	233.36

because the candidate set of SLSH subject to the linear search is orders of magnitude smaller than that of L1LSH.

6.4.3 Multi-Distance Measures

We examine whether using two distance functions to capture similarity (via an ensemble of two predictors or via SLSH) is more beneficial than using a single metric predictor on $\text{Data}_{\text{Lag }300,\ 1\text{x}}$. First, we compare KNN-L1 and KNN-COS to the ensembles of the two. From Table 6.3, we observe that the ensemble of KNN-L1 and KNN-COS with the operator *AND* (0.9627) has a higher prediction accuracy than any of the individual predictors (0.9595 and 0.9374, respectively). However, the ensemble with the *OR* operator (0.9341) does not perform as well as any single predictor. Likewise, with a single level LSH, the ensemble of L1LSH and COSLSH with the *AND* operator has a higher accuracy (0.9510) than any of the two alone (0.9493 and 0.9308, respectively). So the ensembles combining L1 and cosine with the more conservative *AND* operator achieve a superior performance to single metric predictors.

SLSH also effectively combines two distance measures in a hierarchical way. Comparing SLSH to the ensemble of the single layer LSHs (i.e. L1LSH *AND* COSLSH), we observe that SLSH (0.9530) is still more accurate and much faster than the ensemble (0.9510). This implies that to combine two distance metrics, SLSH is a better choice than the ensemble. On the other hand, SLSH has a lower prediction accuracy than the ensemble of KNN-L1 and KNN-COS, but SLSH offers a substantial speed-up gain (74x) compared to the ensemble of two KNNs. The above observations from the ensembles with the *AND* operator and effectiveness of SLSH support our hypothesis that exploiting the data with multiple distance metrics is indeed advantageous.

6.4.4 Commutativity

Having found using two hash families to be more advantageous, it prompts us to ask how sensitive SLSH is to which hash function family it uses at the outer level and at the inner level. We ask whether SLSH construction is commutative (i.e. if the order of distance functions in the outer and the inner SLSH is interchangeable to obtain the same result). The answer is that it is not commutative, and the order of operation greatly matters.

First, changing the order of hash families for the outer and inner SLSH implies different interpretability. With SLSH (L1-COS), among the elements with similar amplitude, we choose the ones with similar shape as our final nearest neighbors. The reverse holds for SLSH (COS-L1). Among the elements with similar shape, we choose the ones with similar amplitude as our final nearest neighbors. Second, from Table 6.3, we observe that SLSH (L1-COS) and SLSH (COS-L1) have different accuracies and querying speeds. SLSH (L1-COS) has a higher accuracy, but is slower than SLSH (COS-L1).

In addition, the change when we build the inner SLSH on top of the best performing outer SLSH (with 1% accuracy drop tolerance) is different in each case. When going from L1LSH to SLSH (L1-COS), the accuracy improves by 0.35 and the querying speed becomes 2.98 times faster. However, when going from COSLSH to SLSH (COS-L1), the accuracy improves by 0.47, but the speed increases by only 1.09 times. The improvement SLSH (COS-L1) makes over COSLSH is smaller in terms of speed compared to SLSH (L1-COS). This behavior is related to the difference in the bucket size distributions of the outer level LSH between by L1LSH and by COSLSH (discussed in Section 5.4.3). For SLSH (COS-L1), the outer level COSLSH stratified the data into more evenly-spread buckets with small sizes (only 7.75% of total data belonged to the top first percentile of hash buckets in terms of bucket sizes) and thus achieved a good speed-up because the bottleneck linear searches were done on small candidate sets. So there was not much room for the inner level SLSH (with L1LSH) to make improvements in terms of speed because data were already stratified into small sized buckets, which often did not qualify for the second level hashing. On the other hand, when using L1LSH as the outer LSH, data stratification was very uneven so that a substantial amount of data belonged to the few largest buckets (20.24% of total data were concentrated in the top first percentile of hash buckets in terms of bucket sizes). So, applying another layer of LSH (with COSLSH) on these large buckets was effective in term of reducing the candidate set size and thus, speeding up the querying time.

6.5 Chapter Conclusion

In this chapter, we proposed multilevel stratified locality-sensitive hashing. We used this method to help address the problem of integrating multiple distance measures in LSH, which previously was not feasible with the standard LSH. Correspondingly, we showed that SLSH, which hybridizes the L1 and the cosine distances, is capable of retrieving nearest neighbors of a query according to a mixture of both amplitude and shape. We compared SLSH against the standard single-level LSHs with the L1 and the cosine distances and found that our method generates higher accuracy and faster querying speed on the problem of predicting acute hypotensive episodes.

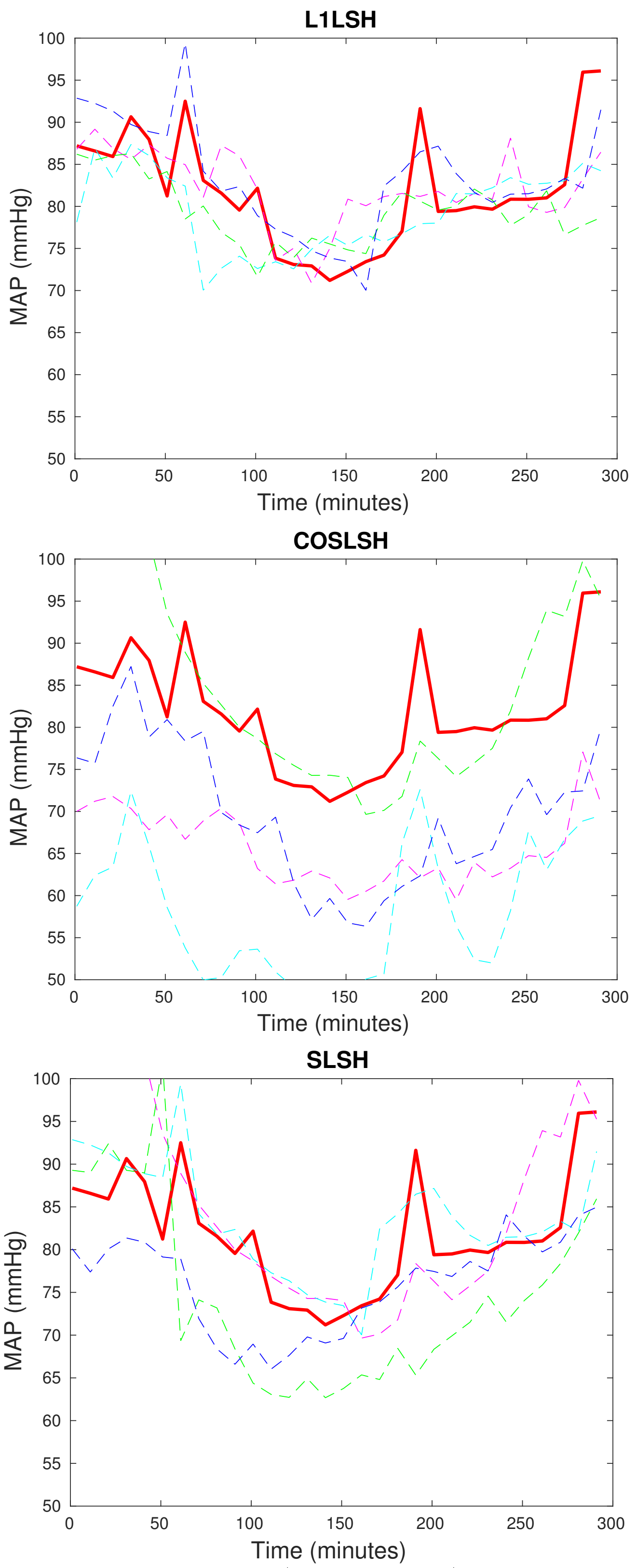


Figure 6-4: The 5-nearest neighbors (dashed lines) of a waveform query (red line) retrieved by (*Top*) the standard L1LSH, (*Middle*) the standard COSLSH, and (*Bottom*) SLSH. Each set is similar to the query in terms of amplitude, shape, and *both* amplitude *and* shape by L1LSH, COSLSH, and SLSH.

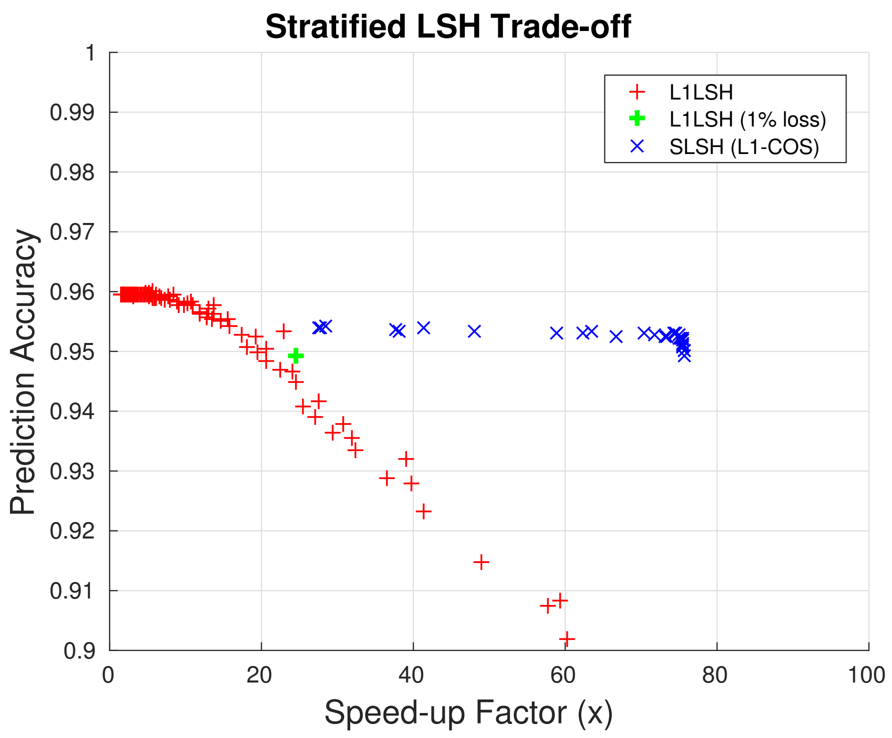


Figure 6-5: Comparison of SLSH (L1-COS) to L1LSH for prediction of AHE with 1-NN on $\text{Data}_{\text{Lag 300, 1x}}$. Given an instance of L1LSH (green) as its benchmark and as the outer layer, SLSH (blue) outperforms for the entire range of (m_{in}, L_{in}). Each point corresponds to a parameter configuration (m, L) and (m_{in}, L_{in}) of L1LSH and SLSH, respectively.

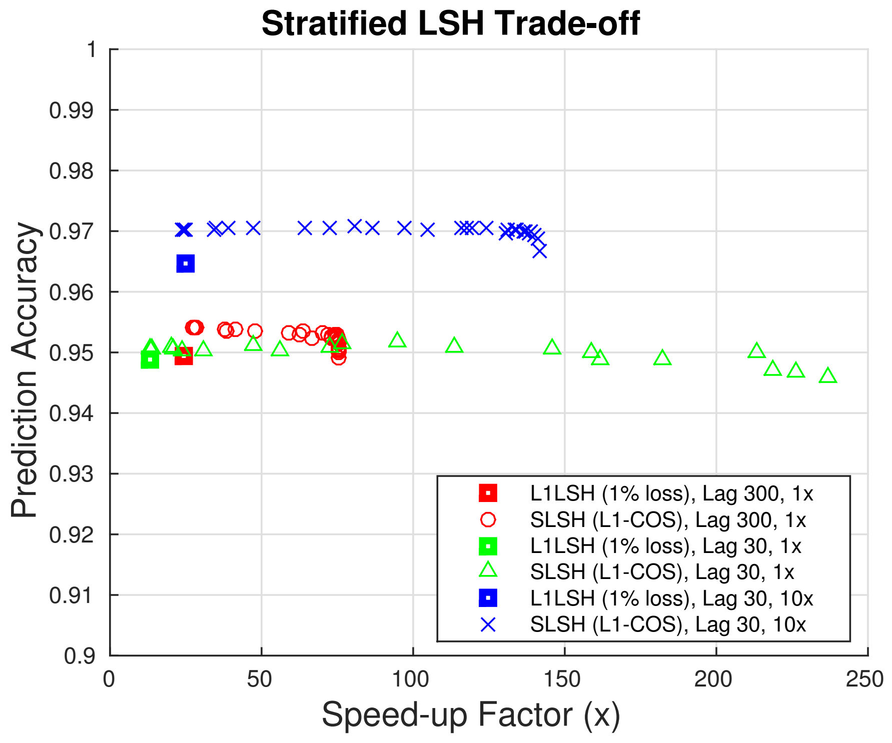


Figure 6-6: SLSH trade-off across all three datasets. Squares indicate L1LSH instances selected by the 1% loss in accuracy criterion. Each non-squared point corresponds to a parameter configuration (m_{in}, L_{in}) of SLSH. Across all datasets, SLSH outperforms L1LSH in terms of accuracy and speed-up.

Chapter 7

Collision Frequency Locality-Sensitive Hashing

This chapter addresses the question of whether the short-listing by calculating the distances between the query and every candidate set element is optimal and whether there exists another effective way of conducting the short-listing. To answer these questions, we propose a new model of locality-sensitive hashing (LSH), namely collision frequency locality-sensitive hashing (CFLSH), to further improve the prediction accuracy without sacrificing any speed. The key idea is that the more frequently an element and query collide across multiple LSH hash tables, the more similar they are. We demonstrate and validate our method on the problem of predicting acute hypotensive episodes with the time series data extracted from the MIMIC II database. The main content of this chapter will be published in [72].

7.1 Motivation

An important concept in LSH is the notion of *candidate set*, a pool of multiple elements preliminarily filtered from the original data via locality-sensitive hash functions. The members of this set are the candidates to eventually become the final nearest neighbors (NNs) of the query. For each element in the candidate set of a query, information on its collision frequency and distance with respect to the query is recorded.

In most conventional variants of LSH, the *short-listing* from the candidate set to the final k NNs is done by ranking the proximity to the query using distance, without utilizing the collision frequency information [54].

In this chapter, we present an alternative way of conducting the retrieval step of LSH and evaluate its impact after extrapolation and prediction. Our modification substitutes the distance based short-listing of the NN set with a short-listing method based on how frequently a query and a point collide during table by table querying. We propose a new model of LSH, namely *collision frequency LSH (CFLSH)*, to further improve the prediction accuracy without sacrificing any speed by introducing an intuitive way of formulating the approximate NN set: the more frequently an element and the query collide in the same hash bucket across multiple LSH hash tables, the more similar they are to each other. Here, the short-listing step is primarily performed with respect to the frequency of collision and then secondarily by distance information when there is a tie. This allows us to take advantage of a broader set of information contained in the candidate set.

Our research question is whether CFLSH is as efficient or superior (in terms of prediction accuracy and querying speed) to the standard LSH with the families for the L1 distance (L1LSH) [40] and the cosine distance (COSLSH) [20]. We evaluate CFLSH on a dataset of mean arterial blood pressure extracted from the MIMIC II database in the context of predicting acute hypotensive episodes (AHE).

7.2 Method

Before we explain CFLSH, we revisit the prediction procedures of LSH. The standard LSH based prediction consists of three steps: *a*) constructing an efficient data structure (hash table) to *index* (hash) the data for fast retrieval to follow (Algorithm 1, Section 4.2.2), *b*) quickly *retrieving* the approximate NNs of the query of interest (Algorithm 2, Section 4.2.3), and *c*) *predicting* the label of the query based on predominance of labels of its NNs (Algorithm 3, Section 5.2).

We are particularly interested in proposing a new method that modifies the retrieval step. For *retrieval*, we 1) hash a query by the same set of hash functions used for indexing, 2) compose *the candidate set*, which is defined as the union of all points contained in the colliding hash buckets of the query from each hash table, and 3) retrieve a group of k points that are most similar to the query by the standard *short-listing* step, which is done by the *linear search by distance* within the candidate set.

For a given query, the candidate set can be represented as a set of triplets $\{(t_i, d_i, f_i)\}$, where t_i is the index of i^{th} element in the data, d_i is its distance to the query, and f_i is how many times it appears (frequency) in the candidate set (i.e. in how many tables the query collides with t_i). It is important to note that, to retrieve a desired number of final approximate NNs, the linear search in the standard LSH does not take account of the frequency f_i information, but only of the distances d_i between the query and the *unique* points in the candidate set.

In CFLSH, we modify the step 3 of retrieval (the *short-listing* step) by replacing the linear search with *collision frequency counting*. We note that the *indexing* and *prediction* parts of CFLSH are identical to the step a and c of the standard LSH above. Unlike the standard LSH which utilizes only the distance information $\{(t_i, d_i)\}$, the query's k nearest neighbors in CFLSH are defined as the top k points among $\{t_i\}$ which most frequently collided with the query (sorted according to f_i) across the entire set of hash tables. When there is a tie, such elements are secondarily sorted according to the distance d_i to the query. Here, the short-listing step of CFLSH is primarily performed with respect to the frequency of collision, seconded by distance information. Therefore, CFLSH exploits the broader range of information $\{(t_i, d_i, f_i)\}$ contained in the candidate set. The detailed procedure of the method is in Algorithm 6.

This is an alternative and intuitive way of constituting the NN set than by performing the linear search (in the step 3 of retrieval). *The more frequently an element and the query belong in the same bucket across multiple tables, the more likely they are to be similar*, because in LSH, each hash function h (and therefore its composite g as well) serves as a different basis of comparing points with a low resolution, "weak" similarity.

Algorithm 6 Collision Frequency Locality-Sensitive Hashing: Querying

```
Require: $q$, the query; $\mathbf{T}$, hash tables; $\mathbf{g}$, hash functions; $k$, number of NNs to
    retrieve
 1: procedure CFLSH-QUERYING($q, \mathbf{T}, \mathbf{g}, k$)
 2:     $\mathcal{C} = \emptyset$                                          ▷ candidate set
 3:     for $l = [1, 2, \ldots, L]$ do                 ▷ $L$ is the number of tables in $\mathbf{T}$
 4:         $\mathcal{S} = \{x \in \mathbf{X} | g_l(x) = g_l(q)\}$          ▷ colliding bucket from table $T_l$
 5:         $\mathcal{C} \leftarrow \mathcal{C} \cup \mathcal{S}$
 6:     end for
 7:     Compute the frequency of each element in $\mathcal{C}$.
 8:     Sort the elements in descending order in terms of frequency.
 9:     If there is a tie, compute distances from $q$ to such elements and sort by distance
    from the smallest.
10:     Return $k$ elements with the highest frequencies.
11: end procedure
```

In case of the hash function families for L1, each g randomly selects m dimensions out of d. As we are adding more tables (i.e. additional g's), a broader set of dimensions out of d are "covered" for similarity comparison. Thus, the more frequently two points collide across multiple tables, the more often they match (sharing the same hash key, thus being considered as equal) under a larger set of dimensions.

For example, let us assume that we have a candidate set $\{(t_i, d_i, f_i)\}$ that is $\{(t_1, 5, 5),\ (t_2, 2, 7),\ (t_3, 3, 10),\ (t_4, 4, 7),\ (t_5, 6, 6),\ (t_6, 7, 1),\ (t_7, 1, 8)\}$. The candidate set sorted according to the frequency f_i is $[t_3, t_7, t_2, t_4, t_5, t_1, t_6]$. In this example, the set of the approximate k nearest neighbors by CFLSH is t_3 and $[t_3, t_7, t_2, t_4, t_5]$ for $k = 1$ and $k = 5$, respectively. On the other hand, with the standard LSH, the candidate set sorted according to the distance is $[t_7, t_2, t_3, t_4, t_1, t_5, t_6]$. The corresponding set of k nearest neighbors is t_7 and $[t_7, t_2, t_3, t_4, t_1]$ for $k = 1$ and $k = 5$, respectively.

It is worth noting that the time complexity of CFLSH is equivalent to the standard LSH. Counting the frequency of each element in the candidate set takes a linear time with respect to its size, which is the same time cost for the linear search step of the standard LSH. However, in real time, CFLSH is a bit faster than the standard LSH because counting takes a shorter time than performing exhaustive distance calculations.

7.3 Experiment

We present our findings on predicting AHE on our time series dataset of mean arterial blood pressure extracted from the MIMIC II database (explained in Section 3.4). We use $\text{Data}_{\text{Lag 300, 1x}}$ for demonstration.

AHE Prediction Similar to the prediction problems we investigated in the previous chapters, we build the prediction models based on CFLSH, the standard LSH, and KNN for the occurrence of AHE within an event window. We build the models with a lag time amount of historical data prior to the window, where each data point x_i has a label y_i indicating the occurrence of AHE. We again formulate this prediction as a supervised binary classification problem, where predicted labels $\hat{y}_i$ describe the patients as AHE-positive or negative.

Performance Measures We evaluate prediction performance with:

- Prediction Accuracy $= (TP + TN)/(TP + FP + TN + FN)$[1], and
- Speed-up factor, which is the average time taken by LSH or CFLSH relative to that by KNN, measured by the inverse of *selectivity* (defined in Section 4.3).

Experimental Procedures We apply L1LSH, COSLSH, CFLSH with L1, and CFLSH with cosine for prediction. For each combination of LSH parameters $m \in [5, 10, \ldots, 50]$ and $L \in [10, 20, \ldots, 100]$, we make a prediction via CFLSH and the standard LSH for L1 and compare them to the result of KNN. For cosine, we use $m \in [1, 3, 5, \ldots, 19]$. We apply the above parameter configurations to retrieve and predict based on 1-NN, 5-NNs, and 10-NNs.

7.4 Results and Discussion

In this section, we compare CFLSH to the standard LSH and conduct sensitivity analysis of the prediction accuracy with respect to the number of hash functions applied per table.

[1] TP = True Positive, FP = False Positive, FN = False Negative, TN = True Negative.

7.4.1 Acute Hypotensive Episode Prediction

Figure 7-1 (Top) presents the prediction accuracy of L1LSH and CFLSH for the L1 distance (CFLSH-L1) with its associated speed-up factor with $k = 1$ (prediction based on 1-NN). Each point corresponds to the average result from a pair of the number of hash functions (m) used per table and the number of hash tables (L) over 10 runs. The leftmost point of L1LSH with no speed-up corresponds to the prediction accuracy (95.95%) of KNN with L1 as its distance function. For configurations of parameters which generate up to 20x speedup, we observe that the standard L1LSH results in slightly higher prediction accuracy than CFLSH-L1. However, beyond the 20x speedup tipping point, whereas the prediction accuracies of L1LSH degrade significantly, those of CFLSH-L1 stay relatively unchanged. If we were to trade 1% accuracy decrease for a faster querying time, we are able to achieve a prediction result via CFLSH up to 100 times faster compared to KNN, where the corresponding speed-up for L1LSH is 25x.

The accuracy-speed trade-off profiles for higher numbers of k-NNs exhibit a different behavior. We remark that for a given (m, L), the speed-ups are the same for all k and only the accuracy changes as k varies. Figure 7-1 (Middle) shows the profile for $k = 5$. We observe that CFLSH outperforms L1LSH over the entire range of parameter configurations. That is, given a speed-up, CFLSH has a higher accuracy than L1LSH. Likewise, given an accuracy, CFLSH is much faster than L1LSH. However, unlike the profile of CFLSH for $k = 1$, whose accuracy range stays relatively unchanged regardless of the magnitude of speed-up, the prediction accuracy for $k = 5$ decreases as the speed-up factor becomes larger. In Figure 7-1 (Bottom), the profile for $k = 10$ exhibits a similar behavior to that of $k = 5$. The difference is that the discrepancy between CFLSH and L1LSH is smaller with $k = 10$.

CFLSH with cosine exhibits a different behavior. Figure 7-2 (Top) presents the prediction accuracy of COSLSH and CFLSH for the cosine distance (CFLSH-COS) with its associated speed-up factor with $k = 1$. From the figure, we observe that CFLSH-COS performs slightly worse than COSLSH for some range of parameters although the discrepancy is negligible. The prediction results based on a higher number of k-NNs exhibit a similar trend to that of $k = 1$. For $k = 5$ and $k = 10$, the accuracy-speed trade-off profiles for CFLSH-COS are almost identical to those of COSLSH, as shown in Figure 7-2 (Middle) and (Bottom), respectively. The fact that CFLSH-COS does not perform better than COSLSH is the major difference between CFLSH-COS and CFLSH-L1. One possible explanation for this behavior is by observing the difference in the interpoint distance distributions under L1 and cosine (Section 3.5). The gap between collision probabilities for close and far points is higher for L1 than cosine, which means that the empirical estimates of those probabilities obtained by collision frequency counting distinguish between close and far points better for L1 than cosine.

7.4.2 Sensitivity Analysis

Figure 7-3 shows the sensitivity analysis of CFLSH and the standard LSH with respect to the number of hash functions applied (m) and the number of hash tables created (L) when the L1 distance is used. Without loss of generality, we illustrate with the case for 1-NN ($k = 1$). In general, the accuracy of the standard LSH decreases with increasing m as the cost of obtaining a higher speed-up when more hash functions are used to define the hash key. Figure 7-3 (Middle) illustrates this behavior for L1LSH. In contrast, CFLSH-L1 exhibits a different behavior. As illustrated in Figure 7-3 (Top), the accuracy increases up to a point and then starts to decrease only slightly with increasing m, across all L, in the range between 0.93 and 0.96. This implies that up to the tipping point, the accuracy and the speed-up both increase, and even with a larger m past that of the tipping point, accuracy does not decrease significantly. We then directly compare CFLSH-L1 to L1LSH. Figure 7-3 (Bottom) shows the change of accuracy as a function of m and L when L1LSH is subtracted from CFLSH-L1. We observe that, although by a very small margin of less than 0.03, L1LSH has a higher accuracy than CFLSH-L1 for m up to 30 across all L. Then, for m greater than 30, CFLSH-L1 shows a significant accuracy improvement over L1LSH. In terms of the accuracy-speed trade-off, this implies that CFLSH-L1 has a significant advantage over L1LSH in the range of m that corresponds to large speed-ups.

We apply the same analysis to CFLSH-COS and COSLSH, shown in Figure 7-4. Following the general expectation that the accuracy of the standard LSH decreases with increasing number of hash functions applied as the cost of obtaining a higher speed-up, COSLSH (Figure 7-4 (Middle)) behaves accordingly. However, in contrast to the case of L1, CFLSH-COS follows this expected general pattern as well (Figure 7-4 (Top)). Almost identical to COSLSH, the accuracy of CFLSH-COS stays relatively unchanged up to a point (m up to 13) and then starts to drop abruptly, across all values of L. When directly comparing CFLSH-COS and COSLSH by subtracting the latter from the former, we observe the trend we saw in the case of L1: the standard LSH performs better in the range of small m, then CFLSH starts to have a higher accuracy for higher values of m. Although by a very small margin, CFLSH-COS has a higher accuracy than COSLSH for m larger than 15 for all values of L. This observation explains the long tail part of the accuracy-speed trade-off in Figure 7-2 (Top).

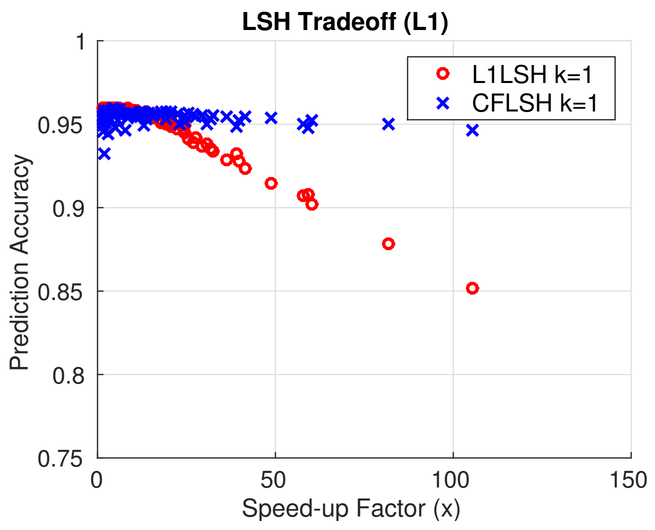
7.4.3 Discussion

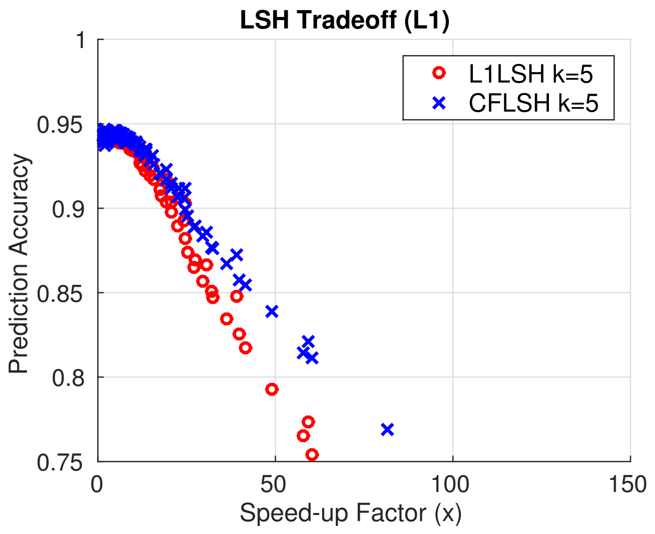
All in all, these results imply that CFLSH performs better than the standard LSH when a large number of hash functions is used to define the hash keys, which corresponds to the situation with a large speed-up. If a user puts an emphasis on querying speed more so than prediction accuracy, our results suggest the use of CFLSH over the standard LSH, especially for L1. But for cosine, one has to be careful which LSH to use as there is a very minimal distinction.

For data in real applications, there is no perfect, ground truth similarity measure because what constitutes similarity is not entirely quantifiable by a single distance metric. A single distance metric cannot capture every aspect of similarity along the axes of matching based on amplitude and shape of time series. Although L1 is the best distance metric for our dataset (i.e. the prediction accuracy being the highest when the similarity is measured by L1 compared to other distance metrics such as cosine), the fact that CFLSH-L1 performs better than L1LSH implies that L1 is not yet a perfect distance metric for measuring similarity.

7.5 Chapter Conclusion

In this chapter, we presented a pragmatic study on a new variant of LSH, namely collision frequency LSH, which short-lists the candidate set by counting the frequency of collision instead of the linear distance calculation. It remains an open question why the trade-off profiles of CFLSH and the standard LSH are different when L1 is used as a distance measure while they are similar in the case of cosine. As a future work, it will be worthwhile to investigate theoretical foundations on why CFLSH performs superior to the standard LSH in certain circumstances.





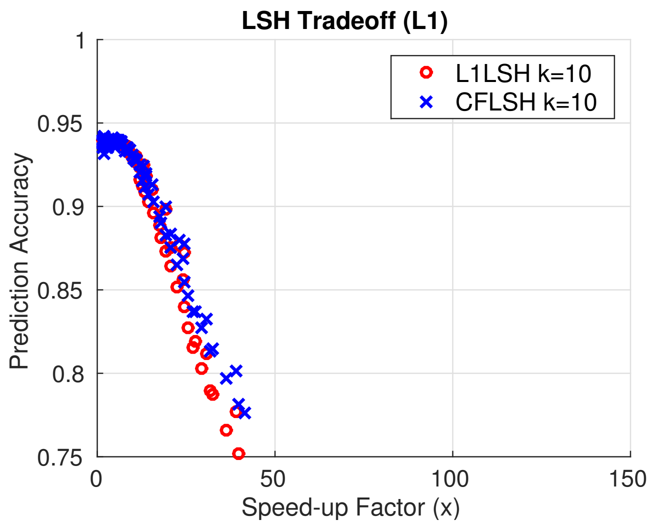


Figure 7-1: Comparison of CFLSH with L1 to L1LSH for (*Top*) $k = 1$, (*Middle*) $k = 5$, and (*Bottom*) $k = 10$. Each point corresponds to a parameter configuration (m, L) of LSH.

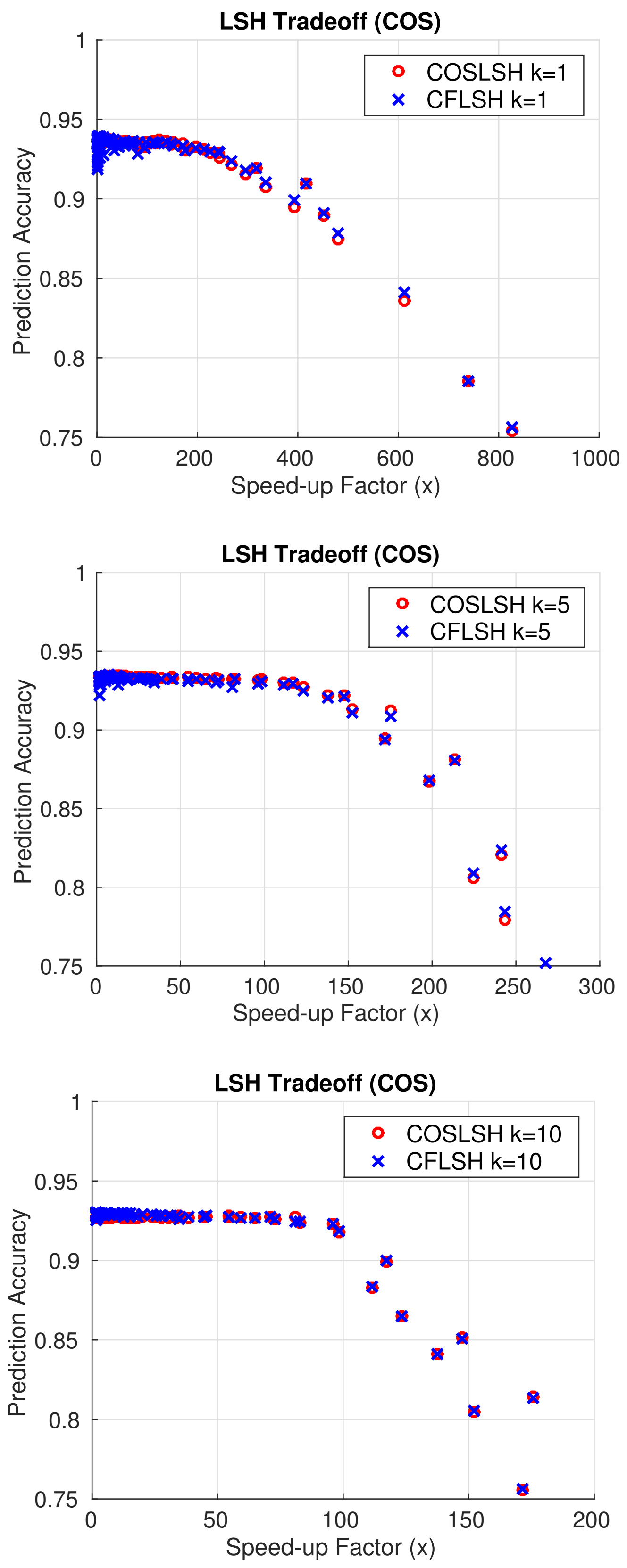


Figure 7-2: Comparison of CFLSH with cosine to COSLSH for (*Top*) $k = 1$, (*Middle*) $k = 5$, and (*Bottom*) $k = 10$. Each point corresponds to a parameter configuration (m, L) of LSH.

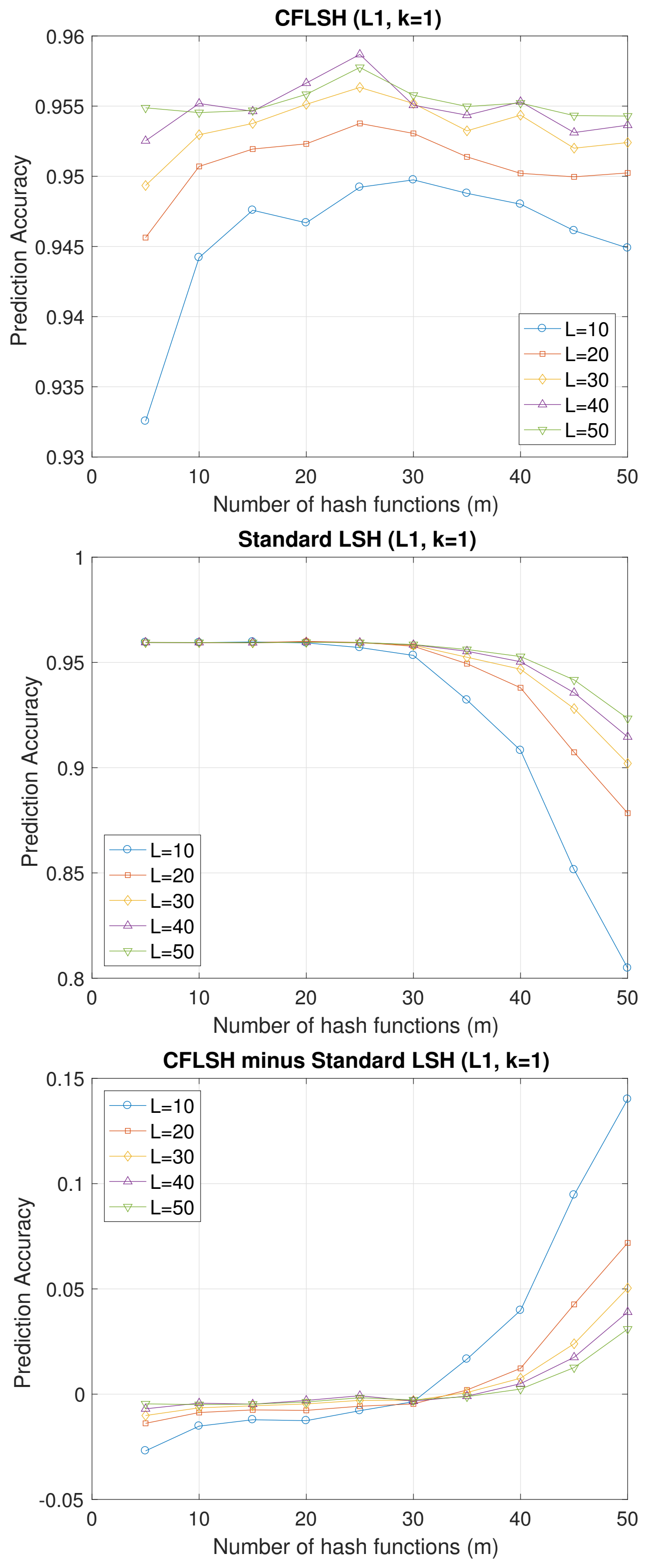


Figure 7-3: Sensitivity analysis of the prediction accuracy against the number of hash functions used per table (m) with L1: (*Top*) CFLSH, (*Middle*) the standard LSH, and (*Bottom*) the difference between the two.

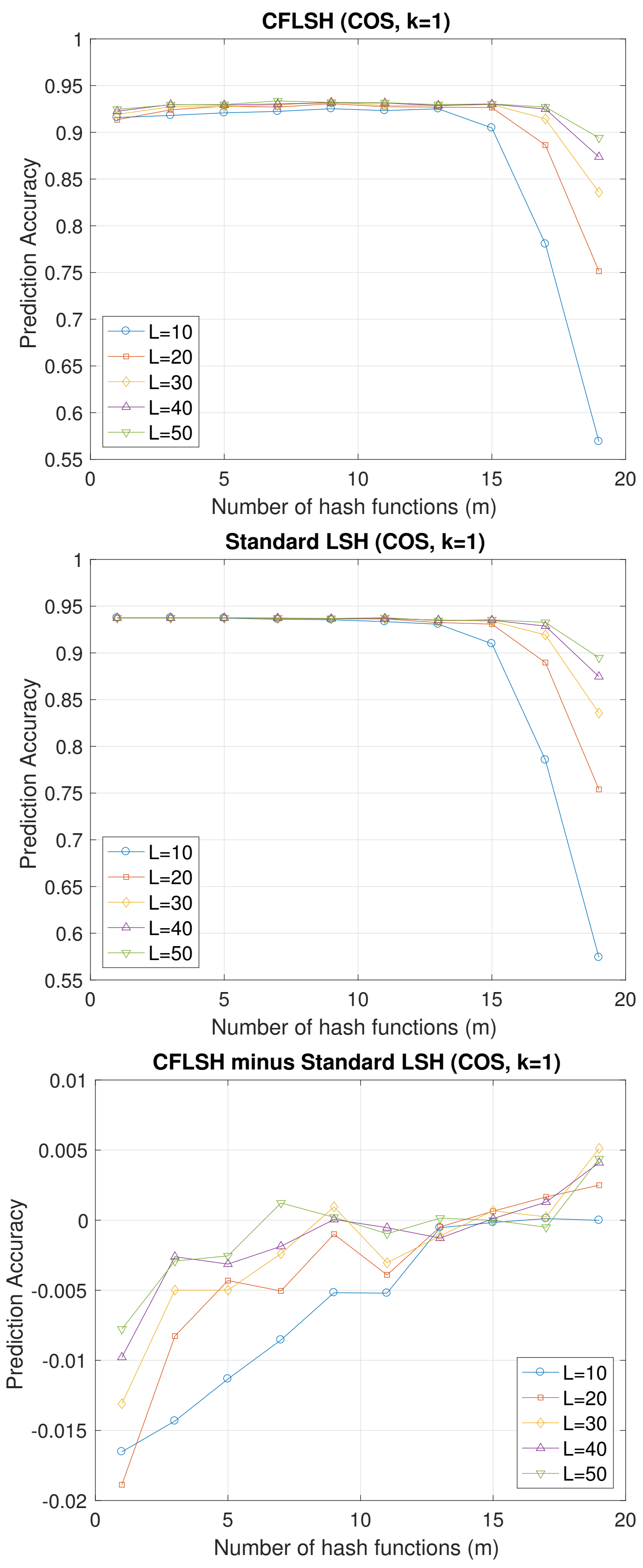


Figure 7-4: Sensitivity analysis of the prediction accuracy against the number of hash functions used per table (m) with cosine: (*Top*) CFLSH, (*Middle*) the standard LSH, and (*Bottom*) the difference between the two.

Chapter 8

Conclusions

In this final chapter, we summarize the contributions of this thesis and discuss future works. We conclude with final remarks.

8.1 Summary of Thesis Contributions

In this thesis, we developed highly efficient methods based on locality-sensitive hashing (LSH) to search through massive databases of physiological time series to identify waveforms that are similar to those from a given individual. Furthermore, we developed and applied the methods that exploit these "*patients with trajectories like mine*" retrievals to support waveform pattern recognition for critical event prediction. We demonstrated our methods on the mean arterial blood pressure dataset extracted from the MIMIC II database in the context of predicting acute hypotensive episodes in intensive care units. To the best of our knowledge, our work to date is the first extensive application of LSH on physiological time series retrieval and event prediction.

The contributions of this thesis are as follows.

- We are the first to apply LSH to the problem of retrieving similar physiological waveform time series. When compared to the exhaustive k-nearest neighbor (KNN) method, our methods based on LSH with the L1 (L1LSH), the cosine (COSLSH), and the Euclidean distances (E2LSH) each largely speed up the retrieval time of similar physiological waveforms without sacrificing significant accuracy. We achieved an order of magnitude speed-up at the cost of decreasing the retrieval accuracy by 5% compared to KNN. We performed the sensitivity analysis of retrieval accuracy and speed-up against the number of hash functions applied per table (m) and the number of hash tables created (L). We found that the retrieval accuracy decreases with increasing m and increases with increasing L, whereas the speed-up increases with increasing m and decreases with increasing L. Additionally, we examined the impact of the data dimension and the data quantity on the LSH performance. We found that using the data with lower dimension and larger quantity each improves retrieval accuracy and querying speed.

- We further extended the LSH based retrieval system to the problem of predicting a medically critical event (acute hypotension) by extrapolating the information of similar waveforms via majority vote. Similar to the retrieval case, compared to using the linear KNN, our LSH based prediction method vastly speeds up the prediction time up to an order (with L1LSH and E2LSH) or two orders (with COSLSH) of magnitude faster while sacrificing less than only 1% of prediction accuracy as a cost. Because prediction accuracy is highly influenced by the data imbalance we faced with our dataset, we additionally performed analysis in terms of correlation and false negative detection. We investigated the underlying factor of the large difference in the speed-ups among L1LSH, E2LSH, and COSLSH. We empirically showed that it originates from the differences in the distribution of bucket sizes (i.e. how uniformly data are indexed across hash buckets). We also examined the impact of lag duration and scaling on the LSH performance and demonstrated that a longer lag and a larger quantity of data each improves prediction accuracy and querying speed.

- We proposed a new variant of LSH, namely stratified locality-sensitive hashing (SLSH), which finds similarity among the data from a more integrated perspective by employing multiple distance metrics in one framework. SLSH is essentially a dual-level LSH where each LSH layer (L1LSH for the outer and COSLSH for the inner LSH) is associated with a distinct distance metric capturing a unique facet of similarity. SLSH overcomes the limitation of the standard LSH that only one distance measure can be used at a time. We visually presented that SLSH is capable of retrieving nearest neighbors according to both amplitude and shape. Comparing SLSH to the standard L1LSH, we demonstrated that SLSH yields a higher prediction accuracy and further shortens the sub-linear querying time of the standard LSH. We showed that this pattern still holds when the lag time of data shortens or the quantity of data scales up. We further examined whether using two distance metrics to capture similarity (via an ensemble of two single-metric predictors or via SLSH) is more beneficial than using a single-metric predictor and empirically proved that exploiting the data with multiple distance metrics is indeed advantageous.

- We proposed another new variant of LSH, namely collision frequency locality-sensitive hashing (CFLSH), to further improve the prediction accuracy without sacrificing any speed based on the key idea is that the more frequently an element and query collide across multiple LSH hash tables, the more similar they are. CFLSH short-lists the candidate set by simply counting the frequency of collision instead of performing the exhaustive distance calculation, which is the main bottleneck of the standard LSH. We empirically demonstrated that CFLSH with the L1 distance has a higher accuracy than L1LSH and further speeds-up the querying time, and that CFLSH is better than the standard LSH in the range of large m which corresponds to significant speed-ups.

8.2 Future Directions

There are multiple areas to further strengthen and extend our proposed methods presented in this thesis. Among many, we discuss three major areas for advancement: noise robust time series representation, LSH for multivariate multi-source data, and data adaptive hashing.

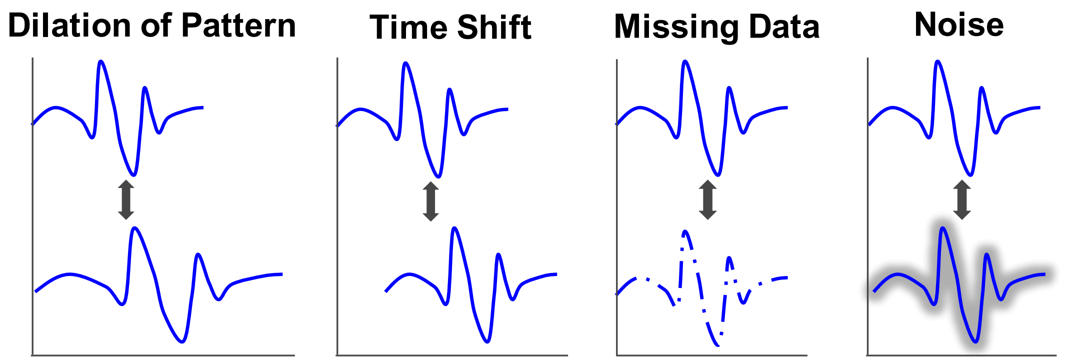


Figure 8-1: Sources of realistic data abnormalities in time series data.

8.2.1 Robust Time Series Representation

Although there have been numerous works in time series analysis, most of them are built on unrealistic assumptions. For example, it is typically assumed (unlike our work) that the training data are perfectly aligned patterns of all equal length with no extra spurious leading or trailing data. However, ICU data are typically very noisy and unaligned because it is more of observational data rather than data from a controlled experiment. This implies they are severely prone to data abnormalities such as noise, missing data, translation, and dilation of patterns (Figure 8-1), which all make calculating similarity within these data very challenging. As discussed in Chapter 2, several works have attempted to overcome such difficulties by finding either effective distance measures (e.g. dynamic time warping) or data representations (e.g. time-frequency transforms). However, while the former approach is effective on handling small local misalignments, it is still vulnerable to high-level (i.e. spanning a longer time period) abnormalities such as phase shifts. In order to find a better representation that captures high-level long term dynamics of physiological time series, which at the same time is robust against low-level, local noise, we propose that a solution based on the use of multi-resolution histograms will be effective.

In multi-resolution histograms, each time series trajectory is represented as a set of histograms. Generated within a window over a given time period, a single histogram keeps only the frequency information that is independent of time. This binning captures the local structure and makes it less sensitive to data abnormalities

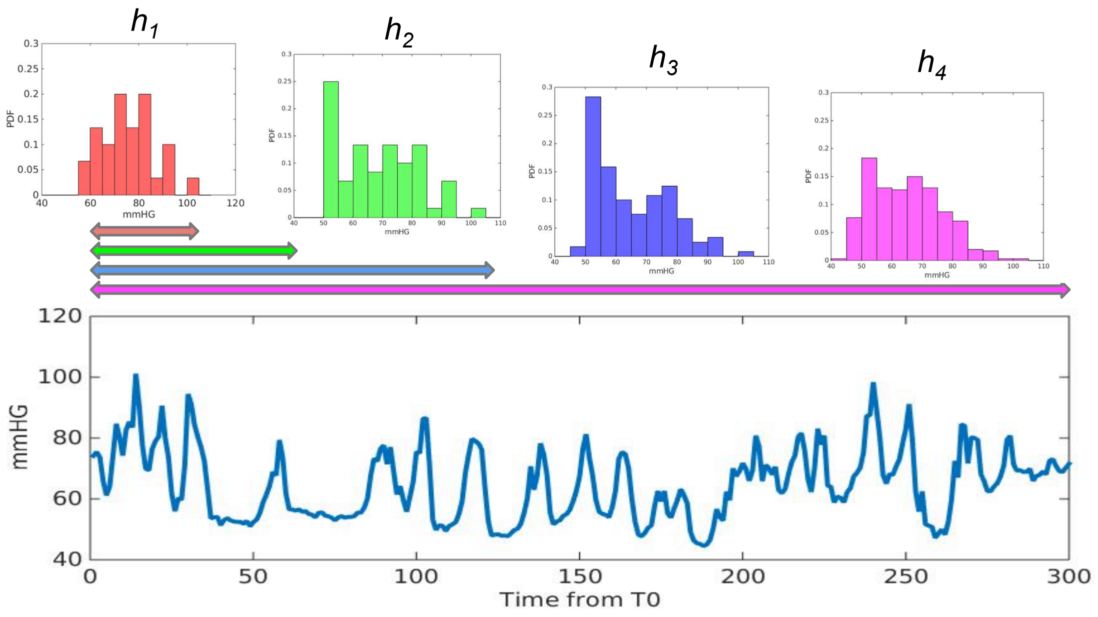


Figure 8-2: Multi-resolution histogram representation $H(x) = \sum_i w_i h_i(x)$ for a time series. Each histogram h_i, with its learned weight w_i, is built over a specified period of time $[t_0, t_i]$.

at the given temporal scale. It can also effectively mitigate the problem of missing data provided the quantity is small.

As shown in Figure 8-2, we hierarchically generate multiple histograms. For a time series x, each histogram $h_i(x)$, expressed as a normalized probability density, is built on the data from t_0 to t_i, where $t_i = m \cdot 2^{i-1}$, for $i = 1, 2, \cdots, n$ (m and n indicate the base window duration and the total number of histograms, respectively). Then, the multi-resolution histogram is $H(x) = \sum_i w_i h_i(x)$, where w_i's are the weights associated at each histogram resolution. By having the multi-scale hierarchical structure over time, we are able to model high-level long term dynamics and behavior. The bin numbers for each histogram, the degree of resolution (i.e. the number of histograms) over time, and the weights at each resolution need to be learned empirically.

For comparing histograms, it is known that the Earth Mover's Distance (EMD) is the most accurate measure. There remains the task of incorporating EMD into LSH to make the multi-resolution histogram based representation even more powerful in this framework. However, for EMD, there is no known family of locality-sensitive hash functions. Thus, we propose to first embed EMD into the space of the L1 distance whose LSH family of hash functions is well-studied [55].

8.2.2 Locality-Sensitive Hashing for Multivariate, Multi-Source Data

Besides measuring arterial blood pressure (which is single source, univariate), there are several other patient monitoring data sources acquired in intensive care units such as electrocardiogram (ECG), body temperature, cardiac output, and amount of oxygen and carbon dioxide in the blood (discussed in detail in Chapter 3). Within each source/channel, there can be multiple features or variables. Thus, in order to leverage multiple sources and features, there is a need to extend our work for multi-source multivariate data.

One possible approach for LSH to handle multivariate data is to hash each variate spanning a single source to a separate hash key of the same size and concatenate it with the keys of other similarly hashed variates to form a higher order bucket index/identifier, effectively allocating a set of multi-variate buckets, one per source, rather than a single bucket. Then for each source/channel, we can repeat this step. Finally, for querying, we can either combine the candidate sets resulting from each source, or take the intersection among them to compose the final candidate set.

It is largely unknown which source among many is best to predict a certain class of event and how reliable each source is. It can be very event-specific and also for a particular event, event-class specific. In many cases, the best subset of time series to use is almost always class-dependent. Investigations in these issues need to be conducted as well.

8.2.3 Data and Task Dependent Hashing

In this thesis, we used hashing methods with distance metrics that are independent of the data properties. Although these methods hold the advantage of strict performance guarantees, they could be more efficient if the hash functions were specifically designed for a certain dataset or task. In other words, even with more integrated LSH methods (such as SLSH), it is possible that there remains a semantic gap, which is the discrepancy between true similarity and what can be captured with distance metrics. Retrieving approximate nearest neighbors in such metric spaces may not lead to good search performance when semantic similarity is represented in a complex way. Therefore, as our future work, we are interested in investigating "learning to hash" methods [122,125] that learn data-dependent and task-specific hash functions to achieve better prediction accuracy and querying speed. Such methods include spectral hashing [128, 129], angular quantization [42], binary reconstructive embedding [79], metric learning hashing [81], semi-supervised hashing [123, 124], and deep-learning based semantic hashing [109].

8.3 Final Remarks

In this thesis, we developed highly efficient methods based on LSH that make it practical to search massive physiological time series repositories to rapidly identify waveforms similar to those from a given individual. We then extended LSH to exploit rapid waveform retrieval to enable critical event prediction in the intensive care unit setting.

Even though many more subsequent works remain to be investigated, we hope and believe that our work based on efficiently finding "*patients with trajectories like mine*" will contribute as a stepping stone toward better diagnostic precision, detection of critical health events, and more individualized treatments and interventions in the near future.

Bibliography

[1] Charu C. Aggarwal, Alexander Hinneburg, and Daniel A. Keim. On the surprising behavior of distance metrics in high dimensional spaces. In *Proceedings of the 8th International Conference on Database Theory*, ICDT '01, pages 420–434. Springer-Verlag, 2001.

[2] Osamah M Al-Qershi and Bee Ee Khoo. Copy-move forgery detection using on locality sensitive hashing and k-means clustering. In *Information Science and Applications (ICISA) 2016*, pages 663–672. Springer, 2016.

[3] Alexandr Andoni and Piotr Indyk. Efficient algorithms for substring near neighbor problem. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1203–1212. Society for Industrial and Applied Mathematics, 2006.

[4] Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pages 459–468. IEEE, 2006.

[5] Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Communications of the ACM*, 51(1):117–122, January 2008.

[6] Alexandr Andoni, Piotr Indyk, Thijs Laarhoven, Ilya Razenshteyn, and Ludwig Schmidt. Practical and optimal LSH for angular distance. In *Advances in Neural Information Processing Systems*, pages 1225–1233, 2015.

[7] Alexandr Andoni, Piotr Indyk, Huy L Nguyen, and Ilya Razenshteyn. Beyond locality-sensitive hashing. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1018–1028. Society for Industrial and Applied Mathematics, 2014.

[8] Alexandr Andoni and Ilya Razenshteyn. Optimal data-dependent hashing for approximate near neighbors. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*, pages 793–801. ACM, 2015.

[9] Alexandr Andoni, Ilya Razenshteyn, and Negev Shekel Nosatzki. LSH forest: Practical algorithms made theoretical. In *Proceedings of the Twenty-Eighth*

Annual ACM-SIAM Symposium on Discrete Algorithms, pages 67–78. SIAM, 2017.

[10] Bahman Bahmani, Ashish Goel, and Rajendra Shinde. Efficient distributed locality sensitive hashing. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, pages 2174–2178. ACM, 2012.

[11] Paul Barach and Stephen D Small. Reporting and preventing medical mishaps: lessons from non-medical near miss reporting systems. *British Medical Journal*, 320(7237):759–763, 2000.

[12] Mayank Bawa, Tyson Condie, and Prasanna Ganesan. LSH forest: self-tuning indexes for similarity search. In *Proceedings of the 14th International Conference on World Wide Web*, pages 651–660. ACM, 2005.

[13] Susanna E Bedell, David C Deitz, David Leeman, and Thomas L Delbanco. Incidence and characteristics of preventable latrogenic cardiac arrests. *JAMA*, 265(21):2815–2820, 1991.

[14] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[15] Konstantin Berlin, Sergey Koren, Chen-Shan Chin, James P Drake, Jane M Landolin, and Adam M Phillippy. Assembling large genomes with single-molecule sequencing and locality-sensitive hashing. *Nature Biotechnology*, 33(6):623–630, 2015.

[16] Donald J Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, pages 359–370. Seattle, WA, 1994.

[17] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time Series Analysis: Forecasting and Control.* John Wiley & Sons, 2015.

[18] Andrei Z Broder. On the resemblance and containment of documents. In *Compression and Complexity of Sequences 1997. Proceedings*, pages 21–29. IEEE, 1997.

[19] Kin-Pong Chan and Ada Wai-Chee Fu. Efficient time series matching by wavelets. In *Data Engineering (ICDE), 1999 IEEE 15th International Conference on*, pages 126–133. IEEE, 1999.

[20] Moses S Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the Thiry-Fourth Annual ACM Symposium on Theory of Computing*, pages 380–388. ACM, 2002.

[21] Lei Chen and Raymond Ng. On the marriage of lp-norms and edit distance. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases*, pages 792–803. VLDB Endowment, 2004.

[22] Lei Chen, M Tamer Özsu, and Vincent Oria. Robust and fast similarity search for moving object trajectories. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, pages 491–502. ACM, 2005.

[23] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on*, 13(1):21–27, 1967.

[24] Abhinandan S Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. Google news personalization: scalable online collaborative filtering. In *Proceedings of the 16th International Conference on World Wide Web*, pages 271–280. ACM, 2007.

[25] MIT Critical Data. MIMIC-II. http://criticaldata.mit.edu/mimic-ii/. Accessed: 2017-03-15.

[26] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the Twentieth Annual Symposium on Computational Geometry*, pages 253–262. ACM, 2004.

[27] Jeffrey Dean and Monika R Henzinger. Finding related pages in the world wide web. *Computer Networks*, 31(11):1467–1479, 1999.

[28] Franck Dernoncourt. BeatDB: An end-to-end approach to unveil saliencies from massive signal data sets. Master's thesis, Massachusetts Institute of Technology, 2014.

[29] Franck Dernoncourt, Kalyan Veeramachaneni, and Una-May O'Reilly. BeatDB: A large scale waveform feature repository. In *NIPS Workshop on Machine Learning for Clinical Data Analysis and Healthcare*, 2013.

[30] Franck Dernoncourt, Kalyan Veeramachaneni, and Una-May O'Reilly. Gaussian process-based feature selection for wavelet parameters: predicting acute hypotensive episodes from physiological signals. In *Computer-Based Medical Systems (CBMS), 2015 IEEE 28th International Symposium on*, pages 145–150. IEEE, 2015.

[31] Hui Ding, Goce Trajcevski, Peter Scheuermann, Xiaoyue Wang, and Eamonn Keogh. Querying and mining of time series data: experimental comparison of representations and distance measures. *Proceedings of the VLDB Endowment*, 1(2):1542–1552, 2008.

[32] David Dobkin and Richard J Lipton. Multidimensional searching problems. *SIAM Journal on Computing*, 5(2):181–186, 1976.

[33] Wei Dong, Zhe Wang, William Josephson, Moses Charikar, and Kai Li. Modeling LSH for performance tuning. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, pages 669–678. ACM, 2008.

[34] Kave Eshghi and Shyamsundar Rajaram. Locality sensitive hash functions based on concomitant rank order statistics. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 221–229. ACM, 2008.

[35] Christos Faloutsos, M. Ranganathan, and Yannis Manolopoulos. Fast subsequence matching in time-series databases. In *Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data*, SIGMOD '94, pages 419–429, New York, NY, USA, 1994. ACM.

[36] GP Findlay, H Shotton, K Kelly, and M Mason. Time to intervene? A review of patients who underwent cardiopulmonary resuscitation as a result of an in-hospital cardiorespiratory arrest. *A report by the National Confidential Enquiry into Patient Outcome and Death*, 2012.

[37] Tak-chung Fu. A review on time series data mining. *Engineering Applications of Artificial Intelligence*, 24(1):164–181, 2011.

[38] Junhao Gan, Jianlin Feng, Qiong Fang, and Wilfred Ng. Locality-sensitive hashing scheme based on dynamic collision counting. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pages 541–552. ACM, 2012.

[39] Jinyang Gao, Hosagrahar Visvesvaraya Jagadish, Wei Lu, and Beng Chin Ooi. DSH: data sensitive hashing for high-dimensional k-nn search. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, pages 1127–1138. ACM, 2014.

[40] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity search in high dimensions via hashing. In *Proceedings of the 25th International Conference on Very Large Data Bases*, VLDB '99, pages 518–529. Morgan Kaufmann Publishers Inc., 1999.

[41] Ary L Goldberger, Luis AN Amaral, Leon Glass, Jeffrey M Hausdorff, Plamen Ch Ivanov, Roger G Mark, Joseph E Mietus, George B Moody, Chung-Kang Peng, and H Eugene Stanley. Physiobank, Physiotoolkit, and Physionet. *Circulation*, 101(23):e215–e220, 2000.

[42] Yunchao Gong, Sanjiv Kumar, Vishal Verma, and Svetlana Lazebnik. Angular quantization-based binary codes for fast similarity search. In *Advances in Neural Information Processing Systems*, pages 1196–1204, 2012.

[43] David Gorisse, Matthieu Cord, and Frederic Precioso. Locality-sensitive hashing for Chi2 distance. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(2):402–409, 2012.

[44] Kristen Grauman and Trevor Darrell. Fast contour matching using approximate earth mover's distance. In *Computer Vision and Pattern Recognition. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 1, pages I–220–I–227. IEEE, 2004.

[45] Xiaoguang Gu, Yongdong Zhang, Lei Zhang, Dongming Zhang, and Jintao Li. An improved method of locality sensitive hashing for indexing large-scale and high-dimensional features. *Signal Processing*, 93(8):2244–2255, 2013.

[46] Antonin Guttman. *R-trees: a dynamic index structure for spatial searching*, volume 14. ACM, 1984.

[47] Alon Halevy, Peter Norvig, and Fernando Pereira. The unreasonable effectiveness of data. *Intelligent Systems, IEEE*, 24(2):8–12, 2009.

[48] James Douglas Hamilton. *Time Series Analysis.* Princeton University Press, 1994.

[49] Sariel Har-Peled, Piotr Indyk, and Rajeev Motwani. Approximate nearest neighbor: Towards removing the curse of dimensionality. *Theory of Computing*, 8(1):321–350, 2012.

[50] JH Henriques and TR Rocha. Prediction of acute hypotensive episodes using neural network multi-models. In *Computers in Cardiology, 2009*, pages 549–552. IEEE, 2009.

[51] Monika Henzinger. Finding near-duplicate web pages: A large-scale evaluation of algorithms. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 284–291. ACM, 2006.

[52] Bing Hu, Yanping Chen, and Eamonn Keogh. Time series classification under more realistic assumptions. In *Proceedings of the 2013 SIAM International Conference on Data Mining*, pages 578–586. SIAM, 2013.

[53] Qiang Huang, Jianlin Feng, Yikai Zhang, Qiong Fang, and Wilfred Ng. Query-aware locality-sensitive hashing for approximate nearest neighbor search. *Proceedings of the VLDB Endowment*, 9(1):1–12, 2015.

[54] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, pages 604–613. ACM, 1998.

[55] Piotr Indyk and Nitin Thaper. Fast Image Retrieval via Embeddings. In *3rd International Workshop on Statistical and Computational Theories of Vision.* ICCV, 2003.

[56] Prateek Jain, Sudheendra Vijayanarasimhan, and Kristen Grauman. Hashing hyperplane queries to near points with applications to large-scale active learning. In *Advances in Neural Information Processing Systems*, pages 928–936, 2010.

[57] Hervé Jégou, Laurent Amsaleg, Cordelia Schmid, and Patrick Gros. Query adaptative locality sensitive hashing. In *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, pages 825–828. IEEE, 2008.

[58] Jianqiu Ji, Jianmin Li, Shuicheng Yan, Qi Tian, and Bo Zhang. Min-max hash for Jaccard similarity. In *Data Mining (ICDM), 2013 IEEE 13th International Conference on*, pages 301–309. IEEE, 2013.

[59] Jianqiu Ji, Jianmin Li, Shuicheng Yan, Bo Zhang, and Qi Tian. Super-bit locality-sensitive hashing. In *Advances in Neural Information Processing Systems*, pages 108–116, 2012.

[60] Ping Jiang, Jonathan Winkley, Can Zhao, Robert Munnoch, Geyong Min, and Laurence T Yang. An intelligent information forwarder for healthcare big data systems with distributed wearable sensors. *IEEE Systems Journal*, 10(3):1147–1159, 2016.

[61] Zhongming Jin, Cheng Li, Yue Lin, and Deng Cai. Density sensitive hashing. *IEEE Transactions on Cybernetics*, 44(8):1362–1371, 2014.

[62] Alistair EW Johnson, Mohammad M Ghassemi, Shamim Nemati, Katherine E Niehaus, David A Clifton, and Gari D Clifford. Machine learning and decision support in critical care. *Proceedings of the IEEE*, 104(2):444–466, 2016.

[63] Alistair EW Johnson, Tom J Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. MIMIC-III, A freely accessible critical care database. *Scientific Data*, 3, 2016.

[64] Alexis Joly and Olivier Buisson. A posteriori multi-probe locality sensitive hashing. In *Proceedings of the 16th ACM International Conference on Multimedia*, pages 209–218. ACM, 2008.

[65] David C Kale, Dian Gong, Zhengping Che, Yan Liu, Gerard Medioni, Randall Wetzel, and Patrick Ross. An examination of multivariate time series hashing with applications to health care. In *Data Mining (ICDM), 2014 IEEE International Conference on*, pages 260–269. IEEE, 2014.

[66] Norio Katayama and Shin'ichi Satoh. The sr-tree: An index structure for high-dimensional nearest neighbor queries. In *ACM SIGMOD Record*, volume 26, pages 369–380. ACM, 1997.

[67] Eamonn Keogh, Kaushik Chakrabarti, Michael Pazzani, and Sharad Mehrotra. Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and Information Systems*, 3(3):263–286, 2001.

[68] Eamonn Keogh and Chotirat Ann Ratanamahatana. Exact indexing of dynamic time warping. *Knowledge and Information Systems*, 7(3):358–386, 2005.

[69] Minje Kim, Paris Smaragdis, and Gautham J Mysore. Efficient manifold preserving audio source separation using locality sensitive hashing. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 479–483. IEEE, 2015.

[70] Yongwook Bryce Kim, Erik Hemberg, and Una-May O'Reilly. Stratified locality-sensitive hashing for accelerated physiological time series retrieval. In *Engineering in Medicine and Biology Society (EMBC), 2016 IEEE 38th Annual International Conference of the*, pages 2479–2483. IEEE, 2016.

[71] Yongwook Bryce Kim, Erik Hemberg, and Una-May O'Reilly. Stratified locality-sensitive hashing for sublinear time critical event prediction. In *Advances in Neural Information Processing Systems (NIPS) Machine Learning in Healthcare Workshop*, 2016.

[72] Yongwook Bryce Kim, Erik Hemberg, and Una-May O'Reilly. Collision frequency locality-sensitive hashing for prediction of critical events. In *Engineering in Medicine and Biology Society (EMBC), 2017 39th Annual International Conference of the IEEE*. IEEE, 2017. To appear.

[73] Yongwook Bryce Kim and Una-May O'Reilly. Large-scale physiological waveform retrieval via locality-sensitive hashing. In *Engineering in Medicine and Biology Society (EMBC), 2015 37th Annual International Conference of the IEEE*, pages 5829–5833. IEEE, 2015.

[74] Yongwook Bryce Kim and Una-May O'Reilly. Analysis of locality-sensitive hashing for fast critical event prediction on physiological time series. In *Engineering in Medicine and Biology Society (EMBC), 2016 IEEE 38th Annual International Conference of the*, pages 783–787. IEEE, 2016.

[75] Yongwook Bryce Kim, Joohyun Seo, and Una-May O'Reilly. Large-scale methodological comparison of acute hypotensive episode forecasting using MIMIC2 physiological waveforms. In *Computer-Based Medical Systems (CBMS), 2014 IEEE 27th International Symposium on*, pages 319–324. IEEE, 2014.

[76] Simon Korman and Shai Avidan. Coherency sensitive hashing. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 1607–1614. IEEE, 2011.

[77] Simon Korman and Shai Avidan. Coherency sensitive hashing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(6):1099–1112, 2016.

[78] Flip Korn, Hosagrahar V Jagadish, and Christos Faloutsos. Efficiently supporting ad hoc queries in large datasets of time sequences. *ACM SIGMOD Record*, 26(2):289–300, 1997.

[79] Brian Kulis and Trevor Darrell. Learning to hash with binary reconstructive embeddings. In *Advances in Neural Information Processing Systems*, pages 1042–1050, 2009.

[80] Brian Kulis and Kristen Grauman. Kernelized locality-sensitive hashing for scalable image search. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 2130–2137. IEEE, 2009.

[81] Brian Kulis, Prateek Jain, and Kristen Grauman. Fast similarity search for learned metrics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(12):2143–2157, 2009.

[82] Thijs Laarhoven. Sieving for shortest vectors in lattices using angular locality-sensitive hashing. In *Annual Cryptology Conference*, pages 3–22. Springer, 2015.

[83] Hongrae Lee, Raymond T Ng, and Kyuseok Shim. Similarity join size estimation using locality sensitive hashing. *Proceedings of the VLDB Endowment*, 4(6):338–349, 2011.

[84] LH Lehman, M Saeed, GB Moody, and RG Mark. Similarity-based searching in multi-parameter time series databases. In *Computers in Cardiology, 2008*, pages 653–656. IEEE, 2008.

[85] Ping Li and Arnd Christian König. Theory and applications of b-bit minwise hashing. *Communications of the ACM*, 54(8):101–109, 2011.

[86] Jessica Lin, Eamonn Keogh, Li Wei, and Stefano Lonardi. Experiencing SAX: a novel symbolic representation of time series. *Data Mining and Knowledge Discovery*, 15(2):107–144, 2007.

[87] Kang Ling and Gangshan Wu. Frequency based locality sensitive hashing. In *Multimedia Technology (ICMT), 2011 International Conference on*, pages 4929–4932. IEEE, 2011.

[88] Qin Lv, William Josephson, Zhe Wang, Moses Charikar, and Kai Li. Multi-probe LSH: efficient indexing for high-dimensional similarity search. In *Proceedings of the 33rd International Conference on Very Large Data Bases*, pages 950–961. VLDB Endowment, 2007.

[89] Gurmeet Singh Manku, Arvind Jain, and Anish Das Sarma. Detecting near-duplicates for web crawling. In *Proceedings of the 16th International Conference on World Wide Web*, pages 141–150. ACM, 2007.

[90] J Frederick McNeer, C Frank Starmer, Alan G Bartel, Victor S Behar, Yihong Kong, Robert H Peter, and Robert A Rosati. The nature of treatment selection in coronary artery disease. *Circulation*, 49(4):606–614, 1974.

[91] GB Moody and LH Lehman. Predicting acute hypotensive episodes: The 10th annual physionet/computers in cardiology challenge. In *Computers in Cardiology, 2009*, pages 541–544. IEEE, 2009.

[92] George B Moody and Roger G Mark. A database to support development and evaluation of intelligent intensive care monitoring. In *Computers in Cardiology, 1996*, pages 657–660. IEEE, 1996.

[93] Abdullah Mueen, Krishnamurthy Viswanathan, Chetan Gupta, and Eamonn Keogh. The fastest similarity search algorithm for time series subsequences under Euclidean distance, August 2015. http://www.cs.unm.edu/~mueen/FastestSimilaritySearch.html.

[94] Kevin Patrick Murphy. *Dynamic bayesian networks: representation, inference and learning.* PhD thesis, University of California, Berkeley, 2002.

[95] Ryan O'Donnell, Yi Wu, and Yuan Zhou. Optimal lower bounds for locality-sensitive hashing (except when q is tiny). *ACM Transactions on Computation Theory (TOCT)*, 6(1):5, 2014.

[96] Ciprian Oprisa, Marius Checiches, and Adrian Nandrean. Locality-sensitive hashing optimizations for fast malware clustering. In *Intelligent Computer Communication and Processing (ICCP), 2014 IEEE International Conference on*, pages 97–104. IEEE, 2014.

[97] Jia Pan and Dinesh Manocha. Bi-level locality sensitive hashing for k-nearest neighbor computation. In *Data Engineering (ICDE), 2012 IEEE 28th International Conference on*, pages 378–389. IEEE, 2012.

[98] Jia Pan and Dinesh Manocha. Fast probabilistic collision checking for sampling-based motion planning using locality-sensitive hashing. *The International Journal of Robotics Research*, 35(12):1477–1496, 2016.

[99] Rina Panigrahy. Entropy based nearest neighbor search in high dimensions. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm*, pages 1186–1195. Society for Industrial and Applied Mathematics, 2006.

[100] Loïc Paulevé, Hervé Jégou, and Laurent Amsaleg. Locality sensitive hashing: A comparison of hash function types and querying mechanisms. *Pattern Recognition Letters*, 31(11):1348–1358, 2010.

[101] PhysiologyWeb. Mean arterial pressure calculator. http://www.physiologyweb.com/calculators/mean_arterial_pressure_calculator.html. Accessed: 2017-03-14.

[102] David Martin Powers. Evaluation: From precision, recall and F-measure to ROC, informedness, markedness & correlation. *Journal of Machine Learning Technologies*, 2(1):37–63, 2011.

[103] Thanawin Rakthanmanon, Bilson Campana, Abdullah Mueen, Gustavo Batista, Brandon Westover, Qiang Zhu, Jesin Zakaria, and Eamonn Keogh. Searching and mining trillions of time series subsequences under dynamic time warping. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 262–270. ACM, 2012.

[104] Ilya Razenshteyn and Ludwig Schmidt. FALCONN. https://falconn-lib.org/. Accessed: 2017-04-12.

[105] Robert A Rosati, J Frederick McNeer, C Frank Starmer, Brant S Mittler, James J Morris, and Andrew G Wallace. A new information system for medical practice. *Archives of Internal Medicine*, 135(8):1017–1024, 1975.

[106] Matti Ryynanen and Anssi Klapuri. Query by humming of MIDI and audio using locality sensitive hashing. In *Acoustics, Speech and Signal Processing (ICASSP), 2008 IEEE International Conference on*, pages 2249–2252. IEEE, 2008.

[107] Mohammed Saeed and Roger G Mark. A novel method for the efficient retrieval of similar multiparameter physiologic time series using wavelet-based symbolic representations. In *AMIA*, 2006.

[108] Mohammed Saeed, Mauricio Villarroel, Andrew T Reisner, Gari Clifford, Li-Wei Lehman, George Moody, Thomas Heldt, Tin H Kyaw, Benjamin Moody, and Roger G Mark. Multiparameter intelligent monitoring in intensive care II (MIMIC-II): A public-access intensive care unit database. *Critical Care Medicine*, 39(5):952, 2011.

[109] Ruslan Salakhutdinov and Geoffrey Hinton. Semantic hashing. *International Journal of Approximate Reasoning*, 50(7):969–978, 2009.

[110] Ludwig Schmidt, Matthew Sharifi, and Ignacio Lopez Moreno. Large-scale speaker identification. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 1650–1654. IEEE, 2014.

[111] Gregory Shakhnarovich, Paul Viola, and Trevor Darrell. Fast pose estimation with parameter-sensitive hashing. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 750–757. IEEE, 2003.

[112] Malcolm Slaney, Yury Lifshits, and Junfeng He. Optimal parameters for locality-sensitive hashing. *Proceedings of the IEEE*, 100(9):2604–2623, 2012.

[113] Narayanan Sundaram, Aizana Turmukhametova, Nadathur Satish, Todd Mostak, Piotr Indyk, Samuel Madden, and Pradeep Dubey. Streaming similarity search over one billion tweets using parallel locality-sensitive hashing. *Proceedings of the VLDB Endowment*, 6(14):1930–1941, 2013.

[114] Zeeshan Syed, Piotr Indyk, and John Guttag. Learning approximate sequential patterns for classification. *Journal of Machine Learning Research*, 10(Aug):1913–1936, 2009.

[115] Kengo Terasawa and Yuzuru Tanaka. Spherical LSH for approximate nearest neighbor search on unit hypersphere. In *Workshop on Algorithms and Data Structures*, pages 27–38. Springer, 2007.

[116] Vikrant Singh Tomar and Richard C Rose. Efficient manifold learning for speech recognition using locality sensitive hashing. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 6995–6999. IEEE, 2013.

[117] Dan Vanderkam, Rob Schonberger, Henry Rowley, and Sanjiv Kumar. Nearest neighbor search in Google correlate. Technical report, Google, 2013.

[118] Vasilis Verroios and Hector Garcia-Molina. Top-k entity resolution with adaptive locality-sensitive hashing. Technical report, Stanford InfoLab, 2017.

[119] Michail Vlachos, George Kollios, and Dimitrios Gunopulos. Discovering similar multidimensional trajectories. In *Data Engineering (ICDE), 2002 IEEE 18th International Conference on*, pages 673–684. IEEE, 2002.

[120] Alexander Waldin, Kalyan Veeramachaneni, and Una-May O'Reilly. Learning blood pressure behavior from large physiological waveform repositories. In *ICML Workshop on Role of Machine Learning in Transforming Healthcare*, 2013.

[121] Jeremy R Wang and Corbin D Jones. Fast alignment filtering of nanopore sequencing reads using locality-sensitive hashing. In *Bioinformatics and Biomedicine (BIBM), 2015 IEEE International Conference on*, pages 127–130. IEEE, 2015.

[122] Jingdong Wang, Heng Tao Shen, Jingkuan Song, and Jianqiu Ji. Hashing for similarity search: A survey. *arXiv preprint arXiv:1408.2927*, 2014.

[123] Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. Semi-supervised hashing for scalable image retrieval. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3424–3431. IEEE, 2010.

[124] Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. Semi-supervised hashing for large-scale search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(12):2393–2406, 2012.

[125] Jun Wang, Wei Liu, Sanjiv Kumar, and Shih-Fu Chang. Learning to hash for indexing big data: A survey. *Proceedings of the IEEE*, 104(1):34–57, 2016.

[126] Zhe Wang, Wei Dong, William Josephson, Qin Lv, Moses Charikar, and Kai Li. Sizing sketches: a rank-based analysis for similarity search. In *ACM SIGMETRICS Performance Evaluation Review*, volume 35, pages 157–168. ACM, 2007.

[127] Roger Weber, Hans-Jörg Schek, and Stephen Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *Proceedings of the 24rd International Conference on Very Large Data Bases*, VLDB '98, pages 194–205, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[128] Yair Weiss, Rob Fergus, and Antonio Torralba. Multidimensional spectral hashing. In *Proceedings of the 12th European Conference on Computer Vision*, ECCV'12, pages 340–353, Berlin, Heidelberg, 2012. Springer-Verlag.

[129] Yair Weiss, Antonio Torralba, and Rob Fergus. Spectral hashing. In *Advances in Neural Information Processing Systems*, pages 1753–1760, 2009.

[130] Jonathan Woodbridge, Bobak Mortazavi, Alex AT Bui, and Majid Sarrafzadeh. High performance biomedical time series indexes using salient segmentation. In *Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE*, pages 5086–5089. IEEE, 2012.

[131] Xindong Wu, Vipin Kumar, J. Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey McLachlan, Angus Ng, Bing Liu, Philip Yu, Zhi-Hua Zhou, Michael Steinbach, David Hand, and Dan Steinberg. Top 10 algorithms in data mining. *Knowledge and Information Systems*, 14(1):1–37, January 2008.

[132] Zhenyu Wu and Ming Zou. An incremental community detection method for social tagging systems using locality-sensitive hashing. *Neural Networks*, 58:14–28, 2014.

[133] Xiaopeng Xi, Eamonn Keogh, Christian Shelton, Li Wei, and Chotirat Ann Ratanamahatana. Fast time series classification using numerosity reduction. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 1033–1040. ACM, 2006.

[134] Byoungkee Yi and Christos Faloutsos. Fast time sequence indexing for arbitrary Lp norms. In *The VLDB Journal*, pages 385–394. Morgan Kaufmann, 2000.

[135] Yi Yu, Suhua Tang, and Roger Zimmermann. Edge-based locality sensitive hashing for efficient geo-fencing application. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 576–579. ACM, 2013.

[136] Xuyun Zhang, Christopher Leckie, Wanchun Dou, Jinjun Chen, Ramamohanarao Kotagiri, and Zoran Salcic. Scalable local-recoding anonymization using locality sensitive hashing for big data privacy preservation. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 1793–1802. ACM, 2016.

[137] Ying Zhang, Huchuan Lu, Lihe Zhang, Xiang Ruan, and Shun Sakai. Video anomaly detection based on locality sensitive hashing filters. *Pattern Recognition*, 59:302–311, 2016.

[138] Yuhui Zhang, Yue-jiao Gong, Huaxiang Zhang, Tian-Long Gu, and Jun Zhang. Towards fast niching evolutionary algorithms: A locality sensitive hashing-based approach. *IEEE Transactions on Evolutionary Computation*, 2016.