

Multi view Subspace Clustering using Good Neighbors

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

Master of Technology
in
Computer Science

by

Abhirup Gupta

[Roll No: CS-1907]

under the guidance of

Dr. Swagatam Das

Associate Professor

Electronics and Communication Sciences Unit

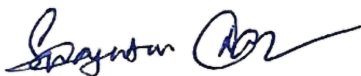


Indian Statistical Institute
Kolkata-700108, India
July, 2021

To my family and friends

CERTIFICATE

This is to certify that the dissertation entitled “**Multi View Subspace Clustering using Good Neighbors**” submitted by **Abhirup Gupta** 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 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 improve my research work.

I would also like to thank all of my friends for their help and support. In particular, I would like to express my appreciation to my classmate Sohan Ghosh for his time and valuable insights during the formation of this work. Last but not the least, I am very much thankful to my parents and family for their everlasting support.

Abhirup Gupta

Abhirup Gupta
Indian Statistical Institute
Kolkata- 700108, India

Abstract

We consider the problem of clustering N data points $\{x_i\}_{i=1}^N \in R^p$, into K number of clusters. We are dealing with high dimensional data points in our scenario where $p \gg N$, i.e. the number of features is much greater than the number of data points.

In our work, we set out to solve this problem using subspace clustering, assuming that our high dimensional data points lie in an union of low dimensional subspaces. We try to solve the problem of clustering in the context of multi view data. We find the self-expression matrices from each of the views using Entropy Norm formulation. Then, we find the consensus self-expression matrix by taking the average of all the individual self-expression matrices. Finally, we apply good neighbors post processing to obtain a sparser and strongly connected self-expression matrix thus resulting in an improved affinity graph. The resultant clusters are obtained using Normalized spectral clustering.

Contents

| | | |
|----------|----------------------------------------------------------------------------|-----------|
| 1 | Introduction | 9 |
| 1.1 | Introduction | 9 |
| 1.2 | Our Contribution | 11 |
| 1.3 | Thesis Outline | 11 |
| 2 | Preliminaries | 12 |
| 2.1 | Multi-View Learning | 12 |
| 2.2 | Sparse Subspace Clustering | 14 |
| 2.3 | General Approach for Multi view subspace clustering | 16 |
| 3 | Related Work | 19 |
| 3.1 | Sparse Subspace clustering using Entropy Norm | 19 |
| 3.2 | Subspace clustering via Good Neighbors | 21 |
| 4 | The proposed Multi View Good Neighbor Subspace Clustering Algorithm | 25 |
| 4.1 | Entropy Norm Formulation | 25 |
| 4.2 | Good Neighbors post processing | 26 |
| 4.3 | The Algorithm | 27 |
| 5 | Results | 29 |
| 5.1 | Complexity Analysis | 29 |
| 5.2 | Setup | 31 |
| 5.3 | Experimental results | 32 |
| 5.3.1 | Parameter Settings | 32 |
| 5.3.2 | On multi-view data sets | 32 |
| 6 | Conclusion and Future Work | 39 |
| 6.1 | Conclusion | 39 |
| 6.2 | Future Work | 39 |

List of Figures

| | | |
|-----|---------------------------------------------------------------------|----|
| 2.1 | Multi view Subspace clustering general procedure | 14 |
| 2.2 | Example of a self-Expression Matrix | 15 |
| 3.1 | Good Neighbors example | 23 |
| 5.1 | Caltech-7 Original Labels | 34 |
| 5.2 | Caltech-7 Predicted Labels | 34 |
| 5.3 | Handwritten Original Labels | 35 |
| 5.4 | Handwritten Predicted Labels | 35 |
| 5.5 | NMI and Purity for Caltech-7 data set | 36 |
| 5.6 | ACC for Caltech-7 and NMI for Handwritten digits data set | 37 |
| 5.7 | Purity and ACC for Handwritten digits data set | 38 |

List of Tables

| | | |
|-----|--------------------------------------------------------------------------------------------|----|
| 5.1 | Description of the data sets. Feature dimensions are provided inside parentheses | 31 |
| 5.2 | Performance evaluation for Caltech-7 data set | 32 |
| 5.3 | Performance evaluation for Caltech-20 data set | 33 |
| 5.4 | Performance evaluation for HandWritten Digits data set | 33 |
| 5.5 | Performance evaluation for Reuters data set | 33 |

List of Algorithms

| | | |
|---|------------------------------------|----|
| 1 | Good Neighbors Algorithm | 23 |
| 2 | Entropy Norm Algorithm | 26 |
| 3 | MVGNSC | 28 |

Chapter 1

Introduction

1.1 Introduction

Data is the buzzword of the 21st century. Every new technology that is introduced leverages the use of data in some form or the other. Data helps us make better and more efficient solutions to many real world problems. A data set normally comprises of a bunch of observations, each characterized by a set of features, which are the defining properties of the data objects. In recent years, *multi view data* has received much interest from researchers. In multi view data, each data point can be represented using multiple feature sets. Each of these feature sets correspond to a distinct *view* of the data points in the multi view data set. Since, there are multiple views a.k.a. multiple feature sets to describe the objects in the data set, these data sets are known as multi view data sets. We have talked in detail about multi view learning in Chapter 2.

The goal of unsupervised learning is to automatically label a set of data points in a data set without human intervention. One of the main research domains in the field of unsupervised learning is *clustering*. The main objective of clustering is to partition the data points into different partitions so that the points in one partition is similar to the points in its own partition and different from points in other partitions. Many researchers have tackled the problem of single view clustering, producing noteworthy results. One of the most famous and widely used single view clustering algorithm is Lloyd's K-means algorithm published in the year 1957. Many other related algorithms to K-means have also been published like K-Medoids [15], KMeans++ [2] being some of the famous ones.

Subspace based clustering methods assume that the data points in the data set lie in a union of low dimensional subspaces. To recover the clusters present in the data set, we need to extract the low dimensional subspaces in which the data points lie. Here, we take advantage of the *self-expressive* property, where each data points is represented as a weighted combination of all the other points in the same subspace. From the self-expressive matrix which is obtained by using the coefficients of the linear representation, we find the affinity matrix, to identify the different clusters. Subspace clustering has been studied in the following papers [10],[30],[23],[24]. There are also some other approaches to subspace based clustering, including statistical approach [29], kernel based approach [33], algebraic-geometry approach like GPCA [26] and iterative methods [5].

The main problem with using the above single view methods is that they consider only the features from a single source, not considering the feature from multiple sources. Different feature sets hold partly *independent* information about the data set. As an example a video can be described as a set of frames which have individual color information along with textual content and other related meta data. We use multi view data to take advantage of the complementary, independent information present in the distinct feature sets.

[12] is one of the first papers that tried to cluster data from multiple views. In the following years, after the idea of using multi view data for clustering gained traction amongst researchers, many papers have been published in this field. We will be highlighting some of the more commonly used methods down below, but the readers are encouraged to go through the survey paper by Yang et al. [32] to gain an in depth idea about the research going on in this field.

In [36], the authors have proposed learning the affinity matrix, consensus representation and final clustering labels matrix in a unified representation using single step optimization. In [16], a joint graph Laplacian is constructed after de-noising the information contained in the individual views. Steifel manifolds and k-means are used to optimize the objective function by minimizing the disagreement between the cluster structures in the consensus view and the individual views. In [28], the authors use manifold learning and tensor Singular Value decomposition(t-SVD) to unearth the clusters by finding the common hidden representation matrix, obtained from the individual affinity matrices in each view. In [19], the authors have proposed a

model which finds the affinity graphs, generates the partitions from the graph, and then learns the weights given to the individual views by generating multiple partitions and combining them using the learnt view specific weights to get a shared partition, from which we can derive the output cluster indicator matrix. Other latest works in this field include [7],[34],[21],[37].

The research done so far on multi view subspace clustering, either focus on optimizing the connectivity or the sparsity of the consensus self-expression matrix, but not both. In our proposed multi view subspace clustering using good neighbors algorithm, we have optimized both the aforementioned properties in the affinity graph, thus producing a consensus self-expression matrix better suited for clustering data in the multi view domain.

1.2 Our Contribution

- We propose a subspace clustering method based on the concept of good neighbors [31], which is also applicable to both single view and multi view data.
- We incorporate the concept of Entropy norm [3] to find out the self-expression matrices in the individual views, which is computationally less demanding in comparison to other related methods of computing the self-expression matrix.

1.3 Thesis Outline

The rest of the thesis is outlined as follows. Chapter 2 contains the preliminary concepts related to our method. Chapter 3 summarizes the related works done in Subspace clustering and Sparse subspace clustering. Chapter 4 discusses our proposed Multi View Good Neighbor Subspace Clustering(MVGNSC) Algorithm. In chapter 5, we provide the results of our experiments, along with discussing the complexity of our method. In chapter 6, we conclude our work and provide some directions in which future work can be done to extend our method.

Chapter 2

Preliminaries

2.1 Multi-View Learning

In this era where high quality data is more easily available, and data retrieval methods have become more sophisticated, data on an object can be generated from multiple different views. These different views can be visualised as different feature sets describing the objects in a data set. So, in Multi view learning, different views of the same data object can be fused together using advanced techniques to get a better idea about the inherent properties present in the data objects.

In multi view data, each view has partial information about some task which is used to gain knowledge about the given data set. Thus, the different views on the same data object provide some complementary information about the objects in the data set which can be exploited in various ways. Multi view learning is divided into two major sub domains, one is supervised learning and the other is unsupervised learning. This dissertation deals with the unsupervised aspect of multi view learning, specifically clustering.

The idea of clustering algorithms is to partition a set of data objects into various clusters that preserve similarity of data within clusters. In the context of multi view clustering, maintaining high quality of clustering in individual views, while maintaining clustering consistency across the different views is very difficult. Multi view clustering can be implemented using some of the following general multi view clustering paradigms

- *Co-training style algorithms*: This performs clustering on the individual views by considering clustering information obtained by clustering other views prior to it and using the corresponding knowledge to achieve consistent clustering across the different

views. Many foundations of Co-training style algorithms were provided in [4] by Bickel and Scheffer in 2004.

- *Multi-kernel learning:* Multi-kernel learning methods use predefined kernels obtained from the different views to obtain the clustering information. The final kernel used to obtain a separation of the data objects into clusters is created through linear or non-linear combination of the individual kernels. MKKM-MR [18] is an example of Multi kernel learning method.
- *Multi-view graph clustering:* These category of methods uses the affinity graph information from each of the individual views to create a fusion graph. The resultant clustering is produced by using some Graph-Cut algorithm like Spectral Clustering [25].
- *Multi-view subspace clustering:* In this category of methods, we find the underlying subspace structures in the multi-view data set, assuming that data objects lying in the same cluster lie on the same underlying low dimensional subspace. We assume that data in all the views share a common hidden feature representation, and represent the data in the individual views using the same representation. A widely used method for solving the multi view subspace clustering problem is Non-negative Matrix Factorization(NMF) [17]. A visual diagram explaining subspace clustering in multi view domain is given in Fig 2.1.
- *Multi-task multi-view clustering:* In these methods, multiple related tasks are performed together. A single view data is a task. The clustering performance is boosted in single view data by utilizing the relationships between these related tasks.

Multi view clustering has two major principles which are *complementary* and *consensus* principles.

Complementary principle states that we need to leverage the power of multiple views to describe a data object from accurately and exhaustively. An example is that for video data, the data objects may be represented with different feature sets like metadata feature set describing the content information in a frame and also various image feature sets like HOG,LBP etc. Now that we have established that we need multiple feature sets aka multiple views to better understand the data object, we move on to the *Consensus* principle.

Consensus principle states that in one view, if data objects x_1 and x_2 are assigned the same cluster, then in the remaining views, the same data points should not be assigned to different clusters. This maintains clustering consistency across the different views, thus minimizing the disagreement between different views.

Our method falls under the category of Multi-view subspace clustering methods. Details about our method have been provided in chapter 4 of this document.

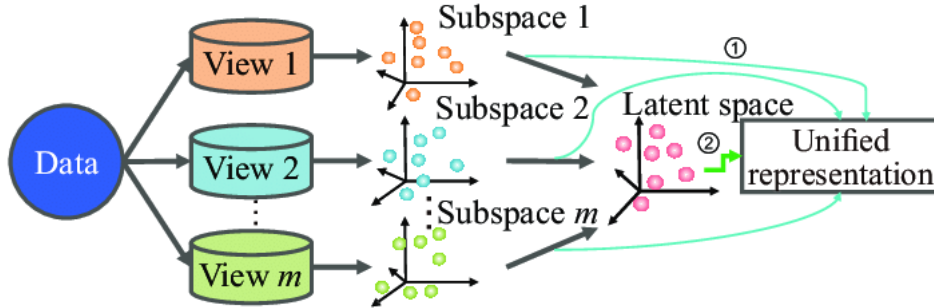


Figure 2.1: Multi view Subspace clustering general procedure

2.2 Sparse Subspace Clustering

High dimensional data like text data, image data, video data often lie close to low dimensional subspaces corresponding to various classes or clusters to which the data objects belong. Sparse subspace clustering can be used as a tool to find the low dimensional subspaces on which the data points lie. Mapping the high dimensional data into lower dimensional subspaces reduces the complexity as well as the memory requirement of the algorithm. Noises which appear in high dimensional data are also absent in the low dimensional representation, thus removing some related irregularities, which improve clustering performance.

Sparse subspace clustering has been studied in many papers like [9] and [10]. We find all the points lying on the same subspace, which helps us identify the clusters. Here, we take advantage of the *self-expressive property* [10] of the data set. The *self-expressive property* forms the foundation of sparse subspace clustering. According to this property, *every data point can be represented as a weighted combination of all the other points lying on a union of low dimensional subspaces*. Sparse representation of a point x_i is the d_l dimensional representation of the point x_i , where x_i is expressed as the weighted sum of d_l other points in the subspace S_l . S_l is the subspace on which the point x_i lies.

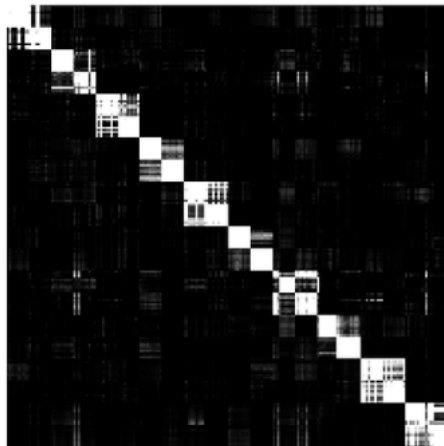


Figure 2.2: Example of a self-Expression Matrix

Mathematically, we can express this self-expressive property in the following fashion.

$$x_i = Xc_i, \quad c_{ii} = 0 \quad (2.1)$$

Here, X is the set of points which lie on a collection of low dimensional subspaces. The constraint $c_{ii} = 0$ eliminates the case where x_i can be expressed as a linear weighted combination of itself. X can be treated as a dictionary, where every point is expressed as a weighted combination of the points in the same subspace. Here, $c_i \triangleq [c_{i1}, c_{i2}, \dots, c_{iN}]^T$ represents the list of coefficients using which x_i can be represented as a weighted combination of X , N being the size of the data set.

In the sparse representation of a data point x_i , if data points x_j lies in a different subspace, $c_{ij} = 0$ and d_l data points which lie on the same subspace as x_i have non-zero coefficients. Now, for each data point x_i , we get a column vector c_i corresponding to the sparse coefficients of the data point. The matrix $C \in \mathbb{R}^{N \times N}$, which we obtain by concatenating all the columns corresponding to the sparse coefficients is known as the *self-expression matrix* as shown in Fig 2.2.

The self-expression matrix can be used to model the affinities between the data points in the form of a matrix like structure. Thus, the self-expression matrix is used as the affinity matrix. Treating the self-expression matrix as the affinity matrix has some issues. In general, the affinity matrix should be symmetric i.e the affinity between x_i and x_j should be same as the affinity between x_j and x_i . This is not the case in C . This is because even if x_j may be present in the sparse representation of x_i , x_i may not be present in the sparse representation of x_j . Thus, to make the affinity matrix symmetric, we use $W = \frac{|C|+|C|^T}{2}$ as the affinity matrix.

The self-expression matrix C is computed by solving the following optimization problem using l1 regularization as defined in [10]

$$\min_C \|X - XC\|_F^2 + \lambda \|C\|_1 \quad s.t. \quad C(i, i) = 0, C^T \cdot 1 = 1 \quad (2.2)$$

Here, $\|X - XC\|_F^2$ is the squared loss which we need to minimize since we want to make $X = XC$. The constraint that the sum of row values of C should be 1 denotes that data points may also lie on a collection of affine subspaces, not just linear subspaces. We have added another constraint $\|C\|_1$ to represent that we mean to minimize the l1-norm of the coefficient matrix C , which promotes sparsity in the coefficient values, thus leading to a sparse solution.

Having formed the similarity graph or the affinity graph, the vertices forming connected components in the graph belong to the same subspace while there are no edges between vertices representing data points from different subspaces. For n subspaces the affinity matrix should have the following *block-diagonal* form

$$W = \begin{bmatrix} C_1 & 0 & \dots & 0 \\ 0 & C_2 & \dots & 0 \\ & & \vdots & \\ 0 & 0 & \dots & C_n \end{bmatrix} \Gamma \quad (2.3)$$

The above block-diagonal form is also visible in Fig 2.2. Here, C_i represents the i th connected component of vertices and Γ represents the permutation matrix.

2.3 General Approach for Multi view subspace clustering

In multi view setting, we have a data set $X_v \in \mathbb{R}^{d_v \times N}$. X_v denotes the set of feature vectors in the v th view ($v = 1, 2, \dots, m$), m being the number of views and N being the number of data points. When we perform subspace clustering on each of the views individually, then we get self-expression matrix Z_v for each of the views, the non-zero elements in Z_v corresponding to points in the same subspace. In most of the solutions for multi view subspace clustering, clustering is done on the individual views, and then the results are unified together to create the consensus clustering self-expression matrix Z . To combine the multi view subspace clustering results, subspace learning on different views

can be performed simultaneously by minimizing the following objective function

$$\begin{aligned} \min_{Z_v, Z} \sum_v \|X_v - X_v Z_v\|_F^2 + \lambda \sum_v \|Z - Z_v\| \\ \text{s.t. } Z_v^T \mathbf{1} = 1, Z_v(i, i) = 0 \end{aligned} \quad (2.4)$$

Here, Z is the consensus self expression matrix or the unified subspace representation matrix. Although this is a viable solution, the data block structures as shown in 2.2 may be dramatically different, so it may not be easy to implement them as given in (2.4). Different strategies have been suggested by different papers to counter the given issue. In the paper [12], using a common indicator matrix F for all of their views has been suggested, to maintain the consistency across the clustering results in different views. They have minimized the following objective function

$$\begin{aligned} \min_{Z_v, F} \sum_v \|X_v - X_v Z_v\|_F^2 + \lambda \sum_v \text{Tr}(F^T (D_v - W_v) F) \\ \text{s.t. } Z_v^T \mathbf{1} = 1, Z_v(i, i) = 0, F^T F = I \end{aligned} \quad (2.5)$$

Here, $W_v = \frac{|Z_v| + |Z_v^T|}{2}$, and D_v represents the diagonal matrix for the v th view.

In the paper [35], instead of reconstructing the data points using original feature values, the aim is to uncover the underlying latent representation present in all the data points. The main idea is that the N data points from different views $\{X_1, X_2, \dots, X_m\}$ share the same latent representation $\{h_i\}_{i=1}^N$ in the latent space H . Since, a shared latent space H is present across all the views, there will be a single unified self-expression matrix Z obtained from the latent representation. We can construct the data points in the different views from the latent space H using models $\{M^{(1)}, M^{(2)}, \dots, M^{(m)}\}$, with the expression $x_i^{(v)} = M^{(v)} h_i + e_i^{(v)}$

The latent representation learning and subspace clustering are done collectively by solving the following optimization function

$$\min_{M, H, Z} L_h(X, MH) + \lambda_1 L_r(H, HZ) + \lambda_2 \Omega(Z) \quad (2.6)$$

Here, L_r represents the loss function associated with the data reconstruction, while L_h representing the loss function related to finding the latent space H that is shared by the data points in the different views. Here, $\lambda_1 > 0$ and $\lambda_2 > 0$ balance the three terms.

As we can see in the above examples, normally subspace clustering requires computation of a consensus self-expression matrix Z across all

the views by minimizing an objective function of the following general form

$$\begin{aligned} \min_{Z_v} \sum_v L(X_v, X_v Z_v) + \lambda \Omega(Z_v) \\ \text{s.t. } Z_v^T \mathbf{1} = 1 \quad Z_v(i, i) = 0 \end{aligned} \quad (2.7)$$

Z_v is the subspace representation structure in each of the views. L is the reconstruction loss and Ω is the regularization which is applied. Ω is normally the l-norm of the subspace representation structure in the views. Different norms lead to different representations of the same subspace clustering problem, constraining the elements of Z_v according to the task at hand.

Chapter 3

Related Work

3.1 Sparse Subspace clustering using Entropy Norm

This work was done in [3]. In this paper, an explicit connection has been provided between sparse subspace clustering and spectral clustering to reach the goal of finding a similarity or an affinity matrix between the data points.

Let us look at spectral clustering from a general perspective. Let us consider X to be a $m \times n$ data matrix, with m being the number of features and n being the number of data points. For spectral clustering, we need to solve the following optimization problem

$$\min_H \Theta = \text{Tr}(H^T L H) \text{ s.t. } H^T H = I \quad (3.1)$$

So, spectral clustering can be in general defined as a *trace minimization problem*. In this case, L is the laplacian matrix where it is defined as $L = D - W$. D being the degree matrix and W being the affinity matrix. The diagonal elements of D are the row sums of the matrix W . H is the $n \times k$ cluster indicator matrix.

In sparse subspace clustering, which has been spoken of in depth in section 2.2, to find the affinity matrix, in order to approximate for noisy representations, the optimization problem is defined as the following

$$\min_Z \|X - XZ\|_F^2 + \lambda \|Z\|_1 \text{ s.t. } \text{diag}(Z) = 0 \quad Z^T \cdot 1 = 1 \quad (3.2)$$

With the self-expression matrix Z , W is computed as $W = |Z| + |Z|^T$. W is symmetric and non negative in this scenario. Now, if we can show Z to be a symmetric and non negative matrix, W is equivalent to Z . Then, we need to solve the following minimization problem

$$\min_Z F \text{ s.t. } Z = Z^T, Z \geq 0, \text{diag}(Z) = 0 \quad (3.3)$$

Here, the objective function F is defined as in sparse subspace clustering of the form $F = L(X, XZ) + \lambda\Omega(Z)$, $\Omega(Z)$ being l1-norm of Z . We use the l1-norm constraint on Z , since we want to find the sparse solutions to the self-expressive representation problem. Entropy norm is selected as the regularization term for the objective function F , we can write the objective function as

$$F = L(X, XZ) + \lambda \sum_{i=1}^n \sum_{j=1}^n z_{ij} \ln z_{ij} \quad (3.4)$$

For the rest of this section, we consider n as the number of elements in the data set. We can use Lagrange multiplier to directly minimize F . After minimizing w.r.t z_{ij} , we obtain

$$z_{ij} = \exp(-1) \exp\left(-\frac{f_{ij}}{\lambda}\right) \quad (3.5)$$

Here, $f_{ij} = \frac{\delta L}{\delta z_{ij}}$. According to equation (3.5), we can see that Z is non negative, and Z is symmetric if $f_{ij} = f_{ji}$. Furthermore, we need to add the constraint $Z^T \cdot 1 = 1$ to the optimization problem, since we are also considering affine subspaces in the picture. We can again use Lagrange multiplier to minimize F to obtain the following solution for z_{ij}

$$z_{ij} = \frac{\exp\left(-\frac{f_{ij}}{\lambda}\right)}{\sum_{h \neq i}^n \exp\left(-\frac{f_{ih}}{\lambda}\right)} \quad (3.6)$$

In the above equation, even if $f_{ij} = f_{ji}$, it is not possible to guarantee that $z_{ij} = z_{ji}$, i.e. Z becomes symmetric. So, to make Z symmetric, the constraints are relaxed, the constraint for the affine subspace is replaced by the constraint $\sum_{h \neq i}^n z_{ih} + \sum_{h \neq j}^n z_{hj} = 2$ for $1 \leq i \neq j \leq n$. Keeping the sum of any row and any column as 2 relaxes the constraints. If we again use the Lagrange multiplier to solve for Z , we get

$$z_{ij} = \frac{2 \exp\left(-\frac{f_{ij}}{\lambda}\right)}{\sum_{h \neq i}^n \exp\left(-\frac{f_{ih}}{\lambda}\right) + \sum_{h \neq j}^n \exp\left(-\frac{f_{hj}}{\lambda}\right)} \quad (3.7)$$

If we use the Euclidean distance $d_{ij} = \|x_i - x_j\|^2$ as f_{ij} , then we can rewrite equation (3.7) as the following

$$z_{ij} = \frac{2 \exp\left(-\frac{\|x_i - x_j\|^2}{\lambda}\right)}{\sum_{h \neq i}^n \exp\left(-\frac{\|x_i - x_h\|^2}{\lambda}\right) + \sum_{h \neq j}^n \exp\left(-\frac{\|x_h - x_j\|^2}{\lambda}\right)} \quad (3.8)$$

Thus, compared to Sparse subspace clustering, Sparse subspace clustering with entropy norm(SSCE) can directly compute the similarity or the affinity matrix using the Gaussian kernel, thus reducing the computation cost. Also, SSCE can obtain a non-negative, symmetric affinity matrix which gives good results after spectral clustering is applied on it.

3.2 Subspace clustering via Good Neighbors

We consider in this section that N is the number of data points. We follow a similar post-processing procedure as given in [31]. This post processing technique is used to optimize both subspace preservation property and connectivity of the self representation system. Before we go further, we must be acquainted with the **Intra-subspace projection dominance**(IPD) [22] property of the self-expression matrix Z . This states that the coefficient between samples in the same subspace are greater than coefficients between samples in different subspaces i.e. $\forall x_p, x_q \in S$ and $x_k \notin S$, we have $z_{pq} \geq z_{pk}$.

IPD supports the subspace preservation property. The generated coefficient matrix must be such that both sparsity is increased to create more compact clusters and connectivity among intra cluster samples is increased. To create a sparser as well as well connected self-expression matrix, we must prune away the weaker connections, while preserving the stronger connections.

By the IPD property, we know that if x_i and x_j lie on the same subspace, their z_{ij} values will be higher compared to the case where x_i and x_j lie on different subspaces. Thus, a good way to prune out erroneous connections in the affinity graph is to eliminate the connections from x_i to the elements x_k which have low z_{ik} values. Eliminating such connections from the affinity matrix still maintains the subspace preservation property.

Thus, we can conserve the top η neighbors of x_i which are computed by maximizing

$$\operatorname{argmax}_{x_k} \sum_{k=1}^N |w_{ik}| \quad (3.9)$$

The concept of η -neighbors is similar to the ones discussed in the paper [22] [24]. w_{ij} is the element in the i th row and j th column of the affinity

matrix W which is computed using

$$W = \frac{|Z| + |Z|^T}{2} \quad (3.10)$$

The need to use W as the affinity matrix instead of Z , arises due to the fact that Z may not be symmetric i.e. even if x_i chooses x_j in its subspace representation, x_j may not necessarily choose x_i in its subspace representation [10]. Thus, to make the affinity matrix symmetric and Non-negative we use the above formulation of W .

This means that we take the η highest values from the i th row of W to find out the η neighbors of x_i . The caveat is that just preserving the η neighbors for each of the elements x_i may lead the algorithm to be easily manipulated by noise in the data, since wrong connections are maintained as maximum weighted edges are sensitive to both noise and outliers. Also, the elements in each cluster may not form a connected component. To improve connectivity, the concept of good neighbors is introduced.

Good Neighbors [31] are chosen among the η -neighbors of an element x_i . x_j will be a good neighbor of x_i , if x_i and x_j have μ common neighbors and x_j is an η -neighbor of x_i . In graph theoretical language, we can say that the path between x_i and x_j contains μ η -neighbors of x_i . Here, let us represent the η neighbors of x_i as $N_\eta(x_i)$.

Good Neighbors x_j is a good neighbor of x_i if there exists μ samples among the η neighbors of x_j , $\{x_{jk}\}_{k=1}^\mu \subset N_\eta(x_j)$ that satisfy

$$\prod_{l=1}^\mu 1_{x_i \in N_\eta(x_{jl})} = 1 \quad (3.11)$$

In our case, since we want to preserve the strongest connections, we have kept the value of $\mu = 1$ in all of our experiments. μ controls the connectivity of the affinity graph, whereas γ controls the sparsity of the affinity graph. More hidden connections between x_i and its good neighbors improves the connectivity in each cluster, while restricting the number of good neighbors to γ improves the sparsity of the graph, thus improving the clustering results.

For every sample x_i , we compute the good neighbors of x_i , the number of good neighbors that we consider for each element being γ , N being the number of elements and store all the good neighbors in a

good neighbor matrix $\mathcal{N} \in \mathbb{R}^{N \times \gamma}$. Each row in the good neighbor matrix corresponds to the γ good neighbors of an element. Let us take a look at the algorithm that we use for calculating the Good Neighbor matrix \mathcal{N} from the self-expression matrix Z .

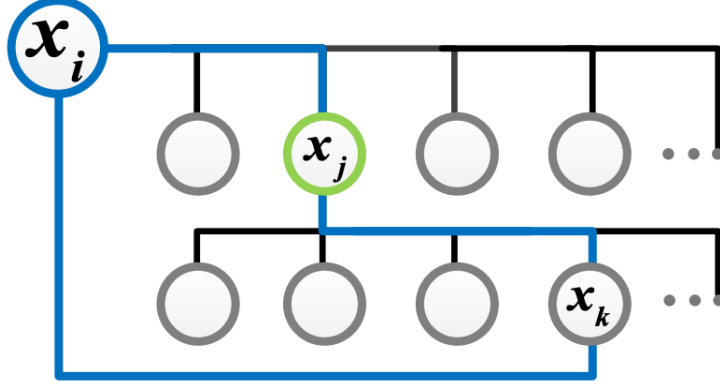


Figure 3.1: Good Neighbors example

In the figure 3.1, x_j is a good neighbor of x_i assuming that the value of μ is 1. Samples with gray circle are η neighbors of the corresponding sample. The blue line shows that the elements x_i, x_j and x_k lie on the same path. The lines represent the connections in the affinity graph.

Algorithm 1: Good Neighbors Algorithm

Input: $Z = [z_1, z_2, \dots, z_N] \in \mathbb{R}^{N \times N}$, γ, η, μ

- 1 The affinity matrix W is computed using (3.10);
- 2 Initialise $\mathcal{N} \in \mathbb{R}^{N \times \gamma}$ with 0's;
- 3 **for** $i=1$ to N **do**
- 4 η neighbors of x_i , $N_\eta(x_i) = \{x_{j_k}\}_{k=1}^\eta$ are computed by (3.9);
- 5 $num = 0$;
- 6 **for** $k=1$ to η **do**
- 7 Compute s_{ik} for $x_{i_k} \in N_\eta(x_i)$ by (3.12);
- 8 **if** $s_{ik} \geq \mu$ and $num < \gamma$ **then**
- 9 $\mathcal{N}_i = \mathcal{N}_i \cup x_{i_k}$;
- 10 $num = num + 1$;
- 11 **if** $num < \gamma$ **then**
- 12 $\mathcal{N}_i = \mathcal{N}_i \cup \{\bar{x}\}_{\gamma-num}$ where $\bar{x}_{\gamma-num}$ consists of elements having the largest $\gamma - num$ similarity in $N_\eta(x_i)$

Output: Good Neighbor Matrix \mathcal{N}

Since, finding out whether x_j is a good neighbor of x_i according to (3.11) is an NP-Hard problem, we find an easier way of traversing the η neighbors of x_i to find the good neighbors. As we have earlier pointed out, for our experiments we set the value of $\mu = 1$, which means that the path connecting x_i and one of its good neighbors must have a different neighbor of x_i in between. The connection score between x_i and x_j is computed as follows

$$s_{ij} = \sum_{m=1}^{\eta} 1_{x_i \in N_{\eta}(x_{jm})} \quad (3.12)$$

The above equation signifies that the similarity score s_{ij} between x_i and x_j , given that $x_j \in N_{\eta}(x_i)$, is calculated by traversing the η -neighbors of x_j and checking how many elements among the η neighbors have x_i as their η -neighbor. If $s_{ij} \geq \mu$, then x_j is a good neighbor of x_i .

After computing the good neighbor matrix \mathcal{N} , we construct Z^* from the affinity matrix W using the following transformation

$$z_{ij}^* = \begin{cases} \frac{w_{ij}}{(\sum_{l=1}^{\gamma} w_{il})}, & \text{for } x_j \in \mathcal{N}_i \\ 0 & \text{for } x_j \notin \mathcal{N}_i \end{cases} \quad (3.13)$$

The numerator consists of the affinity between x_i and x_j , and the denominator consists of the sum of all the affinities from x_i to all its good neighbors. We introduce normalization to make sure that the stronger connections do not affect the clustering results. After constructing Z^* , we construct the affinity matrix W^* using the equation

$$W^* = \frac{|Z^*| + |Z^*|^T}{2} \quad (3.14)$$

After, we get the affinity matrix W^* is calculated, the final clustering is obtained by applying Spectral Clustering [25] on the Laplacian Matrix.

Chapter 4

The proposed Multi View Good Neighbor Subspace Clustering Algorithm

4.1 Entropy Norm Formulation

If we use sparse subspace clustering to compute the affinity matrix W , it takes $O(N^3)$ time. Thus, sparse subspace clustering, being computationally expensive, is not suitable for large scale data.

Sparse subspace clustering with Entropy Norm(SSC-E)[3] is same as spectral clustering using a Gaussian Kernel, if our objective is to learn the affinity matrix. SSC-E produces a closed form solution for computing the affinity matrix using Gaussian Kernel, reducing the computational cost of sparse subspace clustering and producing a non-negative, symmetric representation of the data set.

The self-expression matrix Z obtained from SSC-E forms a lower bound on the normalized Gaussian Kernel

$$Z \leq D^{-\frac{1}{2}}WD^{-\frac{1}{2}} \quad (4.1)$$

We calculate Z from the Gaussian Kernel using the following formulation

$$z_{ij} = \frac{2 e^{-\frac{\|x_i - x_j\|^2}{\lambda}}}{\sum_{h \neq i}^N e^{-\frac{\|x_i - x_h\|^2}{\lambda}} + \sum_{h \neq j}^N e^{-\frac{\|x_j - x_h\|^2}{\lambda}}} \quad (4.2)$$

The Z obtained hence can be treated as the data similarity matrix. Let us look at the algorithm for computing the self-expression matrices from the set of data points $X \in \mathbb{R}^{D \times N}$, D being the intrinsic dimension of the data points, and N being the number of data points.

Algorithm 2: Entropy Norm Algorithm

Input: $X = [x_1, x_2, \dots, x_N] \in \mathbb{R}^{D \times N}$, λ , N

- 1 Initialise $S \in \mathbb{R}^{N \times N}$ with zeros ;
- 2 **for** $i=1$ to N **do**
- 3 **for** $j=1$ to N **do**
- 4 **if** $i=j$ **then**
- 5 $S[i,j]=0$;
- 6 **else**
- 7 $S[i,j] = e^{-\frac{\|x_i-x_j\|^2}{\lambda}}$
- 8 Initialise $Z \in \mathbb{R}^{N \times N}$ with zeros ;
- 9 **for** $i=1$ to N **do**
- 10 **for** $j=1$ to N **do**
- 11 **if** $i=j$ **then**
- 12 $Z[i,j]=0$;
- 13 **else**
- 14 $Z[i,j] = \frac{2S[i,j]}{\sum_i S[i,:] + \sum_j S[j,:]}$

Output: Self Expression Matrix Z

In the above algorithm, we first store the values of the exponentials in the two dimensional matrix S . Then, using the stored values, we compute the self-expression matrix Z . This computes the self-expression matrices much faster, instead of using (3.6) directly to compute the elements of Z .

4.2 Good Neighbors post processing

We perform good neighbors post processing on the self-expression matrices Z_i , which we obtained by Entropy norm formulation using the concepts outlined in the section 4.1 and Algorithm 2. One difference between the post processing technique given in the paper [31] and our method is that, in the paper, they used SMR [13] to create the self-expression matrices, as opposed to Entropy Norm in our paper. The reason of using Entropy Norm instead of SMR was to take advantage of the reduced computational complexity of Entropy Norm.

In our algorithm, we use the good neighbors post processing, outlined in Algorithm 1, to sparsify the already sparse self-expression matrix obtained from the entropy norm formulation. The quality of

the clustering results obtained from Spectral clustering directly depends on the quality of the self-expression matrices. Good neighbors post processing technique improves sparsity, while retaining the strongest connections, hence preserving the subspaces inherent in the data set. The sparse self-expression matrix obtained after the good neighbors post processing thus improves connectivity inside the clusters and also minimizes the connection between clusters, creating a block-like self-expression matrix as given in Fig 2.2.

4.3 The Algorithm

In our algorithm, we leverage the use of multiple views to get better results in standard data sets. Multi view clustering takes advantage of the complementary principle to create a better representation of the data points, by observing them from different perspectives.

For our proposed algorithm, we have as input, multiple views of the same data set. For each of the views in the data set, we create the self-expression matrices Z_v using Entropy Norm [3] formulation, since we get faster generation of the self-expression matrices. In multi view setting, we have m different views for N different multi view data objects. The multi view data set can be represented as X where X is written as

$$X = [X_1, X_2, \dots, X_m], \text{ where } X_i \in \mathbb{R}^{D_i \times N} \quad (4.3)$$

Here, D_i is the dimension of the i th view. After we have generated Z_i from each of the views X_i , $i \in [1, 2, \dots, m]$ using [3], we find a consensus self-expression matrix Z from the views using simple averaging technique i.e $Z = \left(\sum_{i=1}^N Z_i\right) / N$. The consensus self-expression matrix contains clustering information from all the m views. Then we apply a post-processing technique on Z as given in [31], to produce a sparser and more connected self-expression matrix.

The increase in sparsity tends to prune out some of the weaker connections from the affinity matrix and the increase in connectivity helps in reducing the intra-cluster variance, thus making the clusters more compact in nature.

Finally, after we get the sparser self-expression matrix Z^* , we find out the affinity matrix using the formula $W^* = \frac{Z^* + Z^{*T}}{2}$. Then, we apply normalized spectral clustering technique [25] on the affinity matrix W^* to obtain the inherent clusters present in the data. The algorithm which we propose is given as follows:

Algorithm 3: MVGNCS

- Input:** $X = [X_1, X_2, \dots, X_m]$, $X_i \in \mathbb{R}^{D_i \times N}$, $\lambda, \eta, \mu, \gamma, K$,
- 1 **for** $i=1$ to m **do**
 - 2 | Calculate Z_i from X_i using Entropy Norm formulation;
 - 3 Calculate the consensus matrix $Z = \frac{\sum_{i=1}^m Z_i}{m}$;
 - 4 Calculate the sparser self-expression matrix Z^* from Z using [31];
 - 5 Calculate the affinity matrix $W^* = \frac{Z^* + Z^{*T}}{2}$ and compute segmentation from W^* by spectral clustering;
- Output:** Labels of samples $\mathcal{L} \in \mathbb{R}^N$
-

Chapter 5

Results

5.1 Complexity Analysis

For the first part of our MVGNCS algorithm, we need to compute the individual self-expression matrices Z_i for each of the m views. Computing the self-expression matrix from each view takes $O(N^2)$ time, where N is the number of elements present in the data set. This is because the time complexity for obtaining the self-expression matrix through the closed form solution obtained using entropy norm is $O(N^2)$. Since there are m views, the total time taken to generate all the self-expression matrices is $O(mN^2)$. Calculating the consensus matrix takes time equal to $O(m)$, since we need to find the sum of all the individual self-expression matrices before dividing by m .

After this, we find the sparser consensus self-expression matrix from consensus Z using good neighbors post processing. To compute the time complexity of using good neighbors post processing, let's look at Algorithm 1. In the first step, computing the affinity matrix W takes $O(1)$ i.e. constant time. Next, we compute the η neighbors of all the elements $\{x_i\}_{i=1}^N$. Computing the η neighbor of each element takes $O(N)$ time, since we need to make a pass through all the elements in the i th row of W to find the η largest elements. After we have obtained the η neighbors, we need to loop over all the η neighbors to compute the similarity scores with its η neighbors. Finding out the good neighbors by computing the similarity scores takes $O(\eta)$ time. Since, we compute the good neighbors among all the η neighbors, the time complexity for computing the good neighbors among the η neighbors for a single element x_i is $O(\eta^2)$.

As we previously observed, computing the η neighbors takes $O(N)$ time and computing the similarity scores for all the elements in the η

neighbors takes $O(\eta^2)$ time. This is multiplicative, thus computing the good neighbors for each element x_i takes $O(N\eta^2)$ time. Since, we find the good neighbors for all the elements in the data set $\{x_i\}_{i=1}^N$, the total time required to find the good neighbor matrix is $O(N^2\eta^2)$.

After we have the Good Neighbor matrix, to calculate the sparser self expression matrix, we need an additional $O(\gamma N^2)$ time. Since, we normalize the weights for each good neighbor of x_i and make all the other weights 0. Calculating the sparser affinity matrix is a constant time operation. From the sparser affinity matrix, spectral clustering takes $O(N^3)$ time.

The total time complexity for the MVGNCS algorithm is thus $O(mN^2 + m + N^2\eta^2 + \gamma N^2 + N^3)$, which is effectively $O(N^3)$.

5.2 Setup

Configuration. For our experiments, we have used a machine with 128 GB RAM, an AMD Ryzen Threadripper 3960x processor and a RTX 3090 GPU.

Evaluated Methods. We compare the proposed MVGNSC algorithm with state of the art Multi view graph based subspace clustering methods like AMGL[20],MLRSSC[6], LMVSC[14], MSCIAS[27].The parameters are tuned according to instructions given in the corresponding papers to produce the best results.

Datasets. We have performed our experiments on four benchmark data sets: Handwritten, Caltech-101-7, Caltech-101-20, Reuters. The descriptions of the data sets are given in the table below

Table 5.1: Description of the data sets. Feature dimensions are provided inside parentheses

| View | Handwritten | Caltech7/20 | Reuters |
|-----------|---------------------------|---------------------|----------------|
| 1 | Profile Correlations(216) | Gabor(48) | English(21531) |
| 2 | Fourier Coefficients(76) | Wavelet Moments(40) | French(24892) |
| 3 | Karhunen Coefficients(64) | CENTRIST(254) | German(34251) |
| 4 | Morphological(6) | HOG(1984) | Italian(15506) |
| 5 | Pixel Averages(240) | GIST(512) | Spanish(11547) |
| 6 | Zernike Moments(47) | LBP(928) | - |
| Data size | 2000 | 1474/2386 | 18758 |
| Classes | 10 | 7/20 | 6 |

Handwritten Digits data set [8] consists of a collection of handwritten digits of 0 to 9. Two subsets consisting of 7 classes (Caltech-7) and 20 (Caltech-20) classes are chosen for experimentation from the Caltech-101 data set [11]. Reuters [1] data set consists of documents written in English, French, German, Italian and Spanish, 5 different languages. Here, a subset of the entire data set consisting of documents written in English and their translated counterparts are used.

Metrics. For validating our results, we have used 3 commonly used metrics. Accuracy (ACC), Normalized Mutual Information(NMI) and Purity.

5.3 Experimental results

5.3.1 Parameter Settings

Our Algorithm has five parameters $\lambda, \eta, \gamma, \mu$ and K . Among this, as discussed before, we set the value of $\mu = 1$, since the connection between samples decreases in strength as the path length increases. Keeping the value of μ as 1 signifies that there is 1 common neighbor between x_i and x_j or similarly, according to the similarity score s_{ij} we have previously defined, $s_{ij} \geq 1$. The parameter γ controls the sparsity in the resultant affinity graph. In our experiments, keeping the ratio of the value of γ and η to be 2:5, we get identical results. Thus, as suggested by [31], set the value of $\gamma = 8$ and the value of $\eta = 20$.

K represents the number of classes in each of the data sets, which was fixed in each of the data sets. Setting the value of λ in the entropy norm formulation turned out to be a bit tricky. In our experiments, we are setting the same value of λ for each of the views for retrieving the self-expression matrices of each view. First thing of note was that up to a certain point, decreasing the value of λ resulted in improved performance. After that point was reached, the performance of the algorithm started to deteriorate, i.e. the accuracy started to decrease. For our experiments, we found the best performance i.e. the point at which the performance is maximized, by finding the covariance of the data in each of the views, and then choosing a δ value equal to the median value of the co variances from each of the views. Then, we experimented with various values of λ which are selected from the set $\{2\delta, 1.5\delta, \delta, \frac{\delta}{10}, \frac{\delta}{20}, \frac{\delta}{50}\}$, δ being selected as above and chose the value which gave the best results. The results are shown in the figures 5.5, 5.6, 5.7 for the Caltech-7 and the Handwritten Digits data sets.

5.3.2 On multi-view data sets

Table 5.2: Performance evaluation for Caltech-7 data set

| Method | ACC | NMI | Purity |
|--------|--------------|-------------|-------------|
| AMGL | 0.451 | 0.42 | 0.75 |
| MLRSSC | 0.37 | 0.211 | 0.41 |
| MSCIAS | 0.38 | 0.23 | 0.442 |
| LMVSC | 0.726 | 0.51 | 0.75 |
| MVGNSC | 0.762 | 0.56 | 0.84 |

Table 5.3: Performance evaluation for Caltech-20 data set

| Method | ACC | NMI | Purity |
|--------|-------|-------------|-------------|
| AMGL | 0.301 | 0.40 | 0.31 |
| MLRSSC | 0.28 | 0.26 | 0.30 |
| MSCIAS | 0.31 | 0.31 | 0.33 |
| LMVSC | 0.53 | 0.52 | 0.58 |
| MVGNSC | 0.478 | 0.55 | 0.65 |

Table 5.4: Performance evaluation for HandWritten Digits data set

| Method | ACC | NMI | Purity |
|--------|--------------|-------------|--------------|
| AMGL | 0.84 | 0.87 | 0.87 |
| MLRSSC | 0.76 | 0.74 | 0.87 |
| MSCIAS | 0.80 | 0.77 | 0.86 |
| LMVSC | 0.916 | 0.84 | 0.916 |
| MVGNSC | 0.948 | 0.89 | 0.948 |

Table 5.5: Performance evaluation for Reuters data set

| Method | ACC | NMI | Purity |
|--------|-------------|-------------|-------------|
| AMGL | 0.167 | – | – |
| MLRSSC | 0.45 | 0.22 | 0.55 |
| MSCIAS | 0.49 | 0.27 | 0.60 |
| LMVSC | 0.589 | 0.334 | 0.614 |
| MVGNSC | 0.60 | 0.45 | 0.65 |

The value of η and γ has been set as 20 and 8 respectively, in all our experiments. Setting these values gives desirable results, so, we do not need to tweak these values further. For the Caltech-7 data set, the value of λ is set as 50. For the Caltech-20 data set, the value of λ has been set as 0.69, while for the Handwritten digits data set, we set the value of λ as 10. For the Reuters data set, the value of λ was set to 0.29.

The figures 5.1 and 5.2 refer to the original clustering results in the Caltech-7 data set and the obtained clustering result on the same data set by using our method. The figures were obtained using t-SNE technique. Figure 5.3 represents the clustering results obtained by our method on the handwritten digits data set.

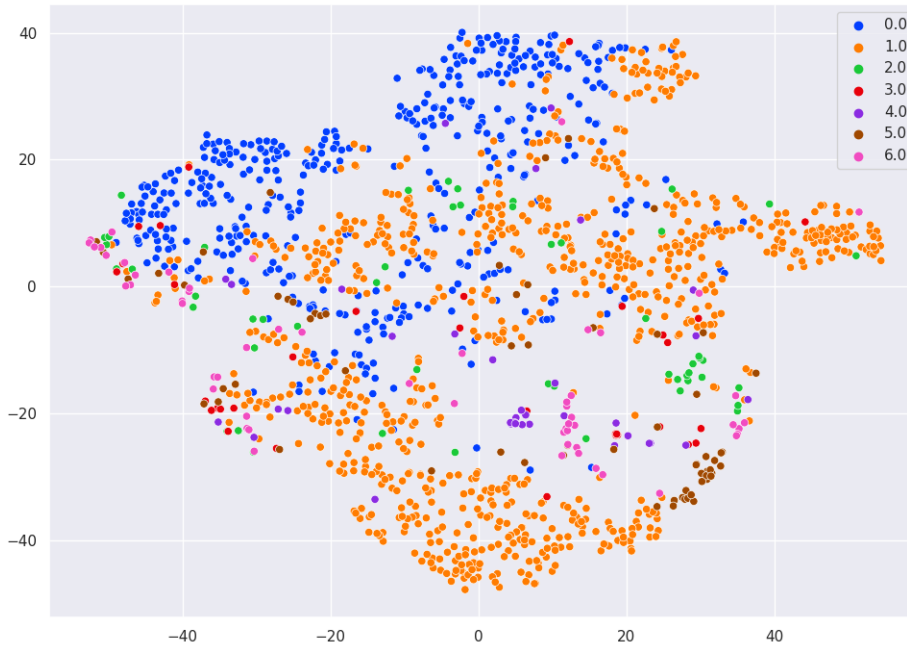


Figure 5.1: Caltech-7 Original Labels

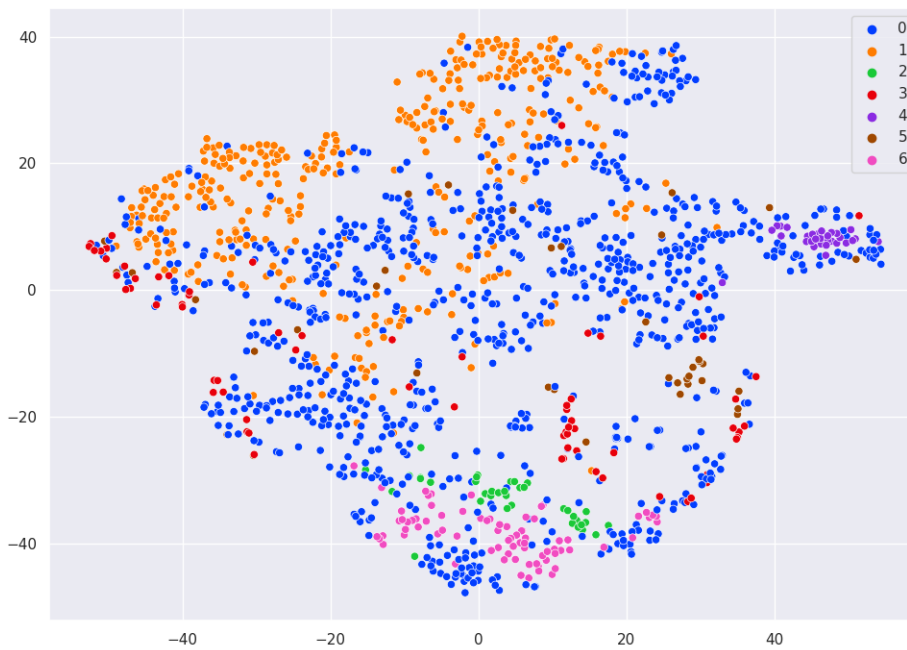


Figure 5.2: Caltech-7 Predicted Labels

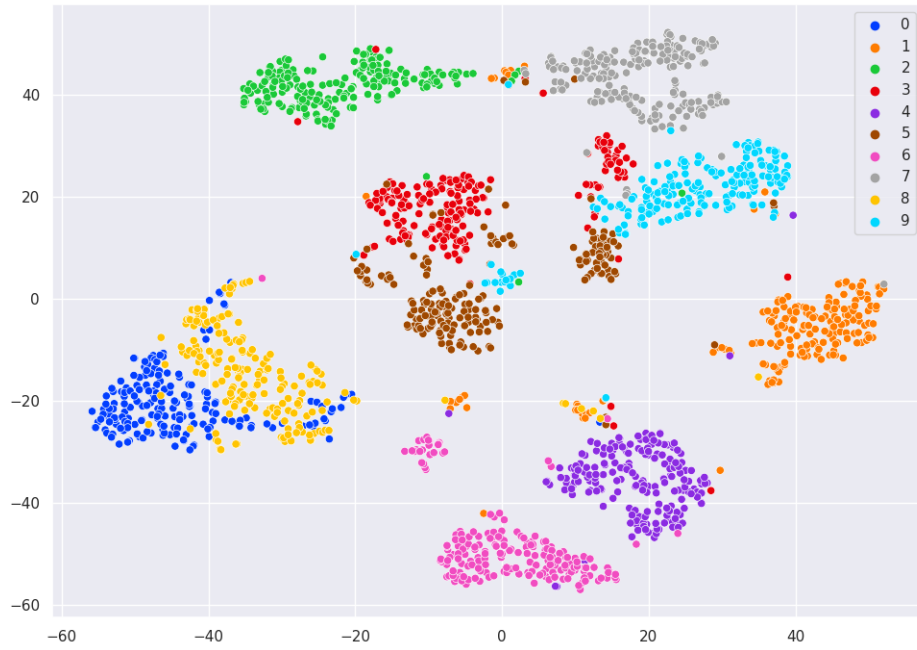


Figure 5.3: Handwritten Original Labels

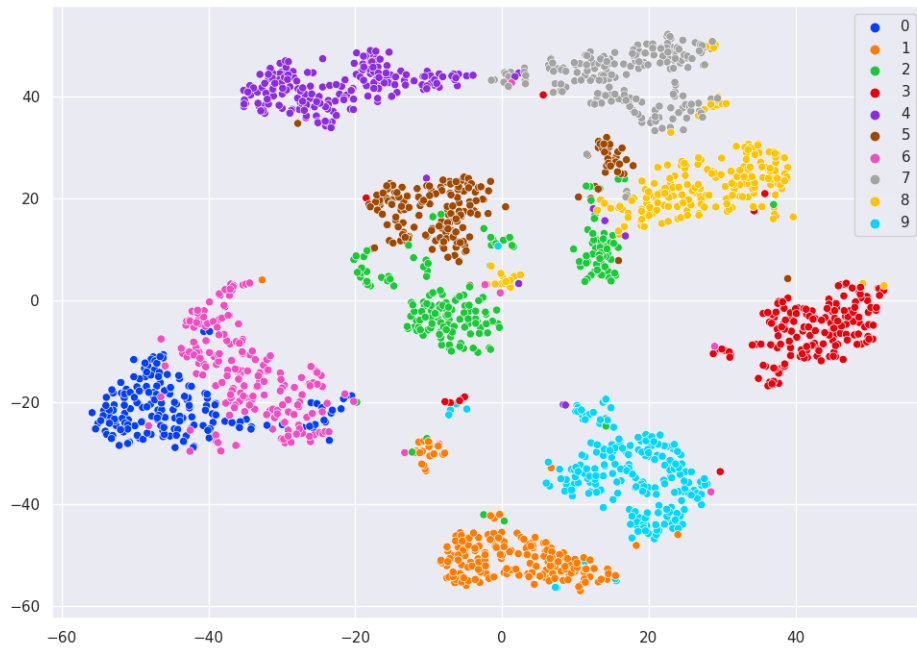


Figure 5.4: Handwritten Predicted Labels

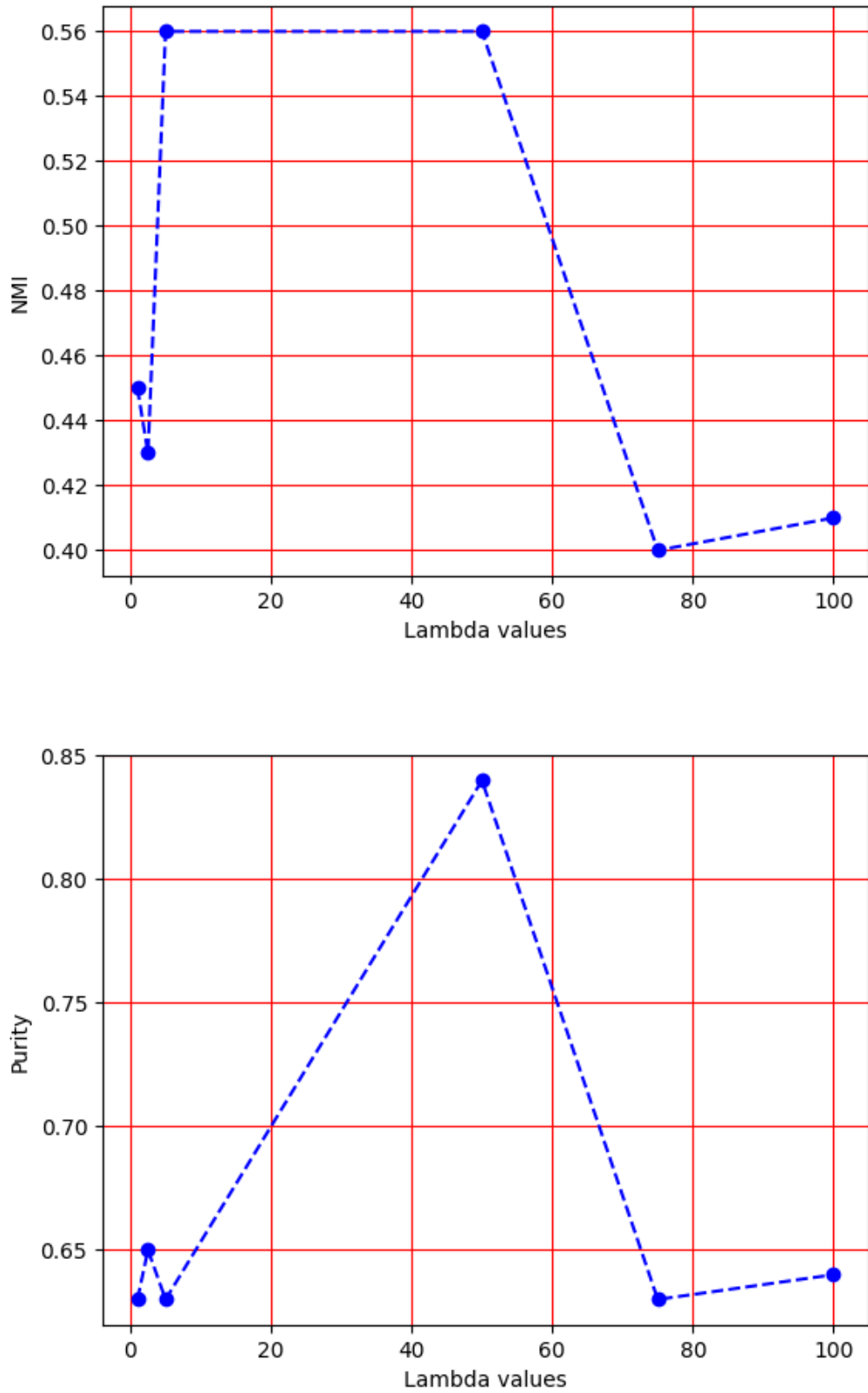


Figure 5.5: NMI and Purity for Caltech-7 data set

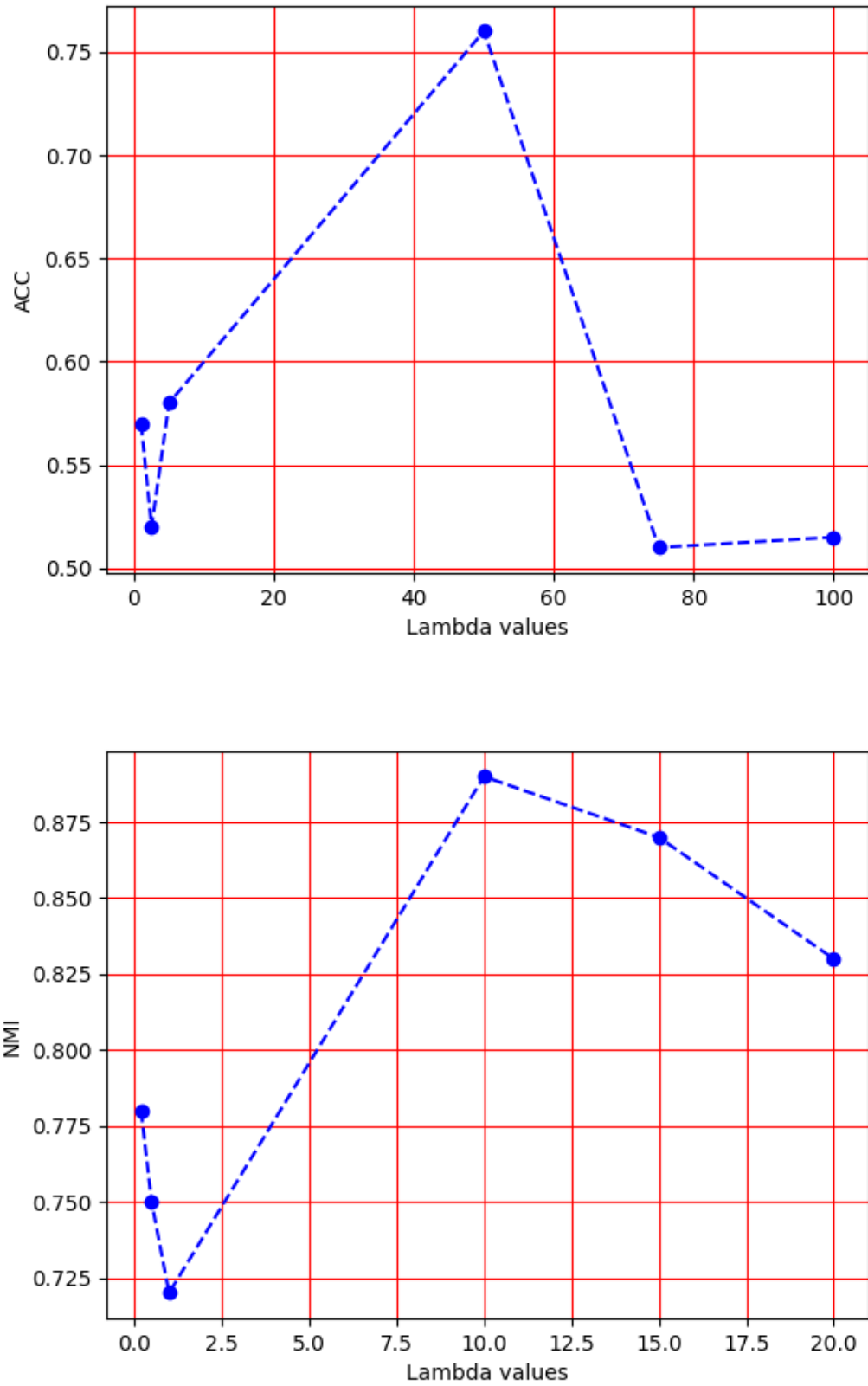


Figure 5.6: ACC for Caltech-7 and NMI for Handwritten digits data set

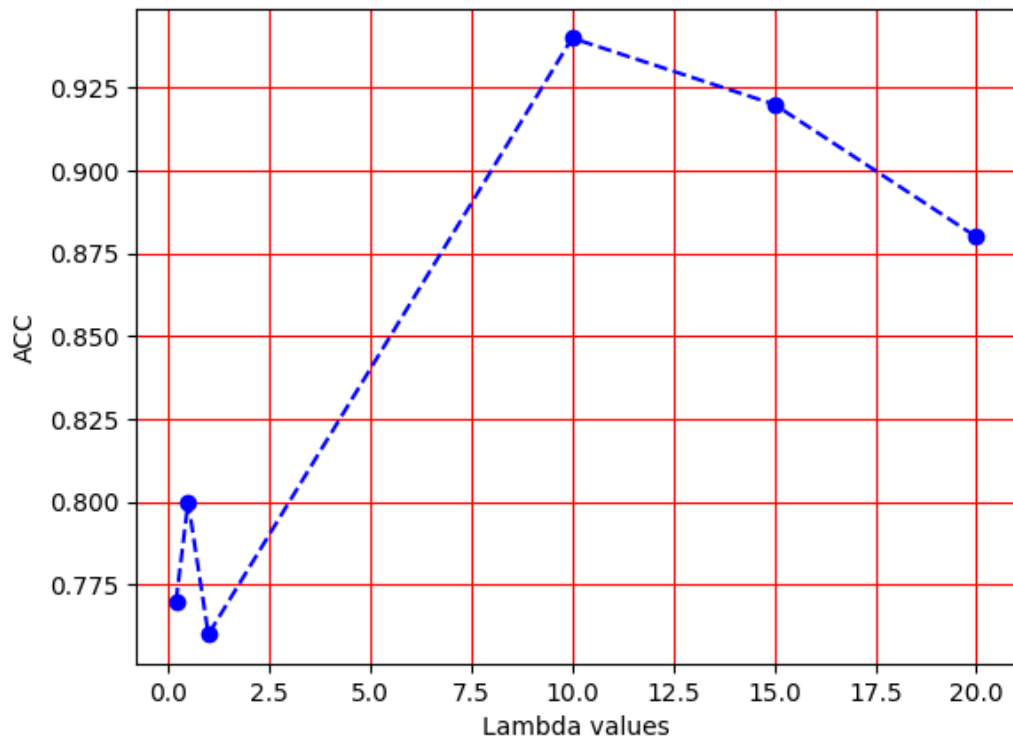
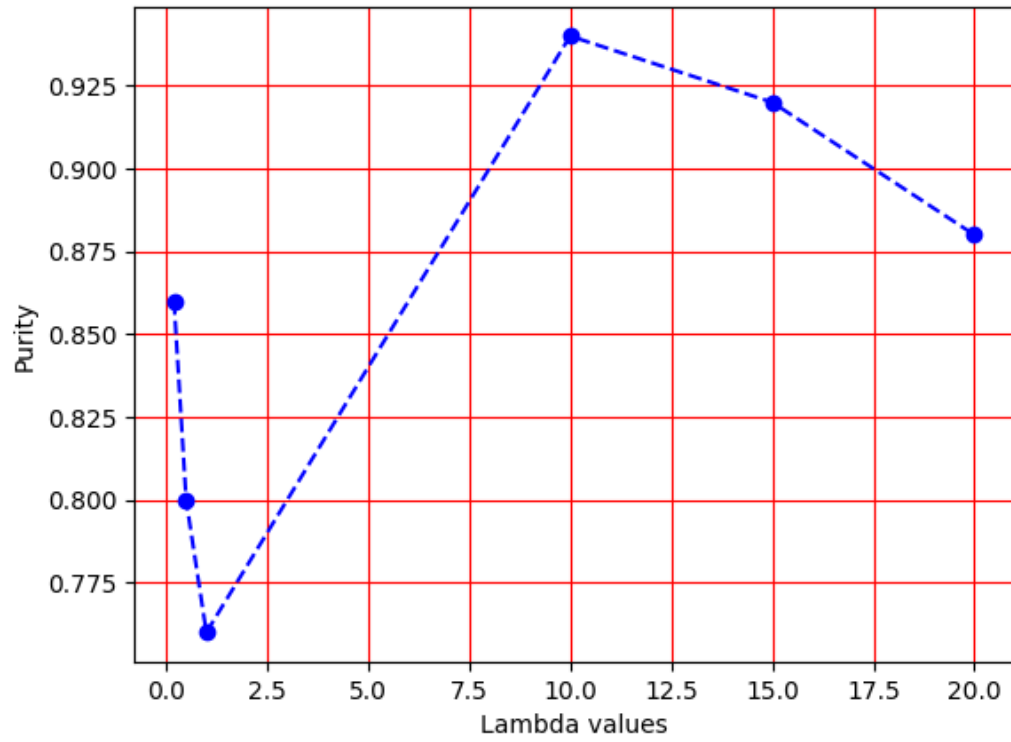


Figure 5.7: Purity and ACC for Handwritten digits data set

Chapter 6

Conclusion and Future Work

6.1 Conclusion

Clustering multi view data is very challenging, since to get good clustering results, we need to maintain both *connectivity* and *sparsity* of the affinity matrix. Most multi view subspace clustering methods deal with optimizing one of them, thus sacrificing the other. In our work, we showed that we can optimize both these properties to improve the quality of our clusters. Our good neighbors algorithm is based on Yang's [31] work.

We also took advantage of the *quadratic* run time of the Entropy Norm algorithm to obtain our self-expression matrix using Gaussian approximation, instead of having to solve an optimization function using solvers like *quadprog*, which are known to be very time consuming.

We also showed that we can leverage the advantage of multiple views to get noteworthy clustering results by comparing to many state-of-the-art algorithms in Multi view subspace clustering, thus showing that a sparse, yet strongly connected affinity matrix can contribute to the improvement in clustering results obtained using Spectral clustering as shown in the section 5.3.

6.2 Future Work

Despite its good performance, our method MVGNCS has its limitations and there is scope for improvement. One of the limitations being that it considers each view has equal contribution to the consensus matrix, since we take a simple average across all the views to get the consensus matrix. This might not be the case, as some of the views may be more noisy and more prone to errors when compared

to other views. We can eliminate this issue by giving different weights to each of the self-expression matrices of the different views and then creating the consensus self expression matrix by considering a linear combination of the individual self-expression matrices. The weights may be learned using optimization. This will significantly improve the clustering results by giving less weights to noisy views and taking advantage of well distributed views.

Bibliography

- [1] Massih-Reza Amini, Nicolas Usunier, and Cyril Goutte. Learning from multiple partially observed views - an application to multilingual text categorization. In *Advances in Neural Information Processing Systems 22 (NIPS 2009)*, pages 28–36, 2009.
- [2] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. Technical report, Stanford, 2006.
- [3] Liang Bai and Jiye Liang. Sparse subspace clustering with entropy-norm. In *International Conference on Machine Learning*, pages 561–568. PMLR, 2020.
- [4] Steffen Bickel and Tobias Scheffer. Multi-view clustering. In *ICDM*, volume 4, pages 19–26. Citeseer, 2004.
- [5] Paul S Bradley and Olvi L Mangasarian. K-plane clustering. *Journal of Global Optimization*, 16(1):23–32, 2000.
- [6] Maria Brbić and Ivica Kopriva. Multi-view low-rank sparse subspace clustering. *Pattern Recognition*, 73:247–258, 2018.
- [7] Man-Sheng Chen, Ling Huang, Chang-Dong Wang, Dong Huang, and S Yu Philip. Multiview subspace clustering with grouping effect. *IEEE Transactions on Cybernetics*, 2020.
- [8] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.
- [9] Ehsan Elhamifar and Rene Vidal. Sparse subspace clustering. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2790–2797, 2009.
- [10] Ehsan Elhamifar and René Vidal. Sparse subspace clustering: Algorithm, theory, and applications. *IEEE transactions on pattern analysis and machine intelligence*, 35(11):2765–2781, 2013.

- [11] Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *2004 conference on computer vision and pattern recognition workshop*, pages 178–178. IEEE, 2004.
- [12] Hongchang Gao, Feiping Nie, Xuelong Li, and Heng Huang. Multi-view subspace clustering. In *Proceedings of the IEEE international conference on computer vision*, pages 4238–4246, 2015.
- [13] Han Hu, Zhouchen Lin Jianjiang Feng, and Jie Zhou. Smooth representation clustering. In *CVPR*, 2014.
- [14] Zhao Kang, Wangtao Zhou, Zhitong Zhao, Junming Shao, Meng Han, and Zenglin Xu. Large-scale multi-view subspace clustering in linear time. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4412–4419, 2020.
- [15] Leonard Kaufman and Peter J Rousseeuw. Partitioning around medoids (program pam). *Finding groups in data: an introduction to cluster analysis*, 344:68–125, 1990.
- [16] Aparajita Khan and Pradipta Maji. Multi-manifold optimization for multi-view subspace clustering. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.
- [17] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.
- [18] Xinwang Liu, Yong Dou, Jianping Yin, Lei Wang, and En Zhu. Multiple kernel k-means clustering with matrix-induced regularization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.
- [19] Juncheng Lv, Zhao Kang, Boyu Wang, Luping Ji, and Zenglin Xu. Multi-view subspace clustering via partition fusion. *Information Sciences*, 560:410–423, 2021.
- [20] Feiping Nie, Jing Li, Xuelong Li, et al. Parameter-free auto-weighted multiple graph learning: a framework for multiview clustering and semi-supervised classification. In *IJCAI*, pages 1881–1887, 2016.

-
- [21] Xi Peng, Zhang Yi, and Huajin Tang. Robust subspace clustering via thresholding ridge regression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.
- [22] Xi Peng, Zhiding Yu, Zhang Yi, and Huajin Tang. Constructing the l2-graph for robust subspace learning and subspace clustering. *IEEE transactions on cybernetics*, 47(4):1053–1066, 2016.
- [23] Xi Peng, Lei Zhang, and Zhang Yi. Scalable sparse subspace clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 430–437, 2013.
- [24] Mostafa Rahmani and George Atia. Innovation pursuit: A new approach to the subspace clustering problem. In *International conference on machine learning*, pages 2874–2882. PMLR, 2017.
- [25] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.
- [26] Rene Vidal, Yi Ma, and Shankar Sastry. Generalized principal component analysis (gpca). *IEEE transactions on pattern analysis and machine intelligence*, 27(12):1945–1959, 2005.
- [27] Xiaobo Wang, Zhen Lei, Xiaojie Guo, Changqing Zhang, Hailin Shi, and Stan Z Li. Multi-view subspace clustering with intactness-aware similarity. *Pattern Recognition*, 88:50–63, 2019.
- [28] Deyan Xie, Wei Xia, Qianqian Wang, Quanxue Gao, and Song Xiao. Multi-view clustering by joint manifold learning and tensor nuclear norm. *Neurocomputing*, 380:105–114, 2020.
- [29] Allen Y Yang, Shankar R Rao, and Yi Ma. Robust statistical estimation and segmentation of multiple subspaces. In *2006 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'06)*, pages 99–99. IEEE, 2006.
- [30] Congyuan Yang, Daniel Robinson, and Rene Vidal. Sparse subspace clustering with missing entries. In *International Conference on Machine Learning*, pages 2463–2472. PMLR, 2015.
- [31] Jufeng Yang, Jie Liang, Kai Wang, Paul L Rosin, and Ming-Hsuan Yang. Subspace clustering via good neighbors. *IEEE transactions*

- on pattern analysis and machine intelligence*, 42(6):1537–1544, 2019.
- [32] Yan Yang and Hao Wang. Multi-view clustering: A survey. *Big Data Mining and Analytics*, 1(2):83–107, 2018.
- [33] Ming Yin, Yi Guo, Junbin Gao, Zhaoshui He, and Shengli Xie. Kernel sparse subspace clustering on symmetric positive definite manifolds. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5157–5164, 2016.
- [34] Cong-Zhe You, Hong-Hui Fan, and Zhen-Qiu Shu. Non-negative sparse laplacian regularized latent multi-view subspace clustering. In *2020 19th International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES)*, pages 210–213. IEEE, 2020.
- [35] Changqing Zhang, Qinghua Hu, Huazhu Fu, Pengfei Zhu, and Xiaochun Cao. Latent multi-view subspace clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4279–4287, 2017.
- [36] Pei Zhang, Xinwang Liu, Jian Xiong, Sihang Zhou, Wentao Zhao, En Zhu, and Zhiping Cai. Consensus one-step multi-view subspace clustering. *IEEE Transactions on Knowledge and Data Engineering*, 2020.
- [37] Guo Zhong and Chi-Man Pun. Subspace clustering by simultaneously feature selection and similarity learning. *Knowledge-Based Systems*, 193:105512, 2020.