Published in final edited form as:

Pervasive Mob Comput. 2016 June ; 28: 69–80. doi:10.1016/j.pmcj.2015.09.006.

Improving Biomedical Signal Search Results in Big Data Case-Based Reasoning Environments

Jonathan Woodbridge¹, Bobak Mortazavi¹, Alex A.T. Bui², and Majid Sarrafzadeh¹

¹Computer Science Department, UCLA, Los Angeles, CA 90095

²Medical Imaging Informatics, UCLA, Los Angeles, CA 90095

Abstract

Time series subsequence matching has importance in a variety of areas in healthcare informatics. These include case-based diagnosis and treatment as well as discovery of trends among patients. However, few medical systems employ subsequence matching due to high computational and memory complexities. This manuscript proposes a randomized Monte Carlo sampling method to broaden search criteria with minimal increases in computational and memory complexities over R -NN indexing. Information gain improves while producing result sets that approximate the theoretical result space, query results increase by several orders of magnitude, and recall is improved with no significant degradation to precision over R -NN matching.

Keywords

Biomedical Signal Search; Locality Sensitive Hashing; Time-Series Subsequence Matching; Monte Carlo Sampling; Case-Based Reasoning

1. Introduction

Medical case-based reasoning (CBR) is a well-studied area for medical diagnosis and treatments [1][2][3]. These systems rely on data mining and machine learning techniques to derive decisions on new cases based on a knowledge-base of previous cases. One major drawback of CBR systems is that the knowledge base must contain relevant cases to correctly make decisions on a current case [4][3]. This is especially difficult for high dimensional measured signals, such as Electrocardiogram (ECG) [5] and accelerometers which exhibit high variability due to noise, sensor displacement, misuse, and other factors that are difficult to control [6]. Systems hope to use such information to accurately classify signals and patients for an accurate medical diagnosis [7], dealing with the complexities as well as the potential issues of missing data [8]. A CBR system based on high dimensional measured signals must be extremely large to not only account for the variability of patients,

jwoodbri@cs.ucla.edu, bobakm@cs.ucla.edu, majid@cs.ucla.edu, buia@mii.ucla.edu

Publisher's Disclaimer: This is a PDF file of an unedited manuscript that has been accepted for publication. As a service to our customers we are providing this early version of the manuscript. The manuscript will undergo copyediting, typesetting, and review of the resulting proof before it is published in its final citable form. Please note that during the production process errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.

but to the variability of the signal type. The memory and computational complexity of such systems can limit the information gain provided. Indeed, as medical devices produce larger quantities of data and more frequently, efficient search becomes paramount in identifying important information and searching for useful and related signals. This work will investigate not only the quality of information presented in the signal search engine developed, but the speed with which such a system returns results, for usefulness in a case-based reasoning environment.

High dimensional subsequence matching, or R nearest neighbor (R -NN), is the process of finding similar segments within a database of high dimensional measured signals. A match is defined as any two segments $u, v \in S$ such that $\text{dist}(u, v) \leq R$ where S is the search space of biomedical signals, dist is a measure of distance between two signals (such as Euclidean distance), and R is a predefined threshold. In practice, R tends to be relatively small, leading to homogeneous result sets. While results may be precise, they offer little information gain. Of course, result sets can be enlarged by increasing R . However, arbitrarily increasing R can destabilize the result set, rendering it meaningless [9].

This paper presents a randomized Monte Carlo approach for improving R -NN search results. This approach enlarges search results while ensuring precision and yielding higher relative information gain. The method is built upon two assumptions: time series databases are extremely large [10] and result sets follow a Gaussian distribution [11][12]. The proposed method consists of two steps. First, a query segment q undergoes m randomizations constructing a set Q of query segments where $|Q| = m$. Next, an R -NN search for each $u \in Q$ is performed using the l_2 norm (Euclidean distance). The Euclidean distance between q and all segments $u \in Q$ follows a Gaussian distribution with a mean μ_Q and standard deviation σ_Q determined by the randomization.

There are several R -NN methods that exist in the literature including spatial indexes [13] [14] [15] and Locality Sensitive Hashing (LSH) [16] [17] [18]. This paper utilizes LSH as the underlying hash-based nearest neighbor search algorithm, but the theoretical contributions of this paper are applicable to most R -NN methods. However, the optimizations proposed by this paper were designed predominately for an LSH scheme.

Results from this paper are shown both theoretically and experimentally. Experiments are run on both synthetic random walk and real-world publicly available datasets. The randomized approach increased the number of search results by several orders of magnitude over LSH alone while keeping similar preciseness. Experimental databases contained tens of millions of indexed subsequences showing both correctness and scalability. However, the proposed algorithm is highly parallelizable, potentially allowing for databases of a much larger scale. The results present important information for a case-based reasoning system.

This paper is an extension of [19] in which further investigation of case-based reasoning environments is presented, as well as scope and evaluation of the method is extended to better represent the performance improvements; results generated in this work also consider the wall-clock time taken in order to perform the tasks outlined. The rest of the paper is organized as follows: Section 2 presents the motivation and related work in signal searching;

Section 3 provides the proposed method as well as its theoretical proof; and Section 3.4 describes the experimental set-up with the results and relating discussion in Section 4. Conclusions are given in Section 5.

2. Background

2.1. Biomedical Signals

Many different strategies to searching efficiently through biomedical time series exist, particularly with ECG and EEG signals. Authors in [20] attempt to approach the problem from a multi-dimensional angle. However, their search uses a dynamic time warping approach to create exact matches for plantar pressure. The proposed method accounts for an extension beyond dynamic time warping and is not truly adaptable to a large database for case-based reasoning due to its time-complexity and information in its results. Many works in ECG and EEG, such as those by authors in [21] and [22] respectively focus on analyzing time-series in biomedical signals but are centered on finding the appropriate features for classification accuracy. While representing time-series of biomedical signals is important in developing an efficient and effective classification algorithm, such systems often have problems with the wide variety of variable signals necessary in a case-based reasoning system that wants to find similar but not exact matches. Work in this paper looks at identifying these similar signals in a large database efficiently.

2.2. Case-Based Reasoning

Many complexities in case-based diagnoses systems need be addressed as data sets grow. Indeed, as patients across various locations submit data through systems, the potential for heterogeneous data sets becomes a problem. Work in [8] highlights an important missing data solution in order to present a comprehensive data set for analysis. Such a system will present even larger data for use and searches of this size of data are those which this manuscript is concerned. Once data sets are completed, authors in [23] use medical signals for case-based reasoning, similar to the intended goal of this work. The system developed uses a k-NN clustering algorithm with Euclidian distance (R -NN based) for matching. Such systems, however, will result in a complex running time, will prune inefficient results, and/or will not find too small a results set. Heuristics can be employed to improve the searches, such as by authors in [24] where a heuristic was developed to improve search for thyroid cancer treatment purposes. As will be shown in this work, the Monte Carlo method presented will result in greater information gain for such medical systems as in [23] using the idea of improving searches such as in [24], presenting improved results necessary for big data environments without increasing memory or computational complexities, shown both theoretically and empirically. This work will consider the exact R -NN matching scenario of previous works and show not only information gain improvement but real-time performance gains.

2.3. Subsequence Matching

Subsequence matching, or signal searching, is synonymous with the R -NN problem in high dimensional space. Nearest neighbors are defined as all objects that fall within distance R to the query object. For the purpose of this paper, distances are defined by the ℓ_2 norm

(Euclidean distance) and objects are represented as points in $\mathbb{R}^d$. Sequential search is a naïve approach to solving the R -NN problem. This, of course, has an $O(n)$ computational complexity that is far too slow for large databases. Hence, an optimal solution would guarantee sub-linear complexity.

The authors in [13] presented a framework for subsequence matching using spatial access methods (such as R^* -trees) to index dimensionally reduced objects. Reduction algorithms for this framework must satisfy the property of minimum bounds. The minimum bounds property guarantees that there are no false dismissals. Therefore, the set of all search results is a superset of all true matches. More formally:

$$\text{dist}(\hat{u}, \hat{v}) \leq \text{dist}(u, v), \quad (1)$$

where $\hat{u}$ and $\hat{v}$ are the dimensionally reduced counterparts of vectors $u, v \in \mathbb{R}^d$. Reduction algorithms satisfying the property of minimum bounds include Piecewise Aggregate Approximation (PAA) [14], Discrete Fourier Transform (DFT) [13], and Chebyshev polynomials [15].

The framework proposed in [13] has two main drawbacks. First, the query results of dimensionally reduced segments are a superset of true matches. Therefore, false positives must be filtered from the result set. Filtering is accomplished by finding the true distances between the original non-reduced query segment and its matches. Filtering can be extremely computationally expensive as no theoretical upper bounds are placed on the number of false positives. Second, the success of spatial indexes are controversial with authors showing both theoretically and experimentally that spatial indexes perform worse than sequential search for data with as little as ten dimensions [25].

Locality Sensitive Hashing (LSH) [16] was introduced as an alternative to spatial indexing schemes. Unlike spatial indexing schemes, LSH has a proven sub-linear computational complexity. LSH is based on a family of hashing functions H that are (r_1, r_2, p_1, p_2) -sensitive meaning that for any $v, q \in \mathcal{S}$:

- if $v \in B(q, r_1)$ then $Pr_H[h(q) = h(v)] \geq p_1$
- if $v \notin B(q, r_2)$ then $Pr_H[h(q) = h(v)] \leq p_2$,

where v and q are segments within search space $\mathcal{S}$, p_1 and p_2 probabilities, $B(q, r)$ represents the set of segments within distance r to q , where $p_1 > p_2$, and $r_2 > r_1$. An LSH scheme guarantees, with probability p_1 , that all segments within distance R to the query segment are returned. In addition, all segments that fall at a distance greater than cR (where C is a predefined constant) are not returned with probability p_2 . The gap between p_1 and p_2 is increased by combining several functions from the same (r_1, r_2, p_1, p_2) -sensitive family. For the purpose of this paper, $r_1 = R$ and $r_2 = cR$ where c is a constant.

LSH results are pruned such that all segments greater than distance R are suppressed. [16] show that the total query runtime is sub-linear and dominated by $O(n^c)$ distance

computations (pruning) where $\rho = \frac{\ln \frac{1/p_1}{1/p_2}}{\ln \frac{1/p_1}{1/p_2}}$. The experimental implementation in this paper uses the LSH scheme based on p -stable distributions defined in [26]. The authors propose the following hash function:

$$h_{a,b}(v) = \left\lfloor \frac{a \cdot v + b}{r} \right\rfloor, \quad (2)$$

where a is a randomized vector following a Gaussian distribution, b is a uniformly randomized vector, and r is a predefined constant. Using the properties of the p -stable distribution, the authors show that the probability of collision is calculated as:

$$p(c) = \int_0^r \frac{1}{c} f_p\left(\frac{t}{c}\right) \left(1 - \frac{t}{r}\right) dx, \quad (3)$$

with c being the distance between two vectors. As can be seen by Equation 3, the probability of collision decreases monotonically as c increases.

In general, an R -NN query suffers from homogeneity as results are extremely similar to the query segment. Raising the value of R is one naïve approach to increasing heterogeneity. However, this approach would add instability to the nearest neighbor problem, rendering the solution meaningless [9]. Therefore, the expansion of search results must be bounded to ensure stability. The solution proposed by this paper is founded on the property that, in terms of classification, classes are composed of multiple sub-groupings that are Gaussian distributed [11][12].

To note, several works have made improvements to the original LSH algorithm such as *locality sensitive B-tree* (LSB-tree) [17], Multi-Probe LSH [18], and entropy-based nearest neighbor search [27]. This paper utilizes the LSH implementation in [16] due to its maturity. However, the proposed approach could be easily adapted to any other method that provides both bounds on the runtime complexity and quality of results.

2.4. Distance Measures

Many authors have suggested the use of elastic measures such as Dynamic Time Warping (DTW), claiming this to be superior to Euclidean distance [28] [29] [30]. However, the authors in [31] show that DTW's ability to compare segments of different lengths have little to no statistical significance. In addition, others in [32] note that DTW's advantages are generally only in small datasets: as datasets grow, there is a higher probability of finding a match and less of a need to warp. Results were shown heuristically and demonstrated how error rates of both DTW and Euclidean distance converge as the size of datasets grow.

Finding an optimal distance function becomes more difficult with an increasing number of dimensions as distance norms have the property to converge with an increasing number of dimensions [9]. It has been shown that this convergence is true for any p -norm and is more

related to the intrinsic dimension than the embedding dimension [33]. However, [11] suggests that L_p norms are effective when considering small neighborhoods.

The usefulness of L_p norms was also shown experimentally in [12]. The Euclidean distance between objects in many datasets also has been shown to follow a Gaussian distribution. In addition, the Euclidean distance between objects shows moderate separation between classes. The authors proposed a new distance measure based on shared nearest neighbors. Under this measure of distance, objects with a large number of shared nearest neighbors (in terms of Euclidean distance) are considered more similar to objects that shared few or no nearest neighbors. While better separation was shown with a nearest neighbor comparison, calculating the sets of nearest neighbors has a computational complexity of $O(n^2)$; far too high for large databases.

3. Method

The following solution, based upon previous work in [19] is founded on two assumptions:

1. Time series databases are extremely large; and
2. Result sets follow a normal distribution.

The first assumption implies that only a subset of true matches is required by a query, and therefore, an accurate sampling is sufficient. The second assumption results from the finding that, in terms of classification, classes are composed of multiple sub-groupings that are Gaussian distributed [11][12]. This assumption allows for a Monte Carlo estimation of the true result set. Distances from a search segment s to its result set S are therefore modeled as a normal distribution $N(\mu_S, \sigma_S)$. m randomizations of a query segment s are used as queries into a time series database using Locality Sensitive Hashing. The randomizations of s are created such that the query results $\hat{S}$ are a Monte Carlo approximation of S and therefore form a normal distribution with μ_S and σ_S .

Each time series segment is assumed to be normally distributed ($N(0, 1)$). The distance between two time series segments s and $\hat{s}$ is defined as follows:

$$\frac{dist(s, \hat{s})}{2} = \sqrt{\sum_{i=0}^n \left(\frac{s(i) - \hat{s}(i)}{2} \right)^2}, \quad (4)$$

where 2 is a normalization factor due to the fact that the distribution of the differences between two $N(0, 1)$ variables is $N(0, 2)$. This results in $dist(s, \hat{s})$ being distributed as a χ distribution with the following property:

$$\lim_{n \rightarrow \infty} \frac{dist(s, \hat{s})/2 - \mu_d}{\sigma_d} \sim N(0, 1), \quad (5)$$

where μ_d and σ_d are the mean and standard deviation, respectively, of the match population of s . This paper assumes a large n of at least several hundred allowing the previous property

to hold approximately true. We therefore can produce a sampling of matches of s by randomizing each time point in s to create $\hat{s}$ such that:

$$s(i) - \hat{s}(i) \sim N(0, \sigma_r^2), \quad (6)$$

where σ_r is the standard deviation of the randomization of s . We define Y to be an independent random variable drawn from $\text{dist}(s, \hat{s})$ and X to be an independent random variable drawn from $s(i) - \hat{s}(i)$. Y is therefore distributed as:

$$Y = \sqrt{\sum_{i=0}^n X_i^2} = \sigma_r \chi_n \quad (7)$$

This yields a mean and standard deviation of:

$$\mu = \sigma_r \mu_{\chi_n} = \sigma_r \sqrt{2} \frac{\Gamma\left(\frac{n+1}{2}\right)}{\Gamma\left(\frac{n}{2}\right)} \quad (8)$$

$$\sigma = \sigma_r \sigma_{\chi_n} = \sigma_r \sqrt{\left(n - \mu_{\chi_n}^2\right)}, \quad (9)$$

where μ_{χ_n} is the mean of the χ distribution, similarly with σ_{χ_n} . ($\Gamma(x)$ being the Gamma function defined as:

$$\Gamma(x) = \int_0^\infty t^x e^{-t} dt \quad (10)$$

A lookup table for μ and σ can be found in Table 1 as these computations can suffer from rounding error; for small n , to avoid this rounding error, and to ensure samples fall in the described distribution, the values are calculated and provided for improved performance and lookup time in smaller datasets. For large n , distances between s and elements in $\hat{S}$ are given by the following distribution:

$$Y \sim N(\sigma_r \mu_{\chi_n}, \sigma_r \sigma_{\chi_n}) \quad (11)$$

However, the likelihood that a randomized segment exists within a database is extremely low. This is especially true as both the granularity of points and length of time series (n) increase. Therefore, a Monte Carlo approximation is unlikely to yield reasonable results unless the number of samples is infinitely large. Therefore, all neighbors within a distance R to a randomized signal are used in the set of returned results. The probability of finding a

segment s with distance in the range of $[x - R, x + R]$ to the search segment can be defined as:

$$f(z) = \frac{1}{2R\sqrt{2\pi\sigma^2}} \int_{x-R}^{x+R} e^{-\frac{(x-\mu)^2}{2\sigma^2}} dx \quad (12)$$

There is no closed form solution to Equation 12, however, as R approaches 0, $f(z)$ approaches a Gaussian distribution. This paper assumes an R that is relatively small. Hence, the error introduced by LSH adds a negligible amount to the bounds of the proposed randomization approach. This is heuristically shown in Fig. 1.

3.1. Complexity

As stated earlier, the algorithmic complexity of LSH is sub-linear and dominated by $O(N^p)$ distance computations (pruning) where $\rho = \frac{\ln}{\ln} \frac{1/p_1}{1/p_2}$ and N being the size of the database. Assuming a segment size of n , the randomization process is $O(mn)$ where m is the number of randomizations. LSH is run m times yielding a complexity dominated by $O(mN^p)$ (as $O(mN^p) > O(mn)$). This limit approximation is still sublinear as $m \ll N$.

3.2. Trivial Matches

This paper uses the formal definition of trivial matches proposed in [34]. Two segments $\hat{i} = [i, i + m]$ and $\hat{j} = [j, j + m]$ are trivial matches if $dist(\hat{i}, \hat{j}) \leq thr$ and there is no segment $\hat{i}' = [i', i' + m]$ where $i' = j$ and $dist(\hat{i}, \hat{j}') > thr$ [34]. Trivial matches add redundancy to the result sets as they represent the same patterns with small translations. This paper probabilistically filters trivial matches by removing any segments that lie immediately next to, or one time point away, from a previously extracted segment. More formally, given a segment u_i returned by an LSH search, the segments u_{i-1} , u_{i-2} , u_{i+1} , u_{i+2} will be filtered as trivial matches (where i is the starting point of segment u). In addition, all trivial matches to previously declared trivial matches will also be removed. The observation is that trivial matches occur in small runs of neighboring segments. The probability of LSH missing one neighboring trivial match (p_2) is reasonably high in practice, but the probability of missing two neighboring segments (p_2^2) is quite low.

3.3. Performance Optimizations

LSH returns all segments with colliding hashes. The result set may contain several false positives. These false positives are filtered by taking the Euclidean distance between the query segment and all matches returned by LSH. Any segment with distance greater than R to the query segment is filtered. During experimentation, it was found that a large proportion of time was spent pruning false positives. Each dataset consists of tens of millions of indexed segments, and therefore, even a small percentage of false positives in the result set is a significant number of segments. This is especially true when aggregated over several random query segments.

To improve performance, results were pruned using the original query segment and not the randomized query segment. Note, the original algorithm searches and prunes results for each

randomized segment independently. In this optimization, results were kept with probability based on their distances to the original query segment using the distribution $Y \sim \mathcal{N}(\sigma_r \mu_{\chi n}, \sigma_r \sigma_{\chi n})$.

This pruning technique also improved the results for LSH (deemed LSH with sampling). Therefore, this paper compares the Monte Carlo approximation with both LSH and LSH with sampling. To note, LSH result sets could be improved by raising the value of R and running LSH with sampling to avoid the overhead added by the Monte Carlo scheme. However, the search runtime using LSH is heavily dominated by the number of distance comparisons during pruning. The Monte Carlo approximation raises the theoretical bound of pruning by a factor of m . On the other hand, raising R exponentially increases the theoretical bound. With the assumption of a uniformly distributed search space, raising R increases the search space by R^d (where d is the number of dimensions).

In addition, two randomizations of the search segment may produce identical hash values. In cases of duplicate hash values, the duplicate hash is searched once and is weighted by the number of occurrences of the respective hash value. Therefore, query results of duplicated hash values have a higher probability of being returned with no additional computational overhead.

3.4. Experimental Setup

This paper leveraged four datasets in its assessment of the proposed method.

These datasets include:

1. MIT-BIH Arrhythmia Database [35] (ECG). This dataset contains several 30-minute segments of two-channel ambulatory ECG recordings. These sample included arrhythmias of varying significance.
2. Gait Dynamics in Neuro-Degenerative Disease Database [36, 37] (GAIT). This dataset contains data gathered from force sensors placed under the foot. Healthy subjects as well as those with Parkinson's disease, Huntington's disease, and amyotrophic lateral sclerosis (ALS) were asked to walk while the data was recorded. Data includes 5-minute segments for each subject.
3. Synthetic Signal (SYN) - This dataset was created using the function $f(x) = \sum_{i=3}^7 \frac{1}{2^i} * \sin(\pi x 2^{3+i}) + rand(1)$. This is a derivative of the pseudo periodic synthetic time series data set presented in [38]. The randomization inserted by $rand(1)$ is to model noise. This dataset helps to assess both LSH and the Monte Carlo approximation's stability with noisy time series signals.
4. Synthetic Shapes (SHAPE) - This dataset was created to assess precision and recall and is composed of ten different shapes with varying levels of Gaussian noise displayed in Fig. 2

Each dataset was indexed and inserted into the the database. A total of 50 segments were randomly chosen as query segments from each dataset. These segments were searched using LSH, LSH with sampling (proposed in Section 3.3), and the full Monte Carlo method

proposed by this paper. Both the standard deviation of the randomization (σ_r) and the number of randomizations (m) were varied for the Monte Carlo experiments. The attributes of each dataset are given in Table 2. The radius R is relative to the normalized distances (i.e., segments are normalized before finding the Euclidean distance).

The percentage improvements of result sizes of the Monte Carlo approximation over LSH and LSH with sampling are shown. The percentage increase is used as a comparison in lieu of raw counts due to the high variability in the size of result sets. The number of results is highly dependent on the random segment extracted from the time series signal. For example, a common heart beat from the ECG signal will generally match to several thousand segments. An anomaly, on the other hand, may only match to a couple of segments.

Precision and recall was assessed only for the SHAPE dataset with varying amount of Gaussian noise. The ECG, GAIT, and SYN datasets are not annotated to label specific neighborhoods. In addition, these datasets have tens of millions of rows and manually creating these labels is not feasible. However, the respective means and standard deviations of the Euclidean distance between the result sets and the query segment are given to show correctness. In addition, example result sets are displayed visually for each dataset for further evidence of correctness.

The runtime of the Monte Carlo approximation was also evaluated. This experiment consisted of 50 random runs using both LSH and the Monte Carlo approximation. Wall clock time (in seconds) is presented. All experiments were run on an Intel(R) Core(TM) i5 CPU 650 processor clocked at 3.2 GHz [39] running Ubuntu 11.10 [40].

The Monte Carlo approximation and LSH were implemented in Python version 2.7 [41] and utilized a MySQL 5.1.58 database [42] for storing, indexing, and retrieving hash values. Raw time series signal were segmented using a sliding window and all unique segments were indexed. An indexed segment in the MySQL database contained a link to the raw time series signal along with its location within the time series. A segment was indexed using its hash values generated by LSH. Segments were retrieved from the file system in batches by sorting segments by their respective host file and location within this file. This is an extremely important optimization as file systems perform poorly under random access.

The implementation of LSH in this paper is based on disk storage unlike that of [26]. [26] assumed a database small enough to be stored in memory; however, the databases used by this paper are extremely large (tens of millions of instances). Hence, the system had to utilize disk based storage.

4. Results

Example Monte Carlo query results are shown for the ECG, GAIT, and SYN datasets in Fig. 3. Five segments are randomly extracted and displayed. The greatest variation between segments can be seen by the ECG and SYN datasets. This is unlike LSH where each result set contained almost no variability, as shown in Fig. 4. In fact, LSH consistently retrieves a dataset size of 1 (exact match) for the SYN dataset. The poor performance of LSH for SYN is due to the randomization added during the construction of SYN. This phenomenon is

important as the randomization of the dataset SYN is similar to the effects of noise. The poor results for SYN demonstrates the degradation of LSH's performance with signal noise.

The percentage increase in the number of results over standard LSH when varying the randomization standard deviation (σ_r) and the number of randomizations (m) is shown in Fig. 5. The results for SYN with varying σ_r are shown separately from the ECG and GAIT datasets as the increase in result size is several orders of magnitude larger. The increase over standard LSH ranged from a couple hundred percent to several thousands. This is expected as standard LSH only returns segments that are very similar to the query segment. The larger increase by the GAIT dataset when varying m is caused by the homogeneity of the dataset (i.e., each step in the dataset is largely similar). The randomizations do a good job of expanding the search space. This is extremely important as this same phenomenon would also be seen in other datasets as the number of indexed segments increases.

Increasing σ_r exhibits larger increases for the SYN dataset over standard LSH. A larger σ_r models a more significant amount of noise. The GAIT and ECG datasets are reasonably clean, and therefore, do not benefit from a larger σ_r . In fact, both the results sets for GAIT and ECG decreased with $\sigma_r > 0.15$. Larger σ_r cause the search space to extend beyond the neighborhood of the query segment resulting in fewer matches. The SYN dataset, on the other hand, has a large amount of noise. Hence, a larger σ_r helps improve the size of result sets (as the neighborhood is larger).

The percentage increase over LSH with sampling of result set sizes when varying σ_r and m is shown in Fig. 6. Most of this improvement is seen with less than 50 randomizations.

As stated earlier, the size of a result set is highly dependent upon the respective query segment. A common pattern will yield a large number of results, while an anomaly will yield small sets. Therefore, both Fig. 5 and Fig. 6 displayed percentage increases. Fig. 7 displays raw result set sizes with $m = 25$ and $\sigma_r = 0.1$ for a reference.

The goodness of the result sets for the Monte Carlo approximation were assessed by finding the mean and standard deviation of the Euclidean distance of the query segments to their respective result segments. Fig. 8 and Fig. 9 display the effects of varying both m and σ_r respectively. When increasing m , the mean and standard deviation converge to the respective theoretical values with the exception of standard deviation for SYN. The SYN dataset's higher standard deviation is due to the addition of uniformly distributed noise that results in neighborhoods that are not normally distributed. A slight divergence in standard deviation is observed when increasing σ_r . This result is expected as increasing σ_r will increase the search outside of a segments immediate neighborhood.

Precision and recall results for the SHAPE dataset are shown in Table 3. All tests resulted in 100% precision. This result is expected as the neighborhoods are known with a radius $R < 1.5$ with high discrimination from other classes. Hence, setting an accurate neighborhood will assure precise results. Recall for LSH with sampling is approximately 91% while recall for Monte Carlo search is near 100% with as little as ten randomizations ($m = 10$).

The run times for both standard LSH and the Monte Carlo approximation are given in Fig. 10. LSH with sampling was not included in this figure as run times for both versions of LSH are identical (i.e., the number of distance comparisons is identical). Theoretically, the algorithm proposed by this paper is m distinct runs of LSH with a theoretical increase in computational and memory complexity of a factor of m . However, the optimizations proposed in section 3.3 allow for a considerable improvement in performance. Runtimes were increased by an approximate factor of three for the GAIT and ECG dataset and less than a factor of two for the SYN dataset. Thus, results show that, as data sets grow, this wall-clock improvement will become of utmost importance in the application of such systems to actual diagnosis engines and use in clinical settings and environments.

Fig. 11 shows the growth of unique hashes with an increase number of randomizations (m). As shown by the figure, the number of unique hashes grows sub-linearly with respect to m . This phenomena is beneficial for two reasons. First, a small number of randomizations is needed to get an accurate representation of the result space. In addition, the average computational and memory complexity is increased by a factor of much less than m than standard LSH.

5. Conclusions

This paper presents a Monte Carlo approximation technique for subsequence matching. The number of results for the Monte Carlo approximation are significantly increased over standard Locality Sensitive Hashing (LSH). This technique adds minimal computational complexity and ensures the preciseness of results. The proposed technique takes in a subsequence as input. The subsequence is randomized m times using a Normal distribution with standard deviation equal to σ_r . The resulting m randomized sub-sequences are provided as input to LSH resulting in m distinct R -NN searches. It was theoretically and experimentally shown that the Euclidean distances of a result set to the respective query segment is bounded by a Normal distribution of $Y \sim \mathcal{N}(\sigma_r \mu_{\chi m}, \sigma_r \sigma_{\chi m})$.

The computational complexity of the Monte Carlo approximation is increased by only a linear factor over LSH (dependent upon the number of randomizations). In practice, however, there was a much smaller decrease in performance. The proposed method ran in approximately twice the time (wall clock) of LSH with $m = 250$. This method can also be run in parallel to further improve run times. Further, while this work considers an LSH implementation, other underlying methods could be considered, depending on runtime and memory constraints of a given system. As a result, the ability to use such a medical signal search engine becomes clear in its improvements over existing case-based reasoning systems. First, and foremost, the information gain is improved without a significant penalty in computational or memory complexities. Further, the actual implementation is presented that improves not only the information gain but the wall clock running time performance of such an algorithm, with the potential of improving the parallel nature of the algorithm that can further improve the speed of such system, which is of great need in increasingly big data environments for medical systems. A system aimed at big data for medical analytics is presented and results validated.

Acknowledgments

This publication was partially supported by Grant Number T15 LM07356 from the NIH/National Library of Medicine Medical Informatics Training Program.

References

1. Baumeister J, Atzmueller M, Puppe F. Inductive learning for case-based diagnosis with multiple faults. *Advances in case-based reasoning*. 2002:173–216.
2. Bichindaritz I, Kansu E, Sullivan K. Case-based reasoning in care-partner: Gathering evidence for evidence-based medical practice. *Advances in case-based reasoning*. 1998:334–345.
3. Marling C, Montani S, Bichindaritz I, Funk P. Synergistic case-based reasoning in medical domains. *Expert systems with applications*. 2014; 41(2):249–259.
4. Nilsson M, Sollenborn M. Advancements and trends in medical case-based reasoning: An overview of systems and system development.
5. Begum S, Barua S, Filla R, Ahmed MU. Classification of physiological signals for wheel loader operators using multi-scale entropy analysis and case-based reasoning. *Expert systems with applications*. 2014; 41(2):295–305.
6. Martinez Orellana, R.; Erem, B.; Brooks, D. Time invariant multi electrode averaging for biomedical signals. *Acoustics, Speech and Signal Processing (ICASSP); 2013 IEEE International Conference on*; 2013; IEEE; p. 1242-1246.
7. Xu, M.; Shen, J. *The 19th International Conference on Industrial Engineering and Engineering Management*. Springer; 2013. Clinical decision support model of heart disease diagnosis based on bayesian networks and case-based reasoning; p. 219-225.
8. Guessoum S, Laskri MT, Lieber J. Respidiag: A case-based reasoning system for the diagnosis of chronic obstructive pulmonary disease. *Expert Systems with Applications*. 2014; 41(2):267–273.
9. Beyer K, Goldstein J, Ramakrishnan R, Shaft U. When is “nearest neighbor” meaningful? *Database Theory-ICDT*. 1999; 99:217–235.
10. Woodbridge, J.; Lan, M.; Sarrafzadeh, M.; Bui, A. Salient segmentation of medical time series signals. *Healthcare Informatics, Imaging and Systems Biology (HISB); 2011 First IEEE International Conference on*; 2011; IEEE; p. 1-8.
11. Bennett, K.; Fayyad, U.; Geiger, D. Density-based indexing for approximate nearest-neighbor queries. *Proceedings of the fth ACM SIGKDD international conference on Knowledge discovery and data mining*; 1999; ACM; p. 233-243.
12. Houle, M.; Kriegel, H.; Kröger, P.; Schubert, E.; Zimek, A. Scientific and Statistical Database Management. Springer; 2010. Can shared-neighbor distances defeat the curse of dimensionality?; p. 482-500.
13. Faloutsos, C.; Ranganathan, M.; Manolopoulos, Y. *SIGMOD '94*. ACM; New York, NY, USA: 1994. Fast subsequence matching in time-series databases, in: *Proceedings of the 1994 ACM SIGMOD international conference on Management of data*; p. 419-429. doi:<http://doi.acm.org/10.1145/191839.191925>. URL <http://doi.acm.org/10.1145/191839.191925>
14. Keogh E, Chakrabarti K, Pazzani M, Mehrotra S. Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and information Systems*. 2001; 3(3):263–286.
15. Cai, Y.; Ng, R. *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*. ACM; 2004. Indexing spatio-temporal trajectories with chebyshev polynomials; p. 599-610.
16. Gionis A, Indyk P, Motwani R. Similarity search in high dimensions via hashing. *Proceedings of the International Conference on Very Large Data Bases*. 1999:518–529.
17. Tao, Y.; Yi, K.; Sheng, C.; Kalnis, P. *Proceedings of the 35th SIGMOD international conference on Management of data*. ACM; 2009. Quality and efficiency in high dimensional nearest neighbor search; p. 563-576.
18. Lv Q, Josephson W, Wang Z, Charikar M, Li K. Multi-probe lsh: efficient indexing for high-dimensional similarity search. *Proceedings of the 33rd international conference on Very large data bases, VLDB Endowment*. 2007:950–961.

19. Woodbridge J, Mortazavi B, Sarrafzadeh M, Bui A. A monte carlo approach to biomedical time series search. *Bioinformatics and Biomedicine (BIBM)*, 2012 IEEE International Conference on. 2012:1–6. doi:10.1109/BIBM.2012.6392646.
20. Moazeni, M.; Mortazavi, B.; Sarrafzadeh, M. Body Sensor Networks (BSN), 2013 IEEE International Conference on. IEEE; 2013. Multi-dimensional signal search with applications in remote medical monitoring; p. 1-6.
21. Wacker M, Witte H. Time-frequency techniques in biomedical signal analysis. a tutorial review of similarities and differences. *Methods of information in medicine*. 52(4)
22. Lawhern, V.; Hairston, WD.; Robbins, K. Foundations of Augmented Cognition. Springer; 2013. Optimal feature selection for artifact classification in eeg time series; p. 326-334.
23. Chattopadhyay S, Banerjee S, Rabhi FA, Acharya UR. A case-based reasoning system for complex medical diagnosis. *Expert Systems*. 2013; 30(1):12–20.
24. Teodorovi D, Šelmi M, Mijatovi L. Combining case-based reasoning with bee colony optimization for dose planning in well differentiated thyroid cancer treatment. *Expert Systems with Applications*. 2013; 40(6):2147–2155.
25. Weber, R.; Schek, H.; Blott, S. Proceedings of the International Conference on Very Large Data Bases. IEEE; 1998. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces; p. 194-205.
26. Datar, M.; Immorlica, N.; Indyk, P.; Mirrokni, V. Proceedings of the twentieth annual symposium on Computational geometry. ACM; 2004. Locality-sensitive hashing scheme based on p-stable distributions; p. 253-262.
27. Panigrahy, R. Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm. ACM; 2006. Entropy based nearest neighbor search in high dimensions; p. 1186-1195.
28. Shou Y, Mamoulis N, Cheung D. Efficient warping of segmented time-series. Tech. rep., HKU CSIS Tech rep, TR-2004-01.
29. Yi, B.; Jagadish, H.; Faloutsos, C. Data Engineering, 1998. Proceedings., 14th International Conference on. IEEE; 1998. Efficient retrieval of similar time sequences under time warping; p. 201-208.
30. Kim S, Park S, Chu W. Efficient processing of similarity search under time warping in sequence databases: an index-based approach. *Information Systems*. 2004; 29(5):405–420.
31. Ratanamahatana C, Keogh E. Everything you know about dynamic time warping is wrong. Third Workshop on Mining Temporal and Sequential Data. 2004:22–25.
32. Shieh J, Keogh E. isax: disk-aware mining and indexing of massive time series datasets. *Data Mining and Knowledge Discovery*. 2009; 19(1):24–57.
33. Francois D, Wertz V, Verleysen M. The concentration of fractional distances, *Knowledge and Data Engineering*. IEEE Transactions on. 2007; 19(7):873–886.
34. Lin J, Keogh E, Patel P, Lonardi S. Finding motifs in time series. *Proc. of the 2nd Workshop on Temporal Data Mining*. 2002:53–68.
35. Mark, R.; Moody, G. Mit-bih arrhythmia database directory. Massachusetts Institute of Technology; Cambridge:
36. Hausdorff J, Lertratanakul A, Cudkowicz M, Peterson A, Kaliton D, Goldberger A. Dynamic markers of altered gait rhythm in amyotrophic lateral sclerosis. *Journal of applied physiology*. 2000; 88(6):2045–2053. [PubMed: 10846017]
37. Hausdorff J, Mitchell S, Firtion R, Peng C, Cudkowicz M, Wei J, Goldberger A. Altered fractal dynamics of gait: reduced stride-interval correlations with aging and huntington's disease. *Journal of Applied Physiology*. 1997; 82(1):262. [PubMed: 9029225]
38. Park, S.; Lee, D.; Chu, W. Knowledge and Data Engineering Exchange, 1999.(KDEX'99) Proceedings. 1999 Workshop on. IEEE; 1999. Fast retrieval of similar subsequences in long sequence databases; p. 60-67.
39. Intel. 2nd generation intel® core™ processor family desktop datasheet. <http://www.intel.com/content/www/us/en/processors/core/2nd-gen-core-desktop-vol-2-datasheet.html> (September 2011)
40. Petersen, R. Ubuntu 11. 10 Desktop: Applications and Administration. Surfing Turtle Press; 2011.

41. Python J. Python programming language. Python (programming language) 1 CPython 13 Python Software Foundation. 2009; 15:1.
42. MySQL, A. Mysql 5.1 reference manual. 2006. Accessible in URL: <http://dev.mysql.com/doc>

Author Manuscript

Author Manuscript

Author Manuscript

Author Manuscript

Figure 1.
Shows the convergence of Equation 12 as $R \rightarrow 0$.

Figure 2.
Displays the ten shapes (classes) in the synthetic shape dataset with Gaussian noise.

Figure 3.

Displays three example query results for each dataset using the Monte Carlo approximation. Five segments were randomly displayed from each result set. The greatest variation between segments can be seen in the second column of ECG and the three columns of SYN.

Figure 4.

Displays example query results for the ECG, GAIT, and SYN datasets using LSH. Five segments were randomly displayed from each result set (when applicable). Almost no variation between segments is observed. The SYN dataset has a result size of one for all queries as shown by the figure.

Figure 5.

Displays the percentage increase in the results sets over standard LSH when varying both m and σ_r . Both a fixed $\sigma_r = 0.1$ with $m = 5, 10, 25, 100, 250$ and a fixed $m = 25$ with $\sigma_r = 0.05, 0.1, 0.15, 0.2$ are displayed. The results for SYN with varying σ_r is shown separately from the ECG and GAIT datasets as the increase in result size is several orders of magnitude larger.

Figure 6.

Displays the percentage increase in the results sets over LSH with sampling when varying both m and σ_r . Both a fixed $\sigma_r = 0.1$ with $m = 5, 10, 25, 100, 250$ and a fixed $m = 25$ with $\sigma_r = 0.05, 0.1, 0.15, 0.2$ are displayed.

Figure 7.
Displays the number of results for 50 randomly selected queries with $m = 25$ and $\sigma_r = 0.1$.

Figure 8.

Displays the mean and standard deviation of the Euclidean distance from query segments to the respective result sets. Queries were run with $\sigma_r = 0.1$ and $m = 5, 10, 25, 100, 250$. The respective theoretical means and standard deviations are also displayed.

Figure 9.

Displays the mean and standard deviation of the Euclidean distance from query segments to the respective result sets. Queries were run with $\sigma_r = 0.05, 0.1, 0.15, 0.2$ and $m = 25$. The respective theoretical means and standard deviations are also displayed.

Figure 10.

Displays the average wall clock time of 50 searches for LSH and the Monte Carlo approximation with $m = 10, 25, 100$. The overhead of the Monte Carlo approximation is minimal (both theoretically and in practice).

Figure 11.

Displays the growth of unique hashes with respect to the number of randomizations. The number of unique hashes grows sub-linearly with respect to the number of randomizations yielding an average runtime much better than the theoretical bound.

Table 1

Lookup table for μ and σ assuming a $\sigma_d = 1$.

n	μ	σ^2
128	11.2916	.4990
256	15.9844	.4995
512	22.6164	.4998
1024	31.9922	.4999
2048	45.2493	.4999

Author Manuscript

Author Manuscript

Author Manuscript

Author Manuscript

Table 2

Dataset Attributes

Dataset	Number of segments	Segment length	<i>R</i>
ECG	70M	512	3
GAIT	15M	512	3
SYN	10M	128	3
SHAPE	1M	600	1.5

Author Manuscript

Author Manuscript

Author Manuscript

Author Manuscript

Table 3Precision Recall for LSH with sampling and Monte Carlo Search ($m = 10$)

Method	Precision	Recall
LSH	1.0	.909089
LSH with sampling	1.0	.912844
Monte Carlo	1.0	.999978