Clustering of Bi-Dimensional and Heterogeneous Times Series: Application to Social Sciences Data

Rémi Gaudin, Sylvaine Barbier, N. Nicoloyannis, Maks Banens

Clustering of Bi-Dimensional and Heterogeneous Times Series: Application to Social Sciences Data

Rémi Gaudin[1], Sylvaine Barnier[1], Nicolas Nicoloyannis[1], Maks Banens[2]
[1]*Laboratoire ERIC, 5 Av. Pierre mendés France, Univ. Lumière - Lyon 2, 69676 Bron. France*
[2] *Centre d'Etudes Démographiques, Institut des Sciences de l'Homme,*
16 avenue Berthelot, 69007 Lyon. France
rgaudin@eric.univ-lyon2.fr

Abstract

We present an application of bi-dimensional and heterogeneous time series clustering in order to resolve a Social Sciences study issue. The dataset is the result of a survey involving more than eight thousand handicapped persons. Sociologists need to know if there are in this dataset some homogeneous classes of people according to two attributes: the idea that handicapped people have about the quality of their life and their couple status (i.e. if they have a partner or not). These two attributes are time series so we had to adapt the k-Means clustering algorithm in order to be efficient with this kind of data. For this purpose, we use the Longest Common Subsequence time series distance for its efficiency to manage time stretching and we extend it to the bi-dimensional and heterogeneous case. The results of our unsupervised process give some pertinent and surprising clusters that can be easily analyzed by sociologists.

1. Introduction

In data mining research, time series represent an actual challenge due to the unique structure of this kind of data. Most classic data mining algorithms, which were initially conceived for classic (i.e. non temporal) data, do not work well for time series. The need to adapt data mining methods to time series has created a new field of research called temporal data mining [3, 32].

Temporal data mining includes association rules [12, 13], indexing (query by content) [23, 47], feature mining [26, 28], the discovery of recurrent or surprising motifs [9, 11, 20, 30], classification [10, 18, 21, 42] and clustering [17, 29, 31, 34, 45].

Time series clustering is a difficult field where numerous papers propose algorithms that work well with artificial data but they are not efficient in real-world dataset problems [19]. Time series clustering using Hidden Markov Models is proposed in [29, 34]. Some approaches perform clustering using *k*-Means algorithm with Euclidean distance measure [31, 45]. A time series clustering algorithm that uses *k*-Means with Dynamic Time Warping distance measure is proposed in [17]. Although it is efficient with several artificial datasets, it does not work with real heterogeneous dataset like the one we have deal with.

Multivariate time series clustering is a more difficult issue where few methods and distances are already proposed. Traditional distances like Weighted Sum SVD [40], Principal Component Analysis similarity factor (S_{PCA}) [25, 41] or Eros [46] are only for numerical data with the same size. Moreover they are often too complex for large dataset and they are basically used for indexing process in databases (e.g. with k-Nearest Neighbors method). In another work, Lee et al propose a method to index sequences of multidimensional points [27]. They extend the ideas presented by Faloutsos et al. in [15] and they use the Euclidean distance.

Some works on indexing moving objects (i.e. bi-dimensional time series) are proposed in order to answer spatial proximity query [1, 24, 35, 36, 39]. Also in [43, 44], an efficient indexing of trajectories is performed by Vlachos et al. using *Longest Common Subsequence* (LCSS) distance. For reasons that we explain in the next section, this method is relatively suitable for our issue. But Vlachos et al. are dealing with numerical data, so we have to make this distance work with heterogeneous data as well. Moreover this technique, and also all the others described above, has not been used for unsupervised clustering with algorithm such *k*-Means yet.

Our work is based on a study carried out by the French National Institute for Statistics and Economic Studies (INSEE). This study is trying to show how living in couples affects the view that handicapped people have on their lives. The dataset owns the result of a survey involving 8403 handicapped persons. For each year all along their life, they had to give a numeric estimation of the quality of their life, and at the same time they also noted down if they had a partner or not (Fig.1). Sociologists need to know if it exists in this dataset some homogeneous classes of people according to its "couple status / life-quality estimation" behaviors. So we have a set of 8403 bi-dimensional and heterogeneous time series that we try to classify in an unsupervised way. The aim is that sociologists can work with our partitioning in order to bring out relevant categories of handicapped people. We can sum up the difficulties of our issue as follows:

- Bi-dimensional and heterogeneous data: Each handicapped person is represented by two time series; one numeric (the life quality estimation) and one symbolic (to have a partner or not).
- Temporal gap: Different persons may have the same bi-dimensional pattern that occurs at different moments in time axis. The process has to match two same patterns despite the potential time axis gap.
- Time series with very different size: Lengths of time series may vary between 2 and 80. The distance measure that we use must be able to manage these differences.
- Unsupervised clustering: All the process must be automatic. This is more a data exploratory analysis than a machine learning problem.

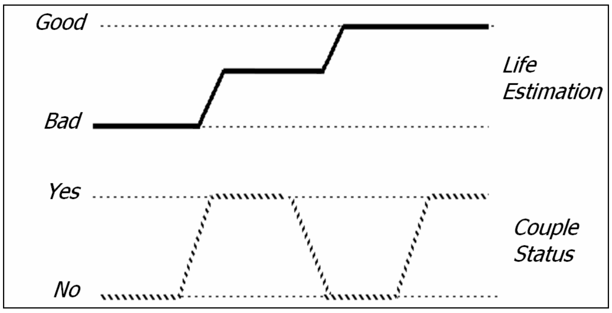


Figure 1. Example of a bi-dimensional time series representing a handicapped person.

So firstly we have to choose an efficient distance measure, and then we have to develop an algorithm that uses the distance measure and performs the clustering in a satisfying way. In section 2, we present the LCSS distance and explain why it is efficient and how we adapt it for our bi-dimensional and heterogeneous issue. In section 3, we describe how we adapt the k-Means clustering algorithm with LCSS and get a complete clustering process. In section 4 we experiment our process on the dataset and present the results. We conclude in section 5.

2. Longest Common Subsequence Distance

Euclidean distance is the most widely used distance measure, even for calculating distances between data such as time series, because it is easy to compute and very fast. The operation consists in matching a given point from a time series with the point from another one that occurs at the same time. The main drawback of Euclidean distance is its inability to manage time axis gap. Two time series with the same shapes that do not occur concurrently on time axis may have a high Euclidean distance. This result is very illogical and it can significantly perturb the clustering [22].

For the particular case of our dataset, we need a distance measure that is able to match some shapes that not occur at the same time. Indeed, some handicapped persons may have similar "couple status / life-quality estimation" patterns at different periods of their life. Among all the measures able to perform time series distance in this way, the *Dynamic Time Warping* (DTW) distance is the most popular. The particularity of DTW is that it compare two time series together by allowing a given point from one time series to be matched with *one or several* points from the other [6, 38]. We choose not to use this distance for two reasons: Firstly, DTW manages only numeric time series. Its adaptation to symbolic data is not obvious and it includes some additional parameters that are difficult to fix. Secondly, DTW forces all elements of each time series to be matched, even if these elements do not have any relevant meaning. Typically for our dataset we have a lot of non relevant periods (e.g. when neither the life-quality appreciation nor the couple status changes for a handicapped person).

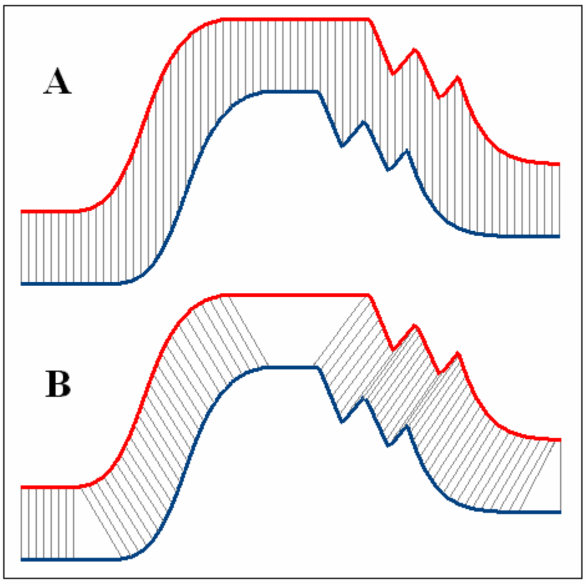


Figure 2. Computation of the distance between two time series with Euclidean distance (fig. A) and LCSS distance (fig. B).

The *Longest Common Subsequence* (LCSS) distance, like DTW, is a time stretching distance. It matches two time series together by allowing them to stretch, without rearranging the sequence of the elements [2, 7, 8, 12]. Whereas in Euclidean and DTW distance *all* elements must be matched, LCSS can keep some elements unmatched by allowing one point of a time series to be matched with *one or zero* point of the other (Fig.2).

In order to force time stretching not to match too distant elements, we may add to LCSS a *warping window* (i.e. a constant δ) that controls how far in time we can go in order to match two points from two different time series. For example, if we set $\delta = 3$, a point that occurs at instant t must only be matched with points from the other time series that occurs at instants t-3, t-2, t-1, t, t+1, t+2 and t+3. In fact, δ is not inevitably a constant and may vary according to time [37], but for simplicity we consider δ to be a constant in this paper. Moreover, we have to set a *spatial window* (i.e. a constant ε) as a matching threshold that defines if two point from two different time series can be matched or not. LCSS distance gives in result a value between 0 (the two time series are perfectly similar) and 1 (no common points between the two time series).

Originally, LCSS is a one-dimensional distance for numeric data. So we have extended its definition to be able to manage time series with two heterogeneous dimensions (one numeric and one symbolic) as follow:

Let A and B be two bi-dimensional time series with size m and n respectively, where $A = \{(a_{x1}, a_{y1}), \dots, (a_{xm}, a_{ym})\}$ and $B = \{(b_{x1}, b_{y1}), \dots, (b_{xn}, b_{yn})\}$. a_{xi} and b_{xi} are the i^{th} value of the numeric time series of A and B respectively. a_{yi} and b_{yi} are the i^{th} value of the symbolic time series of A and B respectively.

Let $Equal(i, j)$ be the matching function between the i^{th} point of A and the j^{th} point of B, where:

$$Equal(i,j) = \begin{cases} \text{True} & if \ |a_{xi} - b_{xj}| \le \varepsilon \ and \ a_{yi} = b_{yj} \\ \text{False} & else \end{cases} \quad (1)$$

Given the recursive function $Sim(i, j)$ that compute the similarity between the subsequence $A_i = \{(a_{x1}, a_{y1}), \dots, (a_{xi}, a_{yi})\}$ and the subsequence $B_j = \{(b_{x1}, b_{y1}), \dots, (b_{xj}, b_{yj})\}$ as follows:

$$Sim(i,j) = \begin{cases} 0 \quad if \ i = 0 \ or \ j = 0 \\ 1 + Sim(i-1, j-1) \\ \quad if \ |i - j| \le \delta \ and \ Equal(i,j) = \text{True} \\ \max\{Sim(i-1, j), Sim(i, j-1)\} \\ \quad otherwise \end{cases} \quad (2)$$

We can now define our *LCSS* distance as follow:

$$LCSS(A,B) = 1 - \frac{Sim(m,n)}{\min\{m,n\}} \quad (3)$$

It’s important to notice that *LCSS* is not a metric distance. Therefore it does not necessary respect the triangular inequality $LCSS(A\ ,B) \le LCSS\ (A,\ C) + LCSS(C,\ B)$.

The recursive definition of our *LCSS* distance has a non computational complexity and needs a dynamic programming approach [4, 5]. Dynamic programming consists in creating a $m \times n$ matrix. Inside the cell (i, j) of the matrix we store 1 if $Equal(i, j)$ = True, 0 otherwise. Once all the cells are filled we search for the best *warping path*. It is the path beginning in cell (1, 1), finishing in cell (m, n) that maximise the sum of the cells it goes through (Fig. 3).

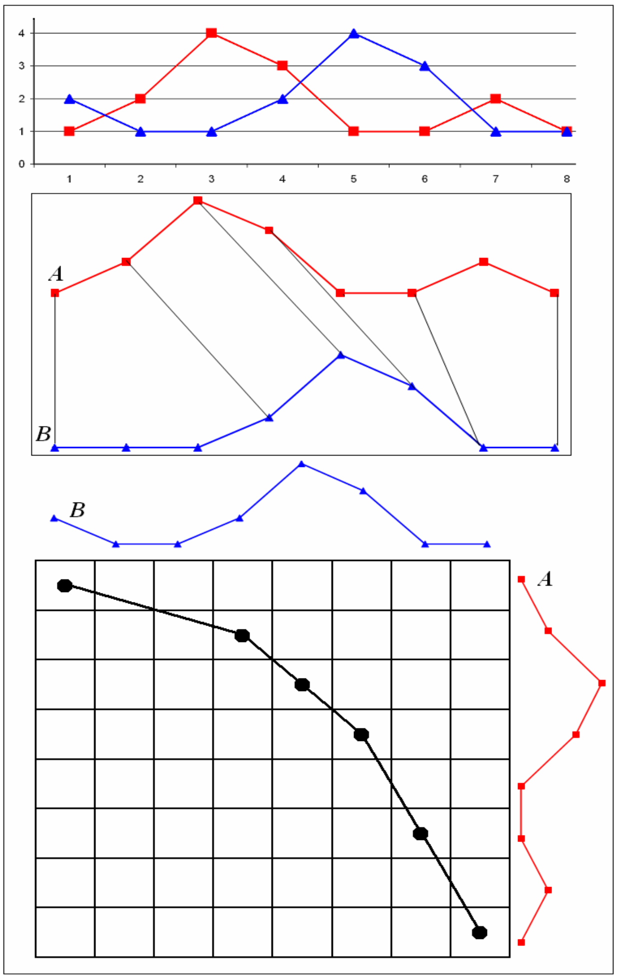


Figure 3. Example of best warping path between two time series A = {1, 2, 4, 3, 1, 1, 2, 1} and B = {1, 1, 1, 2, 4, 3, 1, 1}. The sequence of cells crossing by the best warping path is (1, 1), (2, 4), (3, 4), (4, 6), (6, 7), (8, 8).

Actually, the non recursive algorithm we use to compute the LCSS distance in an optimal and efficient way does not search directly for the best warping path. It simply obtains the LCSS distance value thanks to a cumulative similarity matrix π built as shown in Table 1 (here, for simplicity there is no warping window, but the addition it is trivial to add it).

```
for i = 1 to m, do:
    for j = 1 to n, do:
        if ( (i = 1 or j = 1) and Equal(i, j) = True )
            then π_ij = 1
        if ( (i = 1 or j = 1) and Equal(i, j) = False )
            then π_ij = 0
        if ( i > 1 and j > 1 and Equal(i, j) = True )
            then π_ij = π_{i-1,,j-1} + 1
        if ( i > 1 and j > 1 and Equal(i, j) = False )
            then π_ij = max{π_{i-1,,j} , π_{i,,j-1}}
```

$$LCSS(A,B) = 1 - \frac{\pi_{mn}}{\min\{m,n\}}$$

Table 1. The non recursive algorithm that computes the LCSS distance between two bi-dimensional and heterogeneous time series *A* and *B*.

So the longer step of this algorithm is the completion of the cumulative similarity matrix. This step has a complexity of $O(m \times n)$ ($O(m^2)$ if the two time series have the same length). If we add a warping window with size δ, this algorithm allows to compute LCSS in $O(m \times \delta)$ time (with $\delta << n$).

3. The clustering process

3.1. The k-Means algorithm

A classic way to perform clustering is the use of the k-Means algorithm [33]. This approach is very interesting for us because it generates "spherical" clusters (i.e. each cluster can be considered as a hypersphere inside the multidimensional data space. The center of the hypersphere is the fictive mean between all the objects owned by this cluster. The radius is the distance between the fictive mean and the furthest object in the cluster. Because of admitting relocation after each iteration, using k-means clustering allows poor initial partitions to be corrected at a later stage. So when the fictive mean moves, the sphere-shaped structure of the cluster is conserved and it keeps its homogeneity. This characteristic is empirically observed for non metric distances like LCSS. It permits the creation of homogeneous and proportional clusters that are, for our study, less sensitive to outliers than Hierarchical Clustering clusters. The intuition behind k-Means approach is shown in Table 2.

1	Decide on a value for *k*.
2	Initialize the *k* cluster centres (randomly, if necessary).
3	Decide the class memberships of the *N* objects by assigning them to the nearest cluster centre.
4	Re-estimate the *k* cluster centres, by assuming the memberships found above are correct.
5	If none of the *N* objects changed membership in the last iteration, exit. Otherwise return to step 3.

Table 2. K-Means algorithm

To resolve our catalysis clustering problem, the k-Means approach has one major drawback: At step 4, the algorithm has to re-estimate the *k* cluster centers. This means computing the average of all the time series for each cluster in the multidimensional data space. This is straightforward with non temporal data (we just have to compute the Euclidean average) but illogical for temporal data like time series. We will resolve this difficulty with by using a variant of this algorithm proposed by Didey [14].

3.2. Using k-Means with LCSS distance

K-Means algorithm is a variant of the Forgy algorithm [16]. The Forgy algorithm resumes the basic intuition behind all partitional clustering algorithms like k-Means. It creates clusters with only two parameters: the number of clusters noted *k* and the size of seeds noted *c*. For the reasons explained in the previous section, the distance measure used by our method is LCSS, so we have adapted this algorithm to make time series clustering work with this distance (Table 3).

Each cluster is characterized by a seed of *c* time series. Seeds are used to compute distances between time series and clusters as well as distances between each cluster. The *c* times series of a cluster are those that minimize the *Inertia* function. This function is also used to compute the intra-class variance between all clusters that evaluates the quality of the clustering.

The exact complexity of this algorithm ca not be determined because it depends on the relative size of each cluster during the iterations. However, for a dataset of *N* time series, it has a complexity inferior or equal to $O(kPL.(N^2 + N))$, where *k* is the number of clusters specified by the user, *P* is the number of iterations until convergence, and *L* is the duration of one LCSS calculation (i.e. $O(m \times \delta)$).

It is obvious that the final time series clustering depends on the seeds initialization. Ideally, each seed should be initialized only with the time series that belong to the same cluster, but if we do not have any *a priori* knowledge about the dataset (as it is usually the case in unsupervised knowledge data discovery), we have to initialize the seeds at random. For a dataset with k time series clusters (with the same number of time series for each cluster), the probability to have a perfect initialization at random is approximately equal to $\frac{k!}{k^k}$ (for example if $k = 6$ then the probability is equal to 0.015, i.e. 1 in 65 to have a perfect initialization). Here we use intra-class variance to evaluate the quality of the seeds initialization: if a seed initialization does not give an intra-class variance inferior to a threshold, then another initialization is tried.

1	Let X be a set of N time series that we aim to split in k clusters, where $X = \{x_1,\ldots,x_n\}$.
2	Let S be a set of k seeds, where $S = \{S_1, \ldots, S_k\}$. Each seed S_j is composed of c time series chosen among the initial set X (randomly if necessary). One time series ca not belong to more than one seed.
3	Given $L(c, S_j)$, the distance between the time series x_i and the seed S_j as follows: $$L(x_i, S_j) = \frac{1}{c}\sum_{y \in S_j} LCSS(x_i, y)$$
4	For $i = 1,\ldots, N$ do: For $j = 1,\ldots, k$ do: Compute $L(x_i, S_j)$
5	Assign to each time series x_i its nearest seed (i.e. the seed S_j that minimize $L(x_i, S_j)$).
6	Let C be a set of k clusters, where $C = \{C_1, \ldots, C_k\}$. Each cluster C_j is made of all time series that have S_j as nearest seed.
7	Redefine a new set of seeds $S' = \{S'_1, \ldots, S'_k\}$. Each new seed S'_j is made of the c time series x_i from C_j that minimize: $$Inertia(x_i, C_j) = \sum_{y \in C_j} LCSS(x_i, y)$$
8	To estimate the clustering quality, calculate the intra-class variance $Var(C)$ as follows: $$Var(C) = \frac{1}{n}\sum_{j=1}^{k}\sum_{x_i \in C_j} Inertia(x_i, C_j)$$ If the value of $Var(C)$ does not decrease between the iteration p and the iteration p+1 (or decrease less than an arbitrary threshold), then stop the process. Else restart a new iteration at step 4 with $S = S'$.

Table 3. Didey's k-Means generalization with LCSS distance

Because of the nature of our data, we do not use a temporal window because similar patterns that we search in two handicapped person can occur anywhere in time axis. So it leaves us only with three parameters: the number of clusters k, the size of seeds c and the spatial window ε for LCSS.

Only the numeric variable of our bi-dimensional time series needs the ε parameter. This parameter must be fixed by user according to the maximum difference that he considers that two persons have the same life estimation.

We notice that the value of c does not influence the intra-class variance defined in our algorithm (contrary to δ and k). So we can consider that the optimal value of c minimizes this variance.

We ca not use the same method to find the optimal value of k because this parameter influences the intra-class variance final value (the more k increases, the more the intra-class variance decreases). This limitation can be minimized by attempting all values of k within a large range.

4. Application to handicapped people dataset

We apply our algorithm on our handicapped people dataset. We split the data in two parts according to sex (i.e. one part with 4617 women and one part with 3786 men). Sociologists need to know if the "direct link" hypothesis (i.e. to have a partner implies an increase of life-quality estimation for handicapped people) is valid or not. Results of our process give, for each sex, surprising clusters that contradict this expectation. In spite of the very large multiplicity of patterns, we are able to bring out some homogeneous classes, in particular if we cluster the dataset with $k = 4$ (Table. 4).

The first surprising fact that we notice is the quasi-perfect symmetry between the two sexes. The difference between the proportional cluster sizes for the two sexes is always inferior or equal to 1 %. That logically means that living in couples affects in the same way the view that handicapped men and women have on their lives.

The other surprising fact is the frequency of handicapped people that have no variation of their life-quality estimation in spite of their couple-status change (72 % for men and 73 % for women). The second largest cluster is composed of people without a couple-status change. People that directly relate life quality with couple status (i.e. to find a partner implies a life quality increase) are in the third cluster. With a frequency of 6 % for the two sexes, this cluster is surprisingly small. In the same way, the cluster with people that have an opposite relation between the two variables is relatively insignificant (3 % for each sex).

As a conclusion, contrary to what sociologists expect, we can consider that living in couples (or not) is not a determinant variable to explain the increase (or the decrease) in the life quality estimation of handicapped people. This conclusion may be an important decision factor for future assistance programs towards handicapped people.

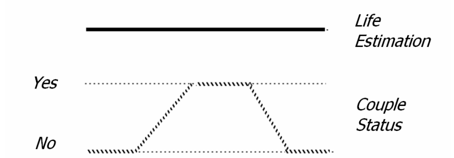
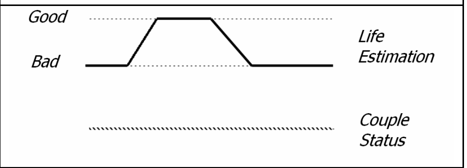
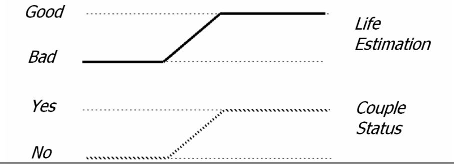
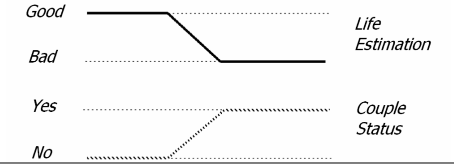
M	W	General tendency
72%	73%	
19%	18%	
6%	6%	
3%	3%	

Table 4. The clustering result of the handicapped people dataset (with $k = 4$). The first and the second columns give the proportional size of each cluster for handicapped men (M) and women (W) respectively. Because of the very large multiplicity of patterns, we only show here for each cluster the main pattern that is the most representative of the general tendency of the cluster.

5. Conclusion

In this paper we present an exploratory analysis on a survey with 8403 handicapped persons. This dataset needs a methodology that is able to manage bi-dimensional, heterogeneous, with different size and temporal gap data. The approach we propose is based on the k-Means algorithm and the Longest Common Subsequence distance. We adapt the LCSS distance to bi-dimensional and heterogeneous data, and adapt the k-Means algorithm to be able to support this distance measure. The results obtained by our process are pertinent and surprising. They can be easily analyzed by sociologists in order to assist them in their work.

References

[1] P. K. Agarwal, L. Arge and J. Erickson, "Indexing moving points," In Proc. of the 19th ACM Symposium on Principles of Database Systems (PODS), 175-186, 2000.

[2] R. Agrawal, K. Lin, H. S. Sawhney and K. Shim, "Fast Similarity Search in the Presence of Noise, Scaling and Translation in Time-Series Databases," In Proc. of VLDB, 490-501, 1995.

[3] C. M. Antunes and A. L. Oliveira, "Temporal data mining: an overview," In Proc. of the Workshop on Temporal Data Mining, at the 7th International Conference on Knowledge Discovery and Data Mining (KDD'01), San Francisco, CA, 1-15, 2001.

[4] R. Bellman, *Dynamic Programming*. New Jersey: Princeton University Press, 1957.

[5] D. J. Berndt and J. Clifford, "Finding Patterns in Time Series: A Dynamic Programming Approach," In Proc. of Advances in Knowledge Discovery and Data Mining, 229-248, 1996.

[6] D. J. Berndt and J. Clifford, "Using Dynamic Time Warping to Find Patterns in Time Series," In Proc. of KDD Workshop, 1994.

[7] B. Bollobas, G. Das, D. Gunopulos and H. Mannila, "Time-Series Similarity Problems and Well-Separated Geometric Sets," In Proc. of the 13th SCG, Nice, France, 1997.

[8] T. Bozkaya, N. Yazdani and M. Ozsoyoglu, "Matching and Indexing Sequences of Different Lengths," In Proc. of the CIKM, Las Vegas, USA, 1997.

[9] J. Buhler and M. Tompa, "Finding motifs using random projections," *Journal of Computational Biology*, vol. 9(2), pp. 225-242, 2002.

[10] J. C. Chappelier, M. Gori and A. Grumbach, "Time in connexionist models," in *Sequence Learning: Paradigms, Algorithms and Applications*, R. S. a. G. L. Giles, Ed.: Springer-Verlag, 2000, pp. 105-134.

[11] B. Chiu, E. Keogh and S. Lonardi, "Probabilistic discovery of time series motifs," In Proc. of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, 493-498, 2003.

[12] G. Das, D. Gunopulos and H. Mannila, "Finding similar time series," In Proc. of Principles of Data Mining and Knowledge Discovery, 1st European Symposium, Trondheim, Norway, 88-100, 1997.

[13] G. Das, K. Lin, H. Mannila, G. Renganathan and P. Smyth, "Rule discovery from time series," In Proc. of the

4th International Conference on Knowledge Discovery and Data Mining, New York, NY, 16-22, 1998.

[14] E. Diday, "The dynamic clusters method in non hierarchical clustering," *International Journal of Computer Sciences*, vol. 2(1), pp., 1973.

[15] C. Faloutsos, M. Ranganathan and I. Manolopoulos, "Fast Subsequence Matching in Time Series Databases," In Proc. of ACM SIGMOD, 419-429, 1994.

[16] Forgy, "Cluster analysis of multivariate data: efficiency versus interpretability of classification," *Biometrics*, vol. 21768-769, 1965.

[17] R. Gaudin and N. Nicoloyannis, "Apprentissage non supervisé de séries temporelles à l'aide des k-means et d'une nouvelle méthode d'agrégation de séries," In Proc. of 5èmes Journées d'Extraction et de Gestion des Connaissances (EGC'05), Paris, France, 201-212, 2005.

[18] D. L. James and R. Miikkulainen, "A self-organizing feature map for sequences," *Advances in NeuralProcessing Systems*, vol. 7577-584, 1995.

[19] E. Keogh and S. Kasetty, "On the need for time series data mining benchmarks: A survey and empirical demonstration," In Proc. of, the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 102-111, 2002.

[20] E. Keogh, S. Lonardi and W. Chiu, "Finding surprising patterns in a time series database in linear time and space," In Proc. of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Edmonton, Alberta, Canada, 550-556, 2002.

[21] E. Keogh and M. Pazzani, "An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback," In Proc. of the 4th International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 239-241, 1998.

[22] E. Keogh and M. Pazzani, "Scaling Up Dynamic Time Warping for Data Mining Applications," In Proc. of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Boston, MA, USA, 285-289, 2000.

[23] E. Keogh and C. A. Ratanamahatana, "Exact indexing of dynamic time warping," *Knowledge and Information Systems: An International Journal (KAIS)*, 2004.

[24] G. Kollios, D. Gunopulos and V. Tsotras, "On Indexing Mobile Objects," In Proc. of the 18th ACM Symposium on Principles of Database Systems (PODS), 261-272, 1999.

[25] W. Krzanowski, "Between-groups comparison of principal components," *JASA*, vol. 74(367), pp., 1979.

[26] D. Kudenko and H. Hirsh, "Feature generation for sequence categorization," In Proc. of the 15th National Conference in Artificial Intelligence (AAAI'98), Menlo Park, California, USA, 733-739, 1998.

[27] S.-L. Lee, S.-J. Chun, D.-H. Kim, J.-H. Lee and C.-W. Chung, "Similarity Search for Multidimensional Data Sequences," In Proc. of ICDE, 599-608, 2000.

[28] N. Lesh, M. J. Zaki and M. Ogihara, "Scalable feature mining for sequential data," *IEEE Intelligent Systems*, vol. 15(2), pp. 48-56, 2000.

[29] F. R. Lin, L. S. Hsieh and S. M. Pan, "Learning clinical pathway patterns by hidden markov model," In Proc. of he 38th Annual Hawaii International Conference on System Sciences (HICSS'05), 2005.

[30] J. Lin, E. Keogh, P. Patel and S. Lonardi, "Finding motifs in time series," In Proc. of the 2nd Workshop on Temporal Data Mining, at the 8th International Conference on Knowledge Discovery and Data Mining (KDD'02), Edmonton, Alberta, Canada, 53-68, 2002.

[31] J. Lin, M. Vlachos, E. Keogh and D. Gunopulos, "Iterative incremental clustering of time series," In Proc. of the IX Conference on Extending Database Technology (EDBT 2004)², Crete, Greece, 2004.

[32] W. Lin, M. A. Orgun and G. J. Williams, "An overview of temporal data mining," In Proc. of The Australasian Data Mining Workshop, Macquarie University and CSIRO Data Mining, 2002.

[33] J. McQueen, "Some Methods for Classification and Analysis of Multivariate Observation," In Proc. of the 5th Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, CA, USA, 281-297, 1967.

[34] T. Oates, L. Firoiu and P. R. Cohen, "Clustering time series with hidden markov models and dynamic time warping," In Proc. of IJCAI-99 Workshop on Sequence Learning, 17-21, 1999.

[35] D. Pfoser, C. Jensen and Y. Theodoridis, "Novel Approaches in Query Processing for Moving Objects," In Proc. of VLDB, Cairo Egypt, 2000.

[36] Y. Qu, C. Wang and X. Wang, "Supporting Fast Search in Time Series for Movement Patterns in Multiple Scales," In Proc. of the ACM CIKM, 251-258, 1998.

[37] C. A. Ratanamahatana and E. Keogh, "Making Time-series Classification More Accurate Using Learned Constraints," In Proc. of SIAM International Conference on Data Mining (SDM '04), Lake Buena Vista, Florida, USA, 11-22, 2004.

[38] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimisation for spoken word recognition," *IEEE Trans. Acoustics, Speech and Signal Processing, ASSP*, vol. 26(1), pp. 43-49, 1978.

[39] S. Saltenis, C. Jensen, S. Leutenegger and M. A. Lopez, "Indexing the Positions of Continuously Moving Objects," In Proc. of the ACM SIGMOD, 331-342, 2000.

[40] C. Shahabi and D. Yan, "Real-time pattern isolation and recognition over immersive sensor data streams," In Proc. of the 9th International Conference On Multi-Media Modeling, 2003.

[41] D. Singhal and A. Seborg, "Clustering of multivariate time-series data," In Proc. of the American Control Conference, 2002.

[42] P. Somervuo and T. Kohonen, "Self-organizing maps and learning vector quantization for feature sequences," *Neural Processing Letters*, vol. 10(2), pp. 151-159, 1999.

[43] M. Vlachos, D. Gunopulos and G. Kollios, "Robust similarity measures for mobile object trajectories," In Proc. of 13th Database and Expert Systems Applications (DEXA), 5th International Workshop on Mobility in Databases and Distributed Systems (MDDS), Aix-en-Provence, France, 721-726, 2002.

[44] M. Vlachos, G. Kollios and D. Gunopulos, "Discovering similar multidimensional trajectories," In Proc. of the 18th International Conference on Data Engineering (ICDE'02), San Jose, CA, USA, 673-684, 2002.

[45] M. Vlachos, J. Lin, E. Keogh and D. Gunopulos, "A Wavelet-Based Anytime Algorithm for K-Means Clustering of Time-Series," In Proc. of Workshop on Clustering High-Dimensionality Data and its Applications, SIAM Datamining, San Francisco, CA, USA, 2003.

[46] K. Yang and C. Shahabi, "A PCA-based similarity measure for multivariate time series," In Proc. of he Second ACM International Workshop on Multimedia Databases, 2004.

[47] B. K. Yi, H. Jagadish and C. Faloutsos, "Efficient retrieval of similar time sequences under time warping," In Proc. of IEEE International Conference on Data Engineering, 201-208, 1998.