IFS $Z = \{z_1, z_2, \dots\}$ $\mu(z): Z \to [0,1]$ memb

$\nu(z): Z \to [0,1]$ non-memb

$\pi(z) = 1 - \mu(z) - \nu(z) \in [0,1]$ hesitation

$\forall Z_i\ V_j$

U_{ij} = cluster memb

	Z_1	Z_2	Z_3	...
V_1				
V_2				
V_3				
$\vdots$				

where $Z = \{Z_1, Z_2, \dots, Z_p\}$ are p IFSs each with n elements, c is the number of clusters ($1 \leqslant c \leqslant p$), and $V = \{V_1, V_2, \dots, V_c\}$ are the prototypical IFSs, i.e., the centroids, of the clusters. The parameter m is the fuzzy factor ($m > 1$), u_{ij} is the membership degree of the jth sample Z_j to the ith cluster, $U = (u_{ij})_{c\times p}$ is a matrix of $c \times p$.

To solve the optimization problem in (12), we employ the Lagrange multiplier method [28]. Let

$$L = \sum_{j=1}^{p}\sum_{i=1}^{c} u_{ij}^m d_1^2(Z_j, V_i) - \sum_{j=1}^{p}\lambda_j\left(\sum_{i=1}^{c} u_{ij} - 1\right) \quad (13)$$

where

$$d_1^2(Z_j, V_i) = \frac{1}{2}\sum_{l=1}^{n} w_l((\mu_{Z_j}(x_l) - \mu_{V_i}(x_l))^2 +$$

Definition 2 [25] For each IFS A in X, if $\pi_A(x) = 1 - \mu_A(x) - v_A(x)$, then $\pi_A(x)$ is called the hesitation degree (or intuitionistic index) of x to A. Obviously, $0 \leqslant \pi_A(x) \leqslant 1$, especially, if $\pi_A(x) = 0$ for all $x \in X$, then the IFS A is reduced to a fuzzy set; if $\mu_A(x) = v_A(x) = 0$, for all $x \in X$, then the IFS A is completely intuitionistic.

Consider that the elements $x_i\ (i = 1, 2, \dots, n)$ in the universe X may have different importance, let $w = (w_1, w_2, \dots, w_n)$ be the weight vector of $x_i\ (i = 1, 2, \dots, n)$, with $w_i \geqslant 0$, $\sum_{i=1}^{n} w_i = 1\ (i = 1, 2, \dots, n)$. Reference [24] defined the following ~~weighted~~ Euclidean distance between the IFSs A and B

$$d_1(A, B) = \left(\frac{1}{2}\sum_{i=1}^{n} w_i((\mu_A(x_i) - \mu_B(x_i))^2 + (v_A(x_i) - v_B(x_i))^2 + (\pi_A(x_i) - \pi_B(x_i))^2\right)^{\frac{1}{2}} \quad (4)$$

$$\mu_{V_i}(x_l) = \frac{\sum_{j=1}^{p} u_{ij}^m \mu_{Z_j}(x_l)}{\sum_{j=1}^{p} u_{ij}^m}$$

$$1 \leqslant i \leqslant c;\ 1 \leqslant l \leqslant n \quad (16)$$

$$v_{V_i}(x_l) = \frac{\sum_{j=1}^{p} u_{ij}^m v_{Z_j}(x_l)}{\sum_{j=1}^{p} u_{ij}^m}$$

$$\left.\frac{u_{ip}(k)}{\sum_{j=1}^{p} u_{ij}(k)}\right\}, \quad 1 \leqslant i \leqslant c$$

AVG $d(V_k, V_{k+1}) < tol \to$ Halt

Step 4 If $\sum_{i=1}^{c} d_1(V_i(k), V_i(k+1))/c < \varepsilon$, then go to Step 5; Otherwise, let $k := k + 1$, and return to Step 2.

Step 5 End.

$$1 \leqslant i \leqslant c;\ 1 \leqslant l \leqslant n \quad (18)$$

For simplicity, we define a weighted average operator for IFSs as follows. Let $A = \{A_1, A_2, \dots, A_p\}$ be a set of IFSs each with n elements, $\omega = \{\omega_1, \omega_2, \dots, \omega_p\}$ be a

For simplicity, we define a weighted average operator for IFSs as follows. Let $A = \{A_1, A_2, \dots, A_p\}$ be a set of IFSs each with n elements, $\omega = \{\omega_1, \omega_2, \dots, \omega_p\}$ be a set of weights for the IFSs respectively with $\sum_{j=1}^{p}\omega_j = 1$. Then the weighted average operator f is defined as

$$\text{II}\quad f(A, \omega) = \left\{ < x_l, \sum_{j=1}^{p}\omega_j\mu_{A_j}(x_l), \sum_{j=1}^{p}\omega_j v_{A_j}(x_l) > |1 \leqslant l \leqslant n\right\} \quad (19)$$

$$\sum_{j=1}^{p} u_{ij} \quad \sum_{j=1}^{p} u_{ij} \quad \sum_{j=1}^{p} u_{ij} \quad (20)$$

$\pi_A(x) = 1$ $\quad w_i = 1$ $\quad X = \{x_1, \dots x_n\}$ UNIV.

più alto π più int. il set

dist(A, B) for $i = 1$ to n

$$\sqrt{\frac{1}{2}\left((\mu_A(x_i) - \mu_B(x_i))^2 + (v_A(x_i) - v_B(x_i))^2 + (\pi_A(x_i) - \pi_B(x_i))\right)^2}$$

$, \dots, V_c\}$ of the

$\mu_{Z_j}(x_l),$

$1 \leqslant i \leqslant c \quad (21)$

interdependent, we exploit an iterative procedure similar to the fuzzy C-means to solve these equations. The steps are as follows:

Procedure 1 The IFCM algorithm

Step 1 ? Initialize seeds $V(0)$, let $k = 0$ (iter), and set $\varepsilon > 0$ (tolerance).

Step 2 Calculate $U(k) = (u_{ij}(k))_{c\times p}$, where

V: centroidi

Z: IFS

(a) If for all j, r, $d_1(Z_j, V_r(k)) > 0$, then

cluster memb Z_j to V_i

$$u_{ij}(k) = \frac{1}{\sum_{r=1}^{c}\left(\frac{d_1(Z_j, V_i(k))}{d_1(Z_j, V_r(k))}\right)^{\frac{2}{m-1}}}$$

V_i con i fissa

$\forall$ IFS j e $\forall$ centroide i fissati scorri V_r centroidi

$$1 \leqslant i \leqslant c;\ 1 \leqslant j \leqslant p$$

IFS j coincide con centr. r

(b) If there exist j, r such that $d_1(Z_j, V_r(k)) = 0$, then let $u_{rj}(k) = 1$ and $u_{ij}(k) = 0$ for all $i \neq r$.

$\sum_{j=1\to c} w_{ij} \leq 1 \quad V_i\ [0,1]$?

Step 3 Calculate $V(k+1) = \{V_1(k+1), V_2(k+1), \dots, V_c(k+1)\}$, where

nuovi centroidi

$$* \quad V_i(k+1) = f(Z, \omega^{(i)}(k+1)), \quad 1 \leqslant i \leqslant c$$

W. AVG.

Step 3

For $i = 1$ to c

I calcola $w^{(i)}$

II calcola V_i

where

$$\text{I}\quad \omega^{(i)}(k+1) = \left\{\frac{u_{i1}(k)}{\sum_{j=1}^{p} u_{ij}(k)}, \frac{u_{i2}(k)}{\sum_{j=1}^{p} u_{ij}(k)}, \dots,\right.$$

set pesi di ogni IFS $\in Z$ $\forall V_i \in V$

set di triple

$* \quad \forall$ centroide $V_i = \{< x, \mu_{V_i}(x), v_{V_i}(x) > | \forall x \in X\}$

IFCM (Z[P], m):

Step 1

```
init V[c]   //centroids
init w[c,P] // IFS weights per centroid
set ε > 0   // tolerance
U = c x P   // cluster membership
k = 1
```

Step 2

```
do
  U^k = -1
  V-Prec = V
  For j = 1 to P
    For i = 1 to c
      IF (U^k_ij = -1)
        sum = 0
        For r = 1 to c
          IF (d(Z_j, V^k_r) = 0)
            U^k_rj = 1
            For l = 1 to c
              IF (l ≠ r)  U^k_lj = 0
            break
          sum += pow (d (Z_j, V^k_i) / d (Z_j, V^k_r), 2/(m-1))
      IF (U^k_ji ≠ -1)
        break
      U^k_ji = 1/sum
```

Step 3

```
  For i = 1 to c
    For l = 1 to P
      sum = 0
      For j = 1 to P
        sum += U^k_ij
      w^{k+1}_il = U^k_il / sum
    V^{k+1}_i = Make-set (Z, w_i)
```

Step 4

```
  sum = 0
  For i = 1 to c
    sum += d (V^{k+1}_i , V-Prec^k_i)
  mov = sum/c
while (mov > ε)
```

Make-set (Z, w_i) → V_i :

Weighted AVG

```
For l = 1 to n   // ∀x ∈ X
  x = V_i.set.get(l)   // get the l-th element in V_i.set
  Nμ = 0
  Nν = 0
  For j = 1 to P
    Nμ += w_ij * Z_j.set.get(l).μ
    Nν += w_ij * Z_j.set.get(l).ν
  X.μ = Nμ * X.μ
  X.ν = Nν * X.ν
```