Weight Entropy Fuzzy Clustering Algorithm For Face Recognition

A Project Report

Submitted in partial fulfillment of the requirements for the award of the degrees

of

BACHELOR OF TECHNOLOGY

in

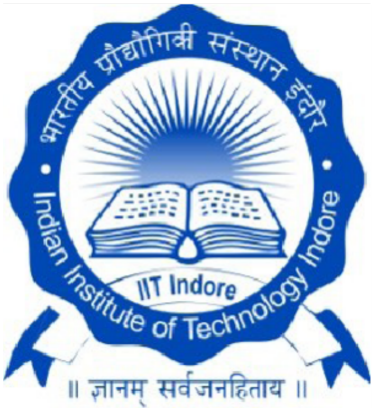
COMPUTER SCIENCE ENGINEERING

Submitted by:

Nitish Raj, 150001020
Paridhi Yadav, 150002024
Chavare Manish Sadashiv, 150001008

Guided by:

Dr. Aruna Tiwari
Associate Professor,
Discipline of Computer Science and Engineering



INDIAN INSTITUTE OF TECHNOLOGY INDORE

December 2018

Candidate's Declaration

We hereby declare that the project entitled **Weight Entropy Fuzzy Clustering Algorithm for Face Recognition** submitted in partial fulfillment for the award of the degree of Bachelor of Technology in Computer Science and Engineering completed under the supervision of **Dr. Aruna Tiwari, Assistant Professor, Computer Science and Engineering**, IIT Indore is an authentic work.

Further, we declare that we have not submitted this work for the award of any other degree elsewhere.

Paridhi Yadav
150002024

Nitish Kumar Raj
150001020

Chavare Manish Sadashiv
150001008

Supervisor's Certificate

This is to certify that the thesis entitled, *"Weight Entropy Fuzzy Clustering Algorithm for Face Recognition"* and submitted by Paridhi Yadav, ID No 150002024, Nitish Kumar Raj, ID No 150001020 and Chavare Manish Sadashiv, ID No 150001008 in partial fulfillment of the requirements of B.Tech Project embodies the work done by them under my supervision.

Dr. Aruna Tiwari
Associate Professor,
Discipline of Computer Science and Engineering,
Indian Institute of Technology Indore

Preface

This report on "Weight Entropy Fuzzy Clustering Algorithm for Face Recognition" is prepared under the guidance of Dr Aruna Tiwari, Assistant Professor, Computer Science and Engineering, IIT Indore.

Through this report, we have tried to provide a detailed description of our approach, design and implementation to apply the Fuzzy Clustering algorithm Weighted Entropy Fuzzy C-Means Clustering for face recognition. We have applied it on a real dataset created by us for face recognition of some students of our college.

Acknowledgement

We wish to thank Dr. Aruna Tiwari for her kind support and invaluable guidance throughout the tenure of this project and providing relentless teaching to develop the skills and concepts required for the development of this project.

We are really grateful to the Institute for the opportunity to be exposed to the systemic research especially Dr. Aruna Tiwari's Lab for providing the necessary hardware utilities to complete the project.

We would also like to express our deep sense of gratitude and convey thanks to every one who knowingly or unknowingly helped us and supported us during the completion and documentation of this project.

Abstract

Clustering unlabelled dataset is one of the most crucial parts of unsupervised learning. Many algorithms have been proposed for clustering till now. In this project, we are studying one of the most effective clustering algorithm known as Weight Entropy Fuzzy C means clustering and implementing it on a real dataset for face recognition. Also by exhaustive testing, we tried to predict the constant values to get optimal results in terms of cluster quality and accuracy of the various dataset. After implementing the algorithm we obtain values of membership matrix, cluster centers and weight matrix stating the importance of each feature while clustering the algorithm.

Using WEFCM we can cluster different images according to their features and when we test with another image, according to belongingness of testing image in different clusters we can recognize in which cluster that image belongs and thus face recognition can be performed using WEFCM.

Apart from implementing WEFCM, to optimize the results, we have studied some feature reduction algorithm and implemented it for optimal results.

Also by exhaustive testing, we tried to estimate the different parameters value and their combination to get optimal results concerning NMI, ARI, and Accuracy.

Table Of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Clustering is an unsupervised learning technique aimed at grouping a set of objects into subsets or clusters. The goal is to create clusters that are coherent internally, but substantially different from each other. In plain words, objects in the same cluster should be as similar as possible, whereas objects in one cluster should be as dissimilar as possible from objects in the other clusters. Organizing data into clusters shows the internal structure of the data. Techniques for clustering is useful in knowledge discovery in data.

The clustering algorithms can be divided into two categories: hard clustering where each object belongs to only one cluster and fuzzy clustering where each object can belong to every cluster to a certain degree [9]. The primary motivation for the introduction of fuzzy clustering as a generalization of crisp or partitioning clustering was to represent partly overlapping clusters better. Data points at the boundary between two clusters should belong partly to both clusters.

K-means algorithm is one of the simplest unsupervised learning algorithms that can deal with most of the clustering problems [4], but this hard clustering algorithm is sensitive to the initial centers. The Fuzzy C-Means clustering algorithm is an extension of hard clustering, in which real membership degrees in unit interval [0,1] are obtained by the

Lagrangian multiplier method. Fuzzy clustering is more robust in the sense that the results seem to be less dependent on the initialisation that is required for many hard clustering algorithms like K-means[9].

The above classical clustering algorithms fail as they take the equal contribution of all features while deciding cluster membership of objects. This is not an ideal condition as it is possible having features which have a more dominating role while determining cluster centres over other features and this fact should be taken into consideration while deciding the cluster architecture. These drawbacks are resolved in Weight Entropy Fuzzy C-Means Clustering [9]. The objective function of FCM is modified by using attribute weighted dissimilarity measure and adding weight entropy regularization term.

1.2 Objective

The objective thus is the development and implementation of Weight Entropy Fuzzy C-Means clustering algorithm. Also use this algorithm for face recognition. The above objective has been divided into following goals:

- Implementation of Weight Entropy Fuzzy C-Means clustering algorithm.
- Creating a real dataset with pictures of students of our college to be used as input for face recognition.
- Data preprocessing of facial images using Principal Component Analysis, Locally Linear Embedding and Extra tree classifier.
- Implementation and testing of face recognition using Weight Entropy Fuzzy C-Mean Clustering.

Chapter 2

Literature Review

2.1 Clustering

Clustering is the process of grouping similar entities together. Clustering is an unsupervised machine learning trick in which a set of observations is divided into several groups based on their similarities, in this way, observations in the same groups are as similar as possible to one another, and different groups are as dissimilar as possible from one another. The clustering algorithms can be divided into two categories: hard clustering where each object belongs to only one cluster and fuzzy clustering where each object can belong to every cluster to a certain degree. We will be primarily focusing on fuzzy clustering.

2.1.1 Fuzzy Clustering

Fuzzy clustering (also referred to as soft clustering) is a form of clustering in which each data point can belong to several clusters with a certain degree of membership. Hard partitioning methods fail if sampled data is not representative of whole data, then centroids overlap. Hence we need soft partitioning methods, i.e., fuzzy clustering. The superiority of fuzzy clustering lies on the fact that it allows a data object to belong to several clusters with a certain degree of membership. The primary motivation for the introduction of fuzzy clustering as a generalization of crisp or partitioning clustering was to represent partly overlapping clusters better. Data points at the boundary between two clusters should belong partly to both clusters.

2.1.2 Fuzzy C-Means

The Fuzzy C-Means algorithm groups dataset into C clusters by minimizing the sum of distances between the objects and the cluster centers [1]. So the objective function to be optimized is:

$$J_{FCM} = \sum_{i=1}^{N}\sum_{j=1}^{C} u_{ij}^{\alpha}||x_i - v_j||^2 \tag{2.1}$$

Subject to $\sum_{i=1}^{C} u_{ij} = 1$, $1 \leq j \leq n$, $0 \leq u_{ij} \leq 1$

Where, $X = x_1, x_2, ..., x_n$ denotes the dataset, where n is the number of objects, and m is the dimensions of a object. $U = [u_{ij}]$ is a $c \times n$ matrix, u_{ij} denotes the degree of membership of the j-th object belonging to the i-th fuzzy cluster. $V = v_1, v_2, ..., v_c$ is a $c \times m$ matrix, v_{il} denotes cluster center of i-th cluster defined by u_{ij}. α is a fuzzification parameter which drastically affects the clustering results. $\alpha \geq 1$ controls the extent of membership sharing between fuzzy cluster [1] [2] [9]. As $\alpha = 1$, FCM converges in theory to the traditional K-means solution. ε denotes the predefined constant.

Algorithm Fuzzy C-Means

Input: X, V, c, α

Output: U, V

Step 1: Randomly initialize cluster centers V_0.

Step 2: Compute cluster membership

$$u_{ic} = \frac{1}{\sum_{j=1}^{C}\left(\frac{||x_i - v_c||^2}{||x_i - v_j||^2}\right)^{\frac{1}{\alpha-1}}}, \forall i, c$$

Step 3: Compute cluster centers

$$v_c = \frac{\sum_{i=1}^{N}(u_{ic})^{\alpha} x_i}{\sum_{i=1}^{N}(u_{ic})^{\alpha}}, \forall c$$

Step 4: If $||V' - V|| < \varepsilon$ then stop, else go to Step 2.

2.2 Face Recognition

A feature is a piece of information which is relevant for solving the computational task related to a certain application. In this project, we have chosen every single pixel as a feature, as a pixel is the simplest element that can be considered as a feature. One penalty for this is that we will get a vast number of features and not every feature is necessary for cluster formation. We need to reduce the features so that we get only the essential features which will be beneficial for our clustering algorithm. And hence, we have used the following feature reduction methods to reduce the features:

- Principal Components Analysis(PCA)
- Locally Linear Embedding(LLE)
- Extra Tree Classifier

2.2.1 Feature Reduction using Principal Components Analysis

PCA is used to reduce the dimensionality of data. PCA transforms the initial data into a new small set of data that too without loosing the most important information in the original data set. This new data corresponds to a linear combination of the original and is called principal components. The dimension reduction is achieved by identifying the principal directions in which the data varies. PCA assumes that the directions with the largest variances are the most important (i.e, the most principal).

Steps to perform Principal Component Analysis:

---

1. Prepare the data :
   (a) Center the data : subtract the mean from each variables. This produces a data set whose mean is zero.

(b) Scale the data : If the variances of the variables in your data are significantly different, its a good idea to scale the data to unit variance. This is achieved by dividing each variables by its standard deviation.

2. Calculate the covariance/correlation matrix

3. Calculate the eigenvectors and the eigenvalues of the covariance matrix

4. Reduce dimensionality and form feature vectors: Eigenvectors are found, now order them in order of eigenvalues, highest to lowest. Now we have components in order of significance. Choose first p eigenvectors, final data set has only p dimensions.

5. compute the new dataset :

   (a) Row feature vector : is the matrix with eigenvector in columns transposed

   (b) Row zero mean data : is the mean adjusted data transposed.

   (c) new data = RowFeatureVector X RowZeroMeanData

2.2.2 Feature Reduction Using Locally Linear Embedding

Locally linear embedding is a technique for nonlinear dimensionality reduction. LLE is used to map high dimensional data into a single global coordinate system of lower dimensionality. The dimensionality reduction by LLE succeeds in identifying the underlying structure of the manifold. The critical feature of LLE is that neighborhood information is maintained. What we mean by it is that the neighbor of a data point in original space should also be a neighbor in low dimensional space.

LLE algorithm psuedocode:

---

1. Find neighbours in X space [b,c].

   for i=1:N

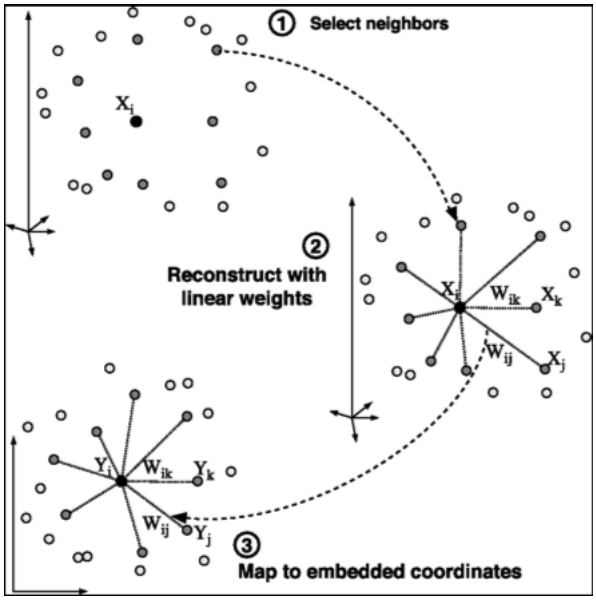


Figure 2.1: Steps of Locally Linear Embedding

compute the distance from X_i to every other point X_j

find the K smallest distances

assign the corresponding points to be neighbours of X_i

end

2. Solve for reconstruction weights W.

for i=1:N

create matrix Z consisting of all neighbours of $X_i[d]$

subtract X_i from every column of Z

compute the local covariance $C = Z' \times Z$ [e]

solve linear system $C \times w = 1$ for w [f]

set $W_{ij} = 0$ if j is not a neighbor of i

set remaining elements in the i^{th} row of W equal to w/sum(w)

end

3. Compute embedding coordinates Y using weights W.

   create sparse matrix $M = (I - W)' \times (I - W)$

   find bottom d+1 eigenvectors of M

   (corresponding to the d+1 smallest eigenvalues)

   set the qth ROW of Y to be the q+1 smallest eigenvector

   (discard the bottom eigenvector [1,1,1,1...] with eigenvalue zero)

2.2.3 Feature Reduction using Extra Tree Classifier

An extra trees classifier, otherwise known as an Extremely randomized trees classifier, is a variant of a random forest. In it splits are selected on random instead of using some criterions. Before explaining its working, few terms are explained:

Entropy: Entropy is the measure of randomness in a dataset.

$HighEntropy \rightarrow LowEntropy \rightarrow ZeroEntropy$

Information Gain: It represents the change in entropy of two levels.

Leaf node carries decision.

Our goal is to lower the entropy to minimum and to achieve this goal we classify data using Decision tree classifier. Decision tree classifier can be explained through the following points:

1. Entropy = $\sum_{i=1}^{k} p(value_i) \times \log_2[p(value_i)]$

Where $p(value_i)$ is the probability of $(value_i)$

2. Entropy of dataset will be calculated after every split to calculate gain.

3. We will try to choose condition that gives us highest gain $\Rightarrow$ split data using each condition and calculate gain that we get.

4. Condition that gives us highest gain $\Rightarrow$ first split

5. Do it till we get single lable.

6. After this, to increase accuracy we will make forest of n such classifiers.

7. In the random forest that we will be forming there can be 'm' number of random features and any number of datapoints in every tree.

8. After forming random forest,we will be extracting most important features(features which cause more information gain) from each tree of random forest-making this algorithm an ensemble algorithm.
9.Like this we can select most important features from dataset and thus reducing the features to significant ones.

2.3 Image Fundamentals

Images are represented as matrix containing intensity value at each pixel of the image. The resolution of an image tells us about the number of pixels in the image. There are generally two kinds of images:

- Color Image(true color image) : A Color image has three intensity values for each pixel. Generally a color image contains an intensity value for Red, Green and Blue each(we consider by default the RGB mode of a picture) there are other formats also like HSV. RGB image is stored as an m-by-n-by-3 data array that defines red, green, and blue color components for each individual pixel. RGB images do not use a palette. The color of each pixel is determined by the combination of the red, green, and blue intensities stored in each color plane at the pixel's location.

- Gray Scale Image: A Gray scale image contains only one type of color i.e. Shades of gray. In digital representation of gray scale image, we assume that the intensity value of each pixel lies within 0 to 255. Each value represents one shade of gray.

2.4 Performance Evaluation

For the evaluation of clusters formed by our algorithm, i.e. to measure the quality of cluster formed we have calculated Normalized Mutual Information and Adjusted Random Index. For the testing of face recognition, i.e. to check whether the face we are classified in the correct cluster or not we have calculated accuracy.

(a) Grey Scale Image

(b) Color Image(RGB)

Figure 2.2: Example of both types of images

2.4.1 Accuracy

Accuracy is one metric for evaluating classification models. Informally, accuracy is the fraction of predictions our model got right. Formally, accuracy has the following definition:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

2.4.2 Normalized Mutual Information

NMI is used to evaluate the clustering results, which measure the agreement of the clustering results produced by an algorithm and the ground truth. If we refer to class as the ground truth and to cluster as the results of a clustering algorithm, the NMI is calculated as follows

$$NMI = \frac{\sum_{c=1}^{k}\sum_{p=1}^{m} n_c^p \log \frac{n.n_c^p}{n_c.n_p}}{\sqrt{\left(\sum_{c=1}^{k} n_c \log\left(\frac{n_c}{c}\right)\right)\left(\sum_{p=1}^{m} n_p \log\left(\frac{n_p}{n}\right)\right)}}$$

where n is the total number of objects, n_c and n_p are the numbers of objects in the c^{th} cluster and the p^{th} class, respectively, and n_c^p is the number of common objects in class p and cluster c.

2.4.3 Adjusted Rand Index

The adjusted Rand index is the corrected-for-chance version of the Rand index which is a measure of the similarity between two clustering. To compute the ARI, we first harden the fuzzy partitions by setting the maximum element in each column of U to 1, and all else to 0. We use ARI to compare the clustering solutions with ground-truth labels (when available)

$$ARI = \frac{\sum_{i,j}\binom{n_{ij}}{2} - \left[\sum_i \binom{n_i}{2} \sum_j \binom{n_j}{2}\right] / \binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{n_i}{2} + \sum_j \binom{n_j}{2}\right] - \left[\sum_i \binom{n_i}{2} \sum_j \binom{n_j}{2}\right] / \binom{n}{2}}$$

Chapter 3

Problem Analysis And Design

For implementing face reduction algorithm firstly we need to pre-process our data. We cannot apply our learning algorithm Weight Entropy Fuzzy C-Means Clustering algorithm on raw data. The first step in data pre-processing will be feature extraction. After extracting features, we will apply the feature reduction algorithm. Finally, the data obtained will be fed to WEFCM to form clusters.

3.1 Feature Extraction

The data that we get for face reduction is face-images. In our data, we have coloured images in RGB format. Before applying any machine learning algorithm on it, we have to represent our face-images as a vector containing pixel values. We have done that in the following steps:

1. We have face-images in the RGB format, in RGB image is stored as an $n \times m \times 3$ data array that defines red, green, and blue colour components for each pixel where $n \times m$ is the dimension of the image. Firstly, we will convert our colour image to greyscale as in the greyscale image stores only one intensity per pixel, and that is the intensity of grey colour. To convert coloured image to greyscale image the following formula is used

$$Y = 0.299R + 0.587G + 0.114B \tag{3.1}$$

where R is the intensity of red at a particular pixel, G is the intensity of green at a particular pixel, B is the intensity of blue at the particular pixel and Y is the output intensity we get for every pixel converted to greyscale.

2. In grayscale images, we have a single intensity value for each pixel. In holistic representation, we take each pixel as a feature. We convert an image of dimension $n \times m$ to a matrix of $n \times m$, where each entry of the matrix is the intensity of the image at that point.

3. Holistic representation is based on the lexicographic ordering of raw pixel values to yield one vector per image. We convert the $n \times m$ matrix into a vector of length $(n.m)$. An image can now be seen as a point in a high dimensional feature space. The dimensionality corresponds directly to the size of the image in terms of pixels. Therefore, an image of size 100×100 pixels can be seen as a point in a 10,000 dimensional feature space.

4. The above steps can be used to convert all the images into a vector. The vectors are concatenated together to form the dataset.

Using the steps as mentioned above, features are extracted from a dataset of images and convert into a format so that feature reduction and other machine learning algorithms can be applied to it.

3.2 Feature Reduction

In Holistic representation, the dimensionality corresponds directly to the size of the image in terms of pixels as raw pixel values are used to yield vectors. Therefore, an image of size 250x250 pixels can be seen as a point in a 62,500-dimensional feature space. This large dimensionality of the problem prohibits the use of any learning to be carried out in such a high dimensional feature space. This is called the curse of dimensionality in the pattern recognition literature [7]. A common way of dealing with it is to employ a dimensionality reduction technique such as Principal Component Analysis PCA to pose the problem into a low-dimensional feature space such that the major modes of variation of the data are

still preserved. Like PCA other dimensionality reduction techniques like Locally Linear Embedding and Extra Tree Classifier are also used.

We have used three different dimensionality reduction algorithms in this project, namely Principal Component Analysis(PCA), Locally Linear Embedding(LLE) and Extra Tree Classifier.

3.3 Weight Entropy Fuzzy C-Means Clustering Algorithm

In this algorithm, we consider that the attribute weight in a cluster represents the probability of contribution of that attribute in forming the cluster. The entropy of the attribute weight represents the certainty of dimensions in the identification of a cluster. Therefore, the objective function of Fuzzy C-Means is modified by using attribute weighted dissimilarity measure and adding the attribute weight entropy term so that we can simultaneously minimize the within-cluster dispersion and maximize the attribute weight entropy to stimulate more important attributes to contribute to the identification of clusters[9]. The new objective function of WEFCM is shown as follows:

$$J_{WEFCM} = \sum_{k=1}^{N}\sum_{i=1}^{C}(u_{ik})^{\alpha}\sum_{l=1}^{M} w_{il}||X_{kl} - V_{il}||^2 + \gamma\sum_{i=1}^{C}\sum_{l=1}^{M} w_{il}\log(w_{il})$$

The first term is a distance-based objective function which controls the shape and size of the clusters and encourages the agglomeration of clusters, while the second term uses the negative attribute weight entropy which regularizes the attribute weights. γ is a regularizing and adjustable parameter. With a proper choice of γ, we can balance the two terms to find a stable solution.

Algorithm: Weighted Entropy Fuzzy C-Means

Input: X, V, c, α, γ

Output: U

Step 1: Randomly initialize cluster centers and weights.

Step 2: Compute cluster membership

$$u_{hj} = \frac{1}{\sum_{i=1}^{C} \left(\frac{D_{hj}^{(1)}}{D_{ij}^{(1)}} \right)^{\frac{1}{\alpha - 1}}}$$

If $D_{hj}^{(1)} \neq 0$ but $D_{ij}^{(1)} = 0$ for some $i \neq h$ then $u_{hj} = 0$

If $D_{hj}^{(1)} = 0$ and $num = i : D_{ij}^{(1)} = 0$ then $u_{hj} = \frac{1}{num}$

Where $D_{hj}^{(1)} = \sum_{l=1}^{M} w_{hl} ||X_{jl} - V_{hl}||^2$, $1 \leq h \leq C, 1 \leq j \leq n$

Step 3: Compute cluster centers

$$V_{il} = \frac{\sum_{j=1}^{N} (u_{ij})^{\alpha} x_{jl}}{\sum_{j=1}^{N} (u_{ij})^{\alpha}}, \text{ if } w_{il} \neq 0$$

$$V_{il} = 0, \text{ if } w_{il} = 0$$

Step 4: Update weights

$$w_{is} = \frac{e^{\frac{-D_{is}}{\gamma}}}{\sum_{l=1}^{M} e^{\frac{-D_{is}}{\gamma}}}$$

If $\sum_{l=1}^{M} e^{\frac{-D_{is}}{\gamma}} = 0$, Then $w_{is} = \frac{1}{m}$

Where $D_{is} = \sum_{j=1}^{N} (u_{ij})^{\alpha} (X_{js} - V_{is})^2$

Step 5: If $||V' - V|| < \epsilon$ then stop, else go to Step 2.

3.4 Testing Algorithm

For the testing of our face recognition algorithm, we applied 10-Fold Cross-Validation. To check the quality of clusters formed by our clustering algorithm we calculate Normalised Mutual Information and Adjusted Random Index. To test whether the faces are getting assigned the correct cluster we compute accuracy. For the computation of accuracy, we need the number of accurate predictions. To get the number of correct predictions we need to label our data, so we can check if the label assigned after clustering matches with the pre-defined label(ground truth).

3.4.1 K-Fold Cross-Validation

In k-fold cross-validation, the original sample is randomly partitioned into k equal sized subsamples. Of the k subsmples, a single subsample is retained as the validation data for testing the model, and the remaining k 1 subsamples are used as training data. The cross-validation process is then repeated k times, with each of the k subsamples used exactly once as the validation data. The k results can then be averaged to produce a single estimation. The advantage of this method over repeated random sub-sampling (see below) is that all observations are used for both training and validation, and each observation is used for validation exactly once. 10-fold cross-validation is commonly used, but in general k remains an unfixed parameter.

We have used 10-fold cross-validation, so k = 10. In 10-fold cross-validation, we randomly shuffle the dataset into ten sets, so that all of them are equal in size (this is usually implemented by shuffling the data array and then splitting it in ten). When k = n (the number of observations), the k-fold cross-validation is exactly the leave-one-out cross-validation. So we train on $d_0 - d_8$ and validate on d_9, similarly we train and validate on each of them.

3.4.2 Labeling Data

For the calculation of accuracy, we need to label our data. We get the membership matrix and cluster centres from Weighted Entropy Fuzzy C-Means. We label cluster centres using

K-Nearest Neighbours.

We have our cluster centres, to label them first we find its K nearest neighbours in term of Euclidean distance. After getting k nearest neighbours, we check the labels of nearest neighbours. The label having the highest frequency among them will be assigned as the label of that cluster centre.

K Nearest Neighbours

For each test data point, we would be looking at the K nearest training data points and take the most frequently occurring classes and assign that class to the cluster centres. K represents the number of training data points lying in proximity to the test data point which we are going to use to find the class.

Algorithm- KNN

---

1. Load the training data and cluster centers
2. Choose the value of K
3. For each point in Cluster Centers:
   (a) find the Euclidean distance to all training data points
   (b) store the Euclidean distances in a list and sort it
   (c) choose the first k points
   (d) assign a class to the cluster center based on the majority of classes present in the chosen points
4. End

Now we have the labels for our predicted data too, so we can calculate accuracy of face recognition algorithm.

3.5 Setup and Implementation

3.5.1 Setup

The implementation of this project is done in Python2 on Ubuntu. Python2 can be easily installed on Ubuntu by running following commands on the terminal one by one in given order-

```
sudo apt update
sudo apt upgrade
sudo apt install python2.7 python-pip
```

The packages used are - os, cv2(openCV), numpy, pandas, matplotlib, sklearn, math, timeit, skfuzz. To install all these packages use the following command for every package separately.

```
pip2 install <package>
```

Here in place of ¡package¿ write the package to be installed.

3.5.2 Implementation

Implementation is done in python. The first step is data preprocessing. It could be understood through the flowchart in the figure 3.1.

After preprocessing feature reduction is done using three different methods PCA, LLE and extra tree classifier. Clustering is done on reduced data using WEFCM. And then finally testing is done. These steps can be understood from the figure 3.2.

Code for WEFCM

We have implemented WEFCM in python. The packages we have used are Numpy, Matplotlib, pandas, cv2(openCV), operator and math. Following is the implementation of some important function needed for implementing WEFCM, and after that we have givin WEFCM's implementation.

```
## Function To update Cluster Centers
```

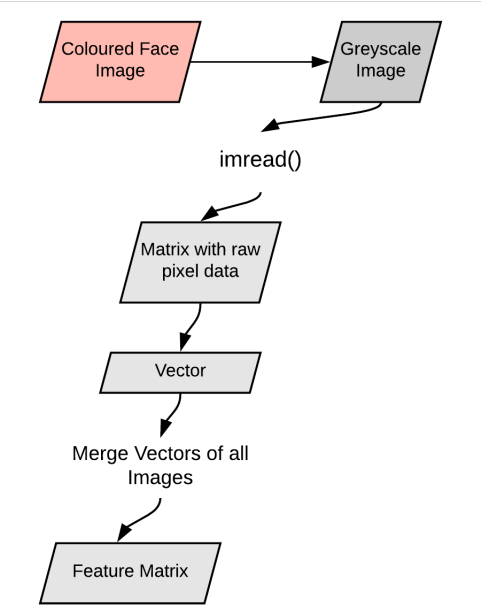


Figure 3.1: Data Preprocessing

```
def calculateClusterCenter(U,sx,k):
        u=np.array(U)
    um=u**m
    sx=np.array(sx)
    um=um.T
    center = um.dot(sx) / np.atleast_2d(um.sum(axis=1)).T
    return center

## Function To Update Membership Matrix

def updateMembershipValue(D,U,k):
        p = float(2/(m-1))
        u=np.power(D,p)
        u1=np.reciprocal(D,dtype=float)
        u2=np.power(u1,p)
        u3=np.sum(u2,axis=1)
```

```
        u=u.T
        u4=np.multiply(u,u3)
        u4=u4.T
        u4=np.reciprocal(u4,dtype=float)
        u4=u4.tolist()
        return u4

## Function To Calculate D1 Which is used to calculate Membership Matrix

def calculate_D1(W,C,X,k):
    D=list()
    x=np.array(X)
    w=np.array(W)
    for i in range(k):
        c=C[i];
        c=np.array(c)
        c1=np.subtract(x,c)
        c2=np.square(c1)
        c3=np.multiply(c2,w[i])
        c4=np.sum(c3,axis=1)
        c5=c4.tolist()
        D.append(c5)
    D=np.array(D)
    D=D.T
    D=D.tolist()
    return D

## Function to calculate D2 which is used to calcualte Weights

def calculate_d2(W,C,U,X,k,D):
    D2=list()
```

```
    x=np.array(X)
    u=np.array(U)
    u=np.power(u,m)
    for i in range(k):
        c1=np.subtract(x,C[i])
        c2=np.square(c1)
        c2=c2.T
        c22=np.array(u[:,i])
        c3=np.multiply(c2,c22)
        c3=c3.T
        c4=np.sum(c3,axis=0)
        D2.append(c4)
    return D2

## Function To Update Weight

def updateweight(W,k,D2,D):
    gamma1=float(-1/gamma)
    D2=np.array(D2)
    D2=np.multiply(D2,gamma1)
    D2=np.exp(D2)
    d3=np.sum(D2,axis=1)
    D2=D2.T
    W=np.divide(D2,d3)
    W=W.T
    W=W.tolist()
    return W

## Weight Entropy Fuzzy C-Means

def WEFCM(Z,k):
```

```
D=len(Z[0])
n=len(Z)
W=initializeWeight(k,D)
U=initializeMembershipMatrix(n,k)
center=calculateClusterCenter(U,Z,k)
i=0
aphselan=np.max(center)
while(i<max_iter):
D1=calculate_D1(W,center,Z,k)
U=updateMembershipValue(D1,U,k)
D2=calculate_d2(W,center,U,Z,k,D)
W=updateweight(W,k,D2,D)
center1=calculateClusterCenter(U,Z,k)
c_d=np.subtract(center1,center)
c_d=np.square(c_d)
c_d1=np.sum(c_d,axis=1)
c_d1=np.sqrt(c_d1)
aphselan=np.max(c_d1)
center=center1
i+=1
return U,center
```

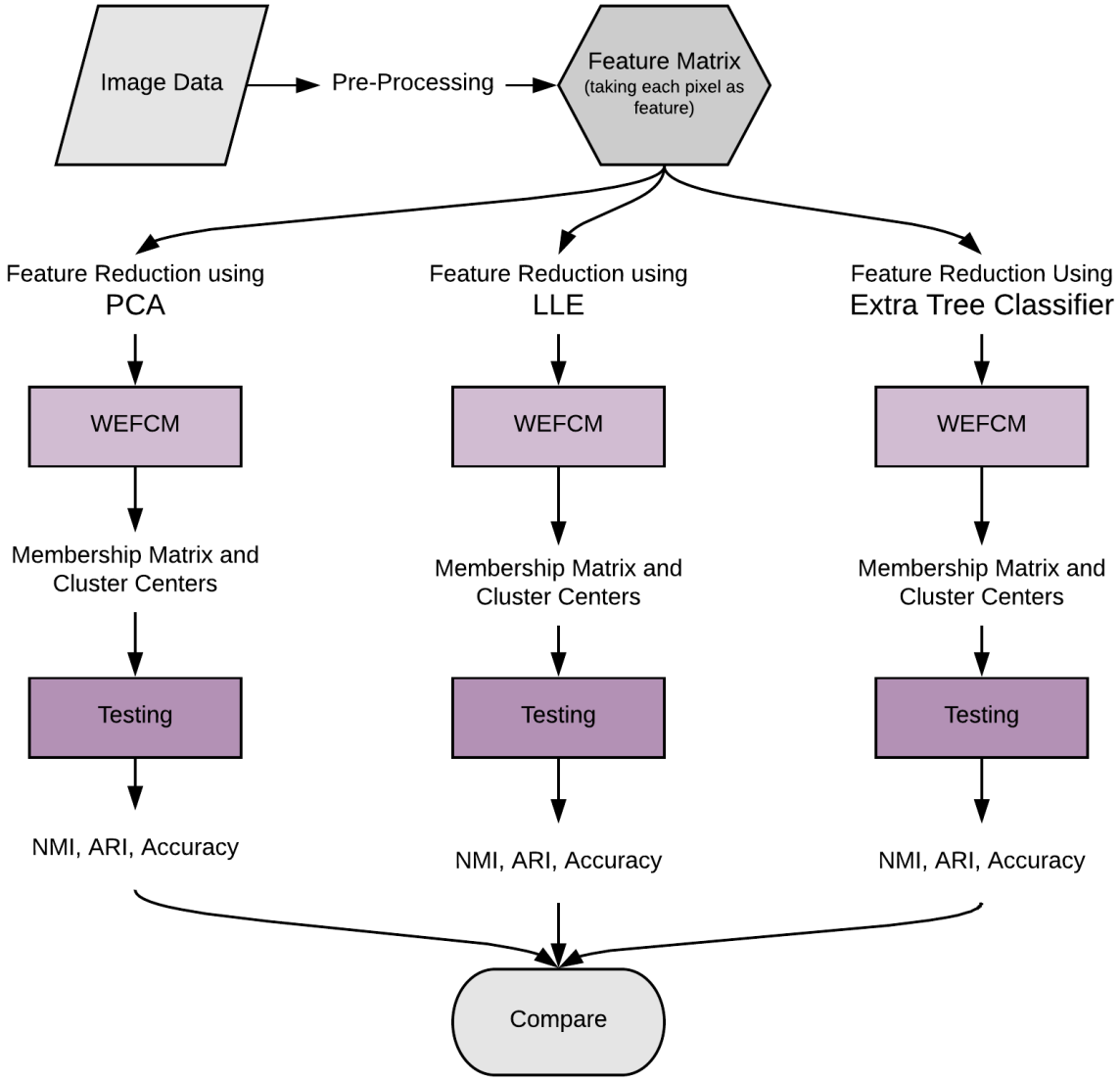


Figure 3.2: Flowchart showing Steps of Implementation

Chapter 4

Experimentation

4.1 Data Description

For the testing of clustering algorithm Weight Entropy Fuzzy C-Means we have used IRIS Dataset. ORL dataset and a real dataset are used for the testing of face recognition.

- ORL Database:

  The ORL Database of Faces contains ten different images of each of 40 distinct subjects. For some subjects, the images were taken at different times, varying the lighting, facial expressions (open / closed eyes, smiling / not smiling) and facial details (glasses / no glasses). All the images were taken against a dark homogeneous background with the subjects in an upright, frontal position (with tolerance for some side movement). The files are in PGM format. The size of each image is 92x112 pixels, with 256 grey levels per pixel.

- Real Dataset:

  We created a real dataset by taking pictures of some students of our college in different angles. Each student has 8 distinct pictures. Each picture has dimenstions 4160-by-3120 pixels.

- Iris Dataset:

It is a benchmark dataset. The data set contains 3 classes of 50 instances each with 5 attributes, where each class refers to a type of iris plant. One class is linearly separable from the other 2; the latter are NOT linearly separable from each other.

4.2 Results

4.2.1 WEFCM

We tested Weight Entropy Fuzzy C-Means against traditional clustering algorithms Fuzzy C-Means and K-means. We compared the three algorithms by clustering them on Iris dataset. Following was the accuracy recorded.

	Mean Accuracy	Max Accuracy	Min Accuracy
K-means	0.82	0.89	0.54
FCM(Alpha = 1.8)	.89	0.89	0.89
WEFCM(Alpha = 1.1)	0.97	0.97	0.98

Table 4.1: Performance Comparison of WEFCM with Traditional algorithm

The table shows the performance analysis of WEFCM. It is clear that WEFCM outperforms traditional clustering methods like FCM and K-means. Hard clustering algorithms are more sensitive to the initial cluster centers, which we can see that for K-means clustering performance fluctuated greatly. WEFCM can outperform FCM because of the introduction of attribute weight factor.

4.2.2 ORL Dataset

After WEFCM outperformed traditionl clustering algorithm, we test WEFCM for face recognition, using ORL Dataset. And also we will try to find optimal values for clustering parameters.

Locally linear Embedding

Following is the result obtained for different values of Alpha, Gamma, n-estimators and Iterations, when Dimention reduction is done by locally linear embedding on ORL Dataset. We have recorded the values of NMI, ARI and Accuracy.

Alpha	Gamma	Iteration	n-components	NMI	ARI	Max Accuracy	Min Accuracy	Mean Accuracy
1.25	-100	160	50	0.88	0.6	0.82	0.76	0.74
1.17	5	100	50	0.88	0.67	0.82	0.64	0.74
1.17	5	100	32	0.88	0.65	0.79	0.76	0.71
1.7	5	100	32	0.89	0.69	0.79	0.74	0.72
1.5	-100	100	50	0.88	0.67	0.76	0.74	0.71

Table 4.2: Result of LLE for ORL

Here, n-components is the dimenstion for manifold for locally linear embedding. We have recorded values for a large number of combinations of alpha, gamma, etc. But we have shown here only the values which were giving optimal results. Here the variation in accuracy is large because of the fact that we are randomly initialising weights.

Extra Tree Classifier

Following is the result obtained for different values of Alpha, Gamma, n-estimators and Iterations, when Dimention reduction is done by Extra Tree Classifier on ORL Dataset. We have recorded the values of NMI, ARI and Accuracy.

Here, n-estimator is the number of trees in the forest. We have recorded values for a large number of combinations of alpha, gamma, etc. But we have shown here only the values which were giving optimal results. Here the variation in accuracy is large because of the fact that we are randomly initialising weights.

Alpha	Gamma	Iteration	n-estimator	NMI	ARI	Max Accuracy	Min Accuracy	Mean Accuracy
1.7	-100	130	50	0.87	0.6	0.77	0.66	0.74
1.5	-100	130	50	0.88	0.67	0.82	0.69	0.76
1.5	-100	140	50	0.87	0.64	0.74	0.61	0.71
1.5	-100	160	50	0.89	0.69	0.79	0.69	0.74
1.5	-100	200	50	0.88	0.67	0.76	0.64	0.71

Table 4.3: Result of Extra Tree Classifier for ORL

Principal Component Analysis

Following is the result obtained for different values of Alpha, Gamma, n-components and Iterations, when Dimention reduction is done by Principal Component Analysis on ORL Dataset. We have recorded the values of NMI, ARI and Accuracy.

Alpha	Gamma	Iteration	n-component	NMI	ARI	Max Accuracy	Min Accuracy	Mean Accuracy
1.7	10	100	75	0.86	0.59	0.74	0.69	0.72
1.7	15	100	75	0.85	0.579	0.74	0.66	0.71
1.7	32	100	75	0.88	0.65	0.74	0.69	0.71
1.7	20	100	75	0.89	0.69	0.79	0.74	0.72
1.5	-100	100	50	0.88	0.67	0.76	0.74	0.71

Table 4.4: Result of PCA for ORL

Here, n-components is the number of components to keep after PCA. We have recorded values for a large number of combinations of alpha, gamma, etc. But we have shown here only the values which were giving optimal results. Here the variation in accuracy is large because of the fact that we are randomly initialising weights.

When we compare the results of the three dimension reduction algorithm we find that Locally linear embedding is the fastest of the three. Although the results of LLE and extra tree classifier were comparable, the result of lle were better in terms of accuracy and speed. But extra tree classifier was showing comparetively less fluctuation than LLE. The

result of LLE were better because it preserves local properties, neighborhood information is maintained.

4.2.3 Real Dataset

Following is the result obtained for different values of Alpha, Gamma, n-components and Iterations, when Dimention reduction is done by Principal Component Analysis on Real Dataset. We have recorded the values of NMI, ARI and Accuracy.

Alpha	Gamma	Iteration	n-component	NMI	ARI	Accuracy
1.7	32	100	25	0.745	0.549	0.894
1.5	1.4	100	25	0.747	0.557	0.894
1.5	5	100	25	0.779	0.5849	0.894
1.7	5	100	25	0.75	0.5411	0.841
1.5	5	30	25	0.758	0.526	0.789

Table 4.5: Result of PCA for Real Dataset

Here, $n-component$ is the number of components to keep after PCA. We have recorded values for a large number of combinations of alpha, gamma, etc. But we have shown here only the values which were giving optimal results. The accuracy for real data set is better as compared to ORL dataset.

Chapter 5

Conclusion and Future Work

In this project we implemented Weight Entropy Fuzzy C-Means and applied it for face recognition. We used three different menthods for dimenstion reduction for this, Locally linear embedding, Extra tree classifier and principal component analysis. Out of these three dimensionality reduction methods, the best results in term of speed and accuracy were given by locally linear embedding. The normalised mutual information for face recognition using locally linear embedding dimension reduction method was coming around 88-89 percent.

In Future work we are planning to extend this project to implement WEFCM in parallel environment. We are planning to implement a scalable version of WEFCM in Apache Spark. And merge this scalable version with Random Fuzzy Clustering Algorithm, which is applied on big data. So we are planning to implement a clustering algorithm for Big data which will have huge number of data points with big number of features.

Bibliography

[1] James C Bezdek, Robert Ehrlich, and William Full. Fcm: The fuzzy c-means clustering algorithm. *Computers & Geosciences*, 10(2-3):191–203, 1984.

[2] Long Chen and Lingning Kong. Fuzzy clustering in high-dimensional approximated feature space. In *Fuzzy Theory and Its Applications (iFuzzy), 2016 International Conference on*, pages 1–6. IEEE, 2016.

[3] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine learning*, 63(1):3–42, 2006.

[4] John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[5] Roweis. Lle publication. `https://cs.nyu.edu/~roweis/lle/publications.html`.

[6] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000.

[7] M Saquib Sarfraz, Olaf Hellwich, and Zahid Riaz. Feature extraction and representation for face recognition. In *Face recognition*. InTech, 2010.

[8] Lawrence K Saul and Sam T Roweis. An introduction to locally linear embedding. *unpublished. Available at: http://www. cs. toronto. edu/˜ roweis/lle/publications. html*, 2000.

[9] Jin Zhou and CL Philip Chen. Attribute weight entropy regularization in fuzzy c-means algorithm for feature selection. In *System Science and Engineering (ICSSE), 2011 International Conference on*, pages 59–64. IEEE, 2011.