CHAPTER 2

Principles and Theories of the Study

In this section, we give the detail of cluster analysis, a description of conventional clustering methods of interest, i.e. hard C-means, fuzzy C-means, fuzzy C-medians, possibilistic C-means, fuzzy possibilistic C-means, possibilistic fuzzy C-means, unsupervised possibilistic C-means and unsupervised possibilistic fuzzy C-means. Moreover, string grammar, string grammar clustering, fuzzy clustering validation techniques and measuring of overlapping data are briefly describing.

2.1 Cluster Analysis

The one of important tool in pattern recognition field is clustering analysis [37]. A clustering is an unsupervised technique used to group data objects in the set based on similarity (or dissimilarity). The algorithm assigns all data objects in dataset to cluster that are more similar. There are various types of clustering algorithm such as partitioning algorithms, hierarchy algorithms, model-based algorithms, etc. In this thesis we only use partitioning algorithms for all clustering methods. Among various techniques that have been proposed in literatures for clustering task, we focus only on prototype-based approaches in this thesis. The prototype-based methods will describe the data set to group by using the prototypes or cluster center of each group. Each prototype will describe the distribution of data in group based on a concept of similarity to the prototype. There are several prototype-based clustering algorithms has been proposed, i.e., hard C-means, fuzzy C-means, fuzzy C-medians, fuzzy possibilistic C-means, fuzzy possibilistic C-means, possibilistic fuzzy C-means, unsupervised possibilistic C-means and unsupervised possibilistic fuzzy C-means, etc.

2.2 Hard Clustering Algorithm

2.2.1 Hard C-Means

Hard C-means or k-means clustering (HCM) [38] is the one of the clustering tool of unsupervised pattern recognition algorithm. To assign each given data in the given set to the group that minimizes the distance between each given data and each other's in the same group. The algorithm of Hard C-means clustering aims at minimizing an objective function given by:

$$J(\mathbf{U}, \mathbf{X}) = \sum_{k=1}^{N} \sum_{i=1}^{C} d_{ik}^2 \tag{2.1}$$

where, $d_{ik}^2 = \|\vec{x}_k - \vec{c}_i\|_1$ is the distance between $\vec{x}_k$ and $\vec{c}_i$, C is the number of cluster centers and N is the number of data points in i^{th} cluster.

However, partitioning of the data of hard clustering such that each data point belongs to exactly one of the partitions, the accuracy and efficiency of hard clustering have been very poor for some applications with overlapping data.

2.3 Fuzzy Clustering Algorithm

Fuzzy sets theory was first proposed by L. A. Zadeh [6]. Let a conventional crisp subset A of a universal set of objects U be regularly determined by the samples of the universe that are members of A. U_A: $U \rightarrow \{0, 1\}$ for all $x \in U$ where

$$U_A(x) = \begin{cases} 1 & x \in A \\ 0 & x \notin A \end{cases} \tag{2.2}$$

When we have the given set $\{\vec{x}_k, ..., \vec{x}_n\}$, fuzzy clustering can identify the group of these samples vector with the membership degree of each data in each c cluster. It is showed by the matrix **U** where $u_{ik} = u_i(\vec{x}_k)$ for $i = 1,\ldots,c,$ and $k = 1, \ldots, n$. The u_{ik} is utilized for showing the membership degree of $\vec{x}_k$ in class i. The properties of membership degree for fuzzy clustering can be described as following equation [39]:

$$\sum_{i=1}^{c} u_{ik} = 1,\ 0 < \sum_{k=1}^{n} u_{ik} < n,\ u_{ik} \in [0,1] \tag{2.3}$$

The benefit of fuzzy membership can describe the detail of each data by using the degree of membership. It has more advantage than crisp clustering. The crisp clustering can just classify by using binary value (zero or one) and especially advantageous if data patterns are overlapping data [39].

2.3.1 Fuzzy C-Means

Fuzzy C-means (FCM) is a method of clustering which allows one of object to belong partially to several clusters which is improved by [9] in 1981. The method is widely used in pattern recognition for a long time. This algorithm is iterative technique that membership value is assigned to each of object. The object that places far away from the cluster center will have a low membership value to that cluster and another object that places close to the cluster center will have a high membership value to that cluster. The FCM is based on minimization of the following objective function which FCM differs from the HCM objective function by the addition of the membership values and the fuzzifier (*m*):

$$J(\mathrm{U},\mathrm{X}) = \sum_{k=1}^{N}\sum_{i=1}^{C} u_{ik}^{m} d_{ik}^{2} \tag{2.4}$$

where $m \in (1,\infty)$, u_{ik} is the membership value of $\vec{x}_k$ in the cluster *i*, $d_{ik}^2 = \|\vec{x}_k - \vec{c}_i\|_2$ is the distance between $\vec{x}_k$ and $\vec{c}_i$.

Fuzzy clustering achieves through an iterative process of the objective function shown in (2.4), with the update of membership u_{ik} and the cluster center c_i are:

$$u_{ik} = \frac{1}{\sum_{j=1}^{C} \left(\frac{d_{ik}}{d_{jk}}\right)^{2/(m-1)}} \tag{2.5}$$

and

$$c_i = \frac{\sum_{k=1}^{N} u_{ik}^{m} \cdot \vec{x}_k}{\sum_{k=1}^{N} u_{ik}^{m}} \tag{2.6}$$

This iteration will stop when $\max_{ik}\left\{|u_{ik}^{(t+1)}-u_{ik}^{(t)}|\right\}<\varepsilon$, where ε is a termination criterion. This algorithm will converge to a local minimum of *objective function*.

2.3.2 Possibilistic C-Means

Fuzzy C-means clustering [9] is sensitive to noises or outliers [2] because of the probabilistic constraint. Krishnapuram and Keller proposed the possibilistic C-means (PCM) [10] algorithm by abandoning the probabilistic constraint of FCM to solve the noise sensitivity of FCM. In [10] suggest to use terminal outputs of Fuzzy c-means as a good way to initialize possibilistic C-means. They also suggest choosing the $\vec{\eta}_i$ by computing:

$$\vec{\eta}_i = K\frac{\sum_{k=1}^{N}(u_{ik}^{m})d_{ik}}{\sum_{k=1}^{N}u_{ik}^{m}};\ K>0 \text{ (Typically } K \text{ is chosen to be 1.)} \qquad (2.7)$$

The objective function of possibilistic C-means algorithm can be described by the following equation:

$$J_{m,\eta}(\mathbf{U},\mathbf{C};X)=\sum_{k=1}^{N}\sum_{i=1}^{C}(u_{ik}^{m})d_{ik}^{2}+\sum_{i=1}^{C}\eta_i\sum_{k=1}^{N}(1-u_{ik})^{m} \qquad (2.8)$$

with the constraint: $m>1$, η and $d_{ik}>0$ where C and N are the total number of clusters and total number of input vectors, respectively, and $d_{ik}^{2}=\|\vec{x}_k-\vec{c}_i\|_2$. Hence, we can use equation (2.9) for updating the membership degree of $\vec{x}_k$ in each cluster.

$$u_{ik}=\frac{1}{1+\left(\frac{d_{ik}^{2}}{\eta_i}\right)^{1/(m-1)}} \qquad (2.9)$$

and the center $\vec{c}_i$ is calculated by:

$$\vec{c}_i = \frac{\sum_{k=1}^{N} (u_{ik}^m)\vec{x}_k}{\sum_{k=1}^{N} (u_{ik}^m)}. \quad (2.10)$$

2.3.3 Fuzzy C-Medians

The fuzzy C-Medians [11, 12] is one of the popular clustering algorithms. Let $\mathbf{X} = \left\{ \vec{x}_j \,\middle|\, j = 1 \ldots N \right\}$ be a set of *N* data. Let $V = (\vec{c}_1, \ldots, \vec{c}_C)$ represent a vector of *C* prototypes. The objective function of fuzzy C-Medians is as follows:

$$J(\mathbf{U}, \mathbf{X}) = \sum_{k=1}^{N} \sum_{i=1}^{C} u_{ik}^m d_{ik} \quad (2.11)$$

with the constraint: $\sum_{i=1}^{C} u_{ik} = 1$ for k = 1 to *N*, where *C* and *N* are the total number of clusters and total number of input vectors, respectively. The membership degree of each data in each cluster for this algorithm is in the range of [0,1] and $d_{ik} = \left\| \vec{x}_k - \vec{c}_i \right\|_1 = \sum_{j=1}^{p} \left| x_{kj} - c_{ij} \right|$. In equation (2.10), $m \in [1,\infty)$ is called the fuzzifier. The membership update equation of $\vec{x}_k$ in each cluster *i* is:

$$u_{ik} = \frac{1}{\sum_{j=1}^{C} \left(\frac{d_{ik}}{d_{jk}} \right)^{1/(m-1)}}. \quad (2.12)$$

The center $\vec{c}_i$ is calculated differ from fuzzy C-median by finding a fuzzy median in cluster *i* with memberships u_{ij}^m using

$$\Psi_{fuzzy}(med_{il}) = \sum_{k=1}^{N} u_{ik}^m \operatorname{sgn}(x_{kl} - med_{il}) \quad \text{for } l = 1, \ldots, p, \quad (2.13)$$

where u_{ik} is the membership of vector $\vec{x}_k$ in cluster *i*. The root (m_{il}) of equation 2.13 is a fuzzy median in l^{th} dimension of cluster *i.*

2.3.4 Fuzzy Possibilistic C-Means

The Fuzzy Possibilistic C-Means (FPCM) [13] is one of the powerful clustering problems. It solves the noise problem of FCM, and also solves the coincident clusters problem of PCM [10]. Let X = $\{\vec{x}_j \mid j = 1...N\}$ be a set of *N* feature vectors in *p*-dimensional feature space. Let $B = (\vec{c}_1,...,\vec{c}_c)$ represent a *C*-tuple of prototypes each of which characterizes one of the *C* clusters. The objective function is as follows:

$$J_{m,\eta}(\mathbf{U},\mathbf{T},\mathbf{C};X) = \sum_{k=1}^{N}\sum_{i=1}^{C}(u_{ik}^{m} + t_{ik}^{\eta})d_{ik}^{2} \tag{2.14}$$

with the constraint: $m > 1, \eta > 1, \sum_{k=1}^{N} u_{ik}^{m} = 1, \sum_{i=1}^{C} t_{ik}^{\eta} = 1$, for $k = 1$ to *N*, where *C* and *N* are the total number of clusters and total number of input vectors, respectively, and $d_{ik}^{2} = \|\vec{x}_k - \vec{c}_i\|_2$. Here, $t_{ik} \in [0,1]$ is the typicality value of data object $\vec{x}_k$ in each cluster. In equation (2.14), *m*is called the fuzzifier and η is suitable positive numbers. The membership update equation of $\vec{x}_k$ in each cluster *i* is:

$$u_{ik} = \frac{1}{\sum_{j=1}^{C}(\frac{d_{ik}}{d_{jk}})^{2/(m-1)}}. \tag{2.15}$$

In each iteration we can use this following equation for updating the typicality of each data in each cluster *i*.

$$t_{ik} = \frac{1}{\sum_{j=1}^{N}(\frac{d_{ik}}{d_{jk}})^{2/(m-1)}}. \tag{2.16}$$

The center $\vec{c}$ is calculated by:

$$\vec{c}_i = \frac{\sum_{k=1}^{N}(u_{ik}^{m} + t_{ik}^{\eta})\vec{x}_k}{\sum_{k=1}^{N}(u_{ik}^{m} + t_{ik}^{\eta})}. \tag{2.17}$$

2.3.5 Possibilistic Fuzzy C-Means Algorithm

The possibilistic fuzzy C-means (PFCM) [14] is the algorithm which solves various problems of FCM, PCM and FPCM. The PFCM can solves the noise problem of FCM, solve the coincident clusters problem of PCM and eliminates the row sum constraints on the typical values of FPCM but retain the column constraint on the membership values. The objective function of PFCM is as follows:

$$J_{m,\eta}(\mathbf{U},\mathbf{T},\mathbf{V};X)=\sum_{k=1}^{N}\sum_{i=1}^{C}(au_{ik}^{m}+bt_{ik}^{\eta})d_{ik}^{2}+\sum_{i=1}^{C}\gamma_i\sum_{k=1}^{N}(1-t_{ik})^{\eta} \tag{2.18}$$

with the constraint: $m>1,\eta>1,a>0,b>0,\gamma>0,\sum_{i=1}^{C}u_{ik}=1$ for k = 1 to N, and $0\le u_{ik},t_k\le 1$ where X = $\{\vec{x}\mid j=1,...,N\}$ be a set of N feature vectors in p-dimensional feature space. V = $(\vec{c}_1,...,\vec{c}_C)$ represent a C-tuple of prototypes each of which characterizes one of the C clusters. C and N are the total number of clusters and total number of input vectors, respectively, and $d_{ik}^2=\|\vec{x}_k-\vec{c}_i\|_2$. In equation (2.18), $m\in$ is called the fuzzifier and η is a suitable positive number, a and b define the relative importance of fuzzy membership and typical values in the objective function. The membership update equation of $\vec{x}_k$ in each cluster i for $1\le i\le C;1\le k\le N$ is:

$$u_{ik}=\frac{1}{\sum_{j=1}^{C}\left(\frac{d_{ik}}{d_{jk}}\right)^{2/(m-1)}}. \tag{2.19}$$

In each iteration we can use this following equation for updating the typicality of each data in each cluster i.

for $1\le i\le C;1\le k\le N$ is:

$$t_{ik}=\frac{1}{1+\left(\frac{b}{\gamma_i}d_{ik}^{2}\right)^{\frac{1}{\eta-1}}}. \tag{2.20}$$

The center $\vec{c}_i$ is calculated by:

$$\vec{c}_i = \frac{\sum_{k=1}^{N} (au_{ik}^m + bt_{ik}^\eta)\vec{x}_k}{\sum_{k=1}^{N} (au_{ik}^m + bt_{ik}^\eta)} \tag{2.21}$$

2.3.6 Unsupervised Possibilistic Clustering Algorithm

The unsupervised possibilistic clustering Algorithm (UPCM) [15] is another posibilistic clustering algorithm which is based on the FCM objective function, the partition coefficient (PC) and partition entropy (PE) validity indexes. The objective function of UPCM is

$$J(\mathbf{U}, X) = \sum_{k=1}^{N}\sum_{i=1}^{C} u_{ik}^m d_{ik}^2 + \frac{\beta}{m^2\sqrt{C}} \sum_{i=1}^{C}\sum_{k=1}^{N} (u_{ik}^m \log u_{ik}^m - u_{ik}^m), \tag{2.22}$$

Where β, m and c are all positive. The membership update equation of $\vec{x}_k$ in each cluster i is:

$$u_{ik} = \exp\left(-\frac{m\sqrt{C}d_{ik}^2}{\beta} \right). \tag{2.23}$$

The parameter β is a normalization term that measure the degree of separation of the dataset, β is define as the sample co-variance is:

$$\beta = \frac{\sum_{j=1}^{n} \left\| x_j - \bar{x} \right\|^2}{n} \text{ with } \bar{x} = \frac{\sum_{j=1}^{n} x_j}{n} \tag{2.24}$$

The center $\vec{c}_i$ is calculated using

$$\vec{c}_i = \frac{\sum_{k=1}^{N} (u_{ik}^m)\vec{x}_k}{\sum_{k=1}^{N} u_{ik}^m}, \tag{2.25}$$

2.3.7 Unsupervised Possibilistic Fuzzy Clustering Algorithm

The unsupervised possibilistic fuzzy C-means (UPFCM) [16] combine possibilistic fuzzy C-Means and UPCM to solve the problem of generating coincident clusters of UPCM. UPFCM combines the characteristics of both fuzzy and possibilistic C-means from properties of PFCM and the partition coefficient (PC) and partition entropy (PE) validity indexes from properties of UPCM. Hence, UPFCM should be able to deal more effectively with noise, overlapping and outliers.

Now, we will briefly describe the unsupervised possibilistic-fuzzy C-means (UPFCM) [24] Suppose X = $\{\vec{x} \mid j = 1,..., N\}$ be a set of N feature vectors in p-dimensional feature space. $V = (\vec{c}_1,...,\vec{c}_C)$ represents a C-tuple of prototypes each of which characterizes one of the C clusters, **U** is a membership matrix $[u_{ik}]_{C\times N}$, **T** is a typicality matrix $[t_{ik}]_{C\times N}$, and $d_{ik} = \|\vec{x}_k - \vec{c}_i\|_2$. Then, the objective function of UPFCM is

$$J_{m,\eta}(\mathbf{U},\mathbf{T},\mathbf{V};X) = \sum_{k=1}^{N}\sum_{i=1}^{C}(au_{ik}^{m} + bt_{ik}^{\eta})d_{ik}^{2} + \frac{\beta}{\eta^2\sqrt{C}}\sum_{i=1}^{C}\sum_{k=1}^{N}(t_{ik}^{\eta}\log t_{ik}^{\eta} - t_{ik}^{\eta}), \quad (2.26)$$

with the constraint: $m > 1, \eta > 1, a > 0, b > 0, \sum_{i=1}^{C} u_{ik} = 1$ for k = 1 to N, and $0 \le u_{ik}, t_k \le 1$ In equation (2.26), m is called the fuzzifier and η is a suitable positive number. Parameters a and b define the relative importance of fuzzy membership and typical values in the objective function. The membership update equation of $\vec{x}_k$ in each cluster i for $1 \le i \le C; 1 \le k \le N$ is:

$$u_{ik} = \frac{1}{\sum_{j=1}^{C}\left(\frac{d_{ik}}{d_{jk}}\right)^{2/(m-1)}}. \quad (2.27)$$

The typicality update equation of $\vec{x}_k$ in each cluster i for $1 \le i \le C; 1 \le k \le N$ is:

$$t_{ik} = \exp\left(-\frac{b\eta\sqrt{C}d_{ik}^{2}}{\beta}\right). \quad (2.28)$$

The center $\vec{c}_i$ is calculated by:

$$\vec{c}_i = \frac{\sum_{k=1}^{N} (au_{ik}^m + bt_{ik}^\eta)\vec{x}_k}{\sum_{k=1}^{N} (au_{ik}^m + bt_{ik}^\eta)} \tag{2.29}$$

2.4 String Grammar

An alphabet *V* is a finite set of symbols, such as the binary set {0, 1} or the set {*A, a, B, b, . . ., Z, z*} of uppercase and lowercase letters. A sentence *x* over alphabet *V* is a string of finite length formed with symbols from *V*. (The words sentence and string are used synonymously.) The length of *x*, denoted by |*x*|, is the number of symbols used in its formation [2]. A sentence (string) over *V* is either a single symbol from *V* or a sequences of symbols formed by concatenation of 0 or more symbol from *V* [2]. A language is a set of sentences over an alphabet; that is, a language over alphabet *V* is a finite or countably infinite subset of *V** (set of all sentences over *V*). [2]

2.4.1 Formal Definitions of Grammar

Formal definition of grammar was originally introduced by Noam Chomsky [2]. A grammar Gr is defined as a four-tuples:

$$\text{Gr} = (N, T, P, S) \tag{2.30}$$

The equation (2.30) consists of the following element:

N finite set of nonterminal symbols which is disjoint set with the strings from grammar

T finite set of terminal symbols, which is disjoint set with the strings from finite set of nonterminal symbols $N \cup T = V$, $N \cap T = \lambda$

ST start symbol, $ST \in N$

P finite set of productions

*W** is the set of all finite length

W(alphabet) is a non-empty set of symbols or finite set

λ is empty string.

the null string, $W^{+} = W^{*} - \{ \lambda \}$.

We have used the traditional notation of syntactic pattern recognition for the terminals (V_N) and non-terminals (V_T) in a grammar G, which are sometimes called the vocabularies of G. The set $V = V_N \cup V_T$, with $V_N \cap V_T$ =empty set called the alphabet or total vocabulary of G.

2.4.2 String Distance Function

In mathematics and computer science, a string distance function is a metric that measures distance between two given strings sequence for approximate string matching or comparison and for fuzzy string searching.

The most widely well-known string metric is a the Levenshtein Distance (also known as Edit Distance) [2]. The Levenshtein distance between the two given strings sequence is equal to the minimum number of symbol which requires to change each string sequence into others with edits operation, i.e., insertion, deletion, or substitution.

The Levenshtein distance is fulfillment of the triangle inequality, whereas the most other string distance measurements this property does not hold. Moreover, this distance can be computed in the quadratic time with respect to the length of the two strings under consideration.

The algorithm of Levenshtein distance [2] between two strings *x, y*:

$$lev_{x,y}(i,j) = \begin{cases} \max(i,j) & \text{,if } \min(i,j) = 0 \\ \min \begin{cases} lev_{x,y}(i-1,j)+1 \\ lev_{x,y}(i,j-1)+1 \\ lev_{x,y}(i-1,j-1)+\left[x_i \neq y_j \right] \end{cases} & \text{,else} \end{cases} \quad (2.31)$$

Where *i* is an index of symbol of string *x* and *j* is a character of string *y*

Here we show the example of Levenshtein distance calculation, the Levenshtein distance between two given string “fast” and “basely” is 4. The steps to

change “fast” into “basely” are: (1) replace “f” with “b”, (2) replace “t” with “e”, (3) insert "l" after “e” and (4) insert "y" at the end.

2.4.3 String Grammar Clustering

The string grammar clustering algorithm [2, 19] so far is the hard clustering (sgHCM). The Levenshtein distance is used to measure distance between strings. One crucial part of the syntactic pattern is the distance metric. One of the distance metrics, that is more preferable because of its calculation flexibility between strings with different length, is the Levenshtein distance The sgHCM use the 1-nearest prototype (1-np) and 1-nearest multiple prototypes (1-nmp) rules, this technique need prototypes. Fu [2] calls the string S_k, selected from the N_i strings $\{ S_k;\ k = 1 \ldots N_i\}$ in the i-th class whose indices satisfy the cluster center of the N_i strings S_k as (2.32).

$$q = \arg\min_{1 \le j \le N_i} \left\{ c_j^i = \sum_{k=1}^{N} \frac{Lev(s_j, s_k)}{N_i} \right\} \tag{2.32}$$

We can describe the process of the sgHCM algorithm as the following psedocode:

Store n unlabeled finite strings $X = \{ \alpha_k; k = 1,...,n\}$

Initial string prototypes for all C classes

Do {

Compute Levenshtein distance (Dik) [1] (a smallest number of transformations needed to derive one string from the other) between input string k and cluster prototype i

Update membership using

$$u_{ik} = \begin{cases} 1 \; ; & D_{ik} < D_{jk} \;\; for \;\; j \neq k \\ 0 \; ; & otherwise \end{cases}$$

Update center string of each cluster i (α_q^i) where

$$q = \underset{1 \le j \le N_i}{\arg\min} \left\{ c_j^i = \sum_{s=1}^{N} \frac{D_{js}}{N_i} \right\}$$

and N_i is number of strings classified in cluster i

}Until (center strings stabilize)

2.5 Fuzzy Clustering Validation Techniques

The aim of clustering algorithm is partition a data set into groups such that the objects within same group are more similar to each other than objects in different group. The cluster validity index can evaluate the goodness of the result of the partitions and enables to determine optimal number of cluster when it is not known in advance in the data set. We calculated cluster validity indices to show the partition goodness [40] However, we only compute the cluster validity indices on the membership because our string grammar fuzzy clustering algorithms use the membership values as the indicator of cluster assignment. We utilize three cluster validity indices, i.e., the partition coefficient (PC) [41], the partition entropy (PE) [42], and Xie and Beni (XB) [43]. These following indices are usually suitable for measuring fuzzy clustering which is described below:

2.5.1 Partition coefficient

The partition coefficient (PC) is the one of validity indices involving only the membership values and it was introduced in [41] as:

$$V_{PC}(\mathbf{U}) = \frac{1}{N}\sum_{i=1}^{C}\sum_{k=1}^{N} u_{ik}^{m} . \tag{2.33}$$

The PC index is the basic fuzzy cluster validity that indicates the average of membership degree of fuzzy subsets in **U**. The PC index value is in the range of $\left[\frac{1}{c},1\right]$. When $V_{PC}=1/c$ the system is entirely fuzzy, since every element belongs to all clusters with the same degree of membership. When $V_{PC}=1$ the system is hard and membership values are either 1 or 0. The larger value of V_{PC} it is, the better accuracy it is.

2.5.2 Partition entropy

The partition entropy (PE) is also the one of validity indices involving only the membership, same as PC, The PE was defined as [42]

$$V_{PE}(\mathbf{U}) = -\frac{1}{N}\sum_{i=1}^{C}\sum_{k=1}^{N} u_{ik}\log_2 u_{ik} \tag{2.34}$$

The PE index is a scalar measure of the amount of fuzziness in a given **U**. The index is computed for values of C greater than 1 and its value is in the range of $\left[0,\log_2 c\right]$. When $V_{PE}=0$ the partition is rigid.When $V_{PE}=\log_2 c$ the fuzziness is maximum. Partition Entropy is directly proportional to the number of partitions. It is more appropriated to validate the best partition among several runs of an algorithm. In this case, the lower value of V_{PE}, the better on tends to be.

2.5.3 Xie and Beni

This validity function is the one of validity indices involving the membership values and the data set, which is proposed by Xie and Beni (XB) [43] and modified by Pal and Bezdek [40]. This index is a measure for evaluating both the compactness and separateness of fuzzy clusters. The XB was defined as

$$XB(\mathbf{U}) = \frac{\sum_{i=1}^{C}\sum_{k=1}^{N} u_{ik}^{m} Lev(s_k, sc_i)}{N d_{\min}} \tag{2.35}$$

where $Lev(s_k,sc_i)$ is the Levenshtein distance between string s_k and cluster center string sc_i of cluster i and

$$d_{\min} = \min_{i,j=1,2,\ldots,C; i\neq j} Lev(sc_i, sc_j). \tag{2.36}$$

This validity index focus on total variation within clusters (compactness) and separation between cluster, the numerator of equation 2.35 indicates the compactness of the fuzzy partition, while the denominator indicates the strength of the separation between clusters. The smaller XB value is, the more compact and better separated clusters is. The XB index value is in the range of $\left[\frac{\sum_{i=1}^{C}\sum_{k=1}^{N} u_{ik} d(s_k, sc_i)}{n d_{\min}}, \infty\right]$.

2.6 Measuring of Overlapping data

The objective of the clustering technique is to partition a data set into groups such that both within group similarity and between group dissimilarity are maximized. However, Most of the real world datasets are overlapping data.

In our work, the *R-value* [44] is utilized to measuring overlapping data. The idea of *R-value* is to find the *K* nearest neighbor string and then count the number of string that belonging to the other classes. If the summation of the count number (*R*) is larger than 0, that means the dataset is overlapping data.

Suppose $C_1, C_2, \ldots, C_n$ are different classes of dataset, **U** is the universal set of whole object on dataset, $S(C_i)$ is set of object that belong to class C_i, p_{im} is m^{th} object of class C_i. The *R-value* of a dataset S is defined as

$$R(f) = \frac{1}{|\mathbf{U}|}\sum_{i=1}^{n}\sum_{i=1}^{|C_i|}\lambda(|kNN(p_{im}, \mathbf{U} - S(\mathrm{C}_i)|) \tag{2.37}$$

where $kNN(p_{im}, \mathbf{U} - S(\mathrm{C}_i))$ is the set of objects in set of K nearest neighbor object for an object p_{im} that not belong to class C_i and $\lambda(|kNN(p_{im}, \mathbf{U} - S(\mathrm{C}_i)|) = 1$ if $|kNN(p_{im}, \mathbf{U} - S(\mathrm{C}_i)| > 0$, else $\lambda(|kNN(p_{im}, \mathbf{U} - S(\mathrm{C}_i)|) = 0$.

The *R-value* can evaluate the degree of overlap. The higher *R-value* indicates that the dataset contains large overlapping area among its class. If *R-value* is equal to zero, the dataset of each class can be well separated. If *R-value* is equal to one, the dataset is full overlapping dataset.

2.7 The modified median string

Martinez, C.D., et al. [23] improved the calculate of prototype of string classification using approximate median string. The approximate median string applies the edition operations, i.e., insertion, deletion and substitution over each symbol of the string, in a set of strings S of cluster i can be calculated as

$$sc_i = \arg\min_{j \in S_i} \sum_{k=1}^{N_i} Lev(s_j, s_k) \quad \text{for } 1 \le i \le C \tag{2.38}$$

where N_i is the number of strings in cluster i.

We can use sc_i as a cluster center i. However, [23] improve finding median string by edition operations (insertion, deletion, and substitution) over each symbol of the string. Let Σ^* be the free monoid over the alphabet set Σ and a set of strings $S \subseteq \Sigma^*$. The selected string (sc_i) will be the one that gives the minimum value, will be

$$sc_i = \arg\min_{j \in \Sigma^*} \sum_{k=1}^{N} Lev(s_j, s_k) \quad \text{for } 1 \le i \le C\,. \tag{2.39}$$

The algorithm of the modified median string is as follows:

Start with the initial string s.

For each position i in the string s

1. Build alternative

Substitution: Set z=s. For each symbol $a \in \Sigma$

a) Set z' to be the result of substituting i^{th} symbol with symbol a.

b) If $\sum_{k=1}^{N} Lev(z', s_k) < \sum_{k=1}^{N} Lev(z, s_k)$, (2.40)

then set $z = z'$.

Deletion: Set y to be the result of deleting the i^{th} symbol of s.

Insertion: Set x=s. For each symbol $a \in \Sigma$

a) Set x' to be the result of adding a at position i^{th} of s.

b) If $\sum_{k=1}^{N} Lev(x', s_k) < \sum_{k=1}^{N} Lev(x, s_k)$, (2.41)

then set $x = x'$.

2. Choose an alternative

Select string s' from the set of strings $\{s,x,y,z\}$ from step 1 using

$$s' = \arg\min_{G \in \{s,x,y,z\}} \sum_{k=1}^{N} Lev(G, s_k). \quad (2.42)$$

Then set $s = s'$.