# An Effective Algorithm for Clustering Time-Series Data by Fuzzy Logic

[1]Sadiq Hussain, [2]Prof. G.C. Hazarika
[1]System Administrator, Dibrugarh University, Assam, India
[2] Director i/c, Centre for Computer Studies, Dibrugarh University, Assam, India

**Abstract**
A variety of techniques currently exist for measuring the similarity between time series datasets. Of these techniques, the methods whose matching criteria are bounded by a specified $\epsilon$ threshold value, such as the LCSS technique, have been shown to be robust in the presence of noise, time shifts, and data scaling. Afterwards, by utilizing an efficient method, clusters are updated incrementally and periodically through a set of fuzzy approaches. So, in this paper, we utilize the clustering of time series data using LCSS and Fuzzy Logic. In addition, we will present the benefits of the proposed system by implementing a real application: using dataset that contains two attributes temperature and humidity.

**Keywords**
Time Series, Data mining, clustering, Fuzzy clustering

## I. Introduction

Techniques for evaluating the similarity between time series data-sets have long been of interest to the database community. New location-based applications that generate time series location trails (called trajectories) have also fueled interest in this topic since time series similarity methods can be used for computing trajectory similarity. One of the critical research issues with time series analysis is the choice of distance function to capture the notion of similarity between two sequences. Past research in this area has produced a number of distance measures, which can be divided into two classes. The first class includes functions based on the L1 and L2 norms. Examples of functions in this class are Dynamic Time Warping (DTW) [1] and Edit Distance with Real Penalty (ERP) [2]. The second class of distance functions includes methods that compute a similarity score based on a matching threshold $\epsilon$ . Examples of this class of functions are the Longest Common Subsequence (LCSS) [19], and the Edit Distance on Real Sequence (EDR) [3]. Previous research [3, 19] has demonstrated that this second class of methods is robust in the presence of noise and time shifting. All of the advanced similarity techniques mentioned above rely on dynamic programming for their evaluation. Dynamic programming requires that each element of one time series be compared with each element of the other; this evaluation is slow. The research community has thus developed indexing techniques such as [3, 5, 8, 11, 20] that use an index to quickly produce a superset of the desired results. However, these indexing techniques still require a refinement step that must perform the dynamic programming evaluation on elements of the superset. Furthermore, time series clustering has also been studied [3, 9, 19], and these clustering techniques require pair wise comparison of all time series in the dataset, which means that indexing methods cannot be used to speed up clustering applications.

To address this problem, a number of techniques have been developed that impose restrictions on the warping length of the dynamic programming evaluation. The Sakoe-Chiba band, studied in [16], uses a sliding window of fixed length to narrow the number of elements that are compared between two time series. The Itakura Parallelogram, studied in [6], also limits the number of comparisons to accomplish a similar effect as the Sakoe-Chiba band. These techniques that constrain the warping factor are faster, but at the expense of ignoring sequence matches that fall outside of the sliding window. If the best sequence match between two time series falls outside of the restricted search area, then these techniques will not find it.

## II. Related Work

There are several existing techniques for measuring the similarity between different time series. The Euclidean measure sums the Euclidean distance between points in each time series. For example, in two dimensions the Euclidean distance is computed as: $\sqrt{\sum_{i=1}^{n}((r_{i,x}-s_{i,x})^2+(r_{i,y}-s_{i,y})^2)}$.
This measure can be used only if the two time series are of equal length, or if some length normalization technique is applied. More sophisticated similarity measures include Dynamic Time Warping (DTW) [1], Edit distance with Real Penalty (ERP) [2], the Longest Common Subsequence (LCSS) [30], and Edit Distance on Real sequences (EDR) [3]. These measures are summarized in Table 2. DTW was first introduced to the database community in [1]. DTW between two time series does not require the two series to be of the same length, and it allows for time shifting between the two time series by repeating elements. ERP [2] creates g, a constant value for the cost of a gap in the time series, and uses the L1 distance norm as the cost between elements. The LCSS technique introduces a threshold value, $\epsilon$, that allows the scoring technique to handle noise. If two data elements are within a distance of $\epsilon$ in each dimension, then the two elements are considered to match, and are given a match reward of 1. If they exceed the $\epsilon$ threshold in some dimension, then they fail to match, and no reward is issued. The EDR [3] technique uses gap and mismatch penalties. It also seeks to minimize the score (so that a score closer to 0 represents a better match). In [1], the authors use the Euclidean distance to measure similarity in time series datasets. The Discrete Fourier Transform is used to produce features that are then indexed in an R-tree. Dimensionality reduction is also studied in [8, 10, 13, 14, 20]. Indexing is also studied in [3], which proposes a generic method built around lower bounding to guarantee no false dismissals. Indexing methods for DTW have been the focus of several papers including [7, 12, 17, 21, 22]. Indexing for LCSS [18] and EDR [3] has also been studied. In this paper, our focus is not on specific indexing methods, but on the design of robust similarity measures, and efficient evaluation of the similarity function. We note that our work is complementary to these indexing methods, since the indexing methods still need to perform a refinement step that must evaluate the similarity function. Traditionally, previous work has not focused on this refinement cost, which can be substantial. Previous works employ a dynamic programming (DP) method

for evaluating the similarity function, which is expensive, especially for long sequences. The Sakoe-Chiba Band [16] and Itakura Parallelogram [6] are both estimation techniques for restricting the amount of time warping to estimate the DTW score between two sequences. A restriction technique similar to the Sakoe-Chiba Band is described for LCSS in [19] and the R-K Band estimate is described in [15]. Time series may be clustered using compression techniques [4,10]. We do not compare our algorithms with these techniques because of their inapplicability for clustering short time series.

## III. 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 [28].

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 [24, 31]. 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. 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 [23, 25, 26, 27]. 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 δ = 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 [30], 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 = {($a_{x1}$, $a_{y1}$), ..., ($a_{xm}$, $a_{ym}$)} and B = {($b_{x1}$, $b_{y1}$), ..., ($b_{xn}$, $b_{yn}$)}. $a_{xi}$ and $b_{xi}$ are the ith value of the numeric time series of A and B respectively. ayi and byi are the ith value of the symbolic time series of A and B respectively. Let Equal(i, j) be the matching function between the ith point of A and the jth point of B, where:

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

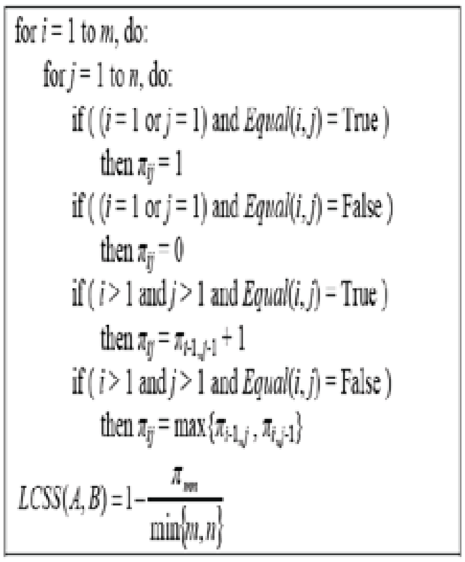
Given the recursive function Sim(i, j) that compute the similarity between the subsequence $A_i$ = {($a_{x1}$, $a_{y1}$), ..., ($a_{xi}$, $a_{yi}$)} and the subsequence $B_j$ = {($b_{x1}$, $b_{y1}$), ..., ($b_{xj}$,$b_{yj}$)} as follows:

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

We can now define our LCSS distance as follow:

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

Table 1: The non recursive alogrithm that computes the LCSS distence between two bi-dimensional and heterogeneous time series A and B.



```
for $i = 1$ to $m$, do:
    for $j = 1$ to $n$, do:
        if ( ($i = 1$ or $j = 1$) and $Equal(i, j)$ = True )
            then $\pi_{ij} = 1$
        if ( ($i = 1$ or $j = 1$) and $Equal(i, j)$ = False )
            then $\pi_{ij} = 0$
        if ( $i > 1$ and $j > 1$ and $Equal(i, j)$ = True )
            then $\pi_{ij} = \pi_{i-1,j-1} + 1$
        if ( $i > 1$ and $j > 1$ and $Equal(i, j)$ = False )
            then $\pi_{ij} = \max\{\pi_{i-1,j}, \pi_{i,j-1}\}$
$LCSS(A, B) = 1 - \frac{\pi_{mn}}{\min\{m,n\}}$
```

It's important to notice that LCSS is not a metric distance. Therefore it does not necessary respect the triangular inequality LCSS(A ,B) ≤ LCSS (A, C) +LCSS(C, B). The recursive definition of our LCSS distance has a non computational complexity and needs a dynamic programming approach. Dynamic programming consists in creating a m×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. 1.

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).

So the longer step of this algorithm is the completion of the cumulative similarity matrix. This step has a complexity of O(m×n) (O(m2) 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×δ) time (with δ << n).

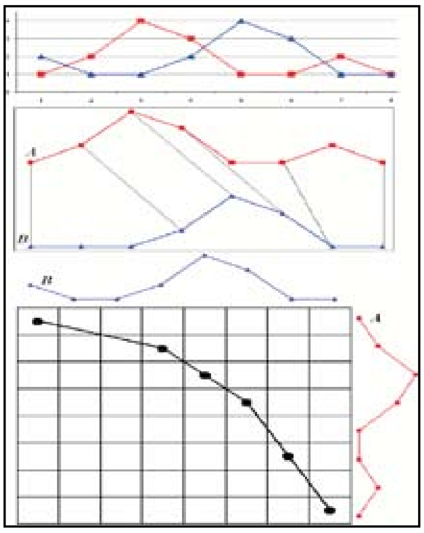


Fig. 1: Example of best warping path between two time series A={1,2,4,3,1,1,2,1} and B={1,1,1,2,4,3,1,1} The sequence of cells crossing by the best warping path is (1,1),(2,4),(3,4),(4,6),(6,7) and (8,8).

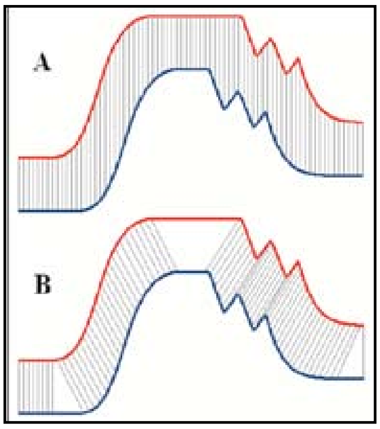


Fig. 2: Computation of the distance between two time series with Euclidean distance fig. (a) and LCSS distance fig. (b)

## IV. Type-1 Fuzzy Logic System (Type-1 FLS)

A combination of Type-1 Fuzzy Logic System (Type-1 FLS) and Gradient descent algorithm is called Type-1 FLS*. Fuzzy logic was developed by Lotfi Zadeh a professor at the University of California, Berkley. Fuzzy system is useful for real world problems where there are different kinds of uncertainty [32]. The idea of fuzzy logic was to show that there is a world behind conventional logic. This kind of logic is the proper way to model human thinking. Fuzzy logic is recently getting the attention of artificial intelligence researchers. It is being used to build expert systems for handling ambiguities and vagueness associated with real world problems. Fig. 3, shows the architecture of Type-1 Fuzzy system.

### A. Gradient descent Algorithm

Gradient descent technique is a training algorithm which was used to tune the membership function parameters in fuzzy system. Using this algorithm the membership function parameters can be optimized and the error rate is reduced to get more accurate results. The details of the algorithm were explained in [33].

### B. Fuzzification Process

According to [34] fuzzifying has two meanings. The first is the process fining the fuzzy value of a crisp one. The second is finding the grade of membership of a linguistic value of a linguistic variable corresponding to a fuzzy or scalar input. The most used meaning is the second. Fuzzification is done by membership functions.

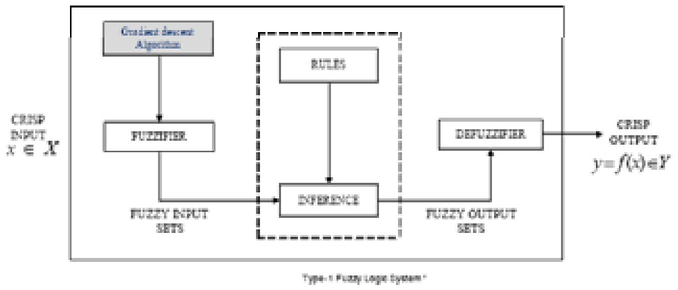


Fig. 3: Structure of Type-1 Fuzzy Logic system*

### C. Inference Process

The next step is the inference process which involves deriving conclusions from existing data [34]. The inference process defines a mapping from input fuzzy sets into output fuzzy sets. It determines the degree to which the antecedent is satisfied for each rule. These results in one fuzzy set assigned to each output variable for each rule. MIN is an inference method. According to [35] MIN assigns the minimum of antecedent terms to the matching degree of the rule. Then fuzzy sets that represent the output of each rule are combined to form a single fuzzy set. The composition is done by applying MAX which corresponds to applying fuzzy logic OR, or SUM composition methods [34].

### D. Defuzzification Process

Defuzzification is the process of converting fuzzy output sets to crisp values [34]. According to [36] there are three defuzzification methods used : Centroid, Average Maximum and Weighted Average. Centroid method of Defuzzification is the most commonly used method. Using this method the defuzzified value is defined by:

$$\text{Centroid}=\frac{\int x\mu(x)dx}{\int \mu(x)dx}$$

Where $\mu(x)$ is the aggregated output member function. The details of Fuzzy system have been explained in [34] [37].

## V. 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

ISSN : 0976-8491 (Online) | ISSN : 2229-4333(Print)

Clustering clusters. The intuition behind k- Means approach is shown in Table 2.

Table 2: K-Means algorithm

| | |
|---|---|
| 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. |

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].

## VI. 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. 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 can 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×δ)).

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 intraclass variance to evaluate the quality of the seeds initialization: if a seed initialization does not give an intraclass variance inferior to a threshold, then another initialization is tried.

### A. The Algorithm

Let $X$ be a set of $N$ time series that we aim to split in $k$ clusters, where $X = \{x_1,\ldots,x_n\}$.

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.

#### 1. Fuzzification Process using Type-1 FLS

Given $L(c, S_j)$, the distance between the time series $x_i$ and the seed $S_i$ as follows:

$$L(x_i, S_j) = \frac{1}{c}\sum_{y \in S_j} LCSS(x_i, y)$$

```
For $i = 1,\ldots, N$ do:
    For $j = 1,\ldots, k$ do:
        Compute $L(x_i, S_i)$
```

Assign to each time series $x_i$ its nearest seed (i.e. the seed $S_i$ that minimize $L(x_i, S_i)$).

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.

#### 2. Inference Process

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:

$$Inertia(x_i, C_j) = \sum_{y \in C_j} LCSS(x_i, y)$$

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 $S = S'$.

### B. Use Z-test also for fitness of Clusters

#### 1. Defuzzication Process

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. 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.

## VII. Results and Discussion

Table 3: The Number of Dataset instances with and without noise

| Attribute | Number of instances with noise | Number of noisy instances | Number of common noisy instances | Total noise | Number of instances without noise |
|---|---|---|---|---|---|
| Temperature 1 | 4600 | 62 | 15 | 90 | 4510 |
| Temperature 2 | 4600 | 43 | | | 4510 |

The dataset for preprocessing is taken from Cetre for Atmospheric Studies, Dibrugarh University that contains two attributes, temperature and humidity. These measures have been taken from a sensor and recorded by a computer at regular intervals of about 15 minutes (the average interval was estimated at 896 seconds) [38].

The dataset possesses 4600 instances. Based on the equation coded in the Matlab, the noise was found and removed from the dataset. The graphic view of the dataset before removing noise is depicted in fig. below.

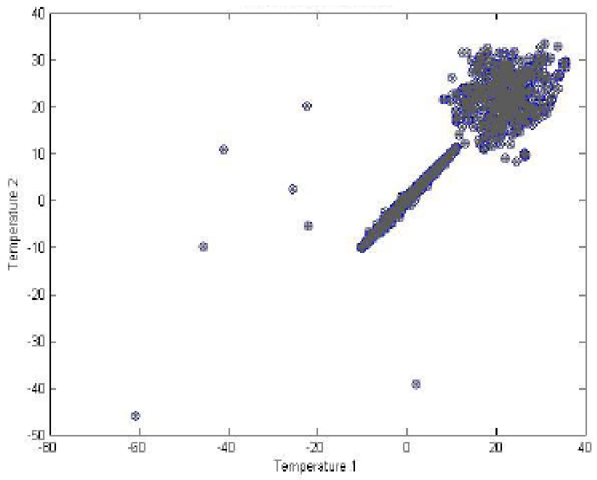


Fig. 4: Graphical View of the Original Dataset

As shown in fig 4, the attributes (temperature 1 and 2) were derived from the device sensor. The extracted data may include noisy data due to device and measurement errors. This would decrease the quality of the data. Therefore, the program in this study was applied based on the definition of the statistic equation on the data to remove the noisy data. The characteristics and results are shown in Table 3.

Table 3, shows that the number of the noisy data in the attribute Temperature 1 is 62 and this is 43 for the attribute Temperature 2; 15 noise instances are common among the attributes Temperature 1 and Temperature 2, and thus the total instances for two attributes after removing noise is 4510.

Table 4: Summary of the Characteristics of the dataset

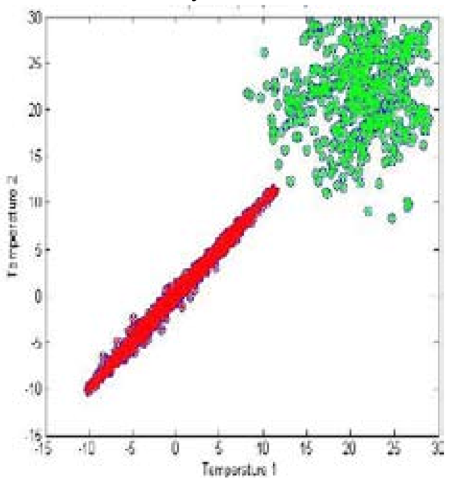


Fig. 5: Clustering the Dataset using the Algorithm

As shown in Table 4, are characteristics of dataset. The dataset has 2 clusters; cluster 1 contains 566 records and cluster 2 has 3944 records. Based on the two dimensions, the centre of cluster 1: Temperature1=20.9173 and Temperature2=21.4507; likewise for cluster 2: Temperature1=0.7194and Temperature2=0.7179.

## References

[1] D. Berndt, J. Clifford, "Using Dynamic Time Warping to Find Patterns in Time Series", In AAAI-94 Workshop on Knowledge Discovery in Databases, pp.359–370, 1994.

[2 ] Chen, R. Ng, "On the Marriage of Lp-norms and Edit Distance", In VLDB, pp. 792–803, 2004.

[3] L. Chen, M. T. Ozsu, V. Oria, "Robust and Fast Similarity Search for Moving Object Trajectories", In SIGMOD, pp. 491–502, 2005.

[4] R. Cilibrasi, P. M. B. Vitanyi, "Clustering by Compression",IEEE Transactions on Information Theory, 51(4): pp. 1523–1545, 2005.

[5] C. Faloutsos, M. Ranganathan, Y. Manolopoulos, "Fast Subsequence Matching in Time-Series Databases", In SIGMOD, pp. 419–429, 1994.

[6] F. Itakura, "Minimum Prediction Residual Principle Applied to Speech Recognition", IEEE Trans. Acoustics, Speech, and Signal Proc., Vol. ASSP-23: pp. 52–72, 1975.

[7] E. Keogh, "Exact Indexing of Dynamic Time Warping", In VLDB, pp. 406–417, 2002.

[8] E. Keogh, K. Chakrabarti, M. Pazzani, S. Mehrotra, "Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases", KAIS, 3(3): pp. 263–286, 2000.

[9] E. Keogh, S. Kasetty, "The Need for Time Series Data Mining Benchmarks", A Survey and Empirical Demonstration. In SIGKDD, pp. 102–111, 2002.

[10] E. Keogh, S. Lonardi, C. A. Ratanamahatana, "Towards Parameter-Free Data Mining" , In SIGKDD, pp. 206–215, 2004.

[11] E. J. Keogh, K. Chakrabarti, S. Mehrotra, M. J. Pazzani, "Locally Adaptive Dimensionality Reduction for Indexing Large Time Series Databases", In SIGMOD, pp. 151–162, 2001.

[12] S.W. Kim, S. Park, W. W. Chu, "An Index-Based Approach for Similarity Search Supporting Time Warping in Large

Sequence Databases", In ICDE, pp. 607–614, 2001.
[13] Korn, H. Jagadish, C. Faloutsos, "Efficiently Supporting Ad Hoc Queries in Large Datasets of Time Sequences", In SIGMOD, pp. 289–300, 1997.
[14] Popivanov, R. Miller, "Similarity Search Over Time Series Data Using Wavelets", In ICDE, pp. 212, 2001.
[15] C. Ratanamahatana, E. Keogh, "Making Time-series Classification More Accurate Using Learned Constraints", In SIAM International Conference on Data Mining, 2004.
[16] H. Sakoe, S. Chiba, "Dynamic Programming Algorithm Optimization for Spoken Word Recognition", IEEE Trans. Acoustics, Speech, and Signal Proc., Vol. ASSP-26(1): pp. 43–49, 1978.
[17] Y. Sakurai, M. Yoshikawa, C. Faloutsos,"FTW: Fast Similarity Search under the Time Warping Distance", In PODS, pp. 326–337, 2005.
[18] M. Vlachos, M. Hadjieleftheriou, D. Gunopulos, E. Keogh, "Indexing Multi-Dimensional Time-Series with Support for Multiple Distance Measures", In SIGKDD, pp. 216–225, 2003.
[19] M. Vlachos, G. Kollios, D. Gunopulos, "Discovering Similar Multidimensional Trajectories. In ICDE, pp. 673–684, 2002.
[20] B.-K. Yi, C. Faloutsos, "Fast Time Sequence Indexing for Arbitrary Lp Norms. In VLDB, pp. 385–394, 2000.
[21] B.-K. Yi, H. V. Jagadish, C. Faloutsos, "Efficient Retrieval of Similar Time Sequences Under Time Warping", In ICDE, pp. 201–208, 1998.
[22] Y. Zhu, D. Shasha, "Warping Indexes with Envelope Transforms for Query by Humming", In SIGMOD, pp. 181–192, 2003.
[23] R. Agrawal, K. Lin, H. S. Sawhney, K. Shim, "Fast Similarity Search in the Presence of Noise, Scaling and Translation in Time-Series Databases", In Proc. of VLDB, pp. 490-501, 1995.
[24] D. J. Berndt, J. Clifford, "Using Dynamic Time Warping to Find Patterns in Time Series", In Proc. Of KDD Workshop, 1994.
[25] B. Bollobas, G. Das, D. Gunopulos, H. Mannila, "Time-Series Similarity Problems and Well-Separated Geometric Sets", In Proc. of the 13th SCG, Nice, France, 1997.
[26] T. Bozkaya, N. Yazdani and M. Ozsoyoglu, "Matching and Indexing Sequences of Different Lengths," In Proc. of the CIKM, Las Vegas, USA, 1997.
[27] 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.
[28] E. Keogh, 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, pp. 285-289, 2000.
[29] 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, pp. 281-297, 1967.
[30] Y. Qu, C. Wang, X. Wang, "Supporting Fast Search in Time Series for Movement Patterns in Multiple Scales", In Proc. of the ACM CIKM, pp. 251-258, 1998.
[31] C. A. Ratanamahatana, 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, pp. 11-22, 2004.
[32] Zadeh, L.A., "Outline of a new approach to the analysis of complex systems and decision processes", IEEE Transactions on Systems, Man, and Cybernetics, 1973. 3: pp. 28-44.
[33] Li-Xin, W., "A course in fuzzy systems and control", prentice Hall, 1997.
[34] Siler, W., J.J. Buckley, "Fuzzy expert systems and fuzzy reasoning", 2005: Wiley-Interscience.
[35] Hwang, G.J., S.T. Huang, "New environment for developing fuzzy expert systems", J INF SCIENG, 1999. 15(1): pp. 53-69.
[36] [Online] Availabe: http://www.jimbrule.com/fuzzytutorial.html. Last accessed 4 November 2006.
[37] [Online] Availabe: http://www.urc.bl.ac.yu/manuals/adv/fuzzy. Lased accessed 3 November 2006.
[38] Mencattini, A., et al, "Local meteorological forecasting by type-2 fuzzy systems time series prediction", 2005.

Sadiq Hussain MCA from Tezpur University, Assam,India in the year 2000 with CGPA 7.85. Currently, he is working as System Administrator of Dibrugarh University. He is in this position since December, 2008. He is in the charge of Computerization of Examination System and MIS of Dibrugarh University. He has published 6 papers with Prof. G.C. Hazarika in international journals and referred conference proceedings.

Prof. G.C. Hazarika Positions held: Director i/c, Centre for Computer Studies, Dibrugarh University, and Professor, Department of Mathematics, Dibrugarh University Academic Positions held: Computer Programmer: Joined as Computer Programmer, Dibrugarh University Computer Centre in Dec, 1977 and served till April, 1985.Lecturer: Joined as Lecturer in the Department of Mathematics, Dibrugarh University in April, 1985. Reader: Joined as Reader in a regular post in June, 1990. Professor: Joined as Professor in a regular post in August, 1998.

He has guided 11 Ph. D students and 9 M Phil students.