COMPUTING CLUSTER CENTERS OF TRAPEZOIDAL FUZZY NUMBERS THROUGH FUZZY C MEANS AND KERNEL BASED FUZZY C MEANS CLUSTERING ALGORITHMS WITH TWO METRIC DISTANCES USING MATLAB

S. Sreenivasan
Research Scholar, School of Advanced Sciences, VIT University, Chennai Campus, Vandalur-Kelambakkam Road, Chennai, Tamil Nadu, India

B. J. Balamurugan
School of Advanced Sciences, VIT University, Chennai Campus, Vandalur-Kelambakkam Road, Chennai, Tamil Nadu, India

ABSTRACT

Fuzzy number type data is a typical class of fuzzy data and it can be regarded as a general form of the interval data and crisp data. In this paper we compute cluster centers of trapezoidal fuzzy numbers through fuzzy c means clustering algorithm and kernel based fuzzy c means clustering algorithm. Two novel distances between the trapezoidal fuzzy numbers are used and the distances are complete metric on the set of trapezoidal fuzzy numbers. A set of thirty trapezoidal fuzzy numbers and a set with the same thirty trapezoidal fuzzy numbers and one more trapezoidal fuzzy number called outlier or noisy point are taken and run the two fuzzy c means algorithms to find the cluster center using MATLAB programming with two metric distances.

Key words: Fuzzy clustering, Kernel function, Trapezoidal fuzzy numbers, Fuzzy C number clustering algorithms.


1. INTRODUCTION

In the literature of fuzzy clustering, fuzzy C means (FCM) clustering procedure proposed by Dunn [3] and extended by Bezdek [1] are most used and discussed. Based on the similar idea of FCM construction, Yang and Ko [4] proposed new type of fuzzy clustering procedure for handling fuzzy data called fuzzy C numbers (FCN) clustering procedure. These FCNs could
well be used for clustering fuzzy data, especially for LR type triangular, trapezoidal and normal fuzzy numbers. However, FCN has the same drawbacks as FCM. It is powerfully influenced by outliers and often gives bad clustering results when the data set includes large numbers of essential cluster sample sizes. A good clustering algorithm should be strong and able to endure situations that may often occur in real life applications.

Hung and Yang [5] provided alternative fuzzy C numbers (AFCN) clustering algorithm for LR type fuzzy numbers based on exponential type distance measure. Based on the concept of robust statistics, the AFCN algorithm improves on the weakness found in FCN.

In this paper we use two novel distances for LR type fuzzy numbers and discuss the robustness of the fuzzy C means type algorithms with these complete metric distances.

Section 2 describes the definition of triangular fuzzy number and the distances $d_{TI}$ and $D_{p,q}$ between two trapezoidal intervals which are complete metric on the set of trapezoidal fuzzy intervals. Two clustering algorithms on the space of all trapezoidal fuzzy intervals are described in Section 3 based on the distances $d_{TI}$ and $D_{p,q}$. Numerical examples are given and comparisons are made between the two algorithms in Section 4. Conclusions will be stated in Section 5.

2. PRELIMINARIES

In this section we refer the following important definitions of trapezoidal fuzzy number[2], and the distances $d_{TI}$[8] and $D_{p,q}$[8]

Definition 1: $X = (m_1, m_2, \alpha, \beta)_{LR}$ with $\alpha > 0, \beta > 0$ is called a LR-type fuzzy number if

$$X(x) = \begin{cases} 
  L \left( \frac{m_1 - x}{\alpha} \right) & \text{for } x \leq m_1 \\
  1 & \text{for } m_1 \leq x \leq m_2 \\
  R \left( \frac{x - m_2}{\beta} \right) & \text{for } x \geq m_2 
\end{cases}$$

We note that the trapezoidal type of fuzzy number is the simplest and most used one among LR-type fuzzy intervals. We shall consider FCN clustering for this special type of fuzzy intervals.

Let $X = (m_1, m_2, \alpha, \beta)_{LR}$ be a LR-type fuzzy interval. If $L$ and $R$ of the form $T(x) = (1 - x, 0 \leq x \leq 1, 0, \text{ Otherwise}$ then $X$ is called a trapezoidal fuzzy interval.

$$X(x) = \begin{cases} 
  1 - \frac{m_1 - x}{\alpha} & \text{for } x \leq m_1 \\
  1 & \text{for } m_1 \leq x \leq m_2 \\
  1 - \frac{x - m_2}{\beta} & \text{for } x \geq m_2 
\end{cases}$$

Let $F_{LR}(R)$ be the set of all LR-type fuzzy intervals. For any $X = (m_{1x}, m_{2x}, \alpha_x, \beta_x)_{LR}$ and $Y = (m_{1y}, m_{2y}, \alpha_y, \beta_y)_{LR}$ in $F_{LR}(R)$, we define a metric $d_f^2(x, y)$ as follows:

$$d_f^2(X, Y) = (m_{1x} - m_{1y})^2 + (m_{2x} - m_{2y})^2 + \left( (m_{1x} - l\alpha_x) - (m_{1y} - l\alpha_y) \right)^2$$
$$+ \left( (m_{2x} + r\beta_x) - (m_{2y} + r\beta_y) \right)^2$$

Where $l = \int_0^1 L^{-1}(\omega)d\omega$ and $r = \int_0^1 R^{-1}(\omega)d\omega$. 

http://www.iaeme.com/IJCIET/index.asp 1323 editor@iaeme.com
Yang and Co[8] proved the following theorems.

**Theorem 1:** \((F_{LR}(R), d_i)\) is metric.

**Theorem 2:** \((F_{LR}(R), d_i)\) is complete.

Based on the distance \(d_i\) defined on \(F(R)\), we define a distance \(d_{T1}(X, Y)\) for any two trapezoidal fuzzy intervals \(X = (m_{1x}, m_{2x}, \alpha_x, \beta_x)_{T1}\) and \(Y = (m_{1y}, m_{2y}, \alpha_y, \beta_y)_{T1}\) in the space \(F_{T1}(R)\) of all trapezoidal fuzzy interval as follows:

\[
d^2_{T1}(X, Y) = (m_{1x} - m_{1y})^2 + (m_{2x} - m_{2y})^2 + \left(\frac{m_{1x} - \alpha_x}{2} - \frac{m_{1y} - \alpha_y}{2}\right)^2
\]

\[
+ \left(\frac{m_{2x} + \beta_x}{2} - \frac{m_{2y} + \beta_y}{2}\right)^2
\]

The \(D_{p,q}\) distance indexed by the parameters \(1 < p < \infty, 1 < q < \infty\), between two fuzzy numbers is a non negative integer. If there is no reason for distinguishing any side to the fuzzy numbers, \(D_{\frac{1}{2},\frac{1}{2}}\) is recommended. For trapezoidal fuzzy numbers \(X = (m_{1x}, m_{2x}, \alpha_x, \beta_x)_{LR}\) and \(Y = (m_{1y}, m_{2y}, \alpha_y, \beta_y)_{LR}\):

\[
D^2_{\frac{1}{2},\frac{1}{2}}(X, Y) = \frac{1}{6}\left\{ (m_{1y} - m_{1x})^2 + (m_{2y} - m_{2x})^2 + (\alpha_y - \alpha_x)^2 + (\beta_y - \beta_x)^2
\]

\[
+ (m_{1y} - m_{1x})(\alpha_y - \alpha_x) + (m_{2y} - m_{2x})(\beta_y - \beta_x)\right\}
\]

Yang and Co[8] proved the following theorems.

**Theorem 3:** \((F_{LR}(R), D_{p,q})\) is metric.

**Theorem 4:** \((F_{LR}(R), D_{p,q})\) is complete.

### 3. FUZZY CLUSTERING ALGORITHMS

In this section we recall fuzzy c means clustering algorithm[1] and kernel based fuzzy c means clustering algorithm[10,11].

Let \(X = \{\tilde{X}_1, \tilde{X}_2, \tilde{X}_3, \ldots, \tilde{X}_n\}\) be a given set of \(n\) fuzzy numbers in \(F(R)\) with \(\tilde{X}_k = (m_{1x_k}, m_{2x_k}, \alpha_{x_k}, \beta_{x_k})\), \(1 \leq k \leq n\). Let \(c\) be the number clusters. Let \(V = \{\tilde{V}_i | 1 < i \leq c\}\) is the set of centers, where \(\tilde{V}_i = (m_{1v_i}, m_{2v_i}, \alpha_{v_i}, \beta_{v_i})\), and \(d_{ik}\) the distance between \(\tilde{X}_k\) and \(\tilde{V}_i\).

#### 3.1. Fuzzy c means Clustering

The fuzzy c means algorithm partitions \(X\) into \(c\) fuzzy subsets by minimizing the following objective function:

\[
J_{FCM}(U, V) = \sum_{i=1}^{c} \sum_{k=1}^{n} u_{ik}^m d_{ik}^2
\]

where \(d_{ik} = d(\tilde{V}_i, \tilde{X}_k)\) and \(u_{ik}\) is the membership of \(\tilde{X}_k\) in class \(i\), satisfying

\[
\sum_{i=1}^{c} u_{ik} = 1
\]

\(m \in [1, \infty)\) is the fuzziness index and \(U = (u_{ik})_{n \times c}\) is a fuzzy c partition matrix.

The Parameters of FCM are estimated by updating min \(J\) step by step according to the formulas below:
Computing Cluster Centers of Trapezoidal Fuzzy Numbers Through Fuzzy C Means and Kernel Based Fuzzy C Means Clustering Algorithms with Two Metric Distances Using Matlab

\[
m_{1vi} = \frac{\sum_{k=1}^{n} u_{ik}^m m_{1x_k}}{\sum_{k=1}^{n} u_{ik}^m} \quad (1)
\]

\[
m_{2vi} = \frac{\sum_{k=1}^{n} u_{ik}^m m_{2x_k}}{\sum_{k=1}^{n} u_{ik}^m} \quad (2)
\]

\[
\alpha_{vi} = \frac{\sum_{k=1}^{n} u_{ik}^m \alpha_{x_k}}{\sum_{k=1}^{n} u_{ik}^m} \quad (3)
\]

\[
\beta_{vi} = \frac{\sum_{k=1}^{n} u_{ik}^m \beta_{x_k}}{\sum_{k=1}^{n} u_{ik}^m} \quad (4)
\]

\[
u_{ik} = \frac{-2}{d_{ik}^{-2}} \sum_{j=1}^{c} d_{jk}^{-2} \quad (5)
\]

Based on these formulas we give a fuzzy c means clustering algorithm for triangular fuzzy numbers.

**Algorithm-1**

Step1: Fix \( m > 1 \), fix \( c \in \{2, 3, 4, \ldots, (n - 1)\} \) and fix any \( \epsilon > 0 \). Choose an initial fuzzy c partition matrix \( U^{(0)} \) and let \( t = 0 \).

Step2: Calculate cluster centers \( V^{(t)} = \{\bar{v}_i^{(t)} | 1 < i \leq c\} \) using \( U^{(t)} \) and equations (1), (2), (3) and (4).

Step3: Update \( U^{(t)} \) by \( U^{(t+1)} \) using \( V^{(t)} \) and equation (5).

Step4: Compute \( E^k = max_{i,k} \left\{ \left| u_{ik}^{(t+1)} - u_{ik}^{(t)} \right| \right\} \), if \( E^k \leq \epsilon \), stop. Otherwise set \( U^{(t+1)} = U^{(t)} \) and go to step 2.

**3.2. Kernel Fuzzy c means Clustering**

A kernel function[4,6,7] is a generalization of the distance metric that measures the distance between two data points as the data points are mapped into a high dimensional space in which they are more clearly separable. By employing a mapping function \( \Phi(\vec{x}) \), which defines a non linear transformation: \( \vec{x} \rightarrow \Phi(\vec{x}) \), the non linearity separable data structure existing in the original data space can possibly be mapped into a linearly separable case in the higher dimensional feature space.

Given the unlabeled data set \( X = \{\vec{x}_1, \vec{x}_2, \vec{x}_3, \ldots, \vec{x}_n\} \) in the \( M \)-dimensional space \( R^M \), let \( \Phi \) be a non linear mapping function from this input space to a high dimensional feature space \( H \):

\[ \Phi: R^M \rightarrow H, \vec{x} \rightarrow \Phi(\vec{x}) \]

The dot product in the high dimensional feature space can be calculated through the kernel function \( K(\vec{x}_i, \vec{x}_j) = \Phi(\vec{x}_i) \cdot \Phi(\vec{x}_j) \)

Examples of kernel function

http://www.iaeme.com/IJCIET/index.asp 1325  editor@iaeme.com
• Linear: $K(\bar{x}_i, \bar{x}_j) = \bar{x}_i^T \bar{x}_j$

• Polynomial: $K(\bar{x}_i, \bar{x}_j) = (y \bar{x}_i, \bar{x}_j + c)^d, y > 0, d \in \mathbb{N}$

• Sigmoid: $K(\bar{x}_i, \bar{x}_j) = \tanh(y \bar{x}_i, \bar{x}_j + c)^d, y > 0$

• Radial basis function (RBF): $K(\bar{x}_i, \bar{x}_j) = \exp(-y \|\bar{x}_i - \bar{x}_j\|^2), y > 0$

where $\gamma, c, d$ are kernel parameters. Since,

$$\|\Phi(\bar{x}_k) - \Phi(\bar{v}_i)\|^2 = (\Phi(\bar{x}_k) - \Phi(\bar{v}_i))^T (\Phi(\bar{x}_k) - \Phi(\bar{v}_i))$$

$$= \Phi(\bar{x}_k)^T \Phi(\bar{x}_k) - \Phi(\bar{x}_k)^T \Phi(\bar{v}_i) - \Phi(\bar{v}_i)^T \Phi(\bar{x}_k) + \Phi(\bar{v}_i)^T \Phi(\bar{v}_i)$$

$$= K(\bar{x}_k, \bar{x}_k) + K(\bar{v}_i, \bar{v}_i) - 2K(\bar{x}_k, \bar{v}_i)$$

when the kernel function is chosen as RBF, $K(\bar{x}_k, \bar{x}_k) = 1, K(\bar{v}_i, \bar{v}_i) = 1$, then

$$\|\Phi(\bar{x}_k) - \Phi(\bar{v}_i)\|^2 = 2(1 - K(\bar{x}_k, \bar{v}_i))$$

The KFCM algorithm modifies the objective function of FCM to

$$J_{KFCM}(U, V) = \sum_{i=1}^{c} \sum_{k=1}^{n} u_{ik}^m (1 - K(\bar{x}_k, \bar{v}_i))$$

The Parameters of KFCM are estimated by updating $\min J$ step by step according to the formulas below:

$$m_{1vi} = \frac{\sum_{k=1}^{n} u_{ik}^m K(\bar{x}_k, \bar{v}_i) m_{1xk}}{\sum_{k=1}^{n} u_{ik}^m K(\bar{x}_k, \bar{v}_i)}$$

$$m_{2vi} = \frac{\sum_{k=1}^{n} u_{ik}^m K(\bar{x}_k, \bar{v}_i) m_{2xk}}{\sum_{k=1}^{n} u_{ik}^m K(\bar{x}_k, \bar{v}_i)}$$

$$\alpha_{vi} = \frac{\sum_{k=1}^{n} u_{ik}^m K(\bar{x}_k, \bar{v}_i) \alpha_{vk}}{\sum_{k=1}^{n} u_{ik}^m K(\bar{x}_k, \bar{v}_i)}$$

$$\beta_{vi} = \frac{\sum_{k=1}^{n} u_{ik}^m K(\bar{x}_k, \bar{v}_i) \beta_{vk}}{\sum_{k=1}^{n} u_{ik}^m K(\bar{x}_k, \bar{v}_i)}$$

$$u_{ik} = \frac{(1 - K(\bar{x}_k, \bar{v}_i))^{-1}}{\sum_{j=1}^{c} (1 - K(\bar{x}_k, \bar{v}_j))^{-1}}$$

Based on these formulas we give a kernel fuzzy $c$ means clustering algorithm for triangular fuzzy numbers.

**Algorithm-2:**

Step1: Fix $m > 1$, fix $c \in \{2,3,4,\ldots,(n-1)\}$ and fix any $\epsilon > 0$. Choose an initial fuzzy $c$ partition matrix $U^{(0)}$ and let $t = 0$. 

http://www.iaeme.com/IJCIET/index.asp 1326 editor@iaeme.com
Computing Cluster Centers of Trapezoidal Fuzzy Numbers Through Fuzzy C Means and Kernel Based Fuzzy C Means Clustering Algorithms with Two Metric Distances Using Matlab

Step 2: Calculate cluster centers \( V^{(t)} = \{\tilde{v}_i^{(t)} | 1 < i \leq c\} \) using \( U^{(t)} \) and equations (6), (7), (8) and (9).

Step 3: Update \( U^{(t)} \) by \( U^{(t+1)} \) using \( V^{(t)} \) and equation (10).

Step 4: Compute \( E^k = \max_{i,k} \left\{ \left| u_{ik}^{(t+1)} - u_{ik}^{(t)} \right| \right\} \), if \( E^k \leq \epsilon \), stop. Otherwise set \( U^{(t+1)} = U^{(t)} \) and go to step 2.

In our experiment, the parameter \( \gamma \) in RBF kernel is defined by
\[
\gamma = \left( \frac{\sum_{k=1}^{n} d^2(\tilde{x}_k, \bar{W})}{n} \right)^{-1}
\]
with \( \bar{W} = (\bar{m}_1, \bar{m}_2, \bar{a}_w, \bar{\beta}_w) \) is the sample mean, where
\[
\begin{align*}
\bar{m}_1 &= \frac{\sum_{k=1}^{n} m_{1x_k}}{n} \\
\bar{m}_2 &= \frac{\sum_{k=1}^{n} m_{2x_k}}{n} \\
\bar{a}_w &= \frac{\sum_{k=1}^{n} a_{x_k}}{n} \\
\bar{\beta}_w &= \frac{\sum_{k=1}^{n} \beta_{x_k}}{n}
\end{align*}
\]

4. NUMERICAL EXAMPLES

In this section, we implement our algorithms with \( m = 2 \) and \( \epsilon = 0.001 \). Consider the data set \( G_1 \) from Yang, M.S and Ko, C.H. [8] consisting of 30 trapezoidal fuzzy numbers.

<table>
<thead>
<tr>
<th>S. No.</th>
<th>Trapezoidal Fuzzy Intervals ( X = (m_1, m_2, a, \beta) )</th>
<th>Memberships ( U = (u_1, u_2, u_3) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(3.34, 5.34, 1.46, 1.30)</td>
<td>(0.0454128, 0.164056, 0.781816)</td>
</tr>
<tr>
<td>2</td>
<td>(9.56, 11.36, 0.27, 1.00)</td>
<td>(0.011139, 0.047289, 0.941572)</td>
</tr>
<tr>
<td>3</td>
<td>(10.56, 13.79, 1.95, 1.93)</td>
<td>(0.004207, 0.020202, 0.975591)</td>
</tr>
<tr>
<td>4</td>
<td>(10.89, 13.24, 0.56, 1.17)</td>
<td>(0.003397, 0.016318, 0.980285)</td>
</tr>
<tr>
<td>5</td>
<td>(13.89, 15.25, 0.89, 0.88)</td>
<td>(0.001020, 0.006212, 0.992768)</td>
</tr>
<tr>
<td>6</td>
<td>(14.78, 16.34, 0.12, 1.21)</td>
<td>(0.007575, 0.040316, 0.953927)</td>
</tr>
<tr>
<td>7</td>
<td>(14.90, 16.89, 1.19, 0.41)</td>
<td>(0.006329, 0.044927, 0.948668)</td>
</tr>
<tr>
<td>8</td>
<td>(15.67, 17.02, 1.82, 0.90)</td>
<td>(0.009451, 0.071329, 0.919219)</td>
</tr>
<tr>
<td>9</td>
<td>(16.87, 17.54, 1.90, 1.85)</td>
<td>(0.018075, 0.158532, 0.823393)</td>
</tr>
<tr>
<td>10</td>
<td>(17.45, 18.14, 1.79, 1.95)</td>
<td>(0.023898, 0.234053, 0.742049)</td>
</tr>
<tr>
<td>11</td>
<td>(19.78, 22.38, 1.47, 0.42)</td>
<td>(0.032126, 0.733173, 0.234700)</td>
</tr>
<tr>
<td>12</td>
<td>(20.67, 23.57, 1.34, 1.10)</td>
<td>(0.022702, 0.868264, 0.109033)</td>
</tr>
<tr>
<td>13</td>
<td>(21.45, 23.67, 0.92, 1.60)</td>
<td>(0.016320, 0.918735, 0.064945)</td>
</tr>
<tr>
<td>14</td>
<td>(22.34, 24.57, 0.04, 1.58)</td>
<td>(0.006287, 0.975737, 0.017976)</td>
</tr>
<tr>
<td>15</td>
<td>(23.47, 25.47, 0.81, 0.51)</td>
<td>(0.001246, 0.995979, 0.002775)</td>
</tr>
<tr>
<td>16</td>
<td>(24.67, 25.25, 0.14, 1.09)</td>
<td>(0.002726, 0.992315, 0.004959)</td>
</tr>
<tr>
<td>17</td>
<td>(25.78, 27.88, 0.39, 1.51)</td>
<td>(0.020890, 0.957697, 0.021413)</td>
</tr>
<tr>
<td>18</td>
<td>(26.45, 28.34, 1.61, 0.92)</td>
<td>(0.029728, 0.942755, 0.027517)</td>
</tr>
<tr>
<td>19</td>
<td>(28.34, 30.56, 1.95, 0.12)</td>
<td>(0.118667, 0.820002, 0.061331)</td>
</tr>
<tr>
<td>20</td>
<td>(32.29, 35.01, 1.66, 1.64)</td>
<td>(0.585195, 0.348492, 0.066313)</td>
</tr>
<tr>
<td>21</td>
<td>(32.77, 34.67, 0.63, 0.47)</td>
<td>(0.592451, 0.342142, 0.065407)</td>
</tr>
<tr>
<td>22</td>
<td>(34.88, 36.89, 1.08, 0.66)</td>
<td>(0.824787, 0.141108, 0.034105)</td>
</tr>
<tr>
<td>23</td>
<td>(35.45, 37.87, 1.48, 1.26)</td>
<td>(0.887354, 0.089429, 0.023217)</td>
</tr>
</tbody>
</table>
We run both algorithm-1 and algorithm-2 on the data set $G_1$ with the two complete metric distances $d_{TL}$ and $D_{2,2}$ and with $c = 3$ using MATLAB 2007. The corresponding results are shown in Table 2 and the cluster centers are shown in Figure 3 and Figure 4.

By adding a point $(98.020, 100.000, 1.630, 0.710)$ to data set $G_1$ we can get another data set $G_2$. This point can be regarded as an outlier. Now we run both algorithm-1 and algorithm-2 on the data set $G_2$ with the two complete metric distances $d_{TL}$ and $D_{2,2}$ and with $c = 3$ using MATLAB. The corresponding results are shown in Table 3 and the cluster centers are shown in Figure 5 and Figure 6.

<table>
<thead>
<tr>
<th></th>
<th>$x$</th>
<th>$y$</th>
<th>$z$</th>
<th>$w$</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>(35.88, 37.89, 1.79, 0.16)</td>
<td>(0.891492, 0.086082, 0.022426)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>(38.88, 40.56, 0.66, 0.64)</td>
<td>(0.997580, 0.001830, 0.000590)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>(40.25, 41.78, 0.52, 1.71)</td>
<td>(0.996313, 0.002733, 0.000954)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>(40.47, 42.35, 1.95, 0.15)</td>
<td>(0.996267, 0.002767, 0.000966)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>(43.56, 45.79, 0.92, 0.63)</td>
<td>(0.936249, 0.045360, 0.018391)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>(43.98, 45.67, 1.74, 1.69)</td>
<td>(0.932316, 0.048068, 0.019619)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>(45.77, 47.56, 1.71, 0.79)</td>
<td>(0.892947, 0.074712, 0.032341)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 1 Data Set $G_1$

Figure 2 Data Set $G_2$

Figure 3 Cluster Centers in $G_1$ With Distance $d_{TL}$
Computing Cluster Centers of Trapezoidal Fuzzy Numbers Through Fuzzy C Means and Kernel Based Fuzzy C Means Clustering Algorithms with Two Metric Distances Using Matlab

Figure 4 Cluster Centers in $G_1$ with distance $D_{2,2}^1$

Figure 5 Cluster Centers in $G_2$ With Distance $d_{TI}$

Figure 6 Cluster Centers in $G_1$ with distance $D_{2,2}^1$

Table 2 Clustering centers in data set $G_1$

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>$m_{1w}, m_{2w}, \alpha_1, \beta_1$</th>
<th>With distance $d_{TI}^2(X, Y)$</th>
<th>With distance $D_{2,2}^1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm-1</td>
<td>$V_1$ (39.401, 41.332, 1.287, 0.875) (39.367, 41.300, 1.288, 0.873)</td>
<td>(39.401, 41.332, 1.287, 0.875) (39.367, 41.300, 1.288, 0.873)</td>
<td>$V_1$ (39.401, 41.332, 1.287, 0.875) (39.367, 41.300, 1.288, 0.873)</td>
</tr>
<tr>
<td></td>
<td>$V_2$ (23.971, 25.986, 0.905, 1.040) (23.893, 25.909, 0.903, 1.042)</td>
<td>(23.971, 25.986, 0.905, 1.040) (23.893, 25.909, 0.903, 1.042)</td>
<td>$V_2$ (23.971, 25.986, 0.905, 1.040) (23.893, 25.909, 0.903, 1.042)</td>
</tr>
<tr>
<td>Algorithm-2</td>
<td>$V_1$ (38.737, 40.659, 1.244, 0.845) (38.646, 40.571, 1.242, 0.845)</td>
<td>(38.737, 40.659, 1.244, 0.845) (38.646, 40.571, 1.242, 0.845)</td>
<td>$V_1$ (38.737, 40.659, 1.244, 0.845) (38.646, 40.571, 1.242, 0.845)</td>
</tr>
<tr>
<td></td>
<td>$V_2$ (23.855, 25.835, 0.829, 1.063) (23.730, 25.714, 0.824, 1.067)</td>
<td>(23.855, 25.835, 0.829, 1.063) (23.730, 25.714, 0.824, 1.067)</td>
<td>$V_2$ (23.855, 25.835, 0.829, 1.063) (23.730, 25.714, 0.824, 1.067)</td>
</tr>
</tbody>
</table>
Table 3 Clustering centers in data set $G_2$

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>$m_{1,0}, m_{2,0}, a_0, \beta_0$ with distance $d_{T1}(X,Y)$</th>
<th>$D_{p,q}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm-1</td>
<td>$V_1$ (98.899,100.100,0.618,1.024)</td>
<td>(98.898,100.099,0.672,1.024)</td>
</tr>
<tr>
<td></td>
<td>$V_2$ (37.536,39.521,286,0.869)</td>
<td>(37.537,39.522,1.289,0.866)</td>
</tr>
<tr>
<td></td>
<td>$V_3$ (16.548,18.400,1.093,1.180)</td>
<td>(16.562,18.413,1.089,1.183)</td>
</tr>
<tr>
<td>Algorithm-2</td>
<td>$V_1$ (39.196,41.120,1.266,0.865)</td>
<td>(39.137,41.062,1.267,0.863)</td>
</tr>
<tr>
<td></td>
<td>$V_2$ (23.971,25.969,0.869,1.049)</td>
<td>(23.868,25.868,0.865,1.052)</td>
</tr>
<tr>
<td></td>
<td>$V_3$ (13.482,15.224,1.140,1.201)</td>
<td>(13.437,15.184,1.136,1.204)</td>
</tr>
</tbody>
</table>

When we run Algorithm-1 and Algorithm-2 on data set $G_1$ with the distances $d_{T1}$ and $D_{p,q}$, the cluster centers are almost same. From Table 3 Algorithm-1 on data set $G_2$ gives bad results. The outlier affects the centers and the centers are away from the clusters. But the centers are almost same when we use Algorithm-2 on the data set $G_2$.

5. CONCLUSIONS

In this paper we used the metric distances $d_{T1}$ and $D_{p,q}$. Based on these distances we run fuzzy $c$ means clustering algorithm and kernel based fuzzy $c$ means clustering algorithm on trapezoidal fuzzy numbers using MATLAB. The algorithm-2 performance was evaluated using the robustness criterion in comparison to algorithm-1. Algorithm-2 performed well for the data set $G_2$.

REFERENCES