Compensation of Translational Displacement in Time Series Clustering using Cross Correlation

Frank Höppner[1] and Frank Klawonn[23]

[1] Department of Economics
University of Applied Sciences Braunschweig/Wolfenbüttel
Robert Koch Platz 10-14, D-38440 Wolfsburg, Germany
[2] Department of Computer Science
University of Applied Sciences Braunschweig/Wolfenbuettel
Salzdahlumer Str. 46/48, D-38302 Wolfenbuettel, Germany
[3] Helmholtz Centre for Infection Research
Department for Cell Biology
Inhoffenstr. 7, D-38124 Braunschweig, Germany

Abstract. Although k-means clustering is often applied to time series clustering, the underlying Euclidean distance measure is very restrictive in comparison to the human perception of time series. A time series and its translated copy appear dissimilar under the Euclidean distance (because the comparison is made pointwise), whereas a human would perceive both series as similar. As the human perception is tolerant to translational effects, using the cross correlation distance would be a better choice than Euclidean distance. We show how to modify a k-means variant such that it operates correctly with the cross correlation distance. The resulting algorithm may also be used for meaningful clustering of time series subsequences, which delivers meaningless results in case of Euclidean or Pearson distance.

1 Introduction

Finding typical patterns within a set of short time series by cluster analysis is a common approach in data analysis. Short time series might arise explicitly, for instance, as growth curves of bacteria or populations under varying conditions, but short time series can also be extracted from one long time series in the form of a sliding window. However, clustering such short time series derived from sliding windows can easily lead to meaningless results [13], a problem that will be addressed in more detail later on in this paper.

Among the applied clustering methods, k-means with Euclidean distance is most frequently used [12]. Typically, normalization of the time series is carried out, so that scaling of the measurements and the basic level of the time series

[0] published in: Advances in Intelligent Data Analysis VIII, 8th International Symposium on Intelligent Data Analysis, IDA 2009, LNCS 5772, pp 71–82, Springer

do not have any influence. The common choice is z-score normalization. In this case, the Euclidean distance corresponds to the Pearson correlation coefficient up to a constant factor – thus, cluster analysis of time series with the Euclidean distance with z-score normalized data as the distance measure is (almost) the same as clustering with the Pearson correlation as the distance measure [2].

Although z-score normalization and the Pearson correlation coefficient rule out differences based on scaling of the measured variable, they do not ensure a suitable time series alignment in case the observed "patterns" do not start at the same time (e.g., depending on the start of recording the data). In order to group time series with a similar patterns that are shifted in time, cross correlation appears to be a better choice. This paper discusses how to incorporate the cross correlation distance into k-means clustering. We want to emphasize, that our main purpose is *not* to advocate the cross correlation distance as the best measure to compare time series[1], but to show, once the decision to use cross correlation has been made, *how* to modify k-means appropriately to guarantee correct objective function-optimization.

For reasons of robustness [15], rather than simple k-means clustering we will use fuzzy clustering which is nothing else than a reformulation of k-means clustering with continuous membership degrees. Nevertheless, the proposed approach does not depend on fuzzy clustering and works in the same way with crisp clustering.

2 Brief Review of Fuzzy c-Means and Noise Clustering

Given a dataset $\{x_1, \ldots, x_n\} \subset \mathbf{R}^p$, k-means as well as fuzzy c-means (FCM) clustering [3] aims at minimizing the following objective function

$$f = \sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}^m d_{ij} \tag{1}$$

under the constraints

$$\sum_{i=1}^{c} u_{ij} = 1 \qquad \text{for all } j = 1, \ldots, n \tag{2}$$

where $u_{i,j}$ is the membership degree of data object x_j to cluster i and $d_{ij} = \|x_j - v_i\|^2$ denotes the squared Euclidean distance between data vector x_j and prototype $v_i \in \mathbf{R}^p$ representing cluster i. While k-means assumes a crisp assignment $u_{i,j} \in \{0, 1\}$, its *continuous* counterpart allows $u_{ij} \in [0, 1]$. c is the chosen number of clusters and m is the so-called fuzzifier, controlling (in case of fuzzy c-means) how much clusters may overlap.

Clustering is thus considered as a nonlinear optimization problem which is usually solved by an alternating scheme. The prototypes are chosen randomly

[1] If the series are dilated, measures such as dynamic time warping may be an option.

in the beginning or by some suitable initialization strategy. Fixing the cluster prototypes, the optimal choice for the membership degrees is given by[2]

$$u_{ij} \;=\; \frac{1}{\sum_{k=1}^{c}\left(\frac{d_{ij}}{d_{kj}}\right)^{\frac{1}{m-1}}} \tag{3}$$

which is used as an update equation for the membership degree. Fixing the membership degrees, the best choice for the prototypes is

$$v_i \;=\; \frac{\sum_{j=1}^{n} u_{ij}^m x_j}{\sum_{j=1}^{n} u_{ij}^m}. \tag{4}$$

The alternating scheme is repeated until the algorithm converges, i.e., no more (or almost no) changes happen.

A very simple extension of k-means and fuzzy c-means clustering to cope with outliers is noise clustering [7]. In the set of prototypes, an additional noise cluster is included. All data have a fixed (usually large) distance d_{noise} to the noise cluster. As soon as the distance of some data x to the nearest cluster p comes close to d_{noise}, the noise cluster gains a considerable fraction of the total membership degree, thereby reducing the influence of x with respect to p. Noise clustering simply requires to exchange (3) by

$$u_{ij} \;=\; \frac{1}{\left(\frac{d_{ij}}{d_{\text{noise}}}\right)^{\frac{1}{m-1}} + \sum_{k=1}^{c}\left(\frac{d_{ij}}{d_{kj}}\right)^{\frac{1}{m-1}}} \tag{5}$$

and represents and effective mean to reduce the influence of noise and extract cluster prototypes more clearly.

In the objective function (1), the number of clusters c must be known or specified in advance. This is, of course, an unrealistic assumption. There are various approaches to determine the number of clusters automatically (for an overview, we refer to [4, 10]). It is also possible to find clusters step by step, extending the idea of noise clustering [8]. A detailed discussion of methods for determining the number of clusters is out of the scope of this paper.

3 Measuring Time Series Similarity

Suppose we have two time series r and s, consisting of T samples each $r = (r_1, r_2, \ldots, r_T) \in \mathbf{R}^T$. The *squared Euclidean distance* between two time series r and s is given by:

$$d_E(r,s) = \sum_{t=1}^{T}(r_t - s_t)^2 \tag{6}$$

[2] If $d_{ij} = 0$ for one or more clusters, we deviate from (3) and assign x_j with membership degree 1 to the or one of the clusters with $d_{ij} = 0$ and choose $u_{ij} = 0$ for the other clusters i.

For time series analysis, it is often recommended to normalize the time series either globally or locally to tolerate vastly differing ranges [12]. Another prominent measure for time series comparison is the Pearson correlation coefficient, which measures the correlation ϱ between two random variables X and Y:

$$\varrho_{X,Y} = \frac{E[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X \sigma_Y} \tag{7}$$

where μ_X denotes the mean and σ_X the standard deviation of X. We obtain a value of ± 1 if X and Y are perfectly (anti-) correlated and a value of ≈ 0 if they are uncorrelated. In order to use the Pearson correlation coefficient as a distance measure for time series it is desirable to generate low distance values for positively correlated (and thus similar) series. The *Pearson distance* is therefore defined as

$$d_P(r,s) = 1 - \varrho_{r,s} = 1 - \frac{\frac{1}{T}\sum_{t=1}^{T}(r_t-\mu_r)(s_t-\mu_s)}{\sigma_r \sigma_s} \tag{8}$$

such that $0 \le d_P(r,s) \le 2$. One can show that k-means clustering (via Euclidean distance) on z-score normalized time series is (almost) equivalent to k-means clustering using the Pearson correlation distance[3] [2].

Sometimes time series do not perfectly align in time and the best correspondence is obtained when shifting both series against each other. Two identical time series, one of them shifted by δ in time, may appear uncorrelated under the Pearson coefficient. The normalized cross correlation takes this shift δ into account and measures the Pearson correlation between r and a series s shifted by δ:

$$\varrho_{r,s}(\delta) = \frac{\frac{1}{T}\sum_{t=-T}^{T}(r_t-\mu_r)(s_{t+\delta}-\mu_s)}{\sigma_r \sigma_s} \tag{9}$$

with $s_t = r_t = 0$ for $t < 1$ and $t > T$. Cross correlation can help to overcome the missing alignment of the series by choosing δ such that the Pearson correlation becomes maximal. We define the cross correlation distance d_X as the best Pearson coefficient we may achieve for an optimal lag of δ where $-T \le \delta \le T$:

$$d_X(r,s) = 1 - \max\{\varrho_{r,s}(\delta) \mid -\varDelta \le \delta \le \varDelta\} \tag{10}$$

4 Compensating Translational Displacement

There are many more distance measures for comparing time series (cf. [12]), but often the proposed distance measures are simply plugged into an existing clustering algorithm without taking influence on the internal steps of the respective algorithm. But at least for objective function-based clustering (such as k-means or fuzzy c-means) replacing the distance measure alone is not sufficient:

[3] It is *almost* equivalent because a normalization of the obtained prototypes is missing, but the absence of this re-scaling is usually neglectable in terms of results.

to operate correctly, the prototype update step has to be adapted to the chosen distance measure. The objective of this paper is therefore not to propose a new distance measure (as Pearson correlation and cross correlation are already well-established), but to modify the prototype update step in fuzzy c-means clustering such that prototypes are optimized w.r.t. cross correlation rather than Euclidean distance.

Let us denote the data series by x_j, $1 \leq j \leq n$, and the prototype series by p_i, $1 \leq i \leq c$. The cross correlation clustering (CCC) algorithm aims at minimizing the objective function

$$f = \sum_{i=1}^{c} \sum_{j=1}^{n} u_{i,j}^{m} d_X(p_i, x_j)$$

subject to (2). We introduce for any pair of time series (p_i, x_j) the lag parameter $\delta_{i,j}$ and reformulate f as

$$f = \sum_{i=1}^{c} \sum_{j=1}^{n} u_{i,j}^{m} (1 - \varrho_{p_i,x_j}(\delta_{i,j}))$$

The optimization will be carried out by alternating optimization in three steps: (a) optimize w.r.t. $\delta_{i,j}$ assuming prototypes and memberships being constant (Sect. 4.1), (b) optimize w.r.t. prototypes assuming lags and memberships being constant (Sect. 4.2), and (c) optimize w.r.t. memberships assuming lags and prototypes to be constant. The last step (c) is independent of the distance and therefore identical to update equations (3) or (5).

4.1 Efficient calculation of the optimal lag

Calculating the cross correlation for a given value of δ is $O(T)$ in case of discrete time series of length T. Exploring the full range of possible δ-values is linear in T, too, so the overall complexity of distance estimation becomes $O(T^2)$. This time can be reduced to $O(T \log T)$ using the Fast Fourier Transform. (Unnormalized) cross correlation can be interpreted as a convolution $r \star s$ of two time series

$$(r \star s)(\delta) = \sum_{t=1}^{T} r_t s_{\delta - t}$$

Note that with convolution we have a factor of $s_{\delta-t}$ whereas cross correlation uses $s_{\delta+t}$. By reversing series s in time ($\bar{s}_t = s_{T-t}$) we overcome this difference.

However, we need to convolve two *normalized series* r and s: Suppose the first half of s correlates perfectly with the second half of r, but the remainder of the series are random noise. If we would normalize the respective halves of r and s individually, and calculate the correlation we would obtain a coefficient of 1.0. But if we would simply normalize r and s once beforehand, a lag of $\delta = T/2$ would deliver a different coefficient because the mean and variance considers also those parts of the time series that are otherwise masked out by the shift.

Therefore, the normalization has to be carried out individually for each possible lag δ.

For notational convenience, we consider all series having indices ranging from $-T+1$ to $2T$ (filled up with zeroes) and denote an offset of δ by $s^{+\delta}$ as shown in this example for $T=4$:

$$\begin{aligned} r &= (\,3\,1\,3\,5\,) \qquad \in \mathbf{R}^4 \\ s &= (\,1\,3\,5\,2\,) \\ s^{+0} &= (0\,0\,0\,0\,1\,3\,5\,2\,0\,0\,0\,0) \\ s^{+1} &= (0\,0\,0\,0\,0\,1\,3\,5\,2\,0\,0\,0) \\ s^{-2} &= (0\,0\,1\,3\,5\,2\,0\,0\,0\,0\,0\,0) \end{aligned}$$

We define $\chi \in \mathbf{R}^T$ as $\chi_t = 1$ for any $1 \leq t \leq T$ and $\chi_t = 0$ otherwise. For instance

$$\begin{aligned} \chi &= (0\,0\,0\,0\,1\,1\,1\,1\,0\,0\,0\,0) \\ \chi^{-2} &= (0\,0\,1\,1\,1\,1\,0\,0\,0\,0\,0\,0) \end{aligned}$$

The calculation of mean and standard deviation and subsequent normalization of two series, say r and s^δ, has to be carried out for the all indices t with $\chi_t \cdot \chi_t^\delta = 1$.

Fortunately, this normalization can also be carried out efficiently: By means of a series of partial sums $\hat{r}_0 = r_0$, $\hat{r}_{t+1} = \hat{r}_t + r_{t+1}$ and $\hat{\hat{r}}_0 = r_0^2$, $\hat{\hat{r}}_{t+1} = \hat{\hat{r}}_t + r_{t+1}^2$ we obtain the mean for the respective subseries $r_{i..j}$ from $(\hat{r}_j - \hat{r}_{i-1})/(j-i+1)$ and the variance from $(\hat{\hat{r}}_j - \hat{\hat{r}}_{i-1})/(j-i+1) - (\hat{r}_j - \hat{r}_{i-1})^2/(j-i+1)^2$ due to $\text{Var}[X] = E[X^2] - (E[X])^2$. Therefore the determination of the optimal lag δ remains $O(T \log T)$ even in the case of normalized correlation coefficients.

Revisiting the last example, we note that series r and s correlate perfectly for different values of δ, for instance, $\delta = +1$ and $\delta = -2$. In general we can expect high values of normalized correlation for $\delta \approx \pm T$ because then only a few values have to correlate incidentally. Therefore we introduce a bias in d_X towards preferable *long* matches. The distance d_X is multiplied by an additional *overlap factor* of $\frac{T-|\delta_{i,j}|}{T}$ where $T - |\delta_{i,j}|$ is the number of valid index positions shared by both series. For small lags we thus obtain an almost unaltered correlation coefficient whereas for large values of δ a possibly high coefficient is penalized due to its limited relevance. In our example, the unique best lag for s and r is now $\delta = 1$.

4.2 Determination of the prototypes

As already mentioned, k-means clustering of z-score normalized series is (almost) identical to clustering via Pearson correlation, that is, the prototypes are obtained by the weighted mean of the series that are associated to the prototype (followed by a normalization step in case of Pearson correlation) [2]. Once the optimal lags for cross correlation have been determined, we consider them in the second step of alternating optimization as being constant and the cross correlation distance reduces to Pearson distance for prototypes p_i and the shifted data series $x_j^{+\delta_{i,j}}$.

As we have seen in the previous section, any two series p_i and $x_j^{+\delta_{i,j}}$ share different ranges of indizes and the prototype calculation has to be carried out pointwise (that is, index by index). Then minimization w.r.t. the prototypes leads to the following update equations (which are a generalization of the weighted mean to shifted series):

$$v_{i,t} = \sum_{j=1}^{n} u_{i,j}^{m} \cdot x_{j,t}^{+\delta_{i,j}}$$
$$w_{i,t} = \sum_{j=1}^{n} u_{i,j}^{m} \cdot \chi_{j,t}^{+\delta_{i,j}}$$
$$p'_{i,t} = \frac{v_{i,t}}{w_{i,t}}$$

This calculation is not restricted to the indices $t = 0 \ldots T$, but carried out for the full range of $t = -T+1 \ldots 2T$. If there are many data series x_j whose second half matches the first half of a prototype p, a higher *overlap factor* may possibly be achieved next time if the prototype would be shifted appropriately. The optimization w.r.t. the overlap factor is accomplished by analyzing the intermediate result $w_{i,t}$: These individual weights per index position indicate how often an optimal match to a data series has involved this index position. Figure 1 shows an example: the graph at the bottom shows the resulting pointwise mean (full width of $3T$) and the graph at the bottom the accumulated weight per index position. The optimal (T-dimensional) prototype subvector of p'_i is then found at

$$t_0 = \text{argmax}_{t_0=-T+1..T} \sum_{t=1}^{T} w_{i,t_0+t}$$

Using a series of partial sums, this optimal position can be found in $O(T)$.

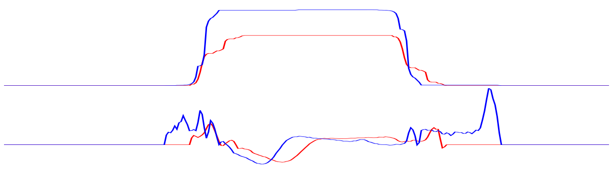


Fig. 1. Example for $w_{i,t}$ (top) and $p'_{i,t}$ (bottom) for $t = -T+1 \ldots 2T$. To locate the best offset t_0 we identify the area of width T with the highest sum of weights ($\text{argmax}_{t_0=-T+1..T} \sum_{t=1}^{T} w_{i,t_0+t}$).

4.3 Interpretation of the Noise Distance

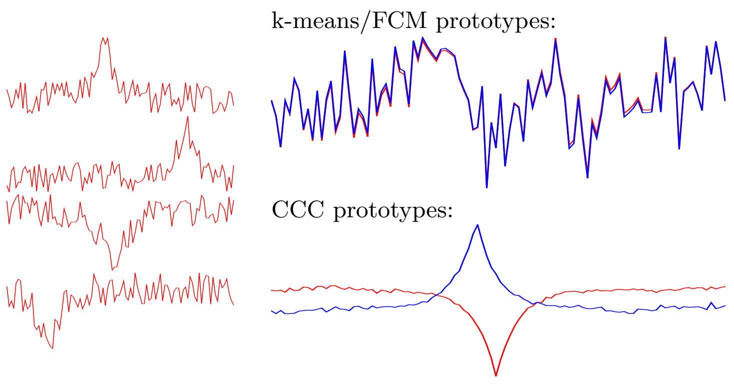
The third step of alternating optimization involves the membership update. For the case of correlation coefficients, the noise clustering approach is particularly appealing. As fuzzy c-means is a partitional clustering algorithm, all data series have to be assigned to the clusters, including those that do not correlate to any of the prototypes (or correlate negatively). By selecting a threshold correlation coefficient of, say, $d_{\text{noise}} = 0.5$ the noise cluster attracts the membership of poorly correlating series, thereby avoiding a contamination of the clusters with poor matches and preventing a blurring of the cluster prototypes. Since the correlation coefficient has a fixed range of $[0, 2]$, the noise distance is easily interpretable.

5 Experimental Evaluation

We demonstrate the proposed method for clustering time series as well as clustering of time series subsequences (STS).

5.1 Clustering (Whole) Time Series

As a first test case, we consider the *hill & valley* dataset from the UCI repository [1]: Each record represents 100 points on a two-dimensional graph. When plotted in order the points will create either a hill (a bump in the terrain) or a valley (a dip in the terrain). We have z-score normalized the data and applied k-means clustering and cross correlation clustering, the results are shown in Fig. 2. As the peaks occur at different places, calculating the mean will eliminate all the individiual peaks and ends up with a noisy prototype. Both k-means prototypes converge roughly to the mean of the full data set, the two classes are not recovered by the prototypes. In contrast, the translational displacement is identified by the CCC algorithm. The hills and valleys are perfectly gathered in the respective clusters.

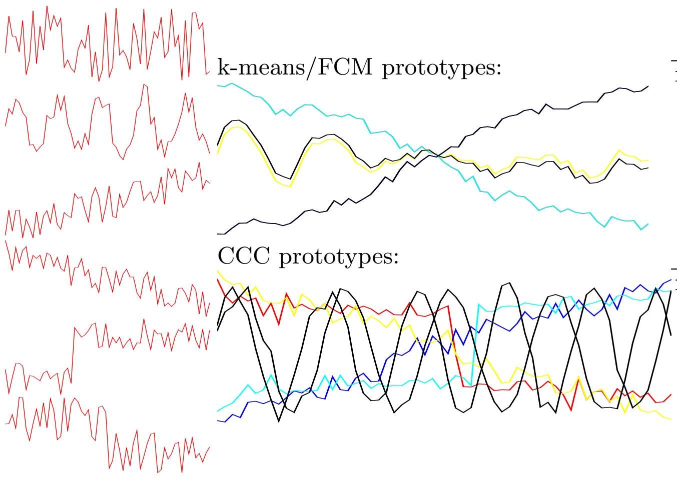


	#0	#1
valley	156	151
hill	150	149

	#0	#1
valley	307	0
hill	0	299

Fig. 2. Results on the hill & valley dataset. On the left, four example series are shown. The final prototypes and confusion matrix is given for k-means/FCM (top) and CCC (bottom).

The *synthetic control chart* dataset (taken from [14]) is also frequently used for the evaluation of time series clustering methods: The dataset contains 600 examples of synthetically generated control charts from six different classes: normal, cyclic, increasing trend, decreasing trend, upward shift, and downward shift (cf. left column of Fig. 3). Although started with 6 clusters, k-means/FCM ends up with three different prototypes only (always two prototypes are almost identical). The increasing and decreasing trend is well recovered, but these clusters also cover many examples from the up-shift and down-shift class. As the abrupt step in these series occurs at different points in time, k-means/FCM cannot detect them as individual clusters. The same argument applies to the cyclic series which have different phases. Again, CCC performs much better, all classes but one are recovered by the respective clusters. The first class (normal) consists of noise only, therefore it does not correlate to any existing cluster nor with other noisy series. Thus, the examples from this case are distributed among the other clusters by chance. The now superfluous sixth cluster is used to split the cyclic cluster into two prototypes. (The "optimal" number of clusters is thus 5 for CCC).



	5	3	1	0	4	2
norm	12	38				
cycl	10	40				
incr			50			
decr				50		
upsh			29		21	
dnsh				43		7

	4	5	1	3	2	0
norm	3	5	6	6	6	16
cycl	24	26				
incr			46		4	
decr				50		
upsh			10		40	
dnsh				2		48

Fig. 3. Results on the synthetic control chart dataset. On the left, one example from each class is shown. The final prototypes and confusion matrix is given for k-means/FCM (top) and CCC (bottom).

The results for the *cylinder-bell-funnel* (CBF) dataset, taken from [14], are shown in Fig. 4. It consists of 20 instances of three translated and dilated basic shapes (five example series in left column of Fig. 4). In the shown example run of k-means/FCM two of the three clusters occassionally collapsed into one prototype, but in general k-means/FCM performs quite well on this dataset. Although the effects of dilation cannot be compensated by the CCC algorithm, the overall quality of the clusters is superior. A comparison between k-means/FCM and CCC with respect to the steepness of the flanks in the bell and funnel patterns reveals that the CCC patterns are much closer to the original patterns while the k-means/FCM clusters are somewhat blurred.

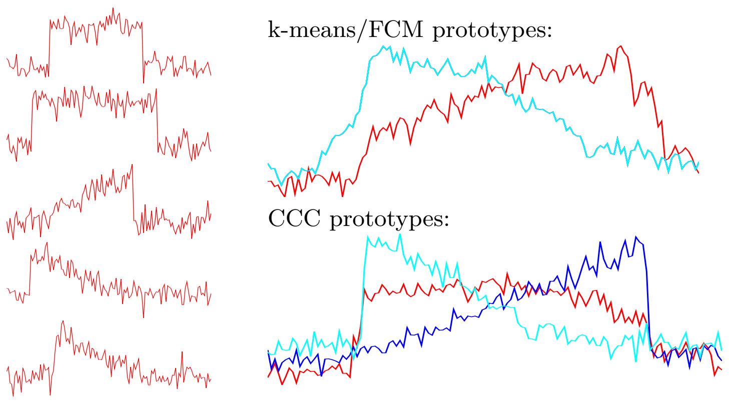


Fig. 4. Results on the cylinder-bell-funnel dataset. On the left, two examples from the cylinder class, one from the bell and two from the funnel class are shown. The final prototypes are presented for k-means/FCM (top) and CCC (bottom).

5.2 Clustering Subsequences of Time Series

Clustering subsequences of time series has been proposed in [6] and thereafter been used by many authors as a tool to extract patterns from time series. The resulting clusters, however, were of poor quality in practice [9], being very similar to translated and dilated trigonometric functions. A deeper analysis led to the conclusion that subsequence time series clustering is completely meaningless [13], because the resulting trigonometric patterns appeared to be independent of the input data.

While there are different attempts to explain this undesired effect [13, 11, 5], an intuitive explanation why subsequence clustering fails is easily given: the input series are obtained by shifting a sliding window of fixed length over the original series. Suppose we have a noisy series with a single bump (cf. hill and valley dataset in Fig. 2) then due to the subsequence generation this bump will re-occur at any location in the input series. The detection of a single cluster or pattern would be the desired result, but k-means clustering with $k = 1$ would average all the series and as the bump never repeats itself at the same spot, the bump gets completely blurred. So the problem is caused by the translation of the original pattern – and the CCC algorithm seems well prepared to compensate this displacement. Therefore, it can be considered as a promising candidate to overcome the problem of meaningless clusters in time series subsequence clustering.

For the hill and valley, CBF and ECG datasets we have concatenated all the series to a single, long time series and created a new dataset by moving a sliding window along the resulting series. Figure 5 shows the result of k-means/FCM and CCC in case of the hill and valley and CBF dataset, and Fig. 6 for the ECG200 dataset. In all cases, the k-means/FCM clustering algorithm delivers trigonometric shapes, whereas the CCC clusters correspond well to the underlying patterns.

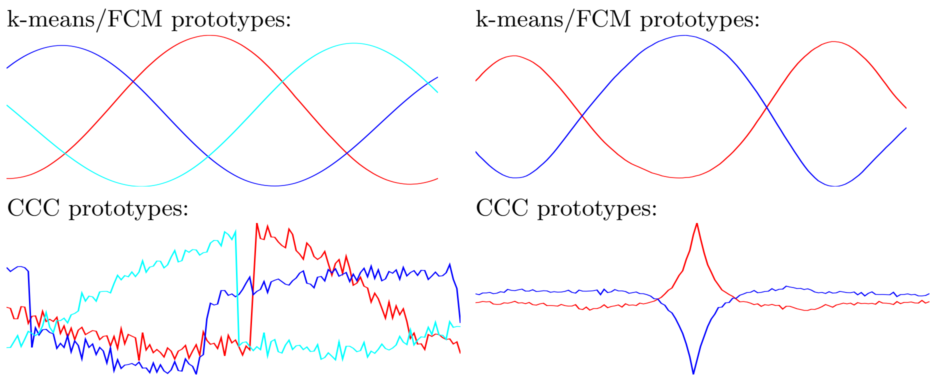


Fig. 5. Clustering of time series subsequences. Left: CBF dataset, right: hill & valley dataset.

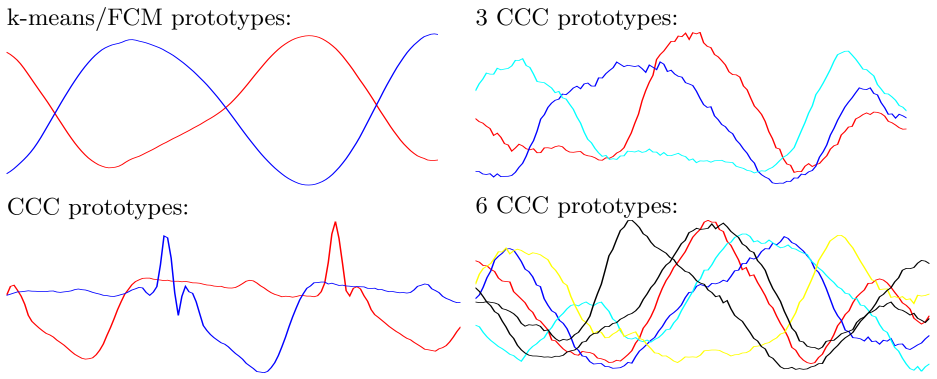


Fig. 6. Clustering of time series subsequences. Left: ECG dataset, right: windstrength dataset.

The CCC algorithm has also been applied to real data, namely wind strength data measured hourly on a small island in the northern sea (shown in Fig. 6).

The sliding window was 5 days long. With real data we do not expect such distinct shapes as in the artificial datasets CBF or hill & valley, because the noise ratio is much higher and different patterns are not separated from each other by a period of time where the series remains constant (which makes it easy to distinguish the *pattern* from the *non-pattern* parts). Furthermore weather phenomena are cyclic in nature (consider the alternation of land and see breeze) and we expect the discovered patterns to exhibit such a cyclic nature. Nevertheless, the resulting prototypes clearly deviate from the sinusoidal prototypes obtained from standard k-means/FCM clustering. For instance, the rise and fall of wind strength have very different slopes in the various prototypes, some patterns contain long periods of still air, etc. The resulting prototypes clearly differ from those obtained from the other datasets.

6 Conclusions

The distance function used by a clustering algorithm should always be adapted carefully with respect to the problem at hand. If time series data has to be clustered and no dilational effects are expected, cross correlation appears to be a promising candidate. For such a situation, we have shown how the objective function-based clustering algorithms k-means/fuzzy c-means have to be modified in order to operate with this distance. The prototype update step of k-means is revised to handle the alignment of time series appropriately. The negative influence of outliers or data series that poorly correlate to any of the clusters is reduced by means of a noise cluster.

The resulting cross-correlation clustering (CCC) algorithm solves the problem of clustering unaligned time series. It can be applied to short time series (whole series clustering), but also to time series subsequence (STS) clustering. This is particularly interesting because most standard clustering algorithms fail with STS clustering.

References

1. A. Asuncion and D. Newman. UCI machine learning repository http://www.ics.uci.edu/∼mlearn/MLRepository.html, 2007.
2. M. R. Berthold and F. Höppner. On clustering time series using euclidean distance and pearson correlation. Technical report, University of Konstanz, 2008.
3. J. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms.* Plenum Press, New York, 1981.
4. J. Bezdek, J. Keller, R. Krishnapuram, and N. Pal. *Fuzzy Models and Algorithms for Pattern Recognition and Image Processing.* Kluwer, Boston, 1999.
5. J. R. Chen. Useful clustering outcomes from meaningful time series clustering. In *AusDM '07: Proceedings of the sixth Australasian conference on Data mining and analytics*, pages 101–109, Darlinghurst, Australia, Australia, 2007. Australian Computer Society, Inc.
6. G. Das, K.-I. Lin, H. Mannila, G. Renganathan, and P. Smyth. Rule discovery from time series. In *Proc. of the 4th ACM SIGKDD Int. Conf. on Knowl. Discovery and Data Mining*, pages 16–22. AAAI Press, 1998.

7. R. Davé. Characterization and detection of noise in clustering. *Pattern Recognition Letters*, 12:657–664, 1991.
8. O. Georgieva and F. Klawonn. Dynamic data assigning assessment clustering of streaming data. *Applied Soft Computing*, 8:1305–1313, 2008.
9. F. Höppner. Time series abstraction methods – a survey. In *Proceedings GI Jahrestagung Informatik, Workshop on Knowl. Discovery in Databases*, Lecture Notes in Informatics, pages 777–786, Dortmund, Germany, Sept. 2002.
10. F. Höppner, F. Klawonn, R. Kruse, and T. Runkler. *Fuzzy cluster analysis.* Wiley, Chichester, 1999.
11. T. Ide. Why does subsequence time-series clustering produce sine waves? In *Proc. Int. Conf. Knowledge Discovery in Databases (PKDD)*, volume 4213 of *LNCS*, pages 211–222, 2006.
12. E. Keogh and S. Kasetty. On the need for time series data mining benchmarks: A survey and empirical demonstration. *Data Mining and Knowledge Discovery*, 7(4):349–371, 2003.
13. E. Keogh, J. Lin, and W. Truppel. Clustering of time series subsequences is meaningless: implications for previous and future research. In *Proc. IEEE Int Conf on Data Mining (ICDM)*, pages 115– 122, 2003.
14. E. Keogh, X. Xi, L. Wei, and C. A. Ratanamahatana. The UCR time series classification/clustering homepage www.cs.ucr.edu/∼eamonn/time_series_data/, 2006.
15. F. Klawonn. Fuzzy clustering: Insights and a new approach. *Mathware and Soft Computing*, 11:125–142, 2004.