CLUSTERING

MUHAMMAD TAJWAR
MATRICULATION No. 888394

June 2021

ARTIFICIAL INTELLIGENCE : KNOWLEDGE REPRESENTATION AND PLANNING

PROFESSOR ANDREA TORSELLO

Contents

1 Problem Introduction

Perform classification of the Semeion Handwritten Digit data set using:

1. Latent Class Analysis
2. Mean Shift
3. Normalized Cut

mean shift and normalized cut assume that the images are vectors in a 256-dimensional Euclidean space.

Provide the code and the extracted clusters as the number of clusters k varies from 5 to 15, for LCA and normalized-cut, while for mean shift vary the kernel width. For each value of k (or kernel width) provide the value of the Rand index :

$$R = 2(a + b)/(n(n - 1))$$

where

- n is the number of images in the dataset.
- a is the number of pairs of images that represent the same digit and that are clustered together.
- b is the number of pairs of images that represent different digits and that are placed in different clusters.

Then finally we have to explain the main difference between these models

2 Background

2.1 Classification Problem

We are given the Semeion Handwritten Digit Data set which consists of 1593 handwritten digits from approx 80 persons that were scanned and stretched in a rectangular box the pictures are 16x16 in pixels in a grayscale of 256 values for the storing management purpose each pixel of the images was scaled into (1/0) Boolean's value using a fixed threshold
People were asked to write on paper all the digits from 0 to 9 in two different ways first they wrote all the digits in a normal way that is typing the digits very accurately and then again write all the digits in fast manners hence with less accuracy.
Now our task is to perform three different kinds of classification models on this data set latent class analysis, Mean Shift, and Normalized Cut, and analyze how they behave on the data set.

2.2 Latent Class Analysis

The purpose of LCA is to classify your dataset into latent classes/groups. It is more advantageous compared to traditional cluster analysis due to its output for model fit. Then, you can judge whether your classification is appropriate or not. Latent class model(LCM) mainly used in the statistic that related to observed a set of multivariate variables and convert it to latent variables sets, and in this set, we have a discrete latent variable in a latent class model. The model will give you the results as conditional probabilities in the class, and the class will tell the chances of variables to take certain values.
Latent class analysis(LCA) works on structural equation modeling that is used to categorize multivariate data into subtypes of classes and groups and these are called the latent classes. Lets take an example that will clear your mind to consider disease in the patients the researcher will choose LCA to figure out the data, in the data we have four types of symptoms W, X, Y, Z measured in a range of people with three diff disease A, B, C consider disease A has w,x,y and disease B with x,y,z and disease C with w,y,z
Now the LCA model will detect the presence of disease classes that creates the patterns of association in the symptoms so in this case LCA can be used to classify the patterns in diseases according to maximum likelihood.

2.3 Mean Shift

Mean shift is a famous clustering technique it is also known as the non-parametric feature space analysis technique and it locates the maxima of the density function that why it is often called mode seeking algorithm. Mean shift also does not need to know knowledge of clusters in the dataset and it does not constrain the cluster shape, it can also be used for image segmentation and object tracking based on the nearest neighbor instead of the Parzen kernel density estimator mean shift works as given n data points $x_i, i = 1, \ldots, n$ on a d-dimensional space R^d, it depends on the kernel K which is multivariate kernel density function estimate and window radius h is given by:

$$f(x) = \frac{1}{nh^d} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)$$

it allows much higher dimensionalities and much faster classification it calculates the distance using euclidean distance to measure the distance between data points and search for the clusters. If we use piecewise constant kernels means shift works equivalent to Newton's method that which is bound optimization, in convergence to mean shift it can be further improved by updating the sample data dynamically in the iterations this optimization is aka as Dynamic Mean Shift(DMS) works with the Gaussian kernel

2.4 Normalized Cut

Normalised Cut is a very famous algorithm mainly used for image segmentation and other image processing operations to overcome the problem of unbalanced clusters, Normalized Cut is used it is based on the values of similarity that is measured between pairs of elements. The approach we use is a spectral clustering method is uses the properties of eigenvectors in a form of a matrix which is calculated using similarities.
The normalized cut scheme mainly focuses on the graph partitioning problem like if you have an eigenvector graph that represents the two classes in the data set to normalized cut will cut the graph separating the two classes apart. The graph consists of nodes which are the elements and weights that connect the two nodes according to the similarities measure normalized cut divides the graphs into subgraphs with high similarities between the same subgraph and low between different subgraphs.
let G be a graph with V nodes in it and W be the weight matrix and value W(u,v) is a similarity between the two nodes u,v we can split the graph into two sets A and B so that $A \cup B = V$ and $A \cap B = \emptyset$ by removing the edges connecting two-part and the cost is :

$$cut(A, B) = \sum_{u \in A, v \in B} W(u, v)$$

3 Latent Class Analysis

Latent class analysis(LCA) is also very similar to latent profile analysis(LPA) both techniques have the aim to recover groups that are hidden in the observed data. As aforementioned both the techniques are similar to clustering but with more flexibility, as they work on the explicit model of the data keeping the fact that the groups to be recovered are uncertain. LCA and LPA are very useful when we have two types of data continuous and categorical respectively.

3.1 Mixture Modeling

Mixture modeling refers to the art of unscrambling eggs that if you have observed data you are trying to find hidden groups from it before that you need to know what hidden groups are of kinds what the probability to get back at distributions in groups to find which person belongs to which group, mixture modeling is very useful in two ways:

- we are interested in a specific thing but we measured another for instance if student answer exam question it will be a suggestion that student might learn the material or not.
- we build a model but still find out that it may work differently for specific people and we are interested in how.

We have different kind of latent variable models like in table below:

Observed	Models for means		Regression models	
	Latent		Latent	
	Continuous	Discrete	Continuous	Discrete
Continuous	Factor analysis	**Latent profile analysis**	Random effects	Regression mixture
Discrete	Item response theory	**Latent class analysis**	Logistic ran. eff.	Logistic reg. mix.

According to the table, we have a lot of different variables and complex to handle and extract and we might like to convert these into interpretable groups. If we want a better understanding of how the mixture model works we have to study more about the statistical procedures like capture-recapture models, regression mixture modeling, noncompliance, and randomized response these all are used for image processing computer vision.
The mixture model is sometimes called latent variable modeling because it would help us to find your variables in unobserved data, moreover, it concerns with the discrete latent variable, for the nominal or ordinal variables(unobserved) we defined them into classes\groups or we can say mixture components as shown in the above table rows will tell us about continuous and discrete variables and columns tell us about the latent variables and on the left side of the table we have means models and right side concerns regression type models, but in this paper, we mainly focus on the **latent profile analysis and latent class analysis** these models also called **Gaussian Mixture Model and Binomial Mixture Model** respectively.

3.2 Latent Profile Analysis as Mixture of Categorical Var

We are many focused on the **Latent Class Analysis** but for the better understanding as a literature review we have to explain latent profile analysis with an example, for instance, if we want to measure the height of the people and then separates height for men and women as can show distribution as :

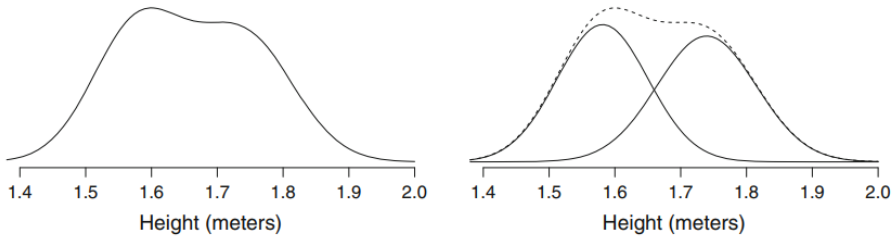

Figure 1: Peoples' height. Left observed distribution. Right men and women separate, with the total shown as a dotted line

Let us first discuss the problem and then we will explain how we can tackle this problem, so if you see the figure we think of the graph as not normal it has two peaks which tell us that it might have two groups we know these two groups likely to be men and women but unfortunately, we haven't recorded the sex of people, one way to separates these curves is using height as women have different(usually small) height as compare to men in most of the cases.
Unfortunately, we don't have information on people' sex and this is might happen usually when collecting data for instance if we are collecting information on attitude, feelings, and people behaviors it is impossible to get correct data, so in this case, we have only heights and there are some hidden groups in the data that we have to recover using latent class analysis.
Now this problem can be solved by mixture modeling our main goal is to find a distribution that is on the right side of Figure 1, we do not know the sex of people so we discover the distribution according to the men and women height, as we gathered more data it is more likely to surmise the distribution. For this we have to assume that height of men and women is normally distributed and of course, we don't know the sex so this assumption is unverifiable so we are left with the left side of Figure 1. For this assumption, a mathematical model would be

$$p(height) = Pr(man)Normal(\mu_{man}, \sigma_{man}) + Pr(women)Normal(\mu_{woman}, \sigma_{woman}) \tag{1}$$

can be written as

$$p(height) = \pi_1^x Normal(\mu_1, \sigma_1) + (1 - \pi_1^x)Normal(\mu_2, \sigma_2) \tag{2}$$

from Figure 1 the probability curve of height is equal to the sum of the two normal curves at this point we have to confront the label switching problem which is we don't know which classes represents which sex so we write π_1^X instead of Pr(man)\Pr(woman) as X will be a dummy variable, moreover π_1^X indicates unknown parameter probability, where X represents the variable takes on the value 1. The next step is to find the means and standard deviations of the distribution i.e $(\mu_1, \mu_2, \sigma_1, \sigma_2)$ which are in class parameters we give the initial value as random and calculates the posterior probability for men and women and we keep on guessing and updating the probabilities until no more change so after doing some calculation in this way we might recover hidden groups from the continuous observed data. Latent profile analysis can be done in many ways including Mplus, Latent Gold, and in R for instance in R

```
library(mclust)
height_fit_mclust <- Mclust(height)
summary(height_fit_mclust, parameters = TRUE)
```

3.3 Latent Class Analysis as Mixture of Categorical Variables

As aforementioned the Latent class analysis has very resemblance to latent profile analysis both the algorithm finds out the hidden groups in the data, from the table above we can see that latent class analysis deals with categorical variables moreover the main difference between LPA and LCA for the variables we assume no particular distribution that is for the observed variables categories of being selected probability is unknown. In LCA we have made some assumptions on the variables that they are distributed in a specific way i.e the observed variables do not have any relation to each other within a class this is called local independence so it creates classes where variables are similar to each other but different from other classes. Now, what the researcher will do is:

- Optimized number of classes K defining observed data.
- summarizes the results obtained.
- relates the similar classes with some external data variables outside the model.

our goal is to use Latent class analysis on the Semeion handwritten data set consisting of images but scaled to 1 and 0 as R language does not take 0 or null so we have to convert it into 1 and 2, the data set contains 256 variables and 1593 instances so we have huge data, one thing we should take into account is about uncertainty in each observation class if not it may produce biased estimates which are often called "three-step modeling" can be performed by software packages like Mplus or Latent Gold.
let's suppose we have three variables as an example but in real cases, we have 256 variables. let's say A, B, and C, then the model would be :

$$\pi_{abc}^{ABC} = \sum_{x} \pi_x^X \pi_a^{A|X} \pi_b^{B|X} \pi_c^{C|X} \tag{3}$$

where X represents the Latent Class variable, π_x^X is the size of class x and $\pi_a^{A|X}\pi_b^{B|X}\pi_c^{C|X}$ are the probabilities that $A, B, C \ldots$ so on takes the value $a, b, c \ldots$ in the latent class x, and conditionally independence would observed in the product $\pi_a^{A|X}\pi_b^{B|X}\pi_c^{C|X}$ Semeion data Representation is :

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14
1	1	1	1	1	1	1	2	2	2	2	2	2	2	2
2	1	1	1	1	1	1	2	2	2	2	2	2	2	2
3	1	1	1	1	1	1	1	1	1	2	2	2	2	2
4	1	1	1	1	2	2	2	2	2	2	2	2	2	1
5	1	1	1	1	1	2	2	2	2	2	2	2	2	1
6	1	1	1	2	2	2	2	2	2	2	2	2	2	1
7	1	1	1	1	2	2	2	2	2	2	2	2	2	2
8	1	1	1	1	1	1	1	1	1	2	2	2	2	2
9	1	1	1	1	1	2	2	2	2	1	1	1	1	1
10	1	1	1	1	1	1	2	2	2	2	2	2	2	2

Figure 2: Data Representation

one of the requirements for this assignment is to vary the number of classes from 5 to 15 K so these are the results I got.

Classes	Log-likelihood	AIC	BIC
5	-217241.2	437050.4	443949.8
6	-213071.9	429225.8	437506.2
7	-210191.9	423979.9	433641.2
8	-207717.6	419545.2	430587.5
9	-205772.6	416169.3	428592.5
10	-202380.5	403703.2	423703.2
11	-201119.9	407891.7	423076.9
12	-200363.2	406892.4	423458.5
13	-197149.1	400978.2	418925.3
14	-195598.7	398391.4	417719.4
15	-194713	**397134**	**417843**

The lowest value for AIC and BIC are in boldface.

Our goal is to find out the best number of class K such that these relationships in the classes are small.
We have the categorical data we apply the latent class analysis to the 1593 observations. I have done the code in R and also in Python just for the results. In R we use the poLCA package to run analysis.

```
library("poLCA")
# select variables
data <- read.delim(file.choose(),header = TRUE)

View(data)

# define function
f<-with(mydata, cbind(V1,V2,V3,V4,V5,V6,V7,V8,V9,V10,V11,)~1) #
    upto V257
```

```
9
10 M1 <- poLCA(f, data=data ,nclass = 5,graphs = TRUE,na.rm = TRUE)
11 M2 <- poLCA(f, data=data ,nclass = 6,graphs = TRUE,na.rm = TRUE)
12 M3 <- poLCA(f, data=data ,nclass = 7,graphs = TRUE,na.rm = TRUE)
13 M4 <- poLCA(f, data=data ,nclass = 8,graphs = TRUE,na.rm = TRUE)
14 M5 <- poLCA(f, data=data ,nclass = 9,graphs = TRUE,na.rm = TRUE)
15 M6 <- poLCA(f, data=data ,nclass = 10,graphs = TRUE,na.rm = TRUE)
16 M7 <- poLCA(f, data=data ,nclass = 11,graphs = TRUE,na.rm = TRUE)
17 M8 <- poLCA(f, data=data ,nclass = 12,graphs = TRUE,na.rm = TRUE)
18 M9 <- poLCA(f, data=data ,nclass = 13,graphs = TRUE,na.rm = TRUE)
19 M10 <- poLCA(f, data=data ,nclass = 14,graphs = TRUE,na.rm = TRUE)
20 M11 <- poLCA(f, data=data ,nclass = 15,graphs = TRUE,na.rm = TRUE)
```

so we fit the model as increasing number of classes from K=5 to K=15 so we have to choose the best model among the classes and there are many ways to measure the model fitness ones being are BIC(Bayesian Information Criterion) and AIC(Akaike Information Criterion). As we see from the table lower the AIC and BIC the better the model is so the best model would be a 15 class model that gives us the solution.

4 Mean Shift

Mean shift algorithm famously known as nonparametric clustering technique which does not have knowledge or require the knowledge of the number of clusters in the data set the algorithm made clusters by calculating the distance between the points and shifts the points accordingly. Given n data points $x_i, i = 1, \ldots, n$ on a d-dimensional space R^d, it depends on the kernel K which is multivariate kernel density function estimate and window radius h is given by:

$$f(x) = \frac{1}{nh^d} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)$$

and if we are working on the radially symmetric kernels then we define the kernel k(x) like this

$$K(x) = c_{k,d} k(||x||^2) \tag{4}$$

where the $c_{k,d}$ is the constant normalization that will intergrates the K(x) to 1.If we want to find the density function we have to look at the zeros of gradient function $\nabla f(x) = 0$
The gradient of density is :

$$\begin{aligned}
\nabla f(x) &= \tfrac{2c_{k,d}}{nh^{d+2}} \textstyle\sum_{i=0}^{n} (x_i - x) g\left(\left|\left|\tfrac{x - x_i}{h}\right|\right|^2\right), \\
&= \tfrac{2c_{k,d}}{nh^{d+2}} \left[\textstyle\sum_{i=1}^{n} g\left(\left|\left|\tfrac{x - x_i}{h}\right|\right|^2\right)\right] \left[\tfrac{\sum_{i=1}^{n} x_i g(||\frac{x - x_i}{h}||^2)}{\sum_{i=1}^{n} g(||\frac{x - x_i}{h}||^2)} - x\right]
\end{aligned}$$

where $g(s) = -k^{'(s)}$ the first term in the equation is to estimate the density at x that is computed by kernel $G(x) = c_{g,d}g(||x||^2)$ and second term will be

$$\mathbf{m}_h(x) = \frac{\sum_{i=1}^{n} x_i g(||\frac{x-x_i}{h}||^2)}{\sum_{i=1}^{n} g(||\frac{x-x_i}{h}||^2)} - x \tag{5}$$

that will be the mean shift, one of the main function of the mean shift is that it always directs us to the maximum increase in density. The means works as

- Computing mean shift vector $m_h(x^t)$
- Computing window translation $x^{t+1} = x^t + m_h(x^t)$

these points will converge you to find the density function where gradient is zero. The picture below illustrates the process of Mean Shift mode.

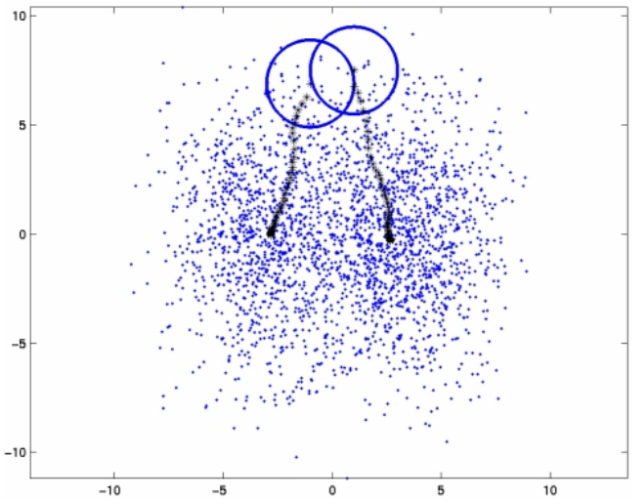

Figure 3: Mean Shift Mode Finding

- starting from the random data points we run the algorithm that will find stationary points of the function
- finding the local maxima meanwhile pruning these points

we define the points on the basin of attraction that the mode will converge and those points which have the same basin of attraction will be in same cluster.

4.1 An Intuition

Our main goal is to find clusters in the data set mean shift doesn't need to know predicted clusters in the data set so it looks for the centroids which the algorithm will determine on the data density points. After the post-processing, the algorithm will prune all the near repeated centroids. The figures below will demonstrate all the working of the algorithms
first, it will find the centers by picking an arbitrary center.

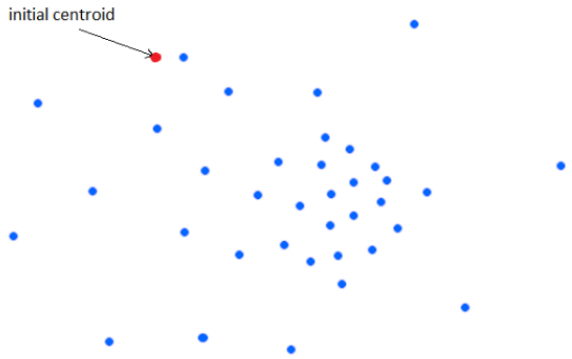

Figure 4: Finding Centroid

The algorithm will move this center point to the direction where data point density is highest and this can be done by calculating mean shift directions meanwhile updating centroid x.

4.2 Mean Shift Vector

One thing the algorithm will do is calculates the mean shift vector we have given amount of data points x_i in n-dimensional space then vector can be expressed as

$$v_s = \frac{1}{K} \sum_{x_i \in S_k} (x_i - x) \tag{6}$$

where v_s is the mean shift vector and S_k are the data points and $(x_i - x)$ is the distance from a point to x less then the radius r in the circle, it can be represents as

$$S_h(x) = \{\ y : (y - x_i)^T (y - x_i) < r^2\} \tag{7}$$

graphically we have this

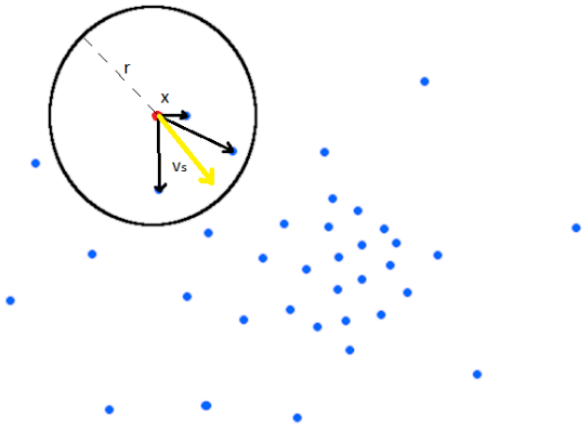

Figure 5: Vector Calculation

4.3 Updating Centroid

The algorithm will update the centroid like this.

$$x := x + v_s \tag{8}$$

which gives

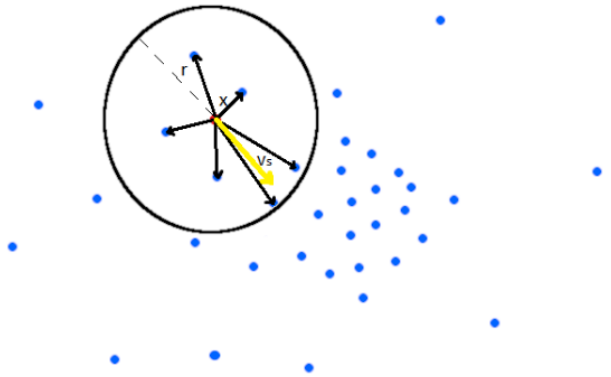

Figure 6: Updating Centroid

This process will loops until no change in x is seen that means the algorithm have found the optimal center with the highest data density, which look like this.

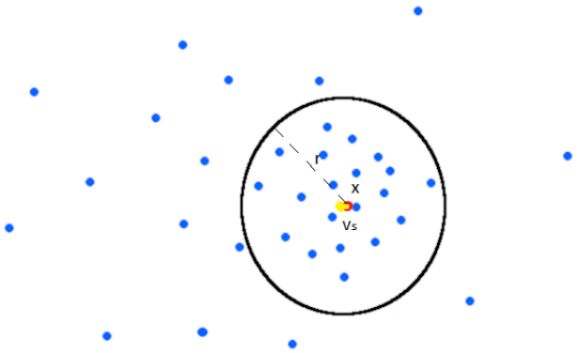

Figure 7: Optimal Centroid

when the mean shift algorithm finds the vector V_s very small this will converged and optimized centroid for cluster has found.

4.4 Kernel Density Estimation

After representing the data in the mathematical manner we have to apply the mean shift algorithm, according to the representation of the data we can choose the different kernels that fit our model. The base of the Mean Shift algorithm depends on the kernel density estimation(KDE) it is a method to estimate the probability density function for a particular data set
The algorithm will place a kernel on each data point it is a weighting function that calculates the weights of vectors, there are many types of the kernel if we sum all the kernels it will produce a probability surface if we vary the kernel bandwidth it will also affect the variance of the density function.

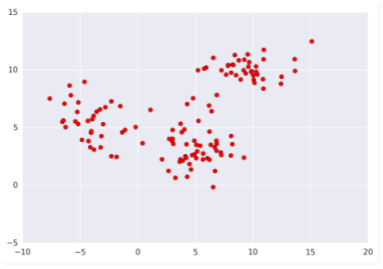

Figure 8: Data Points

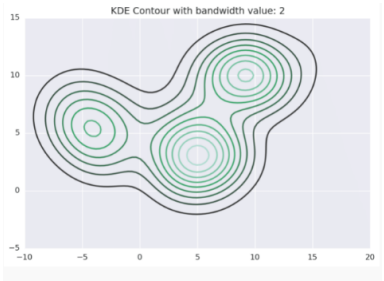

Figure 9: Applying KDE

Before and after using Gaussian kernel with bandwidth 2

4.5 Choices of Kernel Function

We have to choose kernel according to the several regularity conditions that must apply for some function to validate kernel functions, this includes integrated to unity function ($\int K(x))dx = 1$) with no singularities ($sup|K(.)| < \infty$), $\int |K(x)| < \infty$ and $lim_{||x||\to\infty}||x||^k K(x) = 0$

we can define these kernel functions from R to R before giving the data to the kernels we take the norm of the data, we generally use three kind of kernels.

- Gaussian Kernel

Gaussian kernel requires the covariance matrix that being identified by:

$$K(x) = \frac{1}{\sqrt{2\pi}} exp\{ -\frac{1}{2}||x||^2\}$$

- Uniform Kernel

$$K(x) = 1\{ ||x|| < 1\}$$

- Epanechnikov Kernel

$$K(x) = \frac{3}{4}(1 - |x|^2)1\{ |x| < 1\}$$

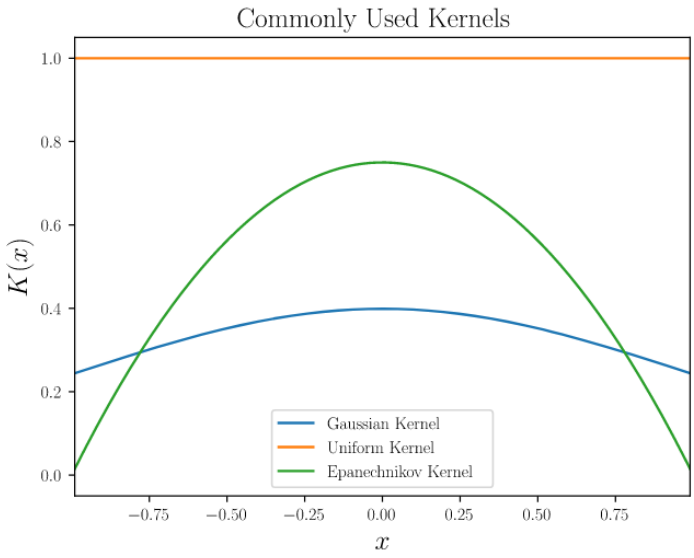

Figure 10: Commonly Used Kernels

4.6 Implementation

One of the tasks of this paper is to implement the Mean shift on the Semeion Handwritten digits data set the mean shift algorithm will tell us the number of clusters that been identified correctly i.e if the picture represents digit 0 the algorithm put this data point in a right cluster or if the algorithm does not classify it correctly and put it in other clusters
However, we also have to vary the number of clusters K from 5 to 15 and with this K value we provide the Rand Index that is

$$R = 2(a+b)/(n(n-1)) \ ,$$

where

- n is the number of images in the dataset.
- a is the number of pairs of images that represent the same digit and that are clustered together.
- b is the number of pairs of images that represent different digits and that are placed in different clusters.

```
import mean_shift as ms

#data = pd.read_csv("handwritten/semeion.data", delimiter=r"\s+",
    header=None)
data = np.genfromtxt('handwritten/semeion.csv', delimiter=',')
mean_shifter = ms.MeanShift()
mean_shift_result = mean_shifter.cluster(data, kernel_bandwidth =
    10)

mean_shifter = ms.MeanShift(kernel='multivariate_gaussian')
mean_shift_result = mean_shifter.cluster(data, kernel_bandwidth =
    [10,20,30,40,50,60,70,80])

original_points =  mean_shift_result.original_points
shifted_points = mean_shift_result.shifted_points
cluster_assignments = mean_shift_result.cluster_ids
print(original_points)
print(shifted_points)
print(cluster_assignments)
```

According to the data we have to use the kernels in our case we can use a multivariate gaussian kernel then the algorithm will calculate the original points and after calculating the distances by using this

```
import math
import NumPy as np


def euclidean_dist(pointA, pointB):
    if(len(pointA) != len(pointB)):
        raise Exception("expected point dimensionality to match")
    total = float(0)
```

```
    for dimension in range(0, len(pointA)):
        total += (pointA[dimension] - pointB[dimension])**2
    return math.sqrt(total)


def gaussian_kernel(distance, bandwidth):
    euclidean_distance = np.sqrt(((distance)**2).sum(axis=1))
    val = (1/(bandwidth*math.sqrt(2*math.pi))) * np.exp(-0.5*((
    euclidean_distance / bandwidth))**2)
    return val


def multivariate_gaussian_kernel(distances, bandwidths):

    # Number of dimensions of the multivariate gaussian
    dim = len(bandwidths)

    # Covariance matrix
    cov = np.multiply(np.power(bandwidths, 2), np.eye(dim))

    # Compute Multivariate gaussian (vectorized implementation)
    exponent = -0.5 * np.sum(np.multiply(np.dot(distances, np.
    linalg.inv(cov)), distances), axis=1)
    val = (1 / np.power((2 * math.pi), (dim/2)) * np.power(np.
    linalg.det(cov), 0.5)) * np.exp(exponent)

    return val
```

Both the kernels will take distance and bandwidth as a parameter and will return the value of the kernels.
We vary the clusters from 5 to 15 and these are the results.

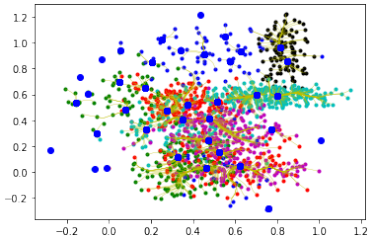

Figure 11: K=5

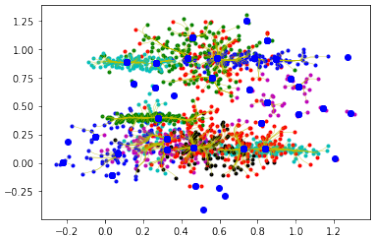

Figure 12: K=6

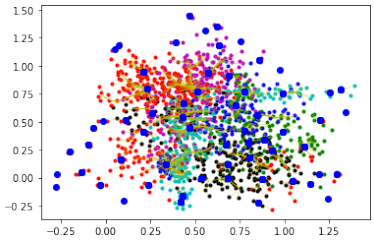

Figure 13: K=7

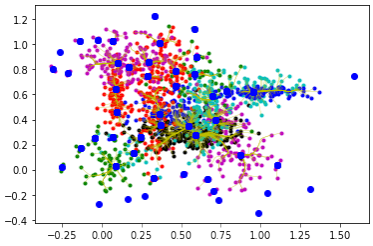

Figure 14: K=8

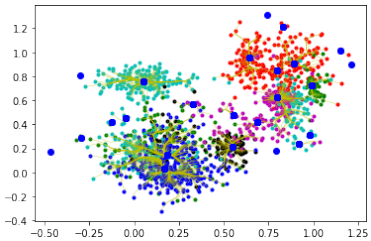

Figure 15: K=9

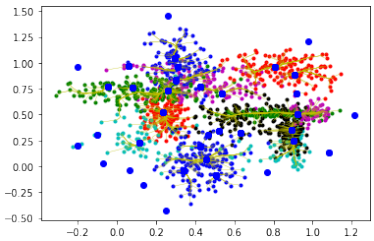

Figure 16: K=10

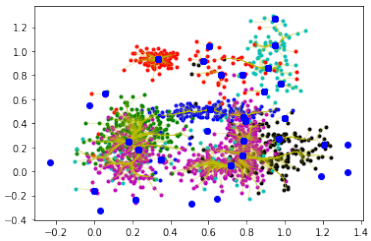

Figure 17: K=11

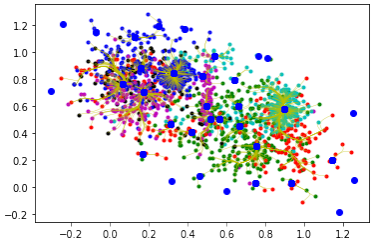

Figure 18: K=12

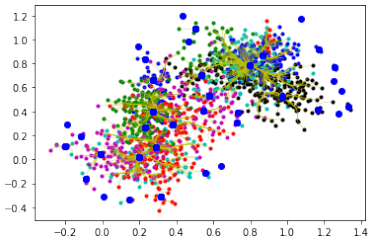

Figure 19: K=13

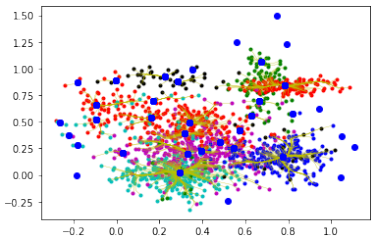

Figure 20: K=14

The Accuracy of the Mean shift when we vary K from 5 to 15 results in the table below :

Clusters	Accuracy
K=5	24.36618
K=6	16.98145
K=7	29.53507
K=8	32.61511
K=9	39.65184
K=10	**55.94336**
K=11	27.65184
K=12	17.90308
K=13	22.82783
K=14	26.78969
K=15	28.10856

The Highest Accuracy is in boldface.

5 Normalized cut

Normalized cut is a graph partitioning problem and is used for segmenting the graph. The criterion used in this algorithm measures both the total dissimilarity between the different groups as well as the total similarity within the groups. One of the several methods to optimize this criterion based on a generalized eigenvalue problem which is also not only an efficient computational technique but also tractable. Normalized cut is usually used for the image segmentation, classification problems, clustering, etc. Figure. 21 shows the difference between the compactness and the connectivity property of a data-set. For the data-set on the left side of Figure. 21, one can use mixture model methods such as K-means clustering or latent class analysis for the clustering and classification purposes, however, one cannot use these algorithms for the data on the right of the Figure. 21. For such kind of data, we employ techniques like a normalized cut that exploits the connectivity property of the data-set for classification.

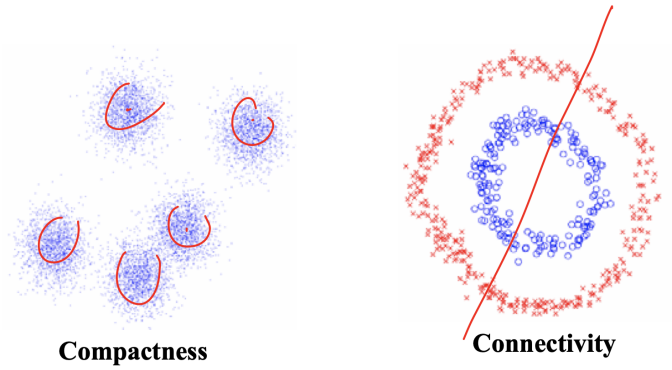

Figure 21: Compactness vs Connectivity

Normalized cut is mostly related to the graph-theoretic formulation of grouping. The set of points in arbitrary feature space are represented as a weighted undirected graph $G = (V, E)$, where the nodes of the graph are the points in the feature space, and an edge is formed between every pair of nodes. The weight on each edge, $w(i, j)$, is a function of the similarity between nodes i and j.

Our goal is to partition the set of vertices into several disjoint sets $\mathbf{V}_1$, $\mathbf{V}_2$, ..., $\mathbf{V}_m$, whereby some measure the similarity among the vertices in a set $\mathbf{V}_i$ is high and, across different sets $\mathbf{V}_i, \mathbf{V}_j$ is low. However, there are two main questions while partition that arise:

- What is the precise criterion for a good partition?
- How can such a partition be computed efficiently?

5.1 Grouping as graph partition

A graph $G = (V, E)$ can be partitioned into two disjoint sets, $A, B, A \cup B = V, A \cap B = \emptyset$,by simply removing the edges connecting the two parts. The degree of dissimilarity between these two pieces can be computed as total weight of the edges that have been removed. Mathematically, a cut is defined as follows:

$$cut(A, B) = \sum_{u \in A, v \in B} w(u, v) \ . \tag{9}$$

The optimal cutting of a graph is the one that minimizes this cut value. Although there exists an exponential number of such partitions, however, finding the minimum cut of a graph is a well-studied problem and there exist efficient algorithms for solving it.

In the normalized cut, instead of looking at the value of total edge weight connecting the two partitions, we compute the cut cost as a fraction of the total edge connections to all the nodes in the graph as follows:

$$Ncut(A, B) = \frac{cut(A, B)}{assos(A, V)} + \frac{cut(A, B)}{assos(B, V)} \ , \tag{10}$$

where $assos(A, V) = \sum_{u \in A, t \in V} w(u, t)$ is the total connection from nodes in A to all nodes in the graph.

Similarly, we can define a measure for the total normalized association within groups for a given partition:

$$Nassos(A, B) = \frac{assos(A, A)}{assos(A, V)} + \frac{assos(B, B)}{assos(B, V)} , \tag{11}$$

where $assos(A, A)$ and $assos(B, B)$ are total weights of edges connecting nodes within A and B, respectively.

It is straightforward to see the relation between the $Ncut$ and $Nassos$ as follows:

$$\begin{aligned} Ncut(A, B) &= \frac{cut(A, B)}{assos(A, V)} + \frac{cut(A, B)}{assos(B, V)} \\ &= \frac{assos(A, V) - assos(A, A)}{assos(A, V)} + \frac{assos(B, V) - assos(B, B)}{assos(B, V)} \\ &= 2 - Nassos(A, B) . \end{aligned} \tag{12}$$

Therefore, the two partition criteria that in our Normalized cut, minimizing the disassociation between the groups and maximizing the association within the same groups.

The major issue of the normalized cut is the computational complexity. Particularly, the calculation of eigenvectors and weight matrix required a lot of time, therefore, it makes this algorithm impractical when it comes to large images, and defines the least possible node into a single pixel. Even though, efficiency can be dramatically improved by applying the normalized cut theorem by solving it in a matrix-free fashion. However, as long as the resolution of the image goes higher, the computation is always ill-conditioned, which makes the complexity much higher.

5.2 Algorithm

In this subsection, we provide an outline for the algorithm of the normalized cut as follows:

Algorithm 1: Normalized Cut

Result: Write here the result

Given an image generate a graph $G = (V, E)$ with vertices (V) and edges (E) and calculate the weights on each edge (w);

while *While step 4 is True* **do**

- Solve the equation $(D - W)y = \lambda Dy$ for the second smallest eigenvalue and eigenvector;
- Use the second smallest eigenvalue and eigenvector to calculate the minimum normalized cut and partition the graph into two new graphs ;
- Determine whether the two new graphs can be partitioned again given a minimum Ncut condition (which we set);
- If step four is valid we will recursively call the algorithm to partition again;

end

In order to calculate all the normalized cuts necessary we will need to solve the following equation:

$$\min_x Ncut(x) = \min_y \frac{y^T(D - W)y}{y^T Dy} \ , \tag{13}$$

where x be an $N = |V|$ dimensional indicator vector, $x_i = 1$ if node i is in A and -1, otherwise. Let us define the sum $d(i) = \sum_j w(i, j)$, where $w(i, j)$ is the weight of the node between i and j. Let D be an $N \times N$ diagonal matrix with d on its diagonal, W be an $N \times N$ symmetrical matrix with $W(i, j) = w(i, j)$. Moreover, we define k and b as:

$$k = \frac{\sum_{x_i>0} d_i}{\sum_i d_i} \ , \tag{14}$$

and

$$b = \frac{k}{1 - k} \ . \tag{15}$$

Finally, we define y as

$$y = (1 + x) - b(1 - x), \text{ where 1 is vector of ones .} \tag{16}$$

Solving the above problem/algorithm for real values of y implies we need to solve the following eigenvalue system:

$$(D - W)y = \lambda Dy \ . \tag{17}$$

After solving this we obtain two important things: i) the eigenvalues, and ii) eigenvectors of the system. Since we know according to the Rayleigh quotient that given a symmetric matrix the minimum value of an eigenvalue matrix is the second eigenvalue.

Therefore, we have

$$z_1 = \arg \min_{z^T z_0 = 0} \frac{z^T D^{-1/2}(D - W)D^{-1/2}z}{z^T z} \,. \tag{18}$$

The second smallest eigenvalue, where z is defined as $z = D^{1/2}y$,

$$y_1 = \arg \min_{y^T D1 = 0} \frac{y^T (D - W)y}{y^T Dy} \,. \tag{19}$$

In summary, we discussed the normalized cut criterion for graph partitioning and we have shown how this criterion can be computed efficiently by solving a generalized eigenvalue problem.

5.3 Implementation

According to the Normalized cut algorithm we follow the steps and for the better explanation how the algorithm works i can use some examples.
So, generally our dataset is composed of rows and column being features but somehow the normalized cut only be applied on the graphs of connected nodes, for example:

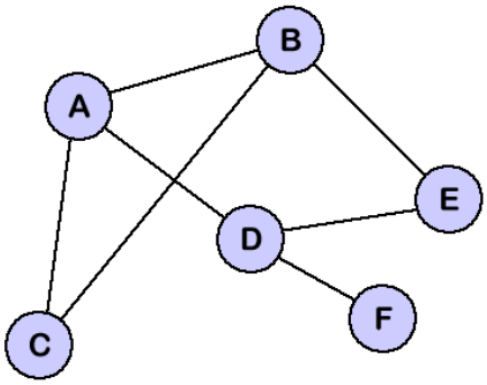

Figure 22: A Graph

so thats why we must transform our data which is in the form of table to a graph in our case the dataset is very big we have almost 1600 images to classify if we plot the images show them in the scatter points like this:

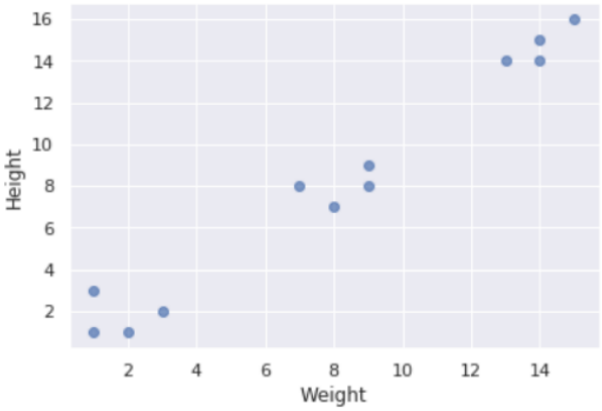

Figure 23: Data Points

According to the algorithm, we first create the similarity matrix which is of size NxN matrix, and afterward, we calculate the distance by using the euclidean formula between the data points.
Next, we copy the data of similarity matrix to build an adjacency matrix but with a constraint of a threshold, we predefined the limit if the distance is greater than then threshold we set it 0 and otherwise 1
With the help of the adjacency matrix, we easily make a graph 1's in the graph that will tell us the edges between two nodes and 0's not connected nodes.

```
W = pairwise_distances(X, metric="euclidean")
vectorizer = np.vectorize(lambda x: 1 if x < 5 else 0)
W = np.vectorize(vectorizer)(W)
print(W)
```

for visualizing graph we use the networkx library.

```
def draw_graph(G):
    pos = nx.spring_layout(G)
    nx.draw_networkx_nodes(G, pos)
    nx.draw_networkx_labels(G, pos)
    nx.draw_networkx_edges(G, pos, width=1.0, alpha=0.5)
```

From this code, we generate a graph and then calculates its adjacency matrix.

```
G = nx.random_graphs.erdos_renyi_graph(10, 0.5)
draw_graph(G)
W = nx.adjacency_matrix(G)
print(W.todense())
```

for example like this :

```
[[0 1 0 0 0 0 0 0 0 0]
 [1 0 1 0 1 0 1 1 0 1]
 [0 1 0 0 0 1 1 0 1 1]
 [0 0 0 0 1 1 0 0 1 0]
 [0 1 0 1 0 1 1 1 0 1]
 [0 0 1 1 1 0 1 0 1 0]
 [0 1 1 0 1 1 0 1 1 0]
 [0 1 0 0 1 0 1 0 0 0]
 [0 0 1 1 0 1 1 0 0 1]
 [0 1 1 0 1 0 0 0 1 0]]
```

Figure 24: Adjacency Matrix

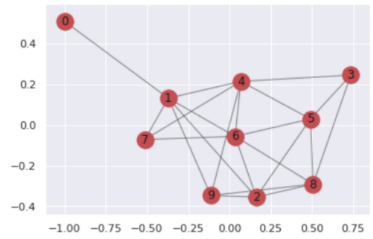

Figure 25: Graph

Now we see the whole graph as a single component and from one node we can transverse to any other node after the adjacency matrix we have to create the degree matrix and this can be done by summing all the numbers in the row of adjacency matrix into the degree matrix then by using the formula $L = D - W$ we compute the Laplacian matrix

```python
# degree matrix
D = np.diag(np.sum(np.array(W.todense()), axis=1))
print('degree matrix:')
print(D)
# laplacian matrix
L = D - W
print('laplacian matrix:')
print(L)
```

```
degree matrix:
[[6 0 0 0 0 0 0 0 0 0]
 [0 3 0 0 0 0 0 0 0 0]
 [0 0 8 0 0 0 0 0 0 0]
 [0 0 0 6 0 0 0 0 0 0]
 [0 0 0 0 5 0 0 0 0 0]
 [0 0 0 0 0 4 0 0 0 0]
 [0 0 0 0 0 0 4 0 0 0]
 [0 0 0 0 0 0 0 6 0 0]
 [0 0 0 0 0 0 0 0 3 0]
 [0 0 0 0 0 0 0 0 0 3]]
laphacian matrix:
[[ 6 -1 -1 -1 -1  0  0 -1 -1  0]
 [-1  3 -1  0 -1  0  0  0  0  0]
 [-1 -1  8 -1 -1 -1 -1 -1  0 -1]
 [-1  0 -1  6 -1  0  0 -1 -1 -1]
 [-1 -1 -1 -1  5  0 -1  0  0  0]
 [ 0  0 -1  0  0  4 -1 -1 -1  0]
 [ 0  0 -1  0 -1 -1  4 -1  0  0]
 [-1  0 -1 -1  0 -1 -1  6  0 -1]
 [-1  0  0 -1  0 -1  0  0  3  0]
 [ 0  0 -1 -1  0  0  0 -1  0  3]]
```

Figure 26: Laplacian Matrix

Laplacian Matrix has very special properties that help to classify the data.

There is the theory of the graph is that if we have K connected nodes then Laplacian L has K eigenvectors which have the 0 eigenvalue.
so now we have to create eigenvalues and eigenvectors, for one component eigenvalue would be 0.

```
e, v = np.linalg.eig(L)
# eigenvalues
print('eigenvalues:')
print(e)
# eigenvectors
print('eigenvectors:')
print(v)
```

```
eigenvalues:
[0.000 9.097 7.871 7.266 6.025 5.460 4.432 2.963 2.336 2.550]
eigenvectors:
[[-0.316 -0.192 0.422 0.619 0.280 0.346 -0.207 0.174 0.131 -0.117]
 [-0.316 -0.107 -0.044 -0.177 0.124 -0.190 0.473 0.145 0.737 -0.127]
 [-0.316 0.919 0.159 -0.012 -0.013 0.107 0.006 -0.077 0.061 0.090]
 [-0.316 -0.189 0.325 -0.726 -0.007 0.274 -0.312 0.216 -0.075 0.078]
 [-0.316 -0.075 -0.369 0.146 -0.641 0.015 -0.477 -0.092 0.297 -0.030]
 [-0.316 -0.165 0.130 0.019 -0.390 0.312 0.559 -0.373 -0.325 -0.221]
 [-0.316 -0.125 0.196 -0.036 0.273 -0.544 -0.232 -0.646 -0.061 0.002]
 [-0.316 -0.044 -0.681 -0.036 0.492 0.360 0.012 -0.128 -0.135 0.165]
 [-0.316 0.089 -0.180 0.021 0.038 -0.379 -0.028 0.463 -0.405 -0.578]
 [-0.316 -0.113 0.040 0.181 -0.156 -0.301 0.205 0.317 -0.225 0.738]]
```

Figure 27: Eigenvalues and Eigenvectors

Ploting these values

```
fig = plt.figure()
ax1 = plt.subplot(121)
plt.plot(e)
ax1.title.set_text('eigenvalues')
i = np.where(e < 10e-6)[0]
ax2 = plt.subplot(122)
plt.plot(v[:, i[0]])
fig.tight_layout()
plt.show()
```

Now we use ncut algorithm to cut the graph and have clusters like this.

```
 cutter = Ncut.dataset
    eigenvector = cutter.EigenSolver_sparse()
    b = eigenvector

    b = b.reshape((50,50)).astype('float64')
    b = (b/b.max())*255
    cv2.print('eigvec',b.astype('uint8'))
```

6 Comparison Between Latent Class Analysis, Mean Shift and Normalized Cut

When we talk about the mean shift is a very simple algorithm and easy to implement all the consequences of the algorithm depends on only one parameter which is kernel bandwidth whereas other clustering approaches such as latent class analysis and k-means requires or must know the knowledge of the clusters well if we talk about in general most of the time the clusters are unspecified. Latent Class Analysis is very different from the other two algorithms as LCA works mainly on conditional probabilities it uses the BIC and AIC values to choose the best model. Mean shift Clustering is also a very clever algorithm it exploits the density points to create the number of clusters whereas the latent class analysis identifies the unobserved groups that differ from the other groups that we often called the classes, comparing both from the normalized cut, the algorithm generates the graphs find the eigenvectors and eigenvalues and then cut the graph in a normalized way. Mean shift algorithms also comes with the disadvantage which is the slowness of the algorithm with a time complexity of $O(N^2)$ as it takes a long time to execute.

7 In a Nutshell

In a nutshell, all the algorithms work very well on different kinds of datasets so we compared the performance of the three clustering algorithms Latent class analysis, Mean Shift and Normalized Cut and studied how they perform in a different situation we have given the dataset more them 1500 people handwritten digit which is scaled to 0/1 value using these three algorithms we have to make 10 clusters where all the 0's and 1's and 2's so on and forth together in the same cluster. We have also discussed some advantages and disadvantages of the three algorithms and their behavior over the clusters values 5 to 15
Firstly we use Latent class analysis on the dataset and LCA tells us which are the unobserved groups then we made 10 models and give each model a different number of classes varying from 5 to 15 and we choose the best according to their BIC and AIC value, smallest of these values would present better model. Secondly we use Mean shift algorithm it is very power full algorithm we used TensorFlow architecture for the better representation of the clusters we vary the clusters values 5 to 15 for comparing the results and when we put K=10 the accuracy was higher. Lastly we use Normalized cut it converted the dataset into the graph and then creates three matrices diagonal matrix, similarity matrix, and laplacian matrix and compute the eigenvectors and values them cut the graph accordingly.