Cluster-Based Similarity Search in Time Series

Leonidas Karamitopoulos, Georgios Evangelidis
Dept. of Applied Informatics, University of Macedonia
Thessaloniki, Greece
e-mail: lkaramit@uom.gr, e-mail: gevan@uom.gr

***Abstract*—In this paper, we present a new method that accelerates similarity search implemented via one-nearest neighbor on time series data. The main idea is to identify the most similar time series to a given query without necessarily searching over the whole database. Our method is based on partitioning the search space by applying the K-means algorithm on the data. Then, similarity search is performed hierarchically starting from the cluster that lies most closely to the query. This procedure aims at reaching the most similar series without searching all clusters. In this work, we propose to reduce the intrinsically high dimensionality of time series prior to clustering by applying a well known dimensionality reduction technique, namely, the Piecewise Aggregate Approximation, for its simplicity and efficiency. Experiments are conducted on twelve real-world and synthetic datasets covering a wide range of applications.**

Keywords-similarity search; clustering; time series; data mining

I. INTRODUCTION

Similarity search is an important task in many applications, such as content-based retrieval, exploratory data analysis, predictive modeling and data mining. The basic problem can be stated as follows: given a set of objects, find the most similar ones to a given query object. For example, one may be interested in retrieving the most similar images to a given one from a database or in identifying those stocks whose prices evolved similarly to a specific one over the last year. Retrieval of these objects is based on "similarity" rather than on "exactness".

The main research in this area is focused on the development of methods that can efficiently support similarity search, since common applications involve very large amounts of data.

One major class of methods for efficient similarity search comprises of multidimensional indexing schemes that can be used for fast access of these points [9]. Although the indexing approach can be extremely fast, its efficiency degrades rapidly with the increase of the dimensionality. There are several research results that demonstrate the negative effects of increasing dimensionality on index structures [10]. For instance, Beyer et al. [4] show that nearest neighbor search can become unstable with as few as 10-20 dimensions. White & Jain [21] report that as the dimensionality increases from 5 to 10, the performance of a nearest neighbor search degrades by a factor of 12 for various multidimensional index structures. This phenomenon, known as dimensionality curse, implies that a simple sequential scan usually performs better at higher dimensionalities than index structures.

One solution for achieving efficient similarity search in the presence of high dimensionality is to condense the data by applying a dimensionality reduction technique (i.e. Principal Component Analysis). The idea is to map the original data into a lower dimension domain without losing substantial amount of information. The approach of dimensionality reduction can be very helpful because it reduces the storage requirements, it potentially allows an efficient implementation of multidimensional indexing structures and it improves the quality of similarity search results.

In this paper, we examine the case of similarity search implemented via one-nearest neighbor (1-NN) on time series data. This type of data differs from other domains in that they have an intrinsically high dimensionality, which necessitates the application of a dimensionality reduction technique. The method of 1-NN requires searching a database for the most similar object (time series) to a given one. The main drawback of this method is that we have to compare a query object to every object in a database in order to find the most similar one. This approach becomes prohibitive, when the reference database is extremely large. The efficiency of this method is affected by the number of objects in the database, as well as, by the dimensionality of these objects, since a distance measure is calculated for measuring the closeness of the corresponding objects.

The objective of our work is to provide a method for the purpose of improving the efficiency of 1-NN similarity search in time series datasets without sacrificing the quality of the results. We introduce an approach that integrates the dimensionality reduction of data and subsequently the reduction of the search space. We call this approach CLUREP (Clustering on Representation).

In particular, the original data is represented in a compressed form by applying any dimensionality reduction technique. There is a wealth of relevant techniques proposed in the literature, such as Discrete Fourier Transform [2], Discrete Wavelet Transform [7], Singular Value Decomposition [14] or Piecewise Aggregate Approximation [13]. Then, the transformed dataset is partitioned by applying a clustering algorithm. The similarity search proceeds at each cluster sequentially

according to specific criteria. Our approach aims at reaching the nearest neighbor without searching the whole database.

Although, a multidimensional index structure can be built on each one of the clusters in order to efficiently apply similarity search, we investigate the effectiveness of our approach with respect to sequential searching. The reason is that the required dimensionality reduction in order to ensure the desired quality performance may require a multidimensional indexing that is not essentially faster than sequential scanning.

In Section II, we provide related work, whereas our approach is analytically presented in Section III. The experimentation framework is described in Section IV and the corresponding results are presented in Section V. Finally, conclusions are provided in Section VI.

II. Related Work

Various methods have been proposed for speeding up nearest neighbor searches that are based on indexing [11]. However, the dimensionality curse affects seriously the performance of high-dimensional similarity search. One solution for achieving scalable performance is to reduce the dimensionality of data. Aggarwal [1] provides a model of the effects of this reduction on high dimensional problems for the purpose of improving the quality of similarity search.

Most of the research is focused on reducing the dimensionality of data in order to apply an effective multidimensional indexing structure and/or reducing the number of objects to be compared against the query. Patella and Ciaccia [17] provide a classification schema for approaches to approximate similarity search.

Chakrabarti and Mehrotra [6] propose to discover correlated clusters in the dataset and reduce the corresponding dimensionalities by applying Principal Component Analysis to each one of them. They also provide a technique to individually index these clusters that guarantees no false positive or false negative answers to be returned to the user. This approach applies local dimensionality reduction assuming that there are subsets of locally correlated data.

Bennett et al. [3] propose to apply an EM (Expectation Maximization) algorithm to cluster data using a mixture-of-Gaussians pdf (probability density function) model. Each cluster corresponds to a specific Gaussian pdf, which is parameterized with a mean vector and a covariance matrix. This approach (DBIN) utilizes the derived density model of the data in order to introduce an indexing scheme that produces a mapping between a query point and an ordering on the clustered index values.

Ferhatosmanoglou et al. [8] introduce a new technique based on clustering that reduces the size of the dataset and the dimensionality of each data object. The authors propose to transform the original d-dimensional data by using the Karhunen-Loève Transformation (KLT). After dimensionality reduction, a modified K-means clustering algorithm is applied on the low dimensional domain. The procedure of approximate nearest neighbor searching starts with the transformation of the query object, the retention of the first r dimensions, and the identification of the cluster this query falls into. Then, similarity search is performed in this cluster and the k nearest neighbors are retrieved.

Another cluster-based approach is presented in [5]. The authors propose an indexing method called Clustering with Singular Value Decomposition (CSVD) that efficiently supports approximate nearest neighbor queries. The construction of a CSVD index involves three steps: partitioning the dataset using a clustering technique (i.e., K-means), applying SVD separately on each cluster to reduce the dimensionality of data, and constructing an index for the transformed space. Thomasian et al. [19] propose an exact algorithm to process k-NN queries on dimensionality reduced clusters produced by CSVD, whereas a general framework for building a persistent main memory index on each cluster is provided in [20].

Li et al. [15] provide a clustering and indexing paradigm (called CLINDEX) for high dimensional spaces. CLINDEX partitions the dataset into clusters. Each cluster is represented by a separate file and all files are sequentially stored on disk. The authors introduce a new technique for constructing clusters that is based on grids. Once the clusters are obtained, an indexing structure is built on them. A different clustering and indexing paradigm for exact nearest neighbor search in image databases is provided in [18].

III. The Proposed Approach

The proposed method consists of three phases that are analytically described hereafter.

The first phase involves the application of a dimensionality reduction technique on the original data for two reasons. First, we expect to improve the quality of the similarity search results. Second, the subsequent clustering analysis becomes more effective, since the problem of high dimensional data is alleviated. Virtually, any technique can be selected at this step; however, this choice is application dependent, since different techniques may result into representations of higher quality in different applications. In this work we propose to apply the Piecewise Aggregate Approximation (PAA) because it is simple, fast to calculate, and it has been shown empirically that it is as efficient as other more sophisticated approaches. In particular, PAA segments a time series into a number of sections and records the corresponding means. The series of these means is the representation of the time series.

The second phase involves the application of a clustering algorithm on the transformed data, namely, K-means [16], which is one of the most researched and popular clustering methods. The output of this phase consists of the centroids (c_i) of the generated clusters along with their radii (r_i). The radius of a cluster is defined as the distance of the farthest object of a cluster to the corresponding centroid. In addition to these, we record the cluster membership of each object and its distance from the corresponding centroid. The objects are reordered in the dataset with respect to their cluster membership and the distance from their cluster's centroid.

Note that the two previous phases constitute the pre-processing that is executed off-line.

The third phase involves the procedure of similarity search in the derived clusters. Given a query object (q), the required steps to be followed are provided below:

1. Calculate the distances of the PAA transformed query object to the centroids of the clusters ($d\left(q,c_i\right)$).

2. Set the cluster with the closest centroid as the current cluster. Let denote this cluster $C^{(i)}$, where $i=1$.

3. Calculate the distance of the query object to each one of the other clusters. This distance is defined to be the difference between the distance of the query to the centroid and the corresponding radius. If the query object lies within the cluster, then this distance is set equal to zero (1).

$$d\left(q,C^{(i)}\right)=max\left\{0\ ,\ d\left(q,c_i\right)-r_i\right\},\quad i=2,3,...,k \quad (1)$$

4. The clusters are sorted in an increasing order with respect to their distances from the query object. Ties may be broken according to the distances of the centroids of the clusters to the query object. Let denote these clusters $C^{(i)}$, where $i=2,3,...,k$ with k corresponding to the cluster that is farthest from the query.

5. Search the current cluster sequentially in order to locate the current nearest neighbor (nn) to the query object. Record the corresponding distance $d\left(q,nn\right)$.

6. If the distance of the query object to the current nearest neighbor is less than or equal to the distance of the query object to the next cluster (2), then the actual nearest neighbor has been found and the algorithm stops.

$$d\left(q,nn\right)\le d\left(q,C^{(i+1)}\right) \quad (2)$$

Otherwise, let $i=i+1$ and move to the next step.

7. Calculate the difference between the distance of the query object to the centroid of $C^{(i)}$ and the distance of the query object to the current nearest neighbor. If this difference is positive, then search the objects of $C^{(i)}$ that their distance to c_i is greater than this difference. Otherwise, search the whole cluster. After this, the current nearest neighbor (nn) has been changed. If $i=k$, the algorithm stops. In this step, we determine a lower bound on the distances of objects from the centroid in order to reduce the search space in the current cluster (3).

$$lower\_bound=max\left\{0\ ,\ d\left(q,c_i\right)-d\left(q,nn\right)\right\} \quad (3)$$

8. Go to step 6.

An example of this procedure is presented in Fig. 1-3. We consider 17 time series of 2 dimensions (two time instances). In this example, the clusters are non-overlapping and the query object lies outside of the clusters; however, the proposed method works for any placement of clusters and queries.

The above algorithm assumes that the Euclidean distance is utilized. Nevertheless, this method holds for any distance that constitutes a metric, especially with respect to the property of the triangle inequality. The proof of the correctness of this method is omitted due to lack of space.

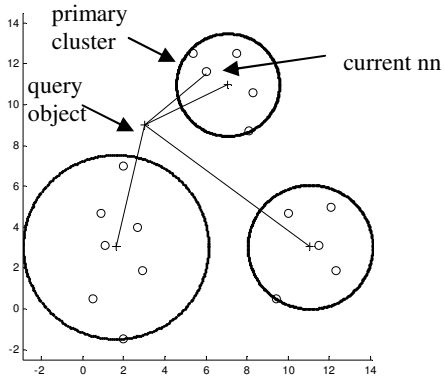

Figure 1. Determination of the primary cluster and location of the current nearest neighbor

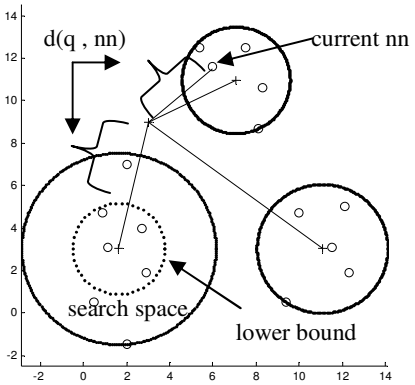

Figure 2. Determination of the next closest cluster, check of whether a better nearest neighbor may exist (in this case, it exists) and determination of the corresponding search space

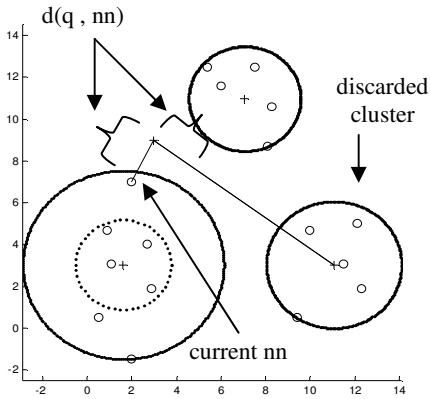

Figure 3. Location of a better nearest neighbor, determination of the next closest cluster and check of whether a better nearest neighbor may exist (in this case, it does not exist and the algorithm ends without searching the next cluster)

IV. Framework of Experimentation

In order to evaluate the performance of the proposed approach, we perform one-nearest neighbor classification (1-NN) and validate it with the leave-one-out procedure. We record the classification accuracy and the percentage of the volume of data that is searched. The corresponding average values are calculated over the total number of series in the dataset.

The experiments are conducted on 12 real world and synthetic datasets, which are available upon request in [12]. The number of classes ranges from 2 to 50, whereas the length of time series ranges from 60 to 427. They are utilized extensively as benchmark datasets for testing classification algorithms and for this reason, they are separated in training and testing sets. In this work, we merge these two sets in order to increase the size of each dataset. All datasets consist of normalized time series, that is, time series of mean equal to zero and standard deviation equal to one. All series are labeled according to the class they belong to. The names of these datasets along with their identification number (ID) are the following: (1) 50words, (2) CBF, (3) ECG200, (4) FaceAll, (5) Gun Point, (6) OSULeaf, (7) SwedishLeaf, (8) SyntheticControl, (9) Trace, (10) TwoPatterns, (11) Wafer, (12) Yoga.

Regarding PAA implementation, the number of sections is set equal to multiples of 2, ranging from 4 to 16. We compare our method (CLUREP) with the complete sequential search and the method that does not restrict the search space of the visited clusters. We call this method CLUREP_all. The algorithm of K-means is applied with a number of clusters that ranges from 2 to 20 in increments of 2. Intuitively, the more the number of clusters, the less the fraction of dataset that need to be searched. Finally, all the necessary codes and experiments are developed in MATLAB.

V. Results

In the first part of the results, we provide the classification error rates and the percentage of the volume of data that is searched for a constant number of clusters ($k = 10$). The objective is to examine their relation under varying dimensionalities.

Table I presents the classification error rates for varying dimensionalities when the number of the generated clusters is set equal to 10. The main observation is that the lowest error rate is achieved at high dimensionalities. In 8 out of 12 datasets, the required dimensionality is the highest (16). The lowest dimensionality is 8 and it is observed in only one dataset. The minimum error rate ranges from 0.00% to 35.75%.

Table II presents the percentages of the volume of data that is searched when the number of the generated clusters is set equal to 10. In 8 out of 12 datasets, there is an increasing trend in the volume of data searched as the dimensionality increases. On the other hand, there are 4 datasets (ID: 5, 9, 11, 12) where the corresponding volume of data remains fairly stable across the dimensionalities. The minimum volume of data ranges from 12.5% to 19.2%.

TABLE I. 1-NN CLASSIFICATION ERROR RATES (%) ($k = 10$)

	Dimensionality						
ID	4	6	8	10	12	14	16
1	65.3	43.9	34.9	31.1	29.6	29.3	27.9
2	10.5	0.4	0.0	0.1	0.0	0.0	0.0
3	25.0	16.5	12.5	12.5	10.5	9.0	10.0
4	59.8	31.7	24.0	13.4	10.0	9.6	7.4
5	17.5	10.5	13.5	7.5	8.0	6.0	5.5
6	72.0	53.9	44.1	40.3	36.7	36.7	35.8
7	63.4	41.4	27.5	23.7	18.8	18.8	17.3
8	14.7	9.0	5.0	1.2	1.2	1.2	0.7
9	20.0	20.0	22.0	22.5	15.0	20.5	18.5
10	45.4	16.8	8.0	3.5	2.8	2.3	0.8
11	0.4	0.4	0.2	0.2	0.2	0.1	0.2
12	24.4	11.3	9.1	8.3	6.7	7.0	6.6

TABLE II. PERCENTAGES OF THE VOLUME OF DATA THAT IS SEARCHED ($k = 10$)

	Dimensionality						
ID	4	6	8	10	12	14	16
1	16.4	21.0	25.6	30.1	33.8	36.7	40.0
2	14.2	15.6	16.3	17.3	18.8	18.3	20.4
3	17.7	20.9	21.9	22.9	24.3	25.8	25.2
4	17.3	26.2	31.8	28.8	28.9	29.5	31.0
5	16.7	16.0	17.5	16.7	17.6	17.1	17.4
6	19.2	28.8	33.6	43.9	48.5	53.2	58.3
7	19.1	23.7	29.6	32.0	35.3	35.7	39.0
8	16.8	19.6	21.3	21.5	25.0	24.6	28.4
9	14.1	13.4	13.5	12.5	13.7	13.8	13.0
10	19.2	19.1	25.1	32.5	37.4	40.9	46.2
11	14.5	15.2	14.9	15.2	15.5	15.3	15.5
12	15.3	15.1	16.4	16.6	16.9	16.8	17.1

The key observation that arises by combining both tables (Table I and Table II) is that as the dimensionality increases, the classification error rate decreases with the cost of an increasing percentage of the volume of data that need to be searched. We also observe that when the dimensionality is set equal to 16, the corresponding volume of data ranges from 13% to 58.3%. In addition to that, the three datasets with the "worst" error rates also present poor results in the volume of data that needs to be searched (ID: 1, 6, 7). Similar observations can be made, if we generate fewer or more clusters than 10. For brevity, results for other values of k are not presented.

In the second part of the results, we provide the percentage of the volume of data that is searched for varying number of clusters. The dimensionality is set equal to that number for which the lowest error rate is achieved (Table I). In general, this number varies across different datasets.

In this set of experiments, we provide results for two methods, namely the CLUREP and the CLUREP_all. Note that the latter differs from the first in that it searches the whole cluster once visited. By definition, CLUREP is expected to provide better results than CLUREP_all.

However, one of our objectives is to experimentally quantify the expected improvement.

In Table III and Table IV, the performance of CLUREP and CLUREP_all is presented with respect to the percentage of volume of data that is searched. The first observation is that when the number of the generated clusters (k) increases, the corresponding volume decreases. In general, the rate of decrease is higher when the value of k increases from 2 to 10 than when k increases from 10 to 20. The second observation is that both methods perform better than sequential scan. However, CLUREP performs consistently better than CLUREP_all across datasets for any number of clusters.

More specifically, when the number of clusters is equal to 20, CLUREP requires searching a fraction of the original datasets that ranges from 7.2% to 45.3% whereas the corresponding fraction of CLUREP_all ranges from 8.0% to 69.9%. On the average across datasets, CLUREP requires searching the 19.6% of the original data volume, whereas the corresponding percentage for CLUREP_all is 39.5%.

TABLE III. PERCENTAGES OF THE VOLUME OF DATA THAT IS SEARCHED (CLUREP)

	Number of Clusters									
ID	2	4	6	8	10	12	14	16	18	20
1	87	67	54	46	40	36	33	30	29	27
2	58	35	25	19	16	14	12	10	10	9
3	69	43	33	29	26	24	21	20	19	17
4	67	48	39	34	31	28	26	25	24	22
5	61	36	27	21	17	16	14	12	11	11
6	86	75	68	63	59	55	52	50	48	45
7	78	59	50	44	39	36	33	31	29	28
8	59	42	36	32	29	26	24	23	22	20
9	51	31	21	17	14	12	10	9	8	7
10	86	68	58	51	46	42	39	36	33	31
11	56	34	24	19	15	13	11	10	9	8
12	61	34	25	20	17	15	13	12	11	10

TABLE IV. PERCENTAGES OF THE VOLUME OF DATA THAT IS SEARCHED (CLUREP_all)

	Number of Clusters									
ID	2	4	6	8	10	12	14	16	18	20
1	99	94	87	79	73	68	63	60	56	54
2	80	55	44	34	27	23	19	17	15	13
3	97	75	61	50	45	39	35	31	29	27
4	99	94	89	84	81	78	74	71	68	65
5	83	60	43	30	26	22	18	17	15	14
6	97	94	91	87	84	81	78	75	72	70
7	100	98	95	90	85	82	78	75	72	69
8	67	60	54	48	44	40	38	35	33	32
9	50	35	24	19	16	13	11	10	9	8
10	100	97	93	90	86	82	77	74	70	66
11	58	56	49	43	39	36	33	30	28	26
12	85	60	54	48	44	40	37	34	32	29

VI. CONCLUSIONS

Nearest neighbor retrieval is a common operation in a wide variety of real applications such as in query by image and video content, in genomic analysis or in object recognition. The increasing amount of stored data necessitates the development of techniques and algorithms that are capable of performing efficiently and accurately nearest neighbor searches.

In this paper, we propose a cluster-based method (CLUREP) for similarity search for the purpose of 1NN classification. Part of this method can be utilized in conjunction with a multidimensional indexing structure. However, we examine the effectiveness of this approach in sequential searching because the dimensionality reduction technique that is applied on data may not be adequate for providing a multidimensional indexing the means for outperforming sequential scanning.

The main conclusion is that, when compared to sequential searching, CLUREP is at least 5 times faster in the majority of the datasets considered in the experiments. In addition to that, CLUREP substantially improves the performance of a similar approach (CLUREP_all) by a factor of 2. A secondary conclusion is that when the number of generated clusters increases, the performance of CLUREP improves. However, experiments show that the rate of improvement "slows down" after the generation of approximately 8 to 12 clusters.

A more general conclusion is that, as the dimensionality of the transformed data increases, the classification error rate decreases with the cost of an increasing percentage of the volume of data that need to be searched. In the data mining context, one can trade-off between accuracy and efficiency with respect to the requirements of the application under consideration.

REFERENCES

[1] C. C. Aggarwal, "On the Effects of Dimensionality Reduction on High Dimensional Similarity Search," Proc. ACM Symp. Principles of Database Systems (PODS 01), ACM Press, May 2001, pp. 256-266, doi.acm.org/10.1145/375551.383213.

[2] R. Agrawal, C. Faloutsos and A. Swami, "Efficient Similarity Search in Sequence Databases," Proc. 4th Int. Conf. Foundations of Data Organization and Algorithms (FODO 93), Springer, Oct. 1993, pp. 69-84.

[3] K. P. Bennett, U. Fayyad and D. Geiger, "Density-based Indexing for Approximate Nearest-neighbor Queries," Proc. 5th Int'l Conf. Knowledge Discovery and Data Mining (SIGKDD 99), ACM, Aug. 1999, pp. 233-243, doi.acm.org/10.1145/312129.312236.

[4] K. Beyer, J. Goldstein, R. Ramakrishnan and U. Shaft, "When is Nearest Neighbor Meaningful?," 7th Int'l Conf. Database Theory (ICDT 99), Springer Verlag, Jan. 1999, pp. 217-235.

[5] V. Castelli, A. Thomasian and C. S. Li, "CSVD: Clustering and Singular Value Decomposition for Approximate Similarity Search in High-dimensional Spaces," IEEE Transactions Knowledge Data Engineering, vol. 15(3), 2003, pp. 671–685, doi:10.1109/TKDE.2003.1198398.

[6] K. Chakrabarti and S. Mehrotra, “Local Dimensionality Reduction: A New Approach to Indexing High Dimensional Spaces,” Proc. 26th Int´l Conf. Very Large Databases (VLDB 00), Morgan Kaufmann, Sep. 2000, pp. 89-100.
[7] K. Chan and A. W. Fu, “Efficient Time Series Matching by Wavelets,” Proc. 15th Int´l. Conf. Data Engineering (ICDE 99), IEEE Computer Society, Mar. 1999, pp. 126-133.
[8] H. Ferhatosmanoglu, E. Tuncel, D. Agrawal and A. El Abbadi, “Approximate Nearest Neighbor Searching in Multimedia Databases,” Proc. 17th IEEE Int'l Conf. Data Engineering (ICDE 01), IEEE Computer Society, Apr. 2001, pp. 503-511.
[9] V. Gaede and O. Gunther, “Mutidimensional Access Methods,” ACM Computing Surveys, vol. 30(2), Jun. 1998, pp. 170–231.
[10] A. Hinneburg, C. C. Aggarwal and D. A. Keim, “What is the Nearest Neighbor in High Dimensional Spaces?,” Proc. 26th Int´l Conf. Very Large Databases (VLDB 00), Morgan Kaufmann, Sep. 2000, pp. 506–515.
[11] G. R. Hjaltason and H. Samet, “Index-driven Similarity Search in Metric Spaces,” ACM Transactions on Database Systems, vol. 28(4), Dec. 2003, pp. 517–580, doi.acm.org/10.1145/958942.958948.
[12] http://www.cs.ucr.edu/~eamonn/time_series_data/
[13] E. Keogh, K. Chakrabarti, M. Pazzani and S. Mehrotra, “Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases,” Knowledge and Information Systems, vol. 3(3), Aug. 2001, pp. 263-286, doi:10.1007/PL00011669.
[14] F. Korn, H. Jagadish and C. Faloutsos, “Efficiently Supporting Ad Hoc Queries in Large Datasets of Time Sequences,” Proc. Int´l Conf. Management of Data (SIGMOD 97), ACM Press, May 1997, pp. 289-300, doi.acm.org/10.1145/253260.253332.
[15] C. Li, H. Garcia-Molina and G. Wiederhold, “Clustering for Approximate Similarity Search in High-dimensional Spaces,” IEEE Trans. Knowl. Data Eng., vol. 14(4), Jul./Aug. 2002, pp. 792–808, doi:10.1109/TKDE.2002.1019214.
[16] J. MacQueen, “Some Methods for Classification and Analysis of Multivariate Observations,” Proc. Fifth Berkeley Symposium on Mathematical Statistics and Probability, Univ. of Calif. Press, vol. 1, 1967, pp. 281-297.
[17] M. Patella and P. Ciaccia, “Approximate Similarity Search: A Multi-faceted Problem,” Journal of Discrete Algorithms, vol. **7**(1), Mar. 2009, pp. 36-48 doi:10.1016/j.jda.2008.09.014.
[18] S. Ramaswamy and K. Rose, “Adaptive Cluster-distance Bounding for Nearest Neighbour Search in Image Databases,” Proc. ICIP, **6**, 2007, pp. 381-384,
[19] A. Thomasian, Y. Li and L. Zhang, “Exact k-NN Queries on Clustered SVD Datasets,” Information Processing Letters, vol. 94(6), Jun. 2005, pp. 247-252, doi:10.1016/j.ipl.2005.03.003.
[20] A. Thomasian, and L. Zhang, “Persistent Clustered Main Memory Index for Accelerating k-NN Queries on High Dimensional Datasets,” Multimedia Tools Applications, vol. 38(2), June 2008, pp. 253-270, doi:10.1007/s11042-007-0179-7.
[21] D. A. White and R. Jain, “Similarity indexing with the SS-Tree,” Proc. 12th Int´l Conf. Data Engineering (ICDE 96), Mar. 1996, IEEE, pp. 516-523, doi:10.1109/ICDE.1996.492202.