Time-series Clustering by Approximate Prototypes

Ville Hautamäki, Pekka Nykänen and Pasi Fränti
Speech and Image Processing Unit,
Department of Computer Science and Statistics,
University of Joensuu, Finland
{*villeh,pnykanen,franti*}*@cs.joensuu.fi*

Abstract

Clustering time-series data poses problems, which do not exist in traditional clustering in Euclidean space. Specifically, cluster prototype needs to be calculated, where common solution is to use cluster medoid. In this work, we define an optimal prototype as an optimization problem and propose a local search solution to it. We experimentally compare different time-series clustering methods and find out that the proposed prototype with agglomerative clustering followed by k-means algorithm provides best clustering accuracy.

1 Introduction

The goal of clustering is to group given N data objects into k clusters. When the data objects are fixed-dimensional feature vectors in an Euclidean space, standard clustering techniques such as k-means can be applied. However, when each data object is a sequence of observations forming a *time-series*, clustering approaches become rather scarce. Recently, traditional *point*-clustering techniques have been generalized for clustering time-series data [6]. Sequences appear commonly for example in bioinformatics [3], handwriting recognition [9], medical data [4] and multimedia data [7].

If all time-series are of equal length (which is not a common case), standard clustering techniques can be applied by considering each time-series as a long vector using Euclidean distance. However, this approach would not take into account the similarities in shape different sequences could possess.

Definition of distance between time-series objects can be divided into three categories: feature, model and shape-based methods [6]. In feature-based distance, equal length feature vector is calculated from each time-series, followed by Euclidean distance measurement. Model-based techniques fit a parametric model to each time-series, where distance can be any suitable model-model distance. Finally, shape- or raw data based-distance tries to match the shape of the two time-series as well as possible, by non-linearly stretching and contracting the time axes.

In this work, we apply the *shape matching* approach and in particular we define the distance between two time-series as the *dynamic time warping* (DTW) distance [8]. Analoguously to the k-means clustering in Euclidean space, we define our clustering cost function to be a sum of DTW distances from each input time-series to it's cluster prototype. Immediately, it follows that we need a way to compute the prototype time-series of a cluster. Or analogously, we need to compute an average time-series from the set of sequences. Due to the difficulty in computing the optimal prototype, typically a *medoid* is used instead. Medoid is the time-series in the cluster that minimizes the sum of distances to other sequences within the same cluster [9].

Moreover, we propose an iterative optimization scheme to compute a locally optimal prototype. In addition, we apply two Euclidean space clustering methods to time-series clustering: *random swap* (RS) [2] and hierarchical clustering followed by k-means finetuning.

2 Clustering methods

Clustering time-series data can be defined as an optimization problem, where the goal is to partition N time-series into k disjoint subsets. Each time-series is a sequence of vectors $s_i = (\boldsymbol{s}_1, \ldots, \boldsymbol{s}_{M_i})$, where the dimensionality (D) of the vectors is fixed, but the length M_i depends on the sequence i. The set of all time-series is denoted by S, and $S_j \subset S$ contains time-series in the cluster j.

We define the cost of the clustering as a sum of dynamic time warping distances between the time-series

978-1-4244-2175-6/08/$25.00 ©2008 IEEE

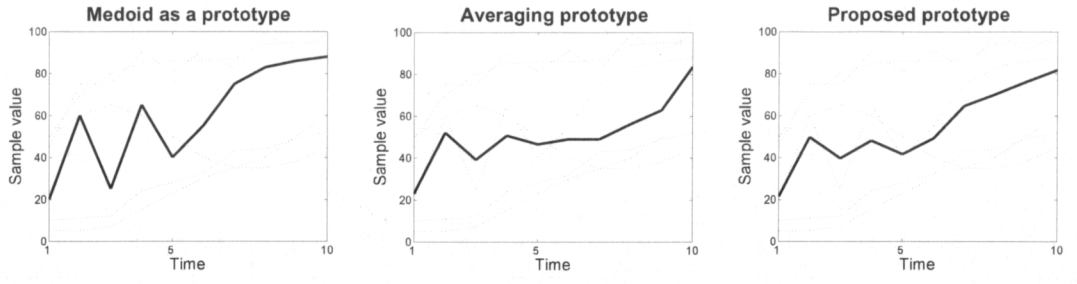

Figure 1. Example of prototype computations for one-dimensional time-series: medoid ($E(S) = 159$), averaging ($E(S) = 138$) and proposed after 10 iterations ($E(S) = 118$).

observation and its cluster prototype:

$$E(S) = \frac{1}{N} \sum_{i=1}^{N} d_{\mathrm{DTW}}(s_i, c_{p(i)}). \quad (1)$$

Here $p(i)$ is the index of the cluster where the s_i belongs to, and c_j is the prototype time-series representing the cluster j.

Dynamic time warping distance between two sequences, s of length M and c of length L, is initialized by first calculating a M-by-L distance matrix. Element (i, j) in the matrix is defined as $d(\boldsymbol{s}_i, \boldsymbol{c}_j) = \|\boldsymbol{s}_i - \boldsymbol{c}_j\|^2$. Warping path between two sequences is defined as $W = (w_1, \ldots, w_K)$, where $w_k = (i, j)_k = d(\boldsymbol{s}_i, \boldsymbol{c}_j)$ from the distance matrix. The goal of the dynamic time warping is to find the monotonically increasing path from $(1, 1)$ to (M, L) that minimizes: $\sum_{k=1}^{K} w_k$.

Optimal path is found using dynamic programming by evaluating the following recurrence:

$$\begin{aligned} \gamma_{i,j} &= d(\boldsymbol{s}_i, \boldsymbol{c}_j) \\ &+ \min\{\gamma_{i-1,j-1} + \gamma_{i-1,j} + \gamma_{i,j-1}\}. \end{aligned} \quad (2)$$

2.1 k-means

The k-means algorithm (KM) consist of two steps that are iterated: i) partition step and ii) prototype step. In partition step, each time-series is mapped to its nearest prototype time-series. Then a new prototype is computed for each subset S_i. The algorithm iterates between these two steps until convergence.

2.2 Random swap

Random swap (RS) [2] has been successfully applied for the clustering problem in Euclidean space. In the following, we apply the same design principles in the case of sequences. The idea of the method is to avoid getting stuck to local optimum by swapping randomly, one cluster to a new location. Cluster swapping is performed by first randomly selecting one cluster to be deleted, and then selecting one time-series from the input data set to be a new prototype of the moved cluster. After swapping, k-means is used to fine tune the solution. If the new solution is better than the previous best, it will replace the current best; otherwise it will be discarded.

2.3 Agglomerative clustering

Agglomerative clustering generates the clusters by a series of merge operations. Agglomeration process starts by initializing each data vector as its own cluster. Average linkage agglomerative clustering has been proposed for the time-series data [4]. In each iteration it will merge two clusters that minimize the distance between the pairs of time-series:

$$d_{\mathrm{AL}}(S_a, S_b) = \frac{1}{|S_a| + |S_b|} \sum_{i=1}^{|S_a|} \sum_{j=1}^{|S_b|} d_{\mathrm{DTW}}(s_i, s_j), \quad (3)$$

where S_a and S_b are two clusters considered for the merge.

After k clusters have been formed, cluster prototype is computed for each partition, we denote this method as *hierarchical clustering* (HC). k-means can also be used to fine tune the solution of the hierarchical methods. This variant we then denote as HC+KM.

3 Prototype computation

In the partitioning stage, each time-series from the input set is given a cluster label. Given these labels, we compute new prototype. Given sequences in a cluster, it is clear that the cluster's prototype c minimizing (1) is

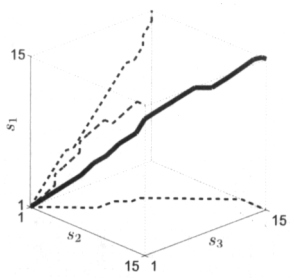

Figure 2. Performing optimal dynamic time warping on three time-series simultaneously (solid line). Pairwise warping paths are also shown (dashed lines).

such that:

$$E(S_j, c) = \sum_{s_i \in S_j} d_{\mathrm{DTW}}(s_i, c), \tag{4}$$

is minimized. Sequence c that minimizes $E(S_j, c)$ is called a *Steiner sequence* [3].

In time-series clustering, most common way to approach this problem is to use cluster medoid as the prototype [5]. Which is a time-series of the cluster, defined as:

$$c_j = \arg\min_{s_j \in S_i} \sum_{s_k \in S_i \setminus s_j} d_{\mathrm{DTW}}(s_k, s_j). \tag{5}$$

Three different ways to compute the cluster prototype are shown in Fig. 1. First panel shows the medoid as a prototype, which is one of the time-series in the cluster. Other two methods use medoid as the initial guess.

3.1 Optimal prototype

Steiner sequence can be computed from the $|S_j|$-dimensional dynamic time warping, where $|S_j|$ is the number of sequences in the cluster. Fig. 2 shows optimal alignment of three time-series s_1, s_2 and s_3 using DTW. Three dimensional DTW alignment will produce warping path $w = ((f_1(1), f_2(1), f_3(1)), \ldots, (f_1(K), f_2(K), f_3(K)))$ where $f_i()$ gives an index to the vector in time-series s_i, and K is the length of the warping path. Optimal prototype is the time-series of length K, where each vector is the average of the original ones given by the warping path.

Unfortunately, the search space grows exponentially as a function of $|S_i|$, as shown in Fig. 2. In fact, finding the Steiner sequence has been proven to be NP-complete in the discrete case [3].

3.2 Averaging method

A common way to compute average time-series of the set of sequences is to combine two sequences at the time, using DTW, until only one time-series is left. Unfortunately, the order in which the pairing is performed affects the final prototype [7].

Abdulla *et al.* [1] proposed a *cross-words reference template* (CWRT), which uses medoid as the reference time-series as follows. First, all sequences are aligned by DTW to a single medoid, and then the average time-series is computed. The resulting time-series has the same length as the medoid, but the method is invariant to the order of processing sequences.

Algorithm 1 Local search prototype (LS)

$c_{\mathrm{old}} \leftarrow$ Compute medoid of the cluster using (5).
repeat
  Compute warping paths to c_{old}
  $c_{\mathrm{new}} \leftarrow$ compute new time-series using paths
until $(E(S, c_{\mathrm{new}}) < E(S, c_{\mathrm{old}}))$

3.3 Prototype by local search

We propose an iterative heuristic to the problem of finding time-series clustering prototype as follows. Starting from the medoid, we iterate between the averaging stage and the mapping stage: i) calculate averaged prototype based on warping paths and ii) calculate new warping paths to the averaged prototype. The averaging stage is the same as discussed in Section 3.2

We can view the method as the generalization of the centroid step in the k-means clustering in Euclidean space. In that case, the result of the proposed method corresponds to cluster centroid.

Table 1. Summary of datasets.

	Char.	Speech	Synthetic
N	925	640	600
k	59	9	6
D	2	12	1
Min. length	15	7	60
Max. length	155	29	60
Avg. length	54	16	60

4 Experiments

Clustering algorithms were tested on three data sets from the UCI Machine Learning Repository[1], including hand-written characters, speech and synthetic data. Summary of data sets is found in the Table 1.

[1] http://www.ics.uci.edu/~mlearn/MLrepository.html

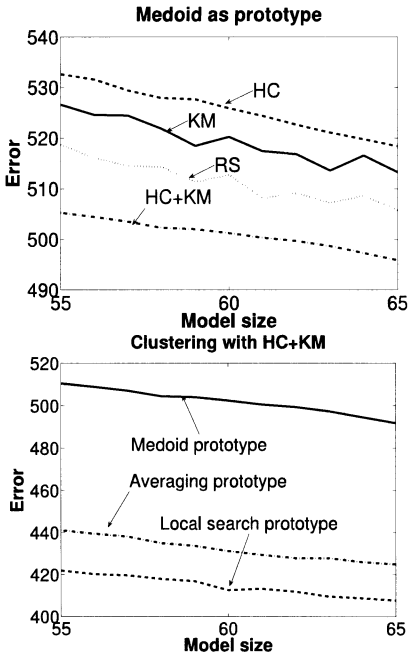

Figure 3. Results for Char. dataset.

All reported results are averages over 10 test runs, using the same setup. We fixed the number of k-medoids iterations for each data set to keep the processing time reasonable. Running until convergence had only little benefit for the clustering quality.

We first compare the clustering algorithms using medoids as prototypes. The results in Fig. 3 show that HC+KM gives the best clustering result among these methods. Using this algorithm, we then compare the methods for calculating the prototypes. The average and local search heuristics clearly outperform the use of medoids.

Results for the other test sets have been summarized in Table 2. and similar observations can be made. For further tests, we select the HC+KM algorithm, and the local search heuristic for calculating the cluster prototypes. The best combination (HC+KM with LS prototypes) is compared with existing methods in Table 3. In case of all data sets, the proposed variant outperforms the others in quality.

5 Conclusions

We formulated the prototype computation problem as a optimization task, and proposed an local search solution to solve it. It provides 10-22% improvement to k-medoids.

Table 2. Clustering accuracy results.

Method	Protype	Char. $k = 59$	Speech $k = 9$	Synthetic $k = 6$
KM	Medoid	537	8.17	160
	Aver.	449	6.72	150
	LS	430	6.68	147
RS	Medoid	523	7.69	156
	Aver.	439	6.57	146
	LS	422	6.56	**144**
HC	Medoid	555	7.80	156
	Aver.	466	6.67	147
	LS	439	6.64	**144**
HC+KM	Medoid	504	7.68	155
	Aver.	434	6.58	147
	LS	**417**	**6.53**	**144**

Table 3. Summary of results, where time is in sec.

	Char. $k = 59$		Speech $k = 9$		Synthetic $k = 6$	
Method	Err.	Time	Err.	Time	Err.	Time
k-medoids [5]	537	25	8.17	3	160	24
HC [4]	555	433	7.80	76	156	399
CWRT [1]	449	76	6.72	20	150	99
Proposed	**417**	515	**6.53**	80	**144**	499

References

[1] W. H. Abdulla, D. Chow, and G. Sin. Cross-words reference template for DTW-based speech recognition systems. In *Proc. TENCON*, volume 2, pages 1576–1579, Bangalore, October 2003.

[2] P. Fränti and J. Kivijärvi. Randomized local search algorithm for the clustering problem. *Pattern Analysis & Applications*, 3(4):358–369, 2000.

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

[4] S. Hirano and S. Tsumoto. Empirical comparison of clustering methods for long time-series database. In *LNCS*, volume 3430, pages 268–286. 2005.

[5] L. Kaufman and P. J. Rousseeuw. *Finding groups in data: an introduction to cluster analysis*. John Wiley Sons, New York, 1990.

[6] T. W. Liao. Clustering of time series data – a survey. *Pattern Recognition*, 38:1857–1874, 2005.

[7] V. Niennattrakul and C. A. Ratanamahatana. On clustering multimedia time series data using k-means and dynamic time warping. In *Proc. MUE*, 2007.

[8] H. Sakoe and S.Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Trans. Ac., Sph., and Sig. Proc.*, 26:143–165, 1978.

[9] V. Vuori and J. Laaksonen. A comparison of techniques for automatic clustering of handwritten characters. In *Proc. ICPR*, volume 2, pages 168–171, 2002.