DOI: http://dx.doi.org/10.26483/ijarcs.v14i3.6991

ISSN No. 0976-5697

Volume 14, No. 3, May-June 2023

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

PPS-FPCM: PRIVACY-PRESERVING SEMI-FUZZY POSSIBILISTIC C-MEANS

Mohamed A. Mahfouz*, Zeinab Abd El Haliem and Ehab Emam Zakaria
College of ComputerScience,
MSA University, Egypt

Abstract: Applying traditional clustering techniques to big data on the cloud while preserving the privacy of the data is a challenge due to the required division and exponential operations in each iteration, which complicate its implementation on encrypted data. Several existing approaches are based on approximating the formulas of centers, weights, and memberships as three polynomial functions according to the multivariate Taylor formula. However, they usually suffer an increase in complexity and a slight drop in accuracy. In this paper, a novel Privacy-Preserving semi-fuzzy clustering algorithm based on the possibilistic paradigm, termed PPS-FPCM, is presented. Its main feature is that it avoids exponentiation and division operations, at each iteration, without losing accuracy. By restricting the typicality to an ordered set of discrete values between zero and one decided by the data owner (DO), the computation is simplified. The second key idea is the use of this soft typicality to detect outliers and compute the corresponding semi-fuzzy memberships, which is used to increase the in-between cluster distance. However, the initial typicality requires a magnitude relation comparison, which is still difficult to do over encrypted data. In this research study, we show how the existing incomplete re-encryption method can be used to tackle this problem. In each iteration, centers and distances to the new centers are computed on a calculator cloud server (CaCS) which is responsible for storing the cipher texts of the (DO)'s data and processing them. Then, CaCS sends the incompletely re-encrypted difference between these distances and iteratively updated bin values that correspond to the discrete possibilistic memberships that are initially decided by the (DO) to the comparator cloud server (CoCS). CoCS decrypts the difference and returns the results of comparisons. When CaCS receives the results of comparison from CoCS, it decides on an appropriate soft typicality or resends the difference of the same distance to another bin value. The required number of comparisons is O(log the number of bins). CaCS iteratively computes the corresponding semi-fuzzy memberships, computes the refined memberships, and updates the centers. In the end, CaCS sends the final soft memberships and centers to the (DO). The proposed algorithm is applicable to normal data and homomorphically encrypted data, is more effective than several related algorithms, and can produce accurate results using large enough (16 or more) discrete values with a high reduction on runtime as the number of comparisons is much less complex than exponential and division operations with added communication cost between CaCS and CoCS.

Keywords: possibilistic clustering, fuzzy clustering, comparison protocol, homomorphic encryption, Incomplete re-encryption

1. INTRODUCTION

The advancement of the Internet of Things, mobile devices, and sensing techniques has resulted in new applications in smart cities, intelligent transportation, and industrial manufacturing [1-3]. The application layer of the IoT is responsible for collecting and analyzing the data collected from the physical layer of the IoT (sensors, RFID, and two-dimensional codes). Machine learning techniques play an important role in the IoT. They help the application layer of the IoT to analyze collected data and provide predictive services and intelligent decisions [4-6]. Clustering is an unsupervised learning technique that is used to group a set of similar objects such that an object is much more similar to objects in its group than to objects in other groups. The correlation between data attributes and the globally distributed data model can be discovered using clustering analysis. The clustering methods can be roughly categorized into five categories: partitioning methods such as K-means [7], fuzzy c-means (FCM) algorithm [8], possibilistic c-means(PCM) [9, 10], mode-based methods such as competitive learning [11], self-organizing map (SOM) [12], grid-based methods [13], density-based methods [14] and hierarchical methods [15]. Partitioning clustering algorithms are the most popular and widely used on the cloud due to their efficiency and simplicity. Compared to FCM, PCM is more robust since it gives low membership to outliers in all clusters.

However,due to its high computational complexity, PCM is not suitable for big data clustering with a large number of objects efficiently. Furthermore, PCM must load all objects into memory, which is constrained on local computing devices. Also, PCM is very sensitive to initialization and may generate coincident clusters.

Reducing the membership of core points to other clusters [16] and the adaptive determination of the penalty parameters [17] can greatly help in reducing the sensitivity to initialization, removing obsolete clusters and avoiding coincident clusters.

Cloud computing offers a scalable and cost-efficient solution for big data clustering by providing tremendous memory space and high processing power [18]. Although, some improved PCM algorithms have been proposed for big data clustering, such as weighted kernel possibilistic c-means [19], incremental WPCM [20, 21], distributed weighted possibilistic c-means [22] and secure weighted possibilistic c-means for privacy-preserving big data clustering on the cloud [23].

These algorithms suffer from several problems, they are based on approximating the required calculations using Taylor's expansion and always produce lower clustering accuracy than their corresponding conventional algorithms. Also, uploading the raw data to the cloud directly, as in [22], poses a serious threat to data security.

To tackle the above problems, this paper has two contributions, first, a semi-fuzzy (soft) possibilistic clustering is proposed, and second, a secure implementation for it based on a judicious integration of the principles of semi-fuzzy [24] and secure multiparty computation (SMC) [25]. Early implementation for semi-fuzzy clustering is found in [24], where three mathematical models for semi-fuzzy clustering with their corresponding algorithms are introduced. The first algorithm limits the fuzziness to a smaller subset of the available clusters, while the other two algorithms allow a pattern to be associated with the proper number of clusters based on a threshold distance or membership value. In [13], a rough-fuzzy c-medoids algorithm is based on the principles of rough sets and fuzzy sets [15]. The concept of soft clustering is applied to fuzzy CLARANS [26] in [27]. Also, in [28], a soft biclustering algorithm is introduced in which the memberships of rows and columns are changed between a set of fixed values as an alternative to hard biclustering.

Secure multiparty computation (SMC) in which mutually doubting users collaborate to compute a function of their private data without revealing any additional information about their data to other users [25] is a good solution for preserving security while doing clustering on the cloud. Several applications require SMC [29]. The algorithms that are based on approximating the used formulas using Taylor expansion not only avoid division and exponentiation but also the comparison of magnitudes, which is difficult to execute over encrypted data.

In the proposed security scheme, two cloud servers are involved, including a calculator (CaCS) and a comparator (CoCS). The calculator stores the encrypted data and is responsible for computing distances to current centers using basic homomorphic operations on the encrypted data. The comparator is in charge of calculating soft memberships based on previously computed bins representing the range of distances corresponding to each discrete value of the memberships. In this way, users can outsource the whole data processing to the cloud servers with minimal participation in communication and computation, which decreases the burden imposed on them. Incomplete re-encryption can be used for secure comparison. In contrast to the general re-encryption, the plaintext decrypted after re-encrypted is different from the original plaintext, but it can maintain the magnitude relation of the original plaintext. However, a secure distributed implementation of the proposed algorithm is out of the scope of this paper.

This paper describes the proposed algorithm and evaluates its performance relative to the traditional algorithm and presents the pseudo-code of the proposed security scheme along with a complexity analysis.

The main contributions of this paper can be summarized as follows:

- By restricting the memberships to a set of discrete values between zero and one that differs by a small step, the proposed algorithm avoids the required exponential and division operations in each iteration. The required multiplications in computing the penalty parameters are reduced to O (c r) where r is the number of discretization levels and c is the number of clusters. Also, computing the typicality costs O(n log r) comparisons.
- The proposed algorithm generalizes the idea in [30] and shows how variations of the PCM algorithm can be securely applied on the cloud.
- Unlike FPCM, the proposed algorithm identifies core objects from their fuzzy memberships while outliers are identified from their typicality. The final memberships of both core and outlier objects are not computed directly as a linear combination of their typicality and fuzzy memberships as in FPCM.
- The proposed protocol is secure without the cloud server's collusion and with minimal participation from data owner (DO). The DO only receives the encrypted penalty parameters in each iteration and applies simple functions on them and returns the results.

The effectiveness of the proposed algorithm is evaluated on several standard datasets and synthetic datasets by comparing PPS-FPCM with hard c-means (HCM), fuzzy c-means (FCM), possibilistic c-means (PCM), possibilistic fuzzy c-means (PFCM), semi-possibilistic c-means SPCM, and the weighted possibilistic c-means (WPCM). The results show that PPS-FPCM is more effective than related fuzzy algorithms and its effectiveness compares favorably to SWPCM. The proposed algorithm is expected to achieve good scalability on the cloud for big data clustering compared to SWPCM, however, the distributed implementation of the proposed algorithm is out of the scope of this paper.

The paper is organized as follows. The possibilistic c-means algorithm along with its related algorithms are reviewed in Section 2 and the proposed algorithm is presented in Section 3. The experimental results are shown in Section 4 and the paper is concluded in Section 5.

2. RELATED WORK

Before describing the proposed algorithm and other related algorithms in the following sections, commonly used abbreviations and symbols in these sections are listed in Table 1.

Table 1 Abbreviations and symbols used in the text

Abbreviation	Description
HCM	Hard c-means algorithm
PCM	possibilistic c-means algorithm
SWPCM	secure weighted possibilistic c-means algorithm
FPCM	fuzzy possibilistic c-means algorithm
SPCM	Efficient semi-possibilistic c-means clustering algorithm
PPS-FPCM	The proposed Privacy-Preserving semi-fuzzy possibilistic c-means algorithm
X	The input dataset consists of *n* objects $x_1,x_2,\ldots\ldots x_n$each having *d* attribute values
u_{ij}	The membership (typicality) of object x_j in cluster *i*
m	The fuzzification factor
c	Input parameter represent the required number of clusters
V	A set of *k* centers $v_1, v_2,\ldots v_k$each of size 1×*d*
maxiter	Maximum number of iterations
ε	Input threshold used in the termination condition
α	The winning prototype is the prototype having the maximum membership and its membership is higher than α
η_i	The Coefficient of the penalty term of the objective function of the possibilistic approach. It is a fixed user-defined parameter in PCM while it is estimated in SPCM.
W	Weight matrix of size *n* × *k* used only in WPCM
U	membership matrix of size *n* × *k*
r	if a fixed step is required then the range of soft memberships is 0..*r* and the corresponding discrete values will be 0,1/r,2/r,……,1 and it is an optional parameter
μ	If an unfixed step is required, S is given as an input parameter as a set of *r*+1 discrete membership values between 0 and 1. Ex. {0,0.1,0.25,0.5,0.7,0.9,1}
μ^m	Are the values in μ raised to power *m*
B_i	A set of *r*-1 bin values for cluster *i* corresponds to μ - {0,1}.
a,*b*	Two parameters represent the weights given to FCM and PCM in PFCM algorithm
C(x)	Cypher text of *x*

a. Possibilistic Clustering Paradigm

The possibilistic approach to clustering [9, 10] assumes that the membership function of a data point in a fuzzyset (or cluster) is absolute, i.e. it is an evaluation of a degree of typicality not depending on the membership values of the same point in other clusters.

Let X={x_1, x_2 ….., x_n} be a set of unlabeled data points, V= {v_1, v_2 …… , v_c} a set of cluster centers (or prototypes) and U= [u_{ij}] the fuzzy membership matrix. In the Possibilistic C-Means (PCM) algorithm, the constraints on the elements of Uare relaxed to:

$u_{ij} \in [0,1]$ *for i=1,2,..c and j=1,2,..n*

$0 < \sum_{j=1`}^{n} u_{ij} < n \quad for\ i = 1,2,..c$

$\vee_i \quad u_{ij} > 0 \qquad for\ j = 1,2,\ldots n$

These requirements simply imply that a cluster cannot be empty and each pattern must be assigned to at least one cluster. This turns a standard fuzzy clustering procedure into a mode-seeking algorithm [9, 10]. The objective function contains two terms; the first one is the objective function of the fuzzy *c*-means [8], while the second is a penalty term considering the entropy of clusters as well as their overall membership values:

$J_m(U,Y) = \sum_{i=1}^{c}\sum_{j=1}^{n} u_{ij}^m \quad \|x_j - v_i\|^2 + \sum_{i=1}^{c} \eta_i \sum_{j=1}^{n}(1-u_{ij})^m$ (1)

where m is a fuzzifier ∈ [1, ∞) given as input and c is the input parameter representing the number of clusters. The parameters $\eta_i$for*i*=1 to c should be estimated before the clustering procedure starts depending on the average size of the *i-th* cluster. The solution that is obtained by minimizing the above objective function will be highly dependent on the parameter η_i. Note that if $\eta_i \to \infty$ *for i = 1,2,…c* (i.e., the second term of$J_m(U,Y)$ is omitted), then a trivial solution is obtained by the minimization of the remaining cost function (i.e., u_{ij}=0 for *i*=1,2,..c and *j*=1,2,..*n*, as no probabilistic constraint is assumed). The pair (*U*; *Y*) minimizes J_m, under the above constraints only if :

$$\boldsymbol{u_{ij}} = \frac{1}{1+\left(d_{ij}^2/\eta_i\right)^{1/(m-1)}}$$ for *i*=1,2,...c and j=1,2,…n(2)

and

$$v_i = \frac{\sum_{j=1}^{n} x_j u_{ij}^m}{\sum_{j=1}^{n} u_{ij}^m} \qquad for\ i = 1,2,\ldots c$$ (3)

Equations (2) and (3) can be used as formulas for recalculating the membership functions and the cluster centers.

The membership $u_{ij}$is the fuzzy membership of x_j in cluster *i*and can be defined heuristically in several ways

Algorithm 1 shows the pseudo-code for traditional possibilistic c-means (PCM).

Algorithm 1: Traditional PCM algorithm.

Input: X,*c, m, maxiter*
Output: U , V
Initialize $v_i$and η_i for *i=1,2,…,c*
repeat
  compute memberships U using Eq. (2)
  V`=V
  compute new centers V using Eq. (3)
until (||V`-V||<ε)
return U, V

b. possibilistic-fuzzy c-means (PFCM*)*

PCM computes possibilistic partitions independently. When the initializations of the corresponding rows are not distinct enough from each other, it is very possible that two or more cluster centers will be moved to the same point if the point is the single optimal point for a cluster center leading to coincident cluster centers. Accordingly, Sensitivity to initializations and coincident clusters are the main drawbacks of PCM. The possibilistic-fuzzy c-means (PFCM) clustering algorithm <Pal, 2005 #17> is an outstanding possibilistic-fuzzy mixture clustering model to avoid the drawbacks of PCM. The objective function of the PFCM is as follows:

$$J_{PFCM} = \sum_{i=1}^{c}\sum_{j=1}^{n}(au_{ij}^{m} + bt_{ij}^{p})d_{ij}^{2}(x_j, v_i) + \sum_{i=1}^{c}\eta_i\sum_{j=1}^{n}(1-t_{ij})^{p} \quad (4)$$

Constrained by $\sum_{i=1}^{c} \quad u_{ij} = 1$, $0 < \sum_{j=1}^{n} t_{ij} < n$, $0 \le u_{ij} \le 1$, $0 \le t_{ij} \le 1$, $1 \le i \le c$

Parameters a and b determine the intensity of the possibilistic and probabilistic term, respectively. The other parameters m, p, η act the same function as that in the PCM and FCM. Moreover, the membership values u_{ij} are updated by Eq. (5) while the typicality t_{ij} are updated using Eq. (6) as follows:

$$\boldsymbol{u_{ij}} = \left(\sum_{k=1}^{c} ((\boldsymbol{d}\ (x_j, v_i)/\, d(x_j, v_k))^{2/(m-1)}\right)^{-1} \quad \text{for } i=1,2,...c \text{ and } j=1,2,\ldots n \quad (5)$$

$$\boldsymbol{t_{ij}} = \frac{1}{1+\left(d_{ij}^{2}\,(x_j,v_i)/\eta_i\right)^{1/(p-1)}} \quad \text{for } i=1,2,...c \text{ and } j=1,2,\ldots n \quad (6)$$

The cluster center is updated by the following formula:

$$v_i = \frac{\sum_{j=1}^{n} x_j\,(au_{ij}^{m} + \ b\,t_{ij}^{p}\ )}{\sum_{j=1}^{n}\,(au_{ij}^{m} + \ b\,t_{ij}^{p}\ )} \quad for\ i = 1,2,\ldots c \quad (7)$$

Algorithm 2: possibilisticfuzzy c-means (PFCM)

Input : X, c, *a, b, m , maxiter*
Output : U , V
Initialize $v_i$and η_i for*i=1,2,…,c*
repeat
  compute membership u_{ij} for *i=1,2,…,c* and *j*= 1,2,..*n* using Eq. (5)
  compute typicality $t_{ij}$for*i=1,2,…,c* and *j*= 1,2,..*n* using Eq. (6)
  V`=V
  compute new centers V using Eq. (7)
until (||V`-V||<ε)
return U, V

c. Secure Weighted Possibilistic c-means (**SWPCM)**

In PCM each object is assigned the same weight, however, the importance of objects differs. Furthermore, PCM cannot yield the desirable clustering results for the datasets including noisy objects or outliers. To tackle this problem, Schneider presented a weighted possibilistic c-means algorithm (WPCM) by assigning a weight value to each object, which leads to an objective function of WPCM [16] :

$$J_m(U,Y) = \sum_{i=1}^{c}\sum_{j=1}^{n} u_{ij}^{m} \quad ||x_j - v_i||^2 + \sum_{i=1}^{c}\frac{1}{\beta_i}\sum_{j=1}^{n}(w_j - u_{ij}) \tag{8}$$

The weights can be calculated using the following formula:

$$w_j = \sum_{i=1}^{c} e^{-\alpha||x_j - v_i||^2} \tag{9}$$

The memberships are computed as follows:

$$u_{ij} = \frac{w_j}{\left(1+||x_j - v_i||^2/\eta_i\right)^{1/(m-1)}} \tag{10}$$

While the centers are computed as in Eq. (3).

In [23], a secure weighted possibilistic c-means algorithm based on a full homomorphic encryption scheme (BGV) [1] is introduced for secure data clustering on the cloud. The BGV scheme allows only addition and multiplication on encrypted data. The key idea of SWPCM is to remove the division and exponential operations from the formulas for calculating the centers, weights, and memberships and approximating them as three polynomial functions according to the multivariate Taylor formula. The approximated formulas are used to obtain the correct clustering result on the encrypted data as shown in algorithm 3. It is clear from Algorithm 3 that the computational complexity of WPCM is dominated by the step for computing the membership matrix which has a computational complexity of *O* (*cn*), resulting in a total computational complexity of *O*(*lcn*) where *l* denotes the number of the iterations. As shown in algorithm 3, SWPCM has to complete all the *maxiter* steps because the centers are encrypted and cannot be compared with the previous iteration. Also, as their experimental results show, in most cases, SWPCM was slightly less accurate than WPCM since the approximation of the functions for calculating the weight values and updating the membership matrix and the clustering centers resulted in the drop of the clustering accuracy.

Algorithm 3: Secure weighted possibilistic c-means scheme (SWPCM)

Initially: both the data objectsX *and initialized parameters are encrypted and uploaded from the client to* the cloud
output: immediate and final *U* and *V can be downloaded from the cloud to the client to decrypt*
for *iteration* = 1 , 2 , . . . , *maxiter***do**
Use BGV to encrypt the membership matrix and the clustering centers;
Upload encrypted membership matrix and centers on the cloud;
On cloud:
Use approximated formula for calculating *C*(V)
Use approximated formula for calculating *C*(W)
Use approximated formula for calculating *C*(U)
Send the encrypted membership matrix and centers to the client
On the client: ;
Decrypt the immediate results to update Uand V

endfor

3. METHODOLOGY

a. Computing the Bin values from the input set of memberships

By limiting the memberships to a set of discrete values between zero and one, the squared distance corresponding to these memberships can be calculated using the original membership calculation formula. Using Eq. (2) of the standard PCM, the squared distance can be expressed as a function of the corresponding membership as follows:

$$\left(d_{ij}^2/\eta_i\right)^{\frac{1}{p-1}} = \left(\frac{1}{t_{ij}} - 1\right)$$

Raising the two sides of the equation to the power (*p*-1) > 0

$$\left(d_{ij}^2/\eta_i\right) = \left(\frac{1}{t_{ij}} - 1\right)^{p-1}$$

Then

$$d_{ij}^2 = \eta_i\left(\frac{1}{t_{ij}} - 1\right)^{p-1} \tag{11}$$

Using Eq. (11), by substituting t_{ij} by one of the discrete membership values that are initially given as input parameter μ, the corresponding bin value for each membership in μ is computed.

According to Eq. (11), the discretization bins at iteration *t* is computed from η_i ,for *i*=1,2,..,*c*, and the set of memberships μ as follows:

$$B_i(t) = \mathrm{C}(\eta_i(t))(\tfrac{1}{\mu} - 1)^{p-1} \tag{12}$$

The discretization bins of cluster *i* at an iteration *t* +1 is updated using the new value of η_i as follows:

$$\boldsymbol{B_i(t+1) = C(\eta_i(t+1))(\frac{1}{\mu} - 1)^{p-1}} \quad \text{for i=1, 2, …, c} \qquad (13)$$

b. Computing initial penalty parameters

After selecting randomly initial *c* cluster centers V from encrypted data X, the encrypted distances D are computed from the encrypted data X and the initial centers V. In order to check if cluster $v_i$isthe nearest cluster center for an object x_j for *j*=1..*n*, the CaCS sends the re-encrypted value D_{kj} to CoCS. Where D_{kj} is computed as follows:-

$D_{kj}=C(d_{kj}) - C(d_{ij})$ where $k \neq i$ (14)

Algorithm 4 secure calculation for initializing the penalty paramters η

for *i*=1 to *c*
$sz_i = 0$
$C(td_i) = C(0)$
for *j*=1 to *n*
if (cls(*j*) is null)
for *k*=*i*+1 to *c*
CaCS: compute $C(D_{kj}) = (C(d_{kj}) - C(d_{ij})$
CoCS: receive re-encrypted $C(D_{kj})$ and return the magnitude relation F to CaCS
CaCS: if (F > 0) or (F=0 and i=c) then
$sz_i = sz_i + 1$
$C(td_i) = C(td_i) + C(d_{ij})$
cls(j) = *i*
endif
end
endif
end
$C(\eta_i) = C(td_i) \times (\frac{1}{sz_i})$
end

Then, CoCS incompletely decrypts re-encrypted (D_{kj}) and send the magnitude relation F to CaCS to compute not encrypted hard membership. If the value of D_{kj} is positive or zero for all *k* this means the center v_i is the nearest center and no need to check next cluster v_k where *k*>*i* and the size of the cluster v_i is incremented and the total distances to it is updated .

c. Computing Soft Typicality using the bin vectors

To compute the typicality *t* of an object in a certain cluster, its distance to the cluster center is compared to the values stored in the bin vector associated with this cluster. Searching for the proper bin given the squared distance costs O (log *r*) comparisons. Each cluster *i* has a bin vector B_i whereB_i is a set of *r*-1 bin values for cluster *i* corresponds to μ-{0,1}. An object x_q having a squared distance d^2 to v_i is assigned a typicality *t* as follows:

$$t = \begin{cases} 0 & d^2 > b_1 \\ 1/r & b_1 \geq d^2 > b_2 \\ 2/r & b_2 \geq d^2 > b_3 \\ \vdots & \vdots \\ (r-1)/r & b_{r-2} \geq d^2 > b_{r-1} \\ 1 & d^2 \leq b_{r-1} \end{cases} \qquad (16)$$

Another approach is to use Trapezoidal membership for categories 0 and 1 and Triangular membership for the other categories as shown in Table 2. In this approach, division is needed to compute the approximated membership from the squared distance d^2 as shown in Eq. (9).

$$t = \begin{matrix} 0 & d^2>2b_1-b_2 \\ (2b_1-b_2-d^2)/(b_2-b_1)r & b_1<d^2 \leq 2b_1-b_2 \\ [2(b_1-d^2)+(d^2-b_2)]/(b_2-b_1)r & b_1 \geq d^2 > b_2 \\ \vdots & \vdots \\ b_{r-1}[r(b_{r-1}-d^2)+(r-1)(d^2-\frac{b_{r-1}}{2})]/2r & b_{r-1} \geq d^2 > b_{r-1}/2 \\ 1 & d^2 \leq b_{r-1}/2 \end{matrix} \qquad (17)$$

Algorithm 5 Calculating the typicality from the discretization bins

Input cipher of a distance C(d_{ij})andbins vector B_i for cluster *i*
l = 1; *h* = *r* - 1
while l<h do
CaCS: *m*= *l*+*(h-l)*/2
CaCS: compute g= C(d_{ij}) – C(B_{im}) and send re-encrypted(g) to CoCS
CoCS: incompletely decrypt g and return F as the magnitude relation to CaCS
CaCS: if (F=0) return *m* end if
if (F<0)
l = *m*
else
h= *m*
end if
if ((*h*-*l*)<=1) return *l* end if
end while

---

d. Computing fuzzy membership of an object in a cluster given its Typicality in all clusters

From Eq. (11)

$$d_{ij} = \sqrt{\eta_i}\left(\frac{1}{t_{ij}} - 1\right)^{\frac{p-1}{2}}$$

$$\left(\frac{d_{ij}}{d_{kj}}\right)^{\frac{2}{m-1}} = \left(\frac{\eta_i}{\eta_k}\right)^{\frac{1}{m-1}}\left(\left(\frac{1}{t_{ij}} - 1\right) \Big/ \left(\frac{1}{t_{kj}} - 1\right)\right)^{p-1/(m-1)}$$

Let

$$z_{ikj} = \left(\left(\frac{1}{t_{ij}} - 1\right) \Big/ \left(\frac{1}{t_{kj}} - 1\right)\right)^{p-1/(m-1)} \quad (17)$$

$$\beta_{ik} = \left(\frac{\eta_i}{\eta_k}\right)^{\frac{1}{m-1}} \quad (18)$$

From equation (5) and (11), the fuzzy membership can be calculated using soft typicality as follows:

$$u_{ij} = 1 \Big/ \sum_{k=1}^{c} \beta_{ik} z_{ikj}$$

for *i*=1,2,...c and *j*=1,2,…*n*(19)

By keeping a matrix Z of size $r \times r$ such that for each pair of values in μ the value z_{ikj} is precomputed knowing the index of t_{ij} and t_{kj} in μ denoted $s_{ij}$and s_{kj} such that z_{ikj} = Z[s_{ij},s_{kj}]

Also, at each iteration, the encrypted values of η are sent to the client to return an unencrypted matrix β such that $\beta[i,k] = \left(\frac{\eta_i}{\eta_k}\right)^{\frac{1}{m-1}}$

$$u_{ij} = 1 \Big/ \sum_{k=1}^{c} \beta[i,k] \;\; Z[s_{ij}, s_{kj}]$$

for *i*=1,2,...c and *j*=1,2,…*n* (20)

e. Updating the penalty parameters

From previous sections, we can compute unencrypted values of the typicality. If we store in the typicality matrix the index of each typicality value t_{ij} in μ, s_{ij}. Also, the values of μ raised to power *m* are stored in a vector φ = μ^m. As we can see from Eq. (15), the exponentiation are avoided also the number of multiplications are reduced to *r* instead of *n* by computing the sum of C(d_{ij}) and the count of x_j where the soft typicality index $s_{ij} = k$ as follows:-

$$\boldsymbol{C(\eta_i) = \sum_{k=1}^{r} \varphi_k[s_{ij}] \sum_{s_{ij}=k} C(d_{ij}) \Big/ \sum_{k=1}^{r} \varphi_k[s_{ij}] \sum_{j=1}^{n} [s_{ij} = k]} \quad (15)$$

f. Computing the final memberships at each iteration

In order to overcome the PCM's coincident clustering problem, core objects need to be identified. In the proposed algorithm, the computed fuzzy memberships are used for identifying core objects and introducing the between-class relationships, and for computing the final membership. From the fuzzy memberships of an object in all clusters, the winner prototype is identified as the prototype having the maximum membership which is higher than a threshold. The between-class relationships are introduced by increasing the difference in the membership of the winner prototype and the membership of non-winner prototypes as in the cutset algorithm that is introduced in [16]. Unlike the procedure used in [17], in the proposed algorithm, the between-class relationships are introduced by setting the non-winner fuzzy membership to 0 and the fuzzy membership of the winner to 1 then the final membership $t_{ij}` = a \times t_{ij} + b \times u_{ij}$,thus, the parameters *a* and b will play an important role in deciding how much the increase in the difference of the membership of the winner prototype and the membership of non-winner prototypes. Such modifications do not affect the typicality of outliers since PCM by its nature gives small typicality values to outliers. In contrast to traditional FPCM, only the membership of objects that are neither core nor outliers are preserved, and their final membership is $a \times t_{ij} + b \times u_{ij}$ where *a* and *b* are input parameters.

Table (c) in Fig. 2 shows the typicality t_{ij} of object *j* in cluster *i* , $t_{ij}$is computed from the discretized distances to centres. The last two objects in this example have very low typicality in all clusters and they can be easily identified as the objects having h_j less than the average of $h_j$where$h_j = \sum_{i=1}^{c} t_{ij}$by three standard deviations. The fuzzy memberships in Table (a) are computed using Eq. (20). Table (b) shows the refined fuzzy memberships, fuzzy memberships of candidate outliers are set to 0 while identified core objects their memberships are set to 1 in the winner class and to zeros in the other classes. The fuzzy membership of any normal object (neither an outlier nor a core) is kept as is. The final memberships at any iteration are computed from the refined fuzzy memberships and typicality. In this example, the parameters a and b are set to 0.5. Table (d) in Fig. 2 shows the final memberships where the between-class relationship is increased and the membership for outliers is decreased.

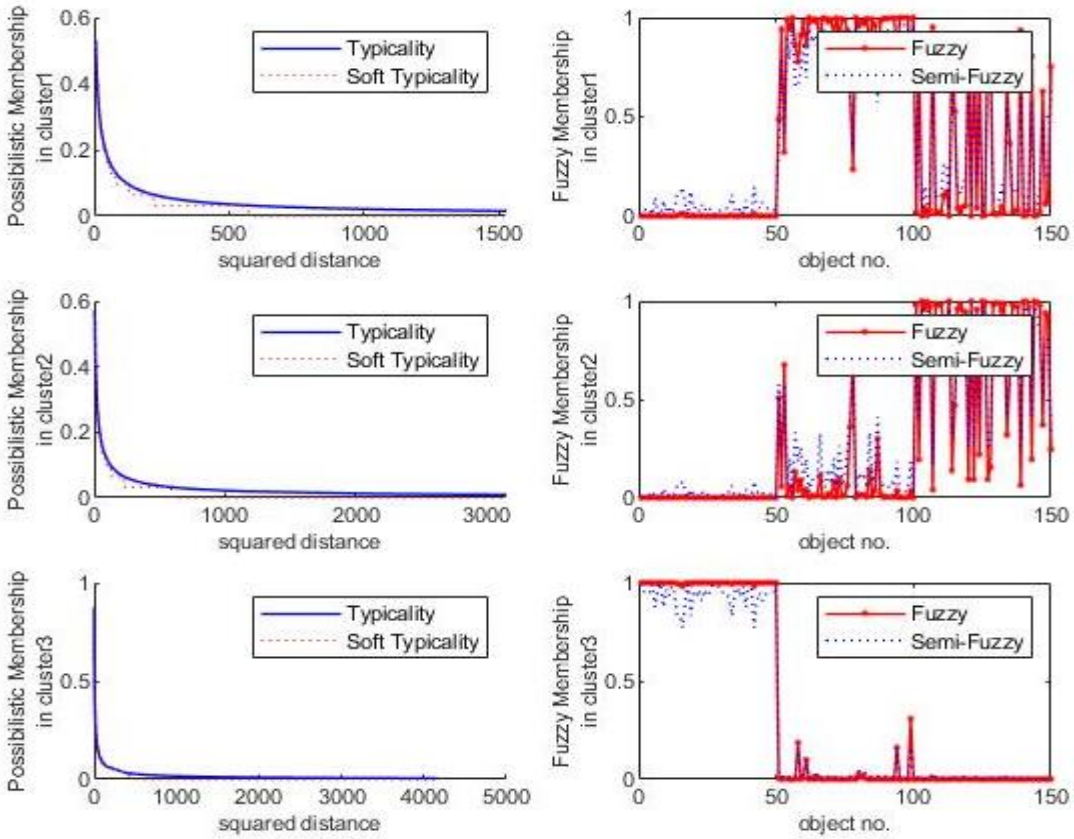
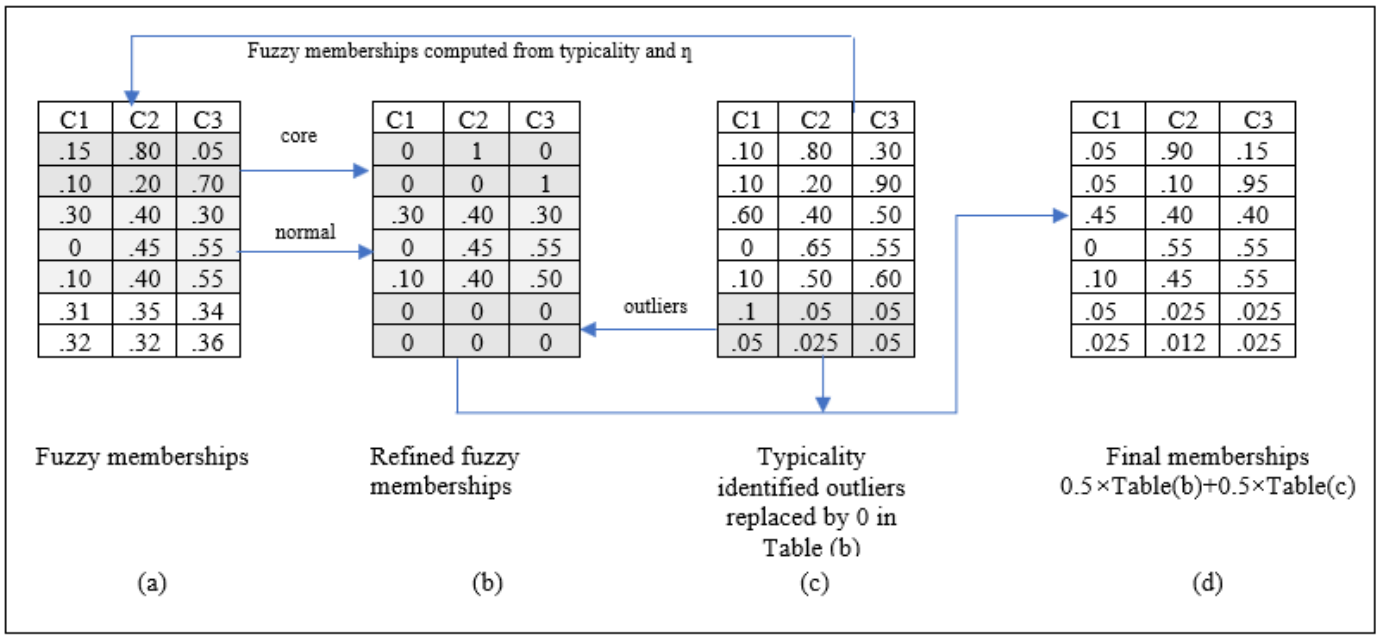


fig. 1 memberships vs. soft memberships



C1	C2	C3
.15	.80	.05
.10	.20	.70
.30	.40	.30
0	.45	.55
.10	.40	.55
.31	.35	.34
.32	.32	.36

C1	C2	C3
0	1	0
0	0	1
.30	.40	.30
0	.45	.55
.10	.40	.50
0	0	0
0	0	0

C1	C2	C3
.10	.80	.30
.10	.20	.90
.60	.40	.50
0	.65	.55
.10	.50	.60
.1	.05	.05
.05	.025	.05

C1	C2	C3
.05	.90	.15
.05	.10	.95
.45	.40	.40
0	.55	.55
.10	.45	.55
.05	.025	.025
.025	.012	.025

fig. 2 flow of memberships in the proposed algorithm

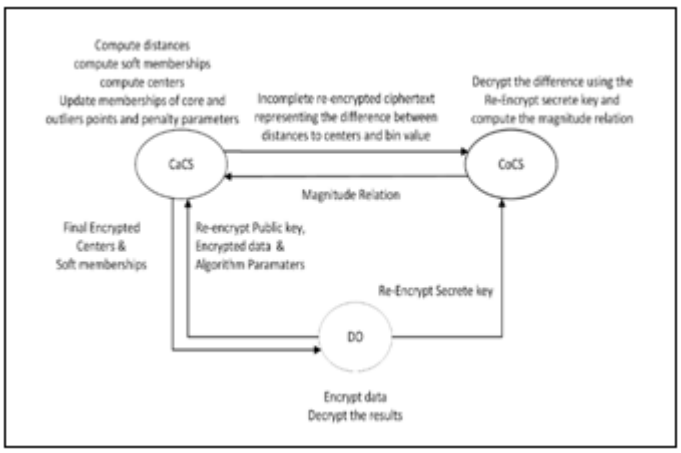


Figure 3

g. Computing cluster centers

The cluster centers are updated using the formula in Eq. (21). The values of the typicality t are selected as one of the values stored unencrypted in a vector $\varphi = \mu^m$ based on the magnitude relation returned by CoCS, where μ stores the set of the input set of discrete values for memberships. The fuzzy memberships u_{ij} are computed unencrypted from typicality, as shown in section 3.5 above. The only encrypted data in the following equation are the owner's encrypted data; thus, no exponentiations or divisions on the encrypted data are required.

$$v_i = \frac{\sum_{j=1}^{n} x_j \left(a u_{ij}^{m} + \; b\, t_{ij}^{p}\right)}{\sum_{j=1}^{n} \left(a u_{ij}^{m} + \; b\, t_{ij}^{p}\right)} \qquad for\ i = 1,2,\dots c \tag{21}$$

h. The proposed Privacy-Preserving semi-fuzzy possibilistic c-means (PPS-FPCM)

In this section, we will describe an algorithm termed PPS-FPCM. The proposed PPS-FPCM is based on secure multiparty computation (SMC) in which mutually doubting users collaborate to compute a function of their private data without revealing any additional information about their private data to the other users [25].

Incomplete re-encryption is a concept based on proxy re-encryption introduced in [29]. Proxy re-encryption is a well-known technology in cryptography [31], [32], and [33]. It enables two entities to share the same content via encryption without revealing the secret key to each other.It allows a semi trusted proxy, given a re-encryption key $pk_{a\rightarrow b}$, to transform a ciphertext under the public key pk_a to a new ciphertext under another public key pk_b. Different from proxy re-encryption, the protocol that is proposed in [29] allows decrypting the result can reflect the magnitude relation of the data without leaking the data. It is similar to general re-encryption, but it outputs plaintext different from the original input, they call it incomplete re-encryption.

The protocol described in [29] is able to get the magnitude relation between the ciphertexts that have even been subjected to multiple homomorphic operations in the ciphertext domain in a cloud computing environment. Following their protocol, a calculator cloud server (CaCS) and a comparator cloud server (CoCS) are involved in our proposed scheme. At first, the client needs to encrypt its data before uploading it to CaCS along with the required parameters, also, it sends an incomplete re-encryption public key to CaCS and an incomplete re-encryption secret key to CoCS. Fig. 3 shows the communications between the components of the system model.

In each iteration, CaCS computes new centers, and then the distances D to the centers are computed using some basic homomorphic operations on encrypted data and centers. The discretization bins B_{PCM} are computed by CaCS using current values of η. The re-encrypted difference between D and B is sent to CoCS. A maximum of a log r comparison is required for each squared distance value to be assigned a soft membership from the available r memberships. The magnitude relation of the squared distance and a bin value is returned by CoCS to CaCS. CaCS will check each received value from CoCS to see if it can assign a soft membership to its corresponding object and exclude it from the next comparison iteration (O (log r) comparisons). After computing the typicality of each object, the fuzzy memberships are computed at CaCS from the computed topicalities and the penalty parameter at this step using Eq. (14). At the end of each iteration, CaCS does a refinement step to compute the final memberships using the assigned soft PCM and FCM memberships. The cluster centers are computed using the refined memberships. The purpose of the refinement step is to reduce the effect of candidate outliers and increase the in-between cluster distances.

Algorithm 6: secure Privacy-Preserving possibilistic c-means (PPS-FPCM))

DO:
  Generate (pk_{DO} ,sk_{DO}) ←KeyGen(σ) // public/secret key pair generated using security parameter σ
  Generate (pk_{Ca} ,sk_{Co}) ←ReKeyGen(sk_{DO}, pk_{DO}) // an incomplete re-encryption public/secret key pair
  Send pk_{Ca} to CaCs // an incomplete re-encryption public key send to CaCS
  Send sk_{Co} toCoCS // an incomplete re-encryption secret key
  {C} ←SH.Enc(X, pk_{DO}) // ciphertext of X
  send {C} to CaCS
send *c*, *m*, α , *r* to CaCS
  CaCS: Select encrypted initial centers V from encrypted data X
  CaCS: Compute encrypted distances D from encrypted data X and encrypted centers V
  CaCS: Compute initial hard memberships and initial penalty parametrs η using algorithm 4
  Repeat:
    CaCS: Send encrypted η to the DO to decrypt and return β as in Eq. (13) and Eq. (18)
    CaCS: Compute encrypted new centers as in Eq. (21)
    CaCS: Compute encrypted distances from encrypted data andencrypted centers
    CaCS: Compute decrypted new bins using Eq. (13)
    CaCS: Compute not encrypted soft typicality as described in algorithm 5
    CaCS: Compute semi-fuzzy memberships from not encrypted soft typicalities and β using Eq. (20)
    CaCS: Compute final memberships from semi-fuzzy memberships and soft typicalities as in section 3.6
    CaCS: Compute encrypted centers from not encrypted final memberships and encrypted data as in Eq. (21)
    CaCS: Compute encrypted η from not encrypted memberships and encrypted distances Eq. (15)
    CaCS: Send encrypted centers and memberships to DO to update U, V
  Until (maximum number of iterations is reached or very small change in cluster centers in two successive iterations)

After computing initial soft memberships, these memberships are unencrypted and we can easily identify core points and update the membership according to section 3.4. Also, in each iteration, the objects having the highest memberships for each cluster are identified and the average distance is computed using their not-encrypted soft memberships and their already computed encrypted distance to these centers to update the penalty parameters η. Given the values η, the bin values are updated using Eq. (7) to be used in the next iteration. Finally, CaCS will send the results to the client.

i. Security Analysis

As stated in [29], DO's data is confidential, if the key is long enough, the attacker cannot obtain the plaintexts in polynomial time. Besides, CoCS has no additional information about DO's data. The leakage of the difference will not disclose any privacy of DO. Also, the magnitude relation that is given as additional information to CaCS is expressed as symbols representing the magnitude relation instead of the accurate difference values that make the protocol still secure. The only possible threat is that CaCS may conclude with CoCS.In this section, we will show how much the proposed scheme prevents the original data, intermediate results, or final results from being leaked.

The first threat issue, CaCS holds the ciphertexts of DO's original data, DO's data is not confidential:ciphertexts of SHE is a key-dependent message thus it is secure under the decisional composite residuality assumption [34] i.e., if the key is long enough, the attacker cannot obtain the plaintexts in polynomial time [35]. Thus, DO's data is confidential.

The second threat issue is that, because CoCS holds the incomplete re-encryption secret key, it may decrypt any ciphertexts it got to obtain DO's private information: only the re-encrypted difference (RD) that represents the difference between the distance and the selected value in the Bin vector is sent to CoCS. It is the ciphertext of the two data's differences. Through numerical analysis, CoCS can eliminate the $x^{\delta-1}$ and get the exact value of the difference. However, it is hard to recover DO's original data. Assuming CoCS gets the exact difference d, it can recover two data in the t-length plaintext space with a probability of 1 /(t−d). The probability decreases as the difference decreases. Besides, CoCS has no additional information about DO's data. Thus, the leakage of the difference will not help CoCS disclose any privacy of DO.

The third threat issue, CaCS will get additional information from the returns of CoCS that may help it access DO's data: the returns from CoCS are only symbols representing the magnitude relation instead of the accurate difference values. They can only be used to help CaCS to assign the appropriate soft memberships. Besides, the bin vector is not related to the data and the distance is a function of the data and centers. Furthermore, because SHE is not an order-preserving encryption, the corresponding plaintext information will not be leaked. Thus, the proposed scheme is still secure.

The fourth threat issue, CaCS may conclude with CoCS: cloud servers are required not to conclude with each other. The proposed scheme is secure without cloud server collusion. If CaCS and CoCS conclude with each other, i.e., if CaCS sends the ciphertexts of DO's data directly to CoCS then it can decrypt the re-encrypted ciphertexts with the incomplete decryption key [29]. It can obtain all DO's data amplified with $x^{\delta-1}$ times. The security of the proposed scheme in the case of cloud servers colluding with each other is out of the scope of this paper.

j. BGV Parameters

For the integer parts of the plaintext vectors that are encrypted in the BGV, the plaintext modulus p establishes an upper bound. For example, setting the plaintext modulus to $p = 255$ means that computing the product of an encrypted 16 and an encrypted 16 will overflow and produce the result $256 - 255 = 1$. The primary functional parameter that controls the scheme's ability to perform encrypted computations is the ciphertext modulus q. A ciphertext in the BGV scheme is composed of an array of $2h$ numbers between 0 and q -1. More operations can be carried out on encrypted data by increasing the parameter q. The security level of the scheme is determined for a given value of q by the ciphertext dimension h, with a bigger h indicating more security. The size of the plaintext vector that is encoded into each ciphertext is also influenced by the ciphertext dimension h at the same time. The plaintext vector's size is typically, but not always, equal to h. for example, to get any kind of security with q=65537 we need at least h=512 and H = ceil((2h+1) log q). It is recommended to choose value of h = 8192 to support q moduli of size b up to 218 bits.

k. Computation and communication cost Analysis

In this section, the complexity of the presented Privacy-Preserving semi-fuzzy possibilistic c-means algorithm will be analyzed in terms of computation cost and communication cost. The computation cost is estimated using the BGV parameters stated above in terms of ADD, MUL and MOD to denote the computation cost of one addition operation, one multiplication operation and one modulus operation on Ring R , respectively. As shown in algorithm 6, the client encrypts the raw objects only once before uploading the encrypted data to CaCS. The client encrypts the dataset with O(nd(h + 1) H (ADD + MUL)) where n is the number of objects and d is the number of attributes of each object. During each iteration, the CaCS calculates the distances to centers on the encrypted data. Also, CaCS compute the difference between distance matrix and the bin vector which costs
nc(log r) (MUL+ADD) where r is the number of discretization levels and re-encrypts the difference and c is the number of clusters. The re-encryption costs O ([nc(log r)(h + 1) H(ADD + MUL)]). CaCS sends the results to CoCS. Also, each time CoCS receives the re-encrypted differences it decrypts the re-encrypted differences and return the results to CaCS.The decryption costs [nc(log r)(h + 1) MUL+ nchADD + 2 nc MOD]. Moreover, the CaCS performs (h + 1)(4 n(c + 1)(2 H + h) + 8 H + 1) MUL, (h + 2)(2 n (c + 1) + 4 n + 1) H ADD and (h + 1)(((6 n (c + 1) + c)(h + 2)) + 4 h) MODfor calculating the distances. While CoCS performs hnc(log r)MOD for computing the magnitude relation in each iteration. The computation of the centers costs (n-1) ADD + n MUL since the dominator is not encrypted the division will be one multiplication with cypher texts. Computing the typicality and fuzzy memberships is done on not encrypted data. Computing the initial penalty parameters costs $2nc^2$ (ADD+MUL).

Communication cost. At the beginning, the client uploads ndmessages with nd(h + 1) b bits to cloud. During each iteration, the client exchanges $2c(d+n)$ messages with $2c(d+n)$(h + 1) b bits with the cloud to decrypt and encrypt the intermediate results of the cluster centers. Also, in each iteration, the CaCS exchanges $2nc$ *(log r)* messages with $2nc$ *(log r)* (h + 1) b bits.

Comparing the complexity of the proposed algorithm to the SWPCM algorithm, we found that calculating the memberships and the centers are much less complex in the proposed algorithm with the added cost of re-encryption and decryption and the communication cost between CaCS and CoCS. Also, the CoCS can be a service that run on the same hardware of the CaCS.

4. EXPERIMENTAL RESULTS

The adopted datasets[36] are described in the first and second columns of Table 2. We used the Iris data in our work's investigation. One of the first and most popular data sets in data mining is iris data. There are 150 samples from three different Iris flower types included in the Iris data set. Four characteristics, including the length and width of the sepal and the length and width of the petal, are present in every sample of iris flowers. The 5800 samples in the shuttle dataset each include 9 properties, all of which are numerical, and they all fall into one of five classifications. A digitized image of a fine needle aspirate (FNA) of a breast mass has 569 samples and 9 numerical features that define properties of the cell nuclei present. These samples make up the Breast dataset. Features in the Wine dataset include the amounts of 13 constituents (including alcohol, malic acid, and others) identified through chemical analysis of 178 samples of wines produced in the same Italian region but coming from three distinct cultivars. For each of the 17 pulses of the Goose Bay system, the 34 characteristics of the Ionosphere dataset correspond to 17 complex values that show the autocorrelation between a pulse's timing and pulse number. There are no missing data and numerical attributes in any of the datasets we used in our experiments. As a metric for sample dissimilarity, Euclidean distance is employed. For any dataset with more than 10 attributes, the Principal Component Analysis algorithm provided in[37] is utilized to limit the dimension to 10.

Table 2 presents the performance indices and runtime of the proposed PPS-FPCM algorithm compared to another six related methods on five standard datasets [38].As shown in Table 2, SPCM outperforms all algorithms on both Wine and Ionosphere datasets in terms of VMI, AMI, and ARI. FCM is the best algorithm on the Breast dataset and PPS-FPCM is the second best. On the Iris dataset, the results of the six algorithms are close to each other, PPS-FPCM is the best in terms of VMI while PFCM in terms of AMI and ARI. The runtime of SPCM is slightly higher than HCM and less than the runtime of any of the other algorithms. Table 3 shows the iteration time, number of iterations, total runtime, silhouette index, maxdist, mindist, and sumdist for five local minima computed using different random initialization on the Iris dataset. The maxdist, mindist, and sumdist correspond to the square root of the maximum, minimum and sum of g_i over i in Eq. (22) where v_i^*denotes the actual center for cluster i. The results show that the proposed algorithm requires a higher number of iterations than WPCM and the cost of a single iteration is about 67% of the cost of a WPCM iteration. PPS-FPCM is the best in terms of silhouette index, AMI and ARI and slightly lower than the best in terms of other indices.

$$g_i = \|v_i^* - v_i\|^2 \quad (22)$$

Table 2 Performance of PPS-FPCM compared to five related clustering algorithms on five standard datasets

Dataset	Samples/ Features/ Classes	Metric	Algorithms						
			HCM	PCM	FCM	PFCM	WPCM	SPCM	PPS-FPCM
Wine	178/13/3	VMI	0.79	0.736	0.424	0.475	0.834	0.842	**0.851**
		AMI	0.79	0.734	0.417	0.468	0.833	0.840	**0.847**
		ARI	0.81	0.747	0.380	0.373	0.854	0.881	**0.883**
		Avg. Runtime (sec)	0.015	0.156	0.156	0.131	0.265	**0.091**	0.244
Iris	150/4/3	VMI	0.65	0.671	0.645	0.691	0.643	0.673	**0.698**
		AMI	0.65	0.668	0.640	0.687	0.639	0.668	**0.693**
		ARI	0.62	0.649	0.607	0.664	0.621	0.658	**0.668**
		Avg. Runtime (sec)	0.015	0.140	0.124	**0.125**	0.374	0.082	0.198
Ionosphere	351/34/2	VMI	0.56	0.581	0.478	0.564	0.609	**0.611**	0.605
		AMI	0.56	0.581	0.477	0.564	0.608	**0.617**	0.601
		ARI	0.68	0.670	0.594	0.684	0.669	0.701	**0.709**
		Avg. Runtime (sec)	0.031	0.203	0.156	0.141	0.296	**0.136**	0.251
`Breast	683/9/2	VMI	0.56	0.609	**0.639**	0.558	0.485	0.560	0.626
		AMI	0.56	0.608	**0.639**	0.558	0.484	0.571	0.611
		ARI	0.68	0.670	**0.750**	0.677	0.568	0.687	0.698
		Avg. Runtime (sec)	0.031	0.281	0.203	0.149	0.359	**0.125**	0.233
Shuttle	58000/9/5	VMI	0.32	0.274	0.105	0.194	**0.312**	0.291	0.234
		AMI	0.32	0.274	0.104	0.194	**0.312**	0.290	0.229
		ARI	0.25	0.217	0.064	0.125	0.147	**0.235**	**0.244**
		Avg. Runtime (sec)	5	208	277	218	233	**143**	455

Table 3: Detailed Performance on Iris Dataset

algorithm	Runtime per iteration(ms)	total runtime (ms)	iteration count	no. local min.	VMI	AMI	ARI	Silhouette index	maxdist	mindist	sumdist
PCM	44.501	2492.078	56	5	0.643	0.639	0.621	0.454	0.766	0.707	2.121
FCM	31.138	3176.032	102	5	**0.684**	0.680	0.653	0.454	**0.523**	**0.379**	**1.138**
SPCM	35.650	1568.614	44	5	0.673	0.668	0.659	0.459	0.791	0.703	2.108
PFCM	48.428	1791.819	37	5	0.607	0.687	0.664	0.451	0.708	0.560	1.681
WPCM	120.981	2419.626	20	5	0.641	0.636	0.611	0.457	0.939	0.787	2.359
HCM	1.1330	19275.234	17009	5	0.673	0.669	0.641	0.448	0.596	0.423	1.168
PPS-FPCM	81.220	1698.512	48	5	0.628	**0.693**	**0.668**	**0.462**	0.843	0.761	2.121

5. CONCLUSION

In this paper, a Privacy-Preserving semi-fuzzy possibilistic c-means algorithm is presented for data clustering. The main idea is to avoid the need for exponentiation and division in each iteration without sacrificing accuracy by limiting the memberships to a set of discrete values between zero and one differ by a small fixed step that simplifies the computation of centers and memberships, deals with noise, and adapts the algorithm parameters during execution. Also, the paper describes a procedure for securely implementing the proposed algorithm on encrypted data using the incomplete reencryption scheme. Experimental results and complexity analysis of the proposed soft possibilistic c-means algorithm (PPS-FPCM) demonstrate the following points:

- Restricting the memberships to a set of discrete values reduces the complexity of PPS-FPCM.
- The algorithm can be securely implemented on the cloud using the existing incomplete re-encryption method which has the advantage that the data owner (DO) does not have to pre-process the data. Only the cipher texts of the data and the incomplete re-encryption key pair are involved.
- However, this protocol is not secure with cloud server's collusion.

In future work, Micro-service implementation will be investigated to implement the PPS-FPCM algorithm on the cloud without the disclosure of private data.

CONFLICTS OF INTEREST

The authors declare that there are no conflicts of interest. No funding is declared.

AUTHOR CONTRIBUTIONS

Mohamed Mahfouz was responsible for the study concept, design, and implementation, while Zeinb was responsible for the preparation and interpretation of input data and the results analysis and Ehab was responsible for critical revision of the privacy and security part. Both authors were responsible for drafting the manuscript and critical revision of the machine learning part of the manuscript.

REFERENCES

1. Al-Fuqaha, A., et al., Internet of things: A survey on enabling technologies, protocols, and applications. IEEE communications surveys & tutorials, 2015. 17(4): p. 2347-2376.
2. Deng, X., et al., Confident information coverage hole healing in hybrid industrial wireless sensor networks. IEEE Transactions on Industrial Informatics, 2017. 14(5): p. 2220-2229.
3. Zhao, Z., et al., Link-correlation-aware data dissemination in wireless sensor networks. IEEE Transactions on Industrial Electronics, 2015. 62(9): p. 5747-5757.
4. ul Islam, F.M.M. and M. Lin, Hybrid DVFS scheduling for real-time systems based on reinforcement learning. IEEE Systems Journal, 2015. 11(2): p. 931-940.
5. Zhang, Q., et al., An improved deep computation model based on canonical polyadic decomposition. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2017. 48(10): p. 1657-1666.
6. Zhang, Q., et al., A survey on deep learning for big data. Information Fusion, 2018. 42: p. 146-157.
7. Khanmohammadi, S., N. Adibeig, and S. Shanehbandy, An improved overlapping k-means clustering method for medical applications. Expert Systems with Applications, 2017. 67: p. 12-18.
8. Bezdek, J.C., Objective function clustering, in Pattern recognition with fuzzy objective function algorithms. 1981, Springer. p. 43-93.

9. Krishnapuram, R. and J.M. Keller, The possibilistic c-means algorithm: insights and recommendations. IEEE transactions on Fuzzy Systems, 1996. 4(3): p. 385-393.
10. Krishnapuram, R. and J.M. Keller, A possibilistic approach to clustering. IEEE transactions on fuzzy systems, 1993. 1(2): p. 98-110.
11. Rumelhart, D.E. and D. Zipser, Feature discovery by competitive learning. Cognitive science, 1985. 9(1): p. 75-112.
12. Yang, M.-S., W.-L. Hung, and D.-H. Chen, Self-organizing map for symbolic data. Fuzzy Sets and Systems, 2012. 203: p. 49-73.
13. Wang, W., J. Yang, and R. Muntz. STING: A statistical information grid approach to spatial data mining. in VLDB. 1997.
14. Ester, M., et al. A density-based algorithm for discovering clusters in large spatial databases with noise. in Kdd. 1996.
15. Murtagh, F. and P. Contreras, Algorithms for hierarchical clustering: an overview, II. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 2017. 7(6): p. e1219.
16. Yu, H. and J. Fan, Cutset-type possibilistic c-means clustering algorithm. Applied Soft Computing, 2018. 64: p. 401-422.
17. Xenaki, S.D., K.D. Koutroumbas, and A.A. Rontogiannis, A novel adaptive possibilistic clustering algorithm. IEEE Transactions on Fuzzy Systems, 2015. 24(4): p. 791-810.
18. Armbrust, M., et al., A view of cloud computing. Communications of the ACM, 2010. 53(4): p. 50-58.
19. Zhang, Q. and Z. Chen, A weighted kernel possibilistic c-means algorithm based on cloud computing for clustering big data. International Journal of Communication Systems, 2014. 27(9): p. 1378-1391.
20. Havens, T.C., et al., Fuzzy c-means algorithms for very large data. IEEE Transactions on Fuzzy Systems, 2012. 20(6): p. 1130-1146.
21. Zhang, Q., et al., A High-Order Possibilistic $ C $-Means Algorithm for Clustering Incomplete Multimedia Data. IEEE Systems Journal, 2015. 11(4): p. 2160-2169.
22. Zhang, Q., et al., Privacy-preserving double-projection deep computation model with crowdsourcing on cloud for big data feature learning. IEEE Internet of Things Journal, 2017. 5(4): p. 2896-2903.
23. Zhang, Q., et al., Secure weighted possibilistic c-means algorithm on cloud for clustering big data. Information Sciences, 2019. 479: p. 515-525.
24. Selim, S.Z. and M.A. Ismail, Soft clustering of multidimensional data: a semi-fuzzy approach. Pattern Recognition, 1984. 17(5): p. 559-568.
25. Lu, Q., et al. Secure collaborative outsourced data mining with multi-owner in cloud computing. in 2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications. 2012. IEEE.
26. Mahfouz, M.A. and M. Ismail. Fuzzy relatives of the CLARANS algorithm with application to text clustering. in Proceedings of World Academy of Science, Engineering and Technology. 2009. Citeseer.
27. Mahfouz, M.A. and M.A. Ismail. Efficient soft relational clustering based on randomized search applied to selection of bio-basis for amino acid sequence analysis. in 2012 Seventh International Conference on Computer Engineering & Systems (ICCES). 2012. IEEE.
28. Mahfouz, M.A. and M.A. Ismail. Soft flexible overlapping biclustering utilizing hybrid search strategies. in International Conference on Advanced Machine Learning Technologies and Applications. 2012. Springer.
29. Jiang, L., et al., An effective comparison protocol over encrypted data in cloud computing. Journal of Information Security and Applications, 2019. 48: p. 102367.
30. Mohamed, A., SPCM: Efficient semi-possibilistic c-means clustering algorithm. 2022.
31. Zhang, Y., et al., Anonymous attribute-based proxy re-encryption for access control in cloud computing. Security and Communication Networks, 2016. 9(14): p. 2397-2411.
32. Xu, P., et al., Conditional identity-based broadcast proxy re-encryption and its application to cloud email. IEEE Transactions on Computers, 2015. 65(1): p. 66-79.
33. Khan, A.N., et al., Incremental proxy re-encryption scheme for mobile cloud computing environment. The Journal of Supercomputing, 2014. 68(2): p. 624-651.
34. Malkin, T., I. Teranishi, and M. Yung. Efficient circuit-size independent public key encryption with KDM security. in Annual International Conference on the Theory and Applications of Cryptographic Techniques. 2011. Springer.
35. Applebaum, B., Key-dependent message security: Generic amplification and completeness. Journal of cryptology, 2014. 27(3): p. 429-451.
36. Dheeru, D. and E.K. Taniskidou, UCI machine learning repository. University of California, Irvine, School of Information and Computer Sciences. 2017.
37. Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011. Available from: https://scikit-learn.org/stable/about.html.
38. UC Irvine Machine Learning Repository available at: http://archive.ics.uci.edu/ml/datasets.

