An approach to Multi View Deep Continuous Clustering using Subspace Projection

DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Technology
in
Computer Science

by

Agnip Mazumder

[Roll No: CS1909]

under the guidance of

Dr. Swagatam Das

Associate Professor
Electronics and Communication Sciences Unit

Indian Statistical Institute
Kolkata-700108, India
July, 2021

To my friends and family

CERTIFICATE

This is to certify that the dissertation entitled "**An approach to Multi View Deep Continuous Clustering using Subspace Clustering**" submitted by **Agnip Mazumder** to Indian Statistical Institute, Kolkata, in partial fulfillment for the award of the degree of **Master of Technology in Computer Science** is a bonafide record of work carried out by him under my supervision and guidance. The dissertation has fulfilled all the requirements as per the regulations of the institute and, in my opinion, has reached the standard needed for submission.

Swagatam Das
Associate Professor,
Electronics and Communication Sciences Unit,
Indian Statistical Institute,
Kolkata-700108, India

Acknowledgements

I would like to express my highest gratitude to my advisor, *Dr. Swagatam Das*, Associate Professor, Electronics and Communication Sciences Unit, Indian Statistical Institute, Kolkata, for his guidance and continuous encouragement. Without his support this work would not have been possible.

My utmost thanks goes to all the teachers of Indian Statistical Institute for their suggestions and discussions whenever I have needed them, which has undoubtedly helped me to improve my research work.

I would also like to thank all of my friends for their help and support. Last but not least, I am very much thankful to my parents and family for their everlasting support.

Agnip Mazumder
Indian Statistical Institute
Kolkata - 700108 , India

Abstract

As we know Clustering is an unsupervised machine learning algorithm, where in a collection of unlabelled data similar data items are grouped under same class and dissimilar data goes to different class. One common algorithm for clustering is K- Means Clustering, which initializes k initial centroids and data, centroids are alternately updated to converge to exact k clusters.

But in real life scenario it is not possible to predetermine the number of clusters from random data. For that we have **Robust Continuous Clustering** [1], which takes representative points for each data points. Initially these representative points are data points itself. Then representative points of data points that are likely to be under a same cluster converge at one point. Gradually the representatives move and collapse into easily distinguishable clusters.

There are more than one ways of looking at a data. Same data-set can be expressed in multiple forms as different data matrices. Such data are called **multi-view data** [2]. We cannot draw a conclusion about the clustering pattern just based on a single view, considering all views is important for clustering them into groups. Examples include a sentence can be visualised in multiple languages, a person can be visualised by their face data, personality,etc. We need to take all these views into consideration to get a bigger picture of the data pattern.

Using **Multi-View Continuous Subspace Clustering** [3], a consensus subspace representation is initialized. Applying the continuous clustering algorithm on subspace of cach view data points and summing the objective function of all views, we arive at a view consensus clustered structure.

High dimensional data poses a challenge as assumptions of many algorithms do not work in higher dimensions. We can use techniques to project the data in lower dimensions, then obtain the cluster structure in lower dimensions. But this algorithm of embedding in proper lower dimension and then obtaining cluster structure has been proven to be less effective than algorithm of dimension reduction and clustering on the low dimension latent spaces simultaneously in each step, also known as **Deep Continuous Clustering** [4]. It considers the reconstruction loss and the clustering loss together in each iteration and reduces both simultaneously. Thus we get a clustering of data points in lower dimensional latent space more effectively.

Now we propose to extend the Deep Continuous Clustering in Multi-View combining the idea of Deep Continuous Clustering and Multi-View Continuous Subspace Clustering, to cluster high dimensional multi-view data without pre-specifying the number of clusters efficiently.

Contents

1 Introduction **7**
- 1.1 Clustering . . . 7
  - 1.1.1 K-Means Algorithm . . . 7
- 1.2 Deep Clustering . . . 7
- 1.3 Multi-view learning . . . 8
- 1.4 Robust Continuous Clustering . . . 9
- 1.5 Multi-view subspace clustering . . . 9
- 1.6 Thesis Outline . . . 9

2 Robust Properties **10**
- 2.1 Properties of Robust Functions . . . 10
- 2.2 Different Robust Functions . . . 10
- 2.3 Family of Robust Functions . . . 12

3 Related Works **14**
- 3.1 Robust Continuous Clustering . . . 14
- 3.2 Deep Continuous Clustering . . . 16
- 3.3 Multi-view Subspace Clustering . . . 17
  - 3.3.1 Single View Subspace Clustering . . . 17
  - 3.3.2 Multi View Subspace Clustering . . . 18
- 3.4 Robust Multi-View Continuous Subspace Clustering . . . 18

4 Proposed Algorithm : Multi-View Deep Subspace Continuous Clustering **20**
- 4.1 Proposal . . . 20
- 4.2 Formulation . . . 20
- 4.3 Modified Algorithm . . . 23

5 Results **24**
- 5.1 Algorithms . . . 24
- 5.2 Experimental Result . . . 25
  - 5.2.1 Caltech101-7 Dataset . . . 25
  - 5.2.2 WebKb Dataset . . . 26
  - 5.2.3 UCI Digits Dataset . . . 26

6 Complexity Analysis **27**
- 6.1 Time Complexity . . . 27
- 6.2 Space Complexity . . . 29

7 Conclusion and Future Works **30**

Chapter 1

Introduction

1.1 Clustering

Clustering is an example of unsupervised machine learning algorithm where we have no knowledge of any labels or class of any data. It categorizes data into distinct set of groups or clusters

The clustering technique can be widely used in various tasks such as Market Segmentation, Social network analysis, Statistical data analysis, Image segmentation, Anomaly detection, etc.

Clustering can be of two types hard clustering, where data point completely belongs to one cluster or soft clustering, where data points can partially belong to multiple clusters.

Clustering methods can be Partitioning Clustering, Distribution-model based clustering, Hierarchical Clustering, Density-based clustering,etc.

1.1.1 K-Means Algorithm

One of the famous clustering algorithm is K-Means algorithm, with is a Partitioning Clustering algorithm which reduces the intracluster variances within the clusters,

Here we take K random , initial centroids , all points are clustered with centroids closest to the data point.Then we do centroids reallocation, centroid of the newly formed clusters.We alternatively continue these processes until the centroids converge.Thus we get exact K no of clusters or partitions.

Disadvantages of the algorithm are firstly that the centroids may converge at a local minimum giving undesirable results and secondly we need to have a prior knowledge of the number of clusters in advance.

1.2 Deep Clustering

Clustering of data in higher dimensional spaces becomes very difficult as many assumptions fail in higher dimension. In higher dimensions as the number of attributes increases, simple distance measure like euclidean distance, makes all point appear equidistant. This is the Curse of Dimensionality. To come up with a solution to cluster high dimensional data, we use **Deep Clustering** [4] [5]. It reduces the dimension of data points by embedding it it low dimensional space with the help of deep autoencoders. Here the reconstruction loss from the autoencoder and the

clustering loss are optimized together as a global objective function. These algorithm outperforms many state-of-the-art deep neural network algorithms used for clustering

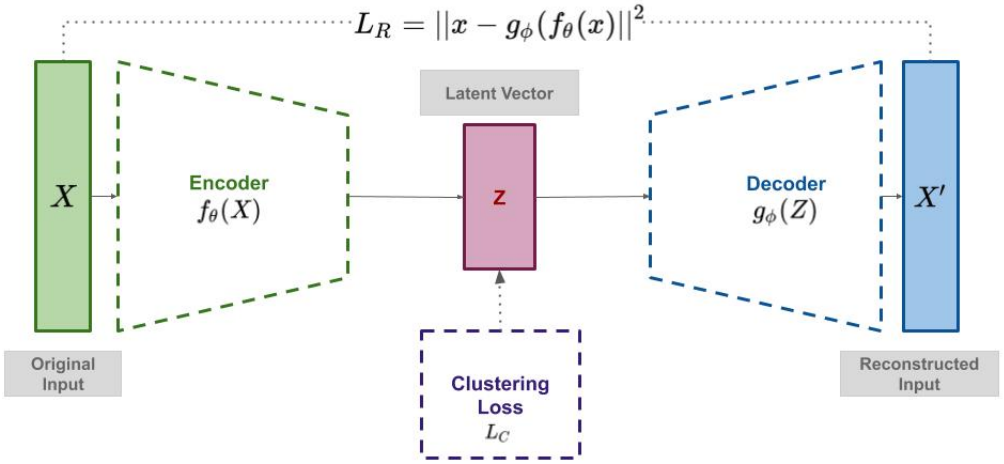

Figure 1.1: Deep clustering architecture

1.3 Multi-view learning

Multi-view data is real-world datasets, where different views describe different perspectives of looking at a same raw data [2]. Examples include a sentence can be visualised in multiple languages, a person can be visualised by their face data, personality,etc. We need to take all these views into consideration to get a bigger picture of the data pattern.

We extract different data matrices from different view, where data points in different view may lie in different feature spaces. We model the different views and learn the consensus data pattern, taking all views into consideration.

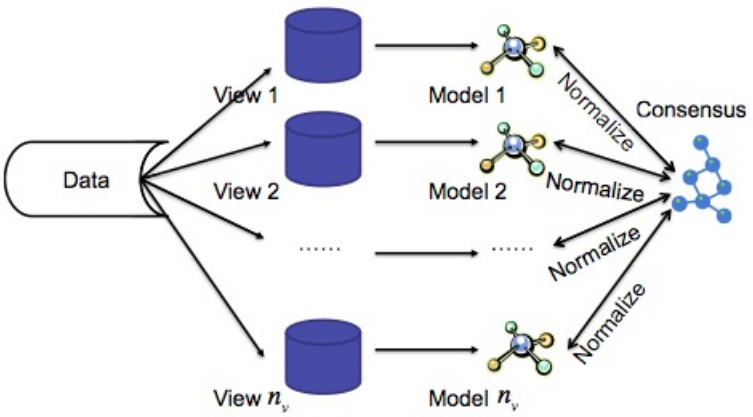

Figure 1.2: Multi-view learning

1.4 Robust Continuous Clustering

This is a clustering technique, where do not need the prior knowledge of the number of clusters in advance [1]. Unlike K-Means algorithm, where we initialize K random centroids and let them find their cluster points, here we assign each data points it's representative points as itself.

Then we let these representative points converge onto each other, whose data-points are likely to be in the same cluster. Finally the resultant distinct non-converged representative points are the cluster centers, and the data-points whose representative points collapsed into a point or are very close belong to the same cluster

1.5 Multi-view subspace clustering

Unlike Multi-view Kernel K-Means (MKKM) where we perform clustering on a common weighted view, we perform simultaneous subspace clustering on each view keeping the subspace representative same for all views [6], thus maintaining consistency, i.e. points in different views follow the same clustering pattern.

For a data-point in a particular view we try to form a subspace or linear combination of all data points in that view to be close to that data point and keeping the subspace representative of the same point constant in all views

Thus the resultant subspace representation points can be used to determine the multi-view clustering pattern.

1.6 Thesis Outline

In our thesis we are trying to cluster Multi-View Data having high dimensional points in different views without having any knowledge about the number of classes or clusters. We project all the view data into a latent space, where we perform a Robust multi-view subspace clustering. There we apply the Robust Continuous Clustering on latent space points of all views, and arrive at a subspace consensus representative of the latent space.Then all views that were embedded into latent space are reconstructed back to their original space. We optimize the data reconstruction loss, robust clustering loss of the subspace representative points of all views together.

The subspace representation of the lower dimensional latent space that we get determines the cluster assignment. By this approach we try to combine idea of Deep Continuous Clustering and Multi-View Continuous Subspace Clustering, to cluster high dimensional multi-view data without pre-specifying the number of clusters efficiently.

Chapter 2

Robust Properties

2.1 Properties of Robust Functions

It is always preferred to have a model that is less influenced by outliers than by inliers. The standard least square solution of minimizing $\sum_i E_i^2$ becomes unstable if there are outliers present in the data. Outlying data strongly affect the minimization process. Hence M-estimators instead of reducing the square of errors it replaces it with a function of the errors

$$min \sum_i \rho(E_i)$$

where $\rho(.)$ is a symmetric, positive-definite function
ρ has the following properties

- $\rho(r) > 0$
- $\rho(0) = 0$
- $\rho(r) = \rho(-r)$
- $\rho(r_1) > \rho(r_2)$ where $\mid r_1 \mid > \mid r_2 \mid$
- The slope of $\rho(r)$ must be less than slope of quadratic function on r.

2.2 Different Robust Functions

Different types of $\rho(.)$ functions [7] [8] [9] used are

- L_1 - Absolute value
  Here absolute value of the residual error is taken.

  $$\rho(r) = \mid r \mid$$

  It reduces the error but as it is not strictly convex, it may lead to instability.

- L_2 - Least square
  This estimator is convex

  $$\rho(r) = \frac{\mid r \mid^2}{2}$$

  But this estimator is not robust

- $L_1 - L_2$
  This maintains a balance between L_1 and L_2 taking the advantages of L1 which takes care of large residual error and of L2 helps maintaining convexity.

$$\rho(r) = 2(\sqrt{1 + \frac{r^2}{2}} - 1)$$

  For smaller values it is L_2 and like L_1 for larger values.

- L_p - Least Power

$$\rho(r) = \frac{\mid r \mid^v}{v}$$

  Value of v has to be small enough to be a robust estimator, i.e less disturbed by outliers.

- Huber's Function
  For normal distributions data that are affected by noise, it provides a min-max solution to them which can be extended to general distributions.

$$\rho(r) = \begin{cases} \frac{r^2}{2}, & \text{if } \mid r \mid \leq c \\ c(\mid r \mid -\frac{c}{2}), & \text{if } \mid r \mid > c \end{cases}$$

  This $\rho(.)$ function is used for multiple situation. New estimators are obtained varying the value of c.

- Modified Huber's Function
  Although Huber's method performs well. But as the second derivative is discontinuous, gradient values are not stable. So Modified Huber's Function is used

$$\rho(r) = \begin{cases} c^2(1 - cos(\frac{|r|}{c})), & \text{if } \frac{|r|}{c} \leq \frac{\pi}{2} \\ c \mid r \mid -c^2(1 - \frac{\pi}{2}), & \text{if } \frac{|r|}{c} > \frac{\pi}{2} \end{cases}$$

  This $\rho(.)$ function is not suitable for complex data and residual due to the presence of trigonometric functions.

- Cauchy's function
  This function is optimal for data following Cauchy's distribution.

$$\rho(r) = \frac{c^2}{2} log(1 + (\frac{\mid r \mid}{c})^2)$$

  This function decreases linearly only for large residual values

- Welsch's function
  This function further reduces the effects of larger error

$$\rho(r) = \frac{c^2}{2}(1 - exp(-(\frac{\mid r \mid}{c})^2))$$

- Geman-McClure's function
  It also reduces the effect of large errors.

$$\rho(r) = \frac{\frac{r^2}{2}}{1 + r^2}$$

2.3 Family of Robust Functions

These robust functions can be derived from a family of robust function[10] [8]

$$\rho(x, \alpha, c) = \frac{\mid \alpha - 2 \mid}{\alpha}\left(\left(\frac{(\frac{x}{c})^2}{\mid \alpha - 2 \mid} + 1\right)^{\frac{\alpha}{2}} - 1\right)$$

Here α is the shape parameter, c is the intrinsic parameter of the function.
When $\alpha = 2$ the function approaches L2 loss

$$\lim_{\alpha \to 2} \rho(x, \alpha, c) = \frac{1}{2}\left(\frac{x}{c}\right)^2$$

When $\alpha = 1$ the function approaches L1 loss

$$\rho(x, 1, c) = \sqrt{\left(\frac{x}{c}\right)^2 + 1} - 1$$

It is also referred as the Pseudo Huber's loss or L1-L2 loss which is L2 loss for smaller residual values to ensure convexity and L1 loss for reducing larger residual values.

When α tends to 0, we get Cauchy loss

$$\lim_{\alpha \to 0} \rho(x, \alpha, c) = log\left(\frac{1}{2}\left(\frac{x}{c}\right)^2 + 1\right)$$

When $\alpha = -2$ we get Geman McClure function

$$\rho(x, 1, c) = \frac{2\left(\frac{x}{c}\right)^2}{\left(\frac{x}{c}\right)^2 + 4}$$

When α tends to negative infinity, it approaches Welsch loss

$$\lim_{\alpha \to -\infty} \rho(x, \alpha, c) = 1 - exp\left(-\frac{1}{2}\left(\frac{x}{c}\right)^2\right)$$

For $\alpha = 2$ and $\alpha = 1$ the derivative function increases linearly and increases with saturating to a fixed value respectively.Here a larger residual will have larger impact.

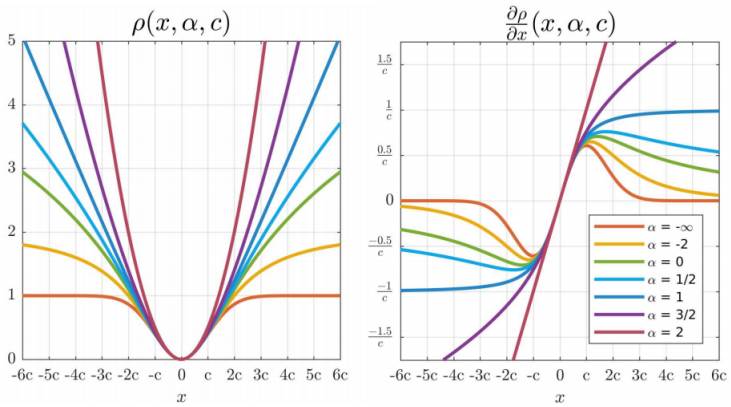

Figure 2.1: This shows the loss function on the left and its derivative function on the right for different values of α. When $\alpha = 2$ it is L2 loss, when $\alpha = 1$ it is Charbonnier loss, when $\alpha = 0$ it is Cauchy loss, when $\alpha = -2$ it is Geman McClure loss, and when $\alpha = -\infty$ it is Welsch loss

Now, for $\alpha < 1$ the derivative starts decreasing after a certain value. So for a larger residual value, impact becomes smaller and smaller as α becomes more and more negative. So for $\alpha = -2$, i.e. Geman McClure function have less impact from outliers and for $\alpha = -\infty$ i.e. Welsch function have very negligible or no impact from outliers.

We can use the concept of graduated non-convexity. First we take the function which is fully convex for the entire data. Over iteration, we introduce more non-convexity to reduce the gradients so that it becomes more and more invariant to outliers.

Chapter 3

Related Works

3.1 Robust Continuous Clustering

Robust Continuous Clustering [1] is an improvement over traditional K-Means algorithm, eliminating the drawbacks of random initialization, pre-specifying the number of clusters in advance. It presents an algorithm that is fast, easy to use which optimizes a continuous objective function. Unlike K-Means algorithm where set of random cluster centers are initialized and it iteratively converges to K clusters, here we assign representative points to the data-points itself and let the representative move and coalesce into easily separable clusters.

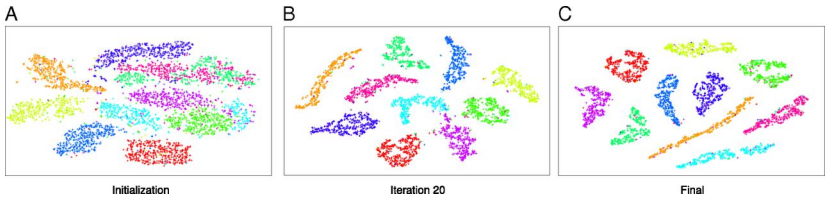

Figure 3.1: Robust Continuous Clustering

Let the input be $X = [x_1, x_2, ..., x_n]$, where $x_i \in R^D$
We take the set of representatives $U = [u_1, u_2, ..., u_n]$, where $u_i \in R^D$
We optimize the objective function on U, which coalesces to reveal the cluster structure.

The objective function of Robust Continuous Clustering is

$$\sum_{i=1}^{n} \parallel x_i - u_i \parallel_2^2 + \lambda \sum_{(p,q)\in\epsilon} w_{p,q}\rho(\parallel u_p - u_q \parallel_2)$$

ϵ is the weighted connectivity graph, for which first we construct the m-KNN graph or the modified K- nearest neighbour graph [11] which provides an edge between i^{th} and j^{th} data point if i^{th} point is one of the K nearest neighbours of j^{th} point and if j^{th} point is one of the K nearest neighbours of i^{th} point. Then to ensure connectivity we augment it with the Minimum Spanning Tree.

We set $w_{i,j}$ to maintain the contribution of every data-point to the pairwise loss.

$$w_{p,q} = \frac{\frac{1}{N}\sum_{k=1}^{n} n_k}{\sqrt{n_p n_q}}$$

Here n_p is the degree of p^{th} data-point in ϵ.
The parameter λ balances the ratio of the data loss and the pairwise loss that needs to be optimized. We set

$$\lambda = \frac{\parallel X \parallel_2}{\parallel A \parallel_2}$$

where

$$A = \sum_{(p,q)\in\epsilon} w_{p,q}(e_p - e_q)(e_p - e_q)^T$$

and $\parallel . \parallel_2$ denotes the spectral norm.

$\rho()$ is a penalty on the regularization term which tries to coalesce the representative points of data-points with higher $w_{p,q}$

So basically the first term tries to keep the representative points closer to the data-points and the second regularization term tries to collapse closer, similar representative points into a single point. More the value of $w_{p,q}$ more likely is the chance the two points will collapse. Here they take scaled Geman-McClure function as the penalty function to ensure robustness or invariant to outliers .

$$\rho(y) = \frac{\mu y^2}{\mu + y^2}$$

The objective function is re-written as

$$\sum_{i=1}^{n} \parallel x_i - u_i \parallel_2^2 + \lambda \sum_{(p,q)\in\epsilon} w_{p,q}(l_{p,q} \parallel u_p - u_q \parallel_2^2 + \Psi(l_{p,q}))$$

$l_{p,q}$ is a variable introduced which tends to 1 when connection is present and tends to 0 when connection is absent.$\Psi(l_{p,q})$ is the penalty imposed on ignoring $l_{p,q}$.

$$\Psi(l_{p,q}) = \mu(\sqrt{l_{p,q}} - 1)^2$$

This is done so that we can perform alternate minimization on U and L. When U is fixed, we compute L

$$l_{p,q} = (\frac{\mu}{\mu + \parallel u_p - u_q \parallel_2^2})^2$$

Although L is a function of U, we assume $l_{p,q}$ to be fixed while updating U
U is calculated as

$$min_U \parallel X - U \parallel_F^2 + \lambda \sum_{(p,q)\in\epsilon} w_{p,q} l_{p,q} \parallel U(e_p - e_q) \parallel_2^2$$

where e_p is the indicator vector with only p^{th} element set to 1.

We continue this alternate minimization and reduce μ every fourth iteration till a threshold. After maximum number of iteration or convergence of objective function we draw an edge between i and j if $\| u_i - u_j \|_2$ is less than a threshold. The connected components gives the clustering result.

This follows graduated non-convexity, where the value of μ is initially set very high to ensure local convexity. After a period of certain interval we reduce the μ value to introduce some non-convexity to the objective function. It helps to attain value closer to global minimum.

3.2 Deep Continuous Clustering

Clustering high dimensional data poses certain problems as assumptions used in certain algorithm break in higher dimensions. It tries to reduce the dimension of the data by embedding it in lower dimensional space. We can first reduce the dimension of data embed into lower dimension first then perform clustering on the lower dimension space.

This method is less effective than performing dimensionality reduction and clustering in lower dimension to be performed together by optimizing a single objective function. Deep Continuous Clustering [4] [12] is an improvement over Robust Continuous Clustering where high dimension data is projected into a lower dimension latent space and Robust Continuous Clustering objective function is applied on latent space data and data is reconstructed back to original data space. Both clustering loss and reconstruction loss are optimized under a single objective function.

$$\| X - G_\omega(Y) \|_F^2 + \sum_i \rho_1(\| z_i - y_i \|_2; \mu 1) + \lambda \sum_{(i,j)\in\epsilon} w_{i,j} \rho_2(\| z_i - z_2 \|_2; \mu 2)$$

where $Y = F_\theta(X)$
First part is the reconstruction loss, where F is the encoding the data into lower dimension Y with the help of parameter θ and Y is reconstructed back to original dimension with decoding function G with parameter ω
z_i is the RCC representative point of y_i data in latent space, so second part is the data loss which ensures y_i to be close to z_i
And third part is the pairwise loss that ensures that z_i i.e. representative point of y_i tends to collapse into a single point if their y_i are similar.

F_θ and G_ω mappings are performed by an autoencoder with fully connected layers performing ReLu or Rectified Linear Unit operation after each layer as activation function.
Weights in *epsilon* measure the connectivity of data points.$\rho 1$ and $\rho 2$ are the M-estimators which keep the representative points close to the data and bring similar data-points representatives to collapse respectively.Here we use Geman-McClure as

the M-estimator to reduce the impact of outlier points on the objective function.

$$\rho_1(x;\mu_1) = \frac{\mu_1 x^2}{\mu_1 + x^2}$$

$$\rho_2(x;\mu_2) = \frac{\mu_2 x^2}{\mu_2 + x^2}$$

First keeping Z constant we update the autoencoder parameters $\{\theta, \omega\}$ which generates an autoencoder latent vector Y and reduces the autoencoder reconstruction loss along with the data loss of the latent vectors with the representative points.

$$\parallel X - G_\omega(Y) \parallel_F^2 + \sum_i \rho_1(\parallel z_i - y_i \parallel_2; \mu 1)$$

Now fixing the parameters hence fixing Y we update Z the representative points

$$\parallel X - G_\omega(Y) \parallel_F^2 + \sum_i \rho_1(\parallel z_i - y_i \parallel_2; \mu 1) + \lambda \sum_{(i,j)\in\epsilon} w_{i,j}\rho_2(\parallel z_i - z_2 \parallel_2; \mu 2)$$

We continue this alternate minimization and reduce $\mu 1$ and $\mu 2$ every fourth iteration till a threshold. After maximum number of iteration or convergence of objective function we draw an edge between i and j if $\parallel z_i - z_j \parallel_2$ is less than a threshold. The connected components gives the clustering result.

3.3 Multi-view Subspace Clustering

Usually Subspace clustering clusters data assuming data-points are drawn from lower dimensional subspaces. Many subspace clustering algorithm are used which takes the subspaces representation of the data-points such that data-points of same cluster lie on the same lower dimensional subspace [6]. Usually these methods uses features from single view data.

They applied subspace clustering on multi-view data feature to uncover subspace structure and perform clustering on multi-view data. Clustering on multiple views are combined to form a consensus clustering on multiple views. MKKM or Multi-view K Kernel Means perform clustering on a weighted common view, whereas here Subspace Clustering is performed on each view, maintaining the subspace consistency among different views [6].This ensures data-points in different views end up in the same cluster.

3.3.1 Single View Subspace Clustering

Let the data-points be $X = [x_1, x_2, ..., x_n]$ where $x_i \in R^d$ and the subspaces are $Z = [z_1, z_2, ..., z_n]$ where $z_i \in R^n$ where z_i is the subspace representation of x_i. $X.z_i$ is the subspace data obtained by computing the i^{th} subspace representation on original dataset and we tend to reduce it from the actual data point.

We need to reduce $\| x_i - X.z_i \|_2^2$ for all data points. It boils down to

$$min_Z \| X - X.Z \|_F^2$$

s.t

$$Z(i,i) = 0, Z^T.1 = 1$$

Solving the optimization problem we get Z, the subspace structure, we get affinity matrix

$$W = \frac{Z + Z^T}{2}$$

$$min_F Tr(F^T.L.F)$$

$L = D - W$ and $d_{i,i} = \sum_j w_{i,j}$, F gives the cluster index

3.3.2 Multi View Subspace Clustering

Here $X_v \in R^{d_v X n}$ denotes the data in the v^{th} view. We perform subspace learning algorithm on each individual view.

We can assume $Z \in R^{nXn}$ as the consensus subspace on all view. Then the objective function becomes

$$min_Z \sum_v \| X_v - X_v.Z \|_F^2$$

s.t

$$Z^T.1 = 1, Z(i,i) = 0$$

To provide more flexibility to the subspaces in different views we assume different subspace representation Z_v for different views and we try to have all the Z_v close to a consensus subspace Z.Hence objective is

$$min_{Z,Z_v} \sum_v \| X_v - X_v.Z_v \|_F^2 + \lambda \sum_v \| Z - Z_v \|_2$$

s.t

$$Z_v^T.1 = 1, Z_v(i,i) = 0$$

Its a relaxed form where we still get the data points in different views closer to the same cluster.

3.4 Robust Multi-View Continuous Subspace Clustering

It is a method to extend the previously shown RCC algorithm to multi-view data [3]. It tries to learn common representative subspace across multiple views [13], alternatively minimizing the clustering loss and common representation subspace pairwise loss. Over the iteration subspace representative points of multi-view data move and collapse into well-defined clusters.

The clustering result from multiple views should be consistent. We take $Z = [z_1, z_2, ..., z_n]$ as the common representative subspace points where $z_i \in R^n$. $X^v \in R^{d_v X n}$ is the v^{th} view data matrix. x_i^v is the i^{th} data from v^{th} view and data in v^{th} view is of dimension d_v. The idea is to form a continuous objective function, which is to be optimized w.r.t the subspace representative Z.

$$\sum_v \{\sum_i \parallel x_i^v - X^v.z_i \parallel_2^2 + \lambda \sum_{(p,q)\in\epsilon} w_{p,q}^v \rho(\parallel z_p - z_q \parallel_2)\}$$

The first part helps to keep the subspace of the data close to the actual data and the second part tries to collapse the subspace representative points of similar data-points.We use Geman-McClure loss function to reduce the impact of outlier points on the objective function.This is summed over all views to form a complete objective function with variable Z.

The objective function is re-written as

$$\sum_v \{\sum_i \parallel x_i^v - X^v.z_i \parallel_2^2 + \lambda \sum_{(p,q)\in\epsilon} w_{p,q}^v (l_{p,q}^v \parallel z_p - z_q \parallel_2^2 + \Psi(l_{p,q}^v)\}$$

$l_{p,q}$ is the connectivity between two subspace representation points which tends to 1 when it is active and tends to 0 when connection is absent.And $\Psi(l_{p,q})$ is the penalty on ignoring a connection.

$$\Psi(l_{p,q}) = \mu(\sqrt{l_{p,q}} - 1)^2$$

Now it becomes a two step minimization problem minimizing $l_{p,q}$ and Z. First of all assuming Z to be constant

$$l_{p,q} = (\frac{\mu}{\mu + \parallel z_p - z_q \parallel_2^2})^2$$

Although $l_{p,q}$ is a function of Z, we assume it to be constant and optimize the function w.r.t Z. We optimize the function

$$\sum_v \left(\parallel X^v - X^v.Z \parallel_F^2 + \lambda \sum_{(p,q)\in\epsilon} w_{p,q}^v l_{p,q}^v \parallel z_p - z_q \parallel_2^2 \right)$$

We continue this alternate minimization and reduce μ every fourth iteration till a threshold. After maximum number of iteration or convergence of objective function we draw an edge between i and j if $\parallel z_i - z_j \parallel_2$ is less than a threshold. The connected components gives the clustering result.

Chapter 4

Proposed Algorithm : Multi-View Deep Subspace Continuous Clustering

4.1 Proposal

We have seen earlier different algorithms, Robust Continuous Clustering which eliminates the need to declare the number of clusters in advance, Deep Continuous Clustering which does RCC in latent space as sometimes clustering on higher dimensions can be difficult, Multi-view Subspace Clustering taking consensus subspace of datapoints in all views and Robust Multi-View Continuous Subspace Clustering Which is extending RCC to multi-view taking subspace of data as representative points and taking the multi-view subspace points to decide the consensus clustering.

Now we propose Multi-View Deep Subspace Continuous Clustering to extend Deep Continuous Clustering to multi-view using the concept of multi-view subspace clustering

4.2 Formulation

Let data be $X^v \in R^{d_v X n}$ be the data in v^{th} view
$Y^v = F_{\theta_v}(X^v)$ is the lower dimensional embedding of data in v^{th} view, and $G_{\omega_v}(Y^v)$ is the reconstructed data of the v^{th} view.
Reconstruction loss is $\parallel X^v - G_{\omega_v}(Y^v) \parallel_F^2$ in v^{th} view, where $Y^v = F_{\theta_v}(X^v)$.
Performing Robust Multi-View Continuous Subspace Clustering on all y_i^v , we assume z_i to be consensus subspace representation of y_i in all views.

We construct the connectivity graph ϵ for all views. We implement m-KNN, which is an improvement over KNN graph where instead of drawing an edge between i and K-nearest neighbour of i, we draw a edge between i and j if they are K-nearest neighbour of each other. This graph will possibly be disconnected. To ensure connectivity we augment it with the Minimum Spanning Tree of the KNN graph. We

calculate the weight of the edges as

$$w_{p,q} = \frac{\frac{1}{N}\sum_{k=1}^{n} n_k}{\sqrt{n_p n_q}}$$

where n_i is the number of edges incident on i^{th} point.

We optimize the objective function

$$\sum_{v}\{\parallel X^v - G_{\omega_v}(Y^v) \parallel_F^2 + \sum_{i} \rho_1(\parallel y_i^v - Y^v.z_i \parallel_2; \mu 1) + \sum_{(p,q)\in\epsilon} \rho 2(\parallel z_p - z_q \parallel_2; \mu 2)\}$$

where the first part is the reconstruction loss when $X^v \in R^{d_v \times N}$ is encoded into $Y^v \in R^{e \times N}$ and decoded back to the original dimension $R^{d_v \times N}$. Second part is the data loss. We get the encoded lower dimensional data points in all views and we assume z_i to be the subspace representation of y_i^v for all $v \in \{1, 2.....,V\}$. We put the subspace data loss in a M-estimator like Geman-McClure [14] and add them for all data in all views. Third part is the pairwise loss, which tries to bring the representative of possibly closer data-points or higher $w_{p,q}$ value in all views nearer and possibly collapse into each other.

$$\rho_1(x; \mu_1) = \frac{\mu_1 x^2}{\mu_1 + x^2}$$

$$\rho_2(x; \mu_2) = \frac{\mu_2 x^2}{\mu_2 + x^2}$$

Keeping Z constant, we can update the auto-encoder parameters $\{\theta_v, \omega_v\}$ for all views. We usually update the auto-encoder parameters to reduce only the reconstruction loss. But here, as changing the auto-encoder parameters changes the latent space vector Y_v the second part or the data loss part also changes with the parameters. Here the first loss we need to optimize is more than just the reconstruction loss of the autoencoder. It's the reconstruction loss added with the data loss of the encoded data that we optimize with the autoencoder parameters.

We update the term $\parallel X^v - G_{\omega_v}(Y^v) \parallel_F^2 + \sum_i \rho_1(\parallel y_i^v - Y^v.z_i \parallel_2; \mu 1)$ in each view individually as they are view independent, using Adam Optimizer.

$$\min_{\{\theta_v, \omega_v\}} \parallel X^v - G_{\omega_v}(Y^v) \parallel_F^2 + \sum_{i} \rho_1(\parallel y_i^v - Y^v.z_i \parallel_2; \mu 1)$$

Now we fix the parameters $\{\theta_v, \omega_v\}$ and Y^v for all views and update Z by updating the objective

$$\min_{Z} \sum_{v}\{\sum_{i} \rho_1(\parallel y_i^v - Y^v.z_i \parallel_2; \mu_1) + \sum_{(p,q)\in\epsilon} \rho 2(\parallel z_p - z_q \parallel_2; \mu_2)\}$$

We initially take μ_1 and μ_2 to be very high to assume convexity on the entire data. After a certain number of iterations we periodically introduce certain non-convexity in the data. After a large number of iterations, we don't let larger residual value

affect the objective function more. So we reduce the μ_1 and μ_2 to it's half every M^{th} iteration till it reaches a threshold. This helps in reaching the global minimum faster.

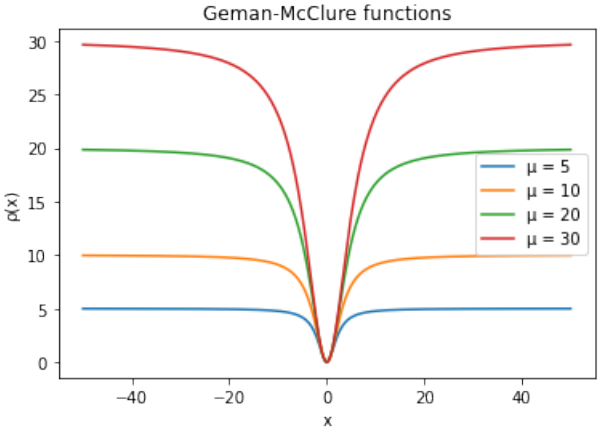

Figure 4.1: Influence of Hyperparameter on Geman-McClure function

We keep iterating till convergence or till *max_iteration* number of epochs. We obtain Z where z_i is the representation point of the i^{th} point. After a certain epoch representative points of the same clusters must have collapsed or be nearer to each other than representative points of other clusters.So we form a graph $G = (V, F)$ where representative points having distance lesser than threshold, i.e. $\| z_i - z_j \|_2 < \delta_2$ are given an edge ,$f_{i,j} = 1$. The connected components of the graph reveal the cluster results.

Algorithm 1: MVDSCC Algorithm

Data: $\{X^v\}_{v=1}^V$ where $X^v \in \mathbb{R}^{d_v \times N}$: the data matrix in all views
Result: Consensus Clustering results of multiple views

Construct connectivity structure ϵ;
Initialize $\{\theta,\omega\}$ and Z;
Precompute $\{\lambda_v\}_{v=1}^V, \mu_1, \mu_2, \delta_1, \delta_2$;
while *Objective converges or iteration* $<$ *max_iteration* **do**
    Update the autoencoder parameters $\{\theta,\omega\}$ to get the suitable Y and reduce the sum of reconstruction and data loss summed over all views ;
    Update Z, the subspace representation to reduce the data loss and pairwise subspace loss summed over all views;
    For every M epoch set $\mu_1 = max(\frac{\mu_1}{2}, \frac{\delta_1}{2})$ and $\mu_2 = max(\frac{\mu_2}{2}, \frac{\delta_2}{2})$;
end
Construct graph $G = (V, F)$ with $f_{i,j} = 1$ if $\| z_i - z_j \|_2 < \delta_2$;
Connected components of G give the clustering result

4.3 Modified Algorithm

Here we may try a more relaxed form of optimization where we don't assume a common subspace representative point for a data-point in all views and optimize w.r.t the common Z. Instead we take Z^v, individual subspace representatives for data-points in each view and then we optimize a consensus subspace representation Z, being close to all the individual subspace representation Z^v. In this way we can optimize the different view subspaces individually and yet maintain a consensus representation.

$$\sum_v \Big(\parallel X^v - G_{\omega_v}(Y^v) \parallel_F^2 + \sum_i \rho_1(\parallel y_i^v - Y^v.z_i^v \parallel_2; \mu 1) + \lambda_1 \sum_{(p,q)\in\epsilon} \rho 2(\parallel z_p^v - z_q^v \parallel_2; \mu 2) + \lambda_2 \parallel Z - Z^v \parallel_F^2 \Big)$$

This becomes a three step optimization instead of a two step optimization. First we update the autoencoder parameters to update Y, second we update all Z^v in all views and third we update Z, the consensus subspace representation of lower dimensional embeddings in all views.

Here we do not have to find a strict equality between the subspace representation in all views. We allow the representative points in all views to relax and form it's representation in it's own view, also maintaining closeness with a common representation. Representation points are not same across views, but share closeness with each other. This is much more flexible algorithm and may yield better result.

Algorithm 2: Relaxed MVDSCC Algorithm

Data: $\{X^v\}_{v=1}^V$ where $X^v \in \mathbb{R}^{d_v \times N}$: the data matrix in all views
Result: Consensus Clustering results of multiple views

Construct connectivity structure ϵ;
Initialize $\{\theta,\omega\}$, $\{Z^v\}_{v=1}^V$ and Z;
Precompute $\{\lambda_1^v\}_{v=1}^V, \lambda_2, \mu_1, \mu_2, \delta_1, \delta_2$;
while *Objective converges or iteration* $<$ *max_iteration* **do**
    Update the autoencoder parameters $\{\theta,\omega\}$ to get the suitable Y and reduce the sum of reconstruction and data loss summed over all views ;
    Update Z^v in all views like a single view ,i.e to reduce data loss and pairwise loss in each view individually but maintaining a closeness with Z the consensus representation;
    Update Z, the consensus subspace representation to have least sum of distance from all Z^v, the view specific representation ;
    For every M epoch set $\mu_1 = max(\frac{\mu_1}{2}, \frac{\delta_1}{2})$ and $\mu_2 = max(\frac{\mu_2}{2}, \frac{\delta_2}{2})$;
end
Construct graph $G = (V, F)$ with $f_{i,j} = 1$ if $\parallel z_i - z_j \parallel_2 < \delta_2$;
Connected components of G give the clustering result

Chapter 5

Results

5.1 Algorithms

- **RMKMC**: RMKMC (Robust Multi View K-Means Clustering) [15] [3] [16] [17] tries to obtain consensus clustering matrix across different views by optimizing a weighted sum of the relaxed K-means objective on each individual view .
- **PC-SPC**: PC-SPC (Pair-wised Co-regularised Multi-modal Spectral Clustering) [3] [18] [19] uses a pair-wised co-regularization term, to form a consensus representation matrix considering data in all views.
- **CC-SPC**: CC-SPC (Centroid Co-regularised Multi-modal Spectral Clustering) [20] [3] uses a centroid-based co-regularization term to form a consensus representation matrix considering data in all views.
- **MVSC**: MVSC (Multi-View Subspace Clustering)[3] [6] performs subspace clustering on each views individually and try to reach a consensus subspace representation.
- **RMVCSC**: RMVCSC (Robust Multi-View Continuous Subspace Clustering) [3] is an extension of RCC into multi-view which performs multi-view clustering without any previous information about the number of clusters. It takes consensus subspace of data in all views.

5.2 Experimental Result

5.2.1 Caltech101-7 Dataset

	Caltech101-7	
No of data points	1474	
No of views	6	
No of classes	7	
	Type of View	**No of Features**
View 1	Gabor	48
View 2	Wavelet Moments	40
View 3	Cenhist	254
View 4	HOG	1984
View 5	GIST	512
View 6	LBP	928

Approach	**AMI**	**NMI**
RMKMC	0.6034	0.5488
PC-SPC	0.6975	0.6547
CC-SPC	0.7047	0.6879
MVSC	0.6034	0.4766
RMVCSC	0.7360	0.7276
MVDSCC	0.4960	0.3188

5.2.2 WebKb Dataset

	WebKb	
No of data points	1051	
No of views	2	
No of classes	2	
	Type of View	**No of Features**
View 1	FullText	2949
View 2	Inlinks	334

Approach	**AMI**	**NMI**
RMKMC	0.8049	0.1592
PC-SPC	0.7659	0.0091
CC-SPC	0.5785	0.0019
MVSC	0.7802	0.0041
RMVCSC	0.9402	0.5694
MVDSCC	0.5258	0.0967

5.2.3 UCI Digits Dataset

	UCI-Digits	
No of data points	2000	
No of views	6	
No of classes	10	
	Type of View	**No of Features**
View 1	Fourier coefficients(FOU)	76
View 2	profile correlations(FAC)	216
View 3	Karhunen-Love coefficients(KAR)	64
View 4	pixel averages(PIX)	240
View 5	Zernike moments(ZER)	47
View 6	morphological features(MOR)	6

Approach	**AMI**	**NMI**
RMKMC	0.7853	0.8125
PC-SPC	0.8682	0.8267
CC-SPC	0.8768	0.8234
MVSC	0.8242	0.8399
RMVCSC	0.9312	0.8867
MVDSCC		0.4678

Chapter 6

Complexity Analysis

6.1 Time Complexity

The first part of the objective function is the reconstruction loss of the autoencoder. For v^{th} view data $X^v \in R^{d_v \times N}$ contains N data each of dimension d_v. x_i^v is a data in X^v of dimension d_v. Dimension d_v is encoded to dimension e and then decoded to dimension d_v.

Let us take a neural network , we have n_i nodes in i^{th} layer and n_{i+1} nodes in $(i+1)^{th}$ layer. $W_{i,i+1} \in R^{n_{i+1} \times n_i}$ is the weight matrix for forward propagation from i^{th} layer to $(i+1)^{th}$ layer. And $Z_i \in R^{n_i \times N}$ is the feedforward data after i^{th} layer and $S_i \in R^{n_i \times N}$ is the i^{th} layer activation function output.

$$Z_{i+1} = W_{i,i+1}.S_i$$

This takes time complexity of $O(n_{i+1} \times n_i \times N)$

$$S_{i+1} = f(Z_{i+1})$$

This takes complexity of $O(n_{i+1} \times N)$
Total complexity at i^{th} layer is $O((n_{i+1} \times n_i \times N) + (n_{i+1} \times N))$
$= O(n_{i+1} \times (n_i + 1) \times N)$
$= O(n_{i+1} \times n_i \times N)$

Similarly for back propagation we back-propagate the Weight matrix elements for N data . It also takes $O(n_{i+1} \times n_i \times N)$ in i^{th} layer.
Total time complexity of Multi-Layer Perceptron is $O(N \times \sum_i (n_{i+1} \times n_i))$

If we take the autoencoder to have no other hidden layers, d_v to e and back to d_v to optimize

$$\min_{\{\theta_v, \omega_v\}} \parallel X^v - G_{\omega_v}(Y^v) \parallel_F^2 + \sum_i \rho_1(\parallel y_i^v - Y^v.z_i \parallel_2; \mu 1)$$

Calculating $\rho_1(\parallel y_i^v - Y^v.z_i \parallel_2; \mu 1)$ takes $O(e \times N^2)$ for multiplication, $O(e \times N)$ for subtraction and $O(e \times N)$ for $\rho()$ operation. Net complexity is $O(e \times N^2)$

Forward propagation takes $O((e \times d_v \times N) + (d_v \times e \times N) + (e \times N^2))$
$= O(e \times N \times (d_v + N))$

Backward propagation of all autoencoder parameters to optimize the loss takes $O((e \times d_v \times N)$
Total time complexity of the first minimization is $O(e \times N \times (2.d_v + N))$ for each view.
$= O(e \times N \times (d_v + N))$
For all view it is

$$O\Big(\sum_{v=1}^{V}\Big(e \times N \times (d_v + N)\Big)\Big)$$

Now for the second optimization

$$\min_{Z} \sum_{v}\Big(\sum_{i} \rho_1(\parallel y_i^v - Y^v.z_i \parallel_2; \mu_1) + \sum_{(p,q)\in\epsilon} \rho 2(\parallel z_p - z_q \parallel_2; \mu_2)\Big)$$

Computing the first part i.e. data loss takes $O(e \times N^2)$. ϵ is the pairwise data weights upper limited by N^2. z_i is N dimensional vector, subtraction takes $O(N)$, norm calculation with $\rho()$ takes $O(N)$ and for all pair of values upper bounded by N^2, time complexity is $O(N^3)$.

Forward calculation of the second minimization takes

$$O\Big(\Big((e \times N^2) + (N^3)\Big) \times V\Big)$$

For backpropagating $Z \in R^{N\times N}$, we have N^2 values used in N data subspace for V views, time complexity is $O(N^3 \times V)$

Total time complexity of the second minimization is

$$O\Big(\Big((e \times N^2) + (N^3)\Big) \times V\Big)$$

Total time complexity is

$$O\Big(\Big((e \times N^2) + (N^3)\Big) \times V\Big) + O\Big(\sum_{v=1}^{V}\Big(e \times N \times (d_v + N)\Big)\Big)$$

$$= O\Big((e \times N^2 \times V) + (N^3 \times V)\Big) + O\Big((e \times N^2 \times V) + (e \times N \times \sum_{v=1}^{V} d_v)\Big)\Big)$$

$$= O\Big((e \times N^2 \times V) + (N^3 \times V) + (e \times N \times \sum_{v=1}^{V} d_v)\Big)\Big)$$

where e is the dimension of the encoded vector, d_v is the dimension of the v^{th} view data, N is the number of data-points and V is the number of views.

6.2 Space Complexity

$X_v \in R^{d_v \times N}$ takes space of $O(d_v \times N)$.For all view it is $O(N \times \sum_{v=1}^{V} d_v)$.
$Y_v \in R^{e \times N}$ takes space of $O(e \times N)$.For all view it is $O(N \times e \times V)$.
$Z \in R^{N \times N}$ takes space of $O(N^2)$.
In each view autoencoder has $(d_v \times e) + (e \times d_v)$ i.e. space complexity of $O(d_v \times e)$ in each view and $O(e \times \sum_{v=1}^{V} d_v)$ in all views.

Total Space Complexity is

$$O(N \times \sum_{v=1}^{V} d_v) + O(N \times e \times V) + O(N^2) + O(e \times \sum_{v=1}^{V} d_v)$$

$$= O\bigg(\Big((N+e) \times \sum_{v=1}^{V} d_v\Big) + \Big(N \times e \times V\Big) + \Big(N^2\Big)\bigg)$$

where e is the dimension of the encoded vector, d_v is the dimension of the v^{th} view data, N is the number of data-points and V is the number of views.

Chapter 7

Conclusion and Future Works

This algorithm of **Multi-View Deep Subspace Continuous Clustering (MVD-SCC)** is an approach to cluster Multi-view Data and arrive at a consensus cluster result after embedding it into a lower dimensional space. It tries to find the suitable subpace of the lower dimensional data of multiple views. As in multi-view data different view data has varying number of features. The number of features differ between multiple view in high magnitude. For eg view 1 is of dimension 1474×512 and view 2 is of dimension 1474×48. Applying same architecture of autoencoder to encode the two views into same encoded vectors can be counterproductive. Encoding both high feature data and low feature data in the same way may not lead to similar data preservation as high feature data requires a deeper network.
We may need a more flexible architecture for autoencoder i.e. different layers and different encoded vectors for different view encoding. This may improve the result.

Hyper-parameters play a major role in the convergence of the algorithm. Initialisation of hyper-parameter can be improved as different view data needs different attention on the ratio of optimization of data representation loss and pairwise representation loss.

Here one of the disadvantage is as the subspace representative of the lower dimensional vector come closer to each other, sometimes presence of few points between the clusters causes large problem. As the cluster results are computed by connected components of the graph, sometimes one edge mistakenly placed between two clusters results in entirely merging the two clusters together. This is how sometimes many clusters gets merged resulting in error. Proper threshold for construction of the graph is difficult as data in different view act differently. It is difficult to find the right threshold for the subspace of lower dimension encoding of varying data matrices.

Bibliography

[1] S. A. Shah and V. Koltun, "Robust continuous clustering," *Proceedings of the National Academy of Sciences*, vol. 114, no. 37, pp. 9814–9819, 2017.

[2] C. Xu, D. Tao, and C. Xu, "A survey on multi-view learning," *arXiv preprint arXiv:1304.5634*, 2013.

[3] J. Ma, R. Wang, W. Ji, J. Zhao, M. Zong, and A. Gilman, "Robust multi-view continuous subspace clustering," *Pattern Recognition Letters*, 2018.

[4] S. A. Shah and V. Koltun, "Deep continuous clustering," *arXiv preprint arXiv:1803.01449*, 2018.

[5] G. C. Nutakki, B. Abdollahi, W. Sun, and O. Nasraoui, "An introduction to deep clustering," in *Clustering Methods for Big Data Analytics*, pp. 73–89, Springer, 2019.

[6] H. Gao, F. Nie, X. Li, and H. Huang, "Multi-view subspace clustering," in *Proceedings of the IEEE international conference on computer vision*, pp. 4238–4246, 2015.

[7] P. Pennacchi, "Robust estimate of excitations in mechanical systems using m-estimators—theoretical background and numerical applications," *Journal of Sound and Vibration*, vol. 310, no. 4-5, pp. 923–946, 2008.

[8] J. T. Barron, "A more general robust loss function," *CoRR*, vol. abs/1701.03077, 2017.

[9] M. J. Black and A. Rangarajan, "On the unification of line processes, outlier rejection, and robust statistics with applications in early vision," *International journal of computer vision*, vol. 19, no. 1, pp. 57–91, 1996.

[10] J. T. Barron, "A general and adaptive robust loss function," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4331–4339, 2019.

[11] H. Parvin, H. Alizadeh, and B. Minaei-Bidgoli, "Mknn: Modified k-nearest neighbor," in *Proceedings of the world congress on engineering and computer science*, vol. 1, Citeseer, 2008.

[12] M. Caron, P. Bojanowski, A. Joulin, and M. Douze, "Deep clustering for unsupervised learning of visual features," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 132–149, 2018.

[13] Y. Li, M. Yang, and Z. Zhang, "A survey of multi-view representation learning," *IEEE transactions on knowledge and data engineering*, vol. 31, no. 10, pp. 1863–1883, 2018.

[14] P. Harjulehto, P. Hästö, and J. Tiirola, "Point-wise behavior of the geman–mcclure and the hebert–leahy image restoration models," *Inverse Problems & Imaging*, vol. 9, no. 3, p. 835, 2015.

[15] C. Chen, Y. Wang, W. Hu, and Z. Zheng, "Robust multi-view k-means clustering with outlier removal," *Knowledge-Based Systems*, vol. 210, p. 106518, 2020.

[16] R. Zhang, F. Nie, and X. Li, "Embedded clustering via robust orthogonal least square discriminant analysis," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2332–2336, IEEE, 2017.

[17] X. Cai, F. Nie, and H. Huang, "Multi-view k-means clustering on big data," in *Twenty-Third International Joint conference on artificial intelligence*, 2013.

[18] W. Zhuge, C. Hou, Y. Jiao, J. Yue, H. Tao, and D. Yi, "Robust auto-weighted multi-view subspace clustering with common subspace representation matrix," *PloS one*, vol. 12, no. 5, p. e0176769, 2017.

[19] M.-S. Chen, L. Huang, C.-D. Wang, and D. Huang, "Multi-view clustering in latent embedding space," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, pp. 3513–3520, 2020.

[20] A. Kumar, P. Rai, and H. Daume, "Co-regularized multi-view spectral clustering," *Advances in neural information processing systems*, vol. 24, pp. 1413–1421, 2011.