Deep Multi-view Clustering
Using Local Similarity Graphs

Shuli Jiang

April 2020

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Artur Dubrawski, Chair
Jeff Schneider

Submitted in partial fulfillment of the requirements
for the degree of Master of Science.

Copyright © 2020 Shuli Jiang
Keywords: Data mining, Multi-view clustering, Canonical correlation analysis, Local similarity graphs, Mutual K nearest neighbors
For my beloved child,
Chester Virgil Jiang.
I wish there is no panleukopenia in Heaven.
Abstract

Multi-view clustering involves clustering data with different, possibly distinct feature sets simultaneously. In many application domains, multi-view data arises naturally. For example, news can be described by both text and pictures, and multimedia segments can be described by their video signals from cameras and audio signals from voice recorders. Multi-view clustering has a wide range of potentially impactful applications. Yet, the benefits of using graph-based local similarity information to learn better representations of data for clustering, and the flexibility of incorporating pairwise constraints which may be accessible to improve clustering performance, are still under-explored in multi-view clustering.

We present Local Similarity Graph based Multi-view Clustering (LSGMC), a new and improved correlation based multi-view clustering approach. The method leverages local similarity graphs constructed by mutual K nearest neighbors. LSGMC uses the graphs to guide search for a better data representation through exploring first order proximity within views, and utilizing complementary information across views. We empirically show that LSGMC can efficiently use information from multiple views to improve clustering accuracy, and outperform state-of-the-art multi-view alternatives on a variety of benchmark and real world datasets, including image data for hand digit recognition, text data for language recognition and acoustic-articulatory data for speech recognition. We further show that LSGMC is flexible in incorporating pairwise constraints and thus it can be naturally extended to handle semi-supervised learning problems.
Acknowledgments

First, I would like to express my deepest gratitude to my advisor Professor Artur Dubrawski, for providing me with the wonderful opportunity to work at the Auton Lab, for opening me to the amazing world of research and for helping me grow as a researcher. I would also like to express my gratitude to Benedikt Boecking, a current CMU Robotics Ph.D. student and a former full-time staff researcher at the Auton Lab, for all the support and advice he gave me on my research throughout the year. Everything during my fifth-year master journey will not be made possible without Professor Dubrawski and Ben Boecking. I am so lucky to have the chance to work with both of them and I really had a wonderful time at the Auton Lab.

I would like to thank a group of wonderful people at the Auton Lab who helped me with my research and who gave me life advice, including Professor Barnabas Poczos, Jieshi (Jessie) Chen, Anthony Wertz, Dr. Eric Lei, Sibi Venkatesan, Chirag Nagpal, Xinyu (Rachel) Li, Nick Gisolfi, Vincent Jeanselme, Jiaxian (Chris) Sheng, Dr. Predrag Punosevac, and many others.

I would like to thank Professor Jeff Schneider at the Auton Lab for being my audience as my second thesis committee member. I am grateful to our program co-ordinator Tracy Farbacher who helped keep everything organized and coordinated with the oral presentation.

I would like to thank all my friends at CMU and my parents for their constant love and support.

Finally, I would like to give special thanks to my advisor Professor Dubrawski, Benedikt Boecking and Professor David P. Woodruff, for not only teaching me essential skills as a researcher but also for greatly influencing me to continue pursuing a career in research. This thesis marks the end of my student life at CMU, and the start of another wonderful journey as a graduate student researcher.
Contents

1 Introduction ................................................. 1
1.1 Motivation ........................................... 1
1.2 Contribution ........................................ 2

2 Related Work .................................................. 3
2.1 Non-centroid Based Clustering .... 3
2.2 Multi-view Clustering Principles .. 3
2.3 Popular Multi-view Clustering Methods . 4
2.4 Local Similarity Graphs .............. 4

3 Background .................................................. 5
3.1 Multi-view Clustering ......................... 5
3.2 Canonical Correlation Analysis (CCA) . 5
3.3 Deep Canonical Correlation Analysis . 6
3.4 Robust Continuous Clustering ....... 6

4 Local Similarity Graph based Multi-view Clustering (LSGMC) .... 9
4.1 Learning Latent Data Representations . 9
4.2 Consistency Between Data Across Views ... 10
4.3 Utilizing Local Similarity Graphs .... 11
4.4 Using Information From Graphs Across Views .... 12
4.5 Injecting Beliefs About Consensus ... 12
4.6 Overall Objective and Optimization .... 13
4.7 Extension to Semi-supervised Clustering .... 13

5 Experiment Setup ................................. 15
5.1 Datasets ............................................ 15
5.1.1 Noisy MNIST .................................. 16
5.1.2 Digit/MNIST-USPS ......................... 16
5.1.3 BBC+The Guardian ....................... 17
5.1.4 XRMB .................................... 17
5.2 Methods for Comparison ............. 17
5.2.1 Deep Canonical Correlation Analysis (DCCA) . 17
5.2.2 Deep Canonically Correlated Autoencoders (DCCAE) . 18
5.2.3 Deep Matrix Factorization (DMF) ..................................................... 18
5.2.4 Low-rank Sparse Subspace Clustering (LRSSC) ................................. 18
5.2.5 Deep Multimodal Subspace Clustering (DMSC) ................................. 18
5.3 Implementation Details ........................................................................... 18
5.4 Evaluation Metrics .................................................................................. 19
  5.4.1 Normalized Mutual Information (NMI) ............................................. 19
  5.4.2 Adjusted RAND Index (ARI) ............................................................ 20
  5.4.3 F1 score ............................................................................................ 20
  5.4.4 Purity ............................................................................................... 21

6 Results ......................................................................................................... 23
  6.1 Performance Comparison ......................................................................... 23
  6.2 Visualization of Local Similarity Graphs ................................................. 25
  6.3 Visualization of Embeddings ................................................................... 25
  6.4 Discussion ............................................................................................... 25
  6.5 Extension to Semi-supervised Clustering ............................................... 31

7 Conclusion ................................................................................................ 35

Bibliography ................................................................................................. 37
List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Architecture Overview: LSGMC uses one autoencoder per view ((f_1, f_2)) to project the data from the respective view to an embedding space ((U^{(1)}, U^{(2)})) and a corresponding decoder ((g_1, g_2)) reconstructs the data representation. LSGMC further constructs local similarity graphs for each view ((G^{(1)}, G^{(2)})) based on the input data. LSGMC merges local similarity graphs into a unified graph ((G_{\text{unified}})) and optionally constructs a graph with common edges between view specific local similarity graphs ((G_{\text{common}})). LSGMC trains the encoders and decoders to keep reconstruction error of each view small, maximize correlation between embeddings of different views to enforce view consistency, all while exploring first order proximity within each view and complementary information across views via (G_{\text{unified}}) and (G_{\text{common}}) to searching for better data representations.</td>
</tr>
<tr>
<td>5.1</td>
<td>Examples of Noisy MNIST data</td>
</tr>
<tr>
<td>5.2</td>
<td>Examples of Digit/MNIST-USPS data</td>
</tr>
<tr>
<td>6.1</td>
<td>Visualization of edges constructed by Mutual K nearest graphs in each view on the two components of TSNE embeddings of the original data</td>
</tr>
<tr>
<td>6.2</td>
<td>TSNE plots of the learned data representation (embedding) on 4000 samples Noisy MNIST dataset (10 classes) by different multi-view clustering approaches. The embedding of view 1 is on the left and view 2 on the right. Note that DMSC applies fusion algorithms on the learned data representation from different views to gain a unified representation. We only plot the final data representation learnt by DMSC.</td>
</tr>
<tr>
<td>6.3</td>
<td>TSNE plots of the learned data representation (embedding) on 169 samples BBC+The Guardian dataset (6 classes) by different multi-view clustering approaches. The embedding of view 1 is on the left and view 2 on the right</td>
</tr>
</tbody>
</table>
List of Tables

5.1 A summary of experiment datasets. .................................................. 16
6.1 Performance on Noisy MNIST dataset. ........................................... 23
6.2 Performance on Digit/MNIST-USPS dataset. ................................. 23
6.3 Performance on BBC+The Guardian dataset. ................................. 24
6.4 Performance on XRMB dataset. ..................................................... 24
6.5 Characterization of local similarity graphs from view 1, view 2, unified graph
and graph with common edges on different dataset. % Total, #True and % denote
the percentage of the number of edges in the local similarity graph among total
edges in a complete graph, the total number of correct edges and the percentage
of correct edges in each local similarity graph. ..................................... 24
6.6 Performance on Noisy MNIST dataset, 50% training data, 50% testing data. .......................... 32
6.7 Performance on Noisy MNIST dataset, 60% training data, 40% testing data. .......................... 32
6.8 Performance on Noisy MNIST dataset, 70% training data, 30% testing data. .......................... 33
6.9 Performance on Noisy MNIST dataset, 50% training data, 50% testing data. .......................... 33
6.10 Performance on Noisy MNIST dataset, 60% training data, 40% testing data. .......................... 33
6.11 Performance on Noisy MNIST dataset, 70% training data, 30% testing data. .......................... 33
Chapter 1

Introduction

1.1 Motivation

Multi-view clustering involves clustering data with different, possibly distinct feature sets simultaneously. In many application domains such multi-view data arises naturally. For example, news can be described by both text and pictures [26], multimedia segments can be described by their video signals from cameras and audio signals from voice recorders [10], a person can be identified by face, fingerprints, signature, or iris, with information obtained from multiple sources [34]. It has further been observed that even artificially splitting features to create multi-view data can improve the performance under multi-view learning compared to single view learning [23]. One might argue that a reasonable and simple way of performing multi-view clustering is to concatenate features from all views in order to convert multi-view clustering into a more familiar single-view setting. However, such concatenation may exacerbate the risk of over-fitting, especially with small training datasets, and it may diminish interpretation of the resulting models since each view often has specific properties [34]. Thus, it is generally preferable to consider methods that can efficiently leverage information from multiple views.

While the use of local similarity graphs–often constructed by K nearest neighbors or mutual K nearest neighbors–has been widely explored and shown to be effective in improving performance in single view clustering, e.g. [3, 11, 21, 27, 28], its use and effect remain under-explored in multi-view clustering. Further, the utility of naturally arising pairwise constraints has also received little attention in multi-view clustering. This information usually comes in the form of must-link and cannot-link pairs and has been shown to be helpful in improving single-view clustering performance [6, 24], where the setting is referred to as semi-supervised clustering or constrained clustering. Existing single view clustering algorithms which are guided by a local similarity graph have been shown to adapt well to the semi-supervised clustering setting where the graph is augmented with known pairwise constraints [11, 28].
1.2 Contribution

We present Local Similarity Graph based Multi-view Clustering (LSGMC), an improved correlation based multi-view clustering approach to address the aforementioned gaps. LSGMC learns an improved data representation in a lower dimensional embedding space through nonlinear maps. To guide this search for a better data representation, LSGMC draws on

1. The ability to reconstruct samples from the low dimensional embedding.
2. Correlation among data across views.
3. First order proximity which preserves the local structure of relationships among samples within views.
4. Complementary information across views through a unified similarity graph which is based on similarity graphs observed in the individual views.

LSGMC is able to naturally adapt to the semi-supervised setting in which we have prior knowledge about pairs of data elements that should belong to the same or different clusters. Our experiments demonstrate that the proposed approach outperforms state-of-the-art multi-view clustering approaches, including canonical correlation analysis (CCA) based deep clustering. Our main contributions are as follow:

1. We explore the usage of local similarity graphs in multi-view clustering, which is under-explored in current literature.
2. We present a new multi-view clustering approach we term LSGMC.
3. We show that LSGMC is able to outperform state-of-art multi-view clustering alternatives on benchmark datasets of various types of data.
4. We further show the flexibility of LSGMC in incorporating pairwise constraints.
Chapter 2

Related Work

2.1 Non-centroid Based Clustering

The clustering framework of the proposed algorithm is related to continuous, non-centroid based clustering approaches in single view clustering such as [11, 28]. These clustering techniques enjoy several benefits, including the flexibility of incorporating pairwise constraints, reduced sensitivity to initialization and effectiveness in high dimensions [28]. Instead of learning a set of cluster centers, non-centroid based clustering methods maintain representatives for each data sample and learn to collapse those representations into clusters under the guidance of local similarity graphs. However, the non-centroid clustering framework and the use of local similarity graphs is under-explored in multi-view setting. LSGMC draws on these ideas in a multi-view setting.

2.2 Multi-view Clustering Principles

In multi-view clustering, two general concepts are exploited: the consensus and complementary principles [34]. Within the consensus principle, the goal is to maximize agreement among multiple views since they are obtained by simultaneously observing the same object. The complementary principle states that each view of the data may contain some knowledge that other views do not have [34]. [10] further divide multi-view clustering algorithms into generative approaches and discriminative approaches. According to this survey, discriminative approaches usually perform better than generative approaches in multi-view clustering [10]. The proposed LSGMC is an unsupervised discriminative approach for multi-view clustering and mainly explore the consensus principle.
2.3 Popular Multi-view Clustering Methods

A popular strategy for multi-view clustering is to first project various feature spaces into similar lower dimensional embedding spaces or learn a unified one while keeping the consistency between views maximized, and then apply basic unsupervised clustering algorithms, such as K means, to cluster the embedding. There are various approaches for maximizing agreement between views, including co-regularization and co-training based methods \cite{15, 16, 32}, matrix factorization based methods \cite{18, 38, 39} and subspace clustering based methods under the general assumption that data from multiple views are generated from the same latent space \cite{9, 12, 19, 45, 50}. Deep learning has also been explored in multi-view clustering. These deep methods usually learn a data representation through autoencoders or convolutional neural networks, which are able to extract nonlinear features from the original data space. Deep clustering approaches have been shown to outperform traditional clustering methods \cite{2, 5, 13, 29, 31, 37}.

2.4 Local Similarity Graphs

Existing work on single view clustering has shown that graph based local similarity can be useful to improve clustering performance \cite{3, 11, 21, 27, 28}. For example, the work in \cite{28} and \cite{11} demonstrates that a connectivity matrix built via a mutual K nearest neighbor approach can bear a useful training signal and that clustering algorithms can overcome the noise contained in such a matrix.
Chapter 3

Background

3.1 Multi-view Clustering

We first formally define the multi-view clustering problem. Consider \( n \) data points available in \( V \) views. For the \( v \)-th view, \( X^{(v)} = [x_1^{(v)}, x_2^{(v)}, \ldots, x_n^{(v)}] \in \mathbb{R}^{n \times d_v} \), where \( x_i^{(v)} \) is the \( i \)-th data point and \( d_v \) is the feature number of view \( v \). Our task is to partition \( n \) data points into \( c \) clusters based on \( \{X^{(v)}\}_{v=1}^{V} \).

3.2 Canonical Correlation Analysis (CCA)

Given a dataset with two views \( X_1, X_2 \), let \( \Sigma_1, \Sigma_2 \) be the covariance matrices for each view and \( \Sigma_{12} \) be the cross-covariance. CCA finds pairs of linear projections of the two views \((w_1^T X_1, w_2^T X_1)\) such that they are maximally correlated:

\[
\max_{w_1, w_2} \frac{w_1^T \Sigma_{12} w_2}{\sqrt{w_1^T \Sigma_1 w_1 w_2^T \Sigma_2 w_2}}
\]

The problem can reformulated as

\[
\max_{w_1, w_2} w_1^T \Sigma_{12} w_2
\]
subject to \( w_1^T \Sigma_1 w_1 = w_2^T \Sigma_2 w_2 = 1 \)

When the aim is to find multiple pairs of projections, CCA is typically formulated to obtain them such that subsequent projections are uncorrelated with previous ones:

\[
\max_{W_1, W_2} Tr(X_1 \Sigma_{12} X_2)
\]

subject to \( W_1^T \Sigma_1 W_1 = W_2^T \Sigma_2 W_2 = I \)

\( w_1^{(i)T} \Sigma_{12} w_2^{(j)} = 0, \forall i, j \in \{1, 2, \ldots, n\}, i \neq j \)
3.3 Deep Canonical Correlation Analysis

With recent advances in deep learning, traditional CCA is parameterized with autoencoders to better extract features and learn data representation. The deep version of CCA (DCCA) [5], is shown to outperform traditional CCA and kernel CCA in clustering two-view data. Let $f_{1;\theta_1}, f_{2;\theta_2}$ denote the autoencoder and its parameters for the first and the second view.

$$\max_{W_1, W_2, \theta_1, \theta_2} \frac{1}{n} Tr(W_1^T f_{1;\theta_1}(X_1) f_{2;\theta_2}(X_2)^T W_2)$$

subject to

$$W_1^T \left( \frac{1}{n} f_{1;\theta_1}(X_1) f_{1;\theta_1}(X_1)^T + r_1 I \right) W_1 = I$$

$$W_2^T \left( \frac{1}{n} f_{2;\theta_2}(X_2) f_{2;\theta_2}(X_2)^T + r_2 I \right) W_2 = I$$

$$u^{(i)}_1 f_{1;\theta_1}(X_1) f_{2;\theta_2}(X_2)^T u^{(j)}_2 = 0, \forall i, j \in \{1, 2, \ldots, n\}, i \neq j$$

where $r_1, r_2 > 0$ are regularization parameters.

An improved version of DCCA, DCCAE [31] combines reconstruction based regularization and CCA. The method optimizes both the correlation between the embedding of two views and the reconstruction error through decoders. DCCAE essentially offers a trade-off between the information captured in the projection to a lower dimensional embedding space, and the information in the relationship across different views. Let $v$ denote the reconstructed data and $x^{(i)}$ denotes the $i$-th sample. The DCCAE objective is as follows:

$$\max_{W_1, W_2, \theta_1, \theta_2} \frac{1}{n} Tr(W_1^T f_{1;\theta_1}(X_1) f_{2;\theta_2}(X_2)^T W_2)$$

$$+ \frac{\lambda}{n} \left( \sum_{i=1}^{n} |x^{(i)}_1 - v^{(i)}_1|^2 + |x^{(i)}_2 - v^{(i)}_2|^2 \right)$$

with the same set of constraints as DCCA. $\lambda > 0$ is a trade-off parameter.

3.4 Robust Continuous Clustering

Robust Continuous Clustering (RCC) [28] uses local information from nearest neighbor graphs based on a relaxation of convex clustering problems. Given $n$ data points of a single view with $d$ features $X \in \mathbb{R}^{n \times d}$, RCC optimizes an embedding space $U \in \mathbb{R}^{n \times d}$ through the following objective:

$$\min_{U} \frac{1}{2} \sum_{i=1}^{n} |x^{(i)} - u^{(i)}|^2 + \frac{\lambda}{n} \sum_{p,q \in E} |w_{p,q} \rho(|u^{(p)} - u^{(q)}|)_2|^2$$

where $E$ is a set of edges in a connectivity graph constructed by a mutual K nearest neighbors algorithm on $X$. $\rho$ is a penalty function on the regularization norm that can be chosen as, for example, the $l_2$ norm. $w_{p,q}$ is the weight balancing the contribution of each data point to the pairwise terms. $\lambda$ is a parameter balancing the two terms in the objective.
Clustering-driven Deep Embedding with Pairwise Constraints (CPAC) \cite{11} is a deep extension of RCC, which learns the embedding space $U$ through deep autoencoders and is able to outperforms RCC on several datasets. In CPAC, $U$ can be a lower dimensional space than the original $X$. Let $X \in \mathbb{R}^{n \times d}$ and $U \in \mathbb{R}^{n \times m}$, where $m < d$. CPAC reformulates the objective as follows, where $v$ denotes reconstructed data points and $x^{(i)}$ denotes the $i$-th sample:

$$
\min_{\theta, \delta} \frac{1}{d} \sum_{i=1}^{n} |v^{(i)} - x^{(i)}|^2 + \frac{\lambda}{m} \sum_{p,q \in \mathcal{E}} w_{p,q} \rho(|u^{(p)} - u^{(q)}|^2)
$$

$\mathcal{E}, \lambda, \rho, w$ have the same definition as in RCC.
Chapter 4

Local Similarity Graph based Multi-view Clustering (LSGMC)

In this section we describe each component of LSGMC in detail. Figure 4.1 provides an overview of the network architecture. We learn a lower dimensional latent embedding for data in each view and maximize the consistency between data across views through canonical correlation analysis (CCA). Additionally, we construct view specific local similarity graphs through mutual K nearest neighbors based on the original data of each view. We describe how we use the graphs to explore first order proximity within views and complementary information across views, which provide additional signals for learning a better data representation. We focus our analysis on data with two views ($V = 2$), but our approach can be extended to more than two views. In the following paragraphs we detail the steps of the proposed clustering algorithm for which pseudo code is provided in Algorithm 1.

4.1 Learning Latent Data Representations

Since data from different views may stem from various different distributions—for example, text data and image data—it is natural to learn a representation of the data in some lower dimensional subspace through nonlinear maps. This allows us to compute additional losses across views in the lower dimensional representation. Autoencoders have been demonstrated as an effective way of modeling feature nonlinearity on a variety of single-view clustering approaches [11, 33], and multi-view clustering approaches [5, 14, 31, 37]. In LSGMC, we use autoencoders to learn a latent embedding for each view. Let $f_{\theta_v}, g_{\phi_v}$ denote the autoencoder and decoder for view $v$, with parameters $\theta_v, \phi_v$ respectively. We fix the dimension of the embedding space to be $p$ for all views. Let $U^{(v)} = f_{\theta_v}(X^{(v)}) \in \mathbb{R}^{n \times p}$ be the latent embedding for view $v$. We learn the embedding by the following reconstruction loss

$$L^{(v)}_{rec} = \| X^{(v)} - g_{\phi_v}(U^{(v)}) \|_2^2$$  \hspace{1cm} (4.1)

Note that the reconstruction loss is an important component to keep the embedding learned through LSGMC meaningful and to prevent inevitable false connections provided in local similarity graphs from corrupting the embedding.
Algorithm 1 LSGMC training procedure

Input: \( \{ X^{(v)} \}_{v=1}^V \), hyperparameters \( \lambda_1, \lambda_2 \), common edge weight update interval \( c \).

Output: cluster labels \( y \).

for \( v \) in 1 \ldots V do
  \( \theta_v, \phi_v \leftarrow \text{weight\_init}() \)
  Construct local similarity graph \( G^{(v)} \).
end for

Compute \( G_{\text{union}} = \bigcup_{v=1}^V G^{(v)} \).
Compute \( G_{\text{common}} = \bigcap_{v=1}^V G^{(v)} \).

for epoch in 1 \ldots \text{max epoch} do
  for all sampled minibatch indices do
    Sample a batch of \( X^{(v)} \), \( \forall v \).
    Compute \( L_{\text{rec}}^{(v)} \), \( \forall v \) by Eq. 4.1.
    Compute \( L_{\text{prox}}^{(v)} \), \( \forall v \) by Eq. 4.3.
    Compute \( L_{\text{corr}} \) by Eq. 4.2.
    \( \theta_v \leftarrow \text{RMSprop}(\theta_v, -\nabla L_{\text{rec}}^{(v)} - \nabla L_{\text{prox}}^{(v)} - L_{\text{corr}}) \)
    \( \phi_v \leftarrow \text{RMSprop}(\phi_v, -\nabla L_{\text{rec}}^{(v)} - \nabla L_{\text{prox}}^{(v)} - L_{\text{corr}}) \)
  end for
  if epoch mod \( c == 0 \) then: \( \triangleright \) Optional
    Uniformly sample a subset of edges \( G' \) from \( G_{\text{common}} \).
    Compute \( L_{\text{com}} \) based on \( G' \) by Eq. 4.4.
    \( \theta_v \leftarrow \text{RMSprop}(\theta_v, -\nabla L_{\text{com}}) \)
    \( \phi_v \leftarrow \text{RMSprop}(\phi_v, -\nabla L_{\text{com}}) \)
  end if
end for

\( y \leftarrow \text{KMeans}(U^{(v)}) \) for any \( v \).

4.2 Consistency Between Data Across Views

One widely accepted assumption in multi-view clustering is that data from different views have certain forms of consistency. Through maximizing a linear correlation between data from two views, CCA is an effective way of maximizing data consistency across views. Since CCA has a closed form solution [5], we can maximize view consistency through the following correlation loss. Let

\[
\bar{U}^{(v)} = U^{(v)} - \frac{1}{n} U^{(v)} 1
\]

be the re-centered embedding for view \( v \),

\[
\Sigma^{(v)} = \frac{1}{n-1} \bar{U}^{(v)T} \bar{U}^{(v)}
\]

the covariance matrix of view \( v \)'s embedding, and

\[
\Sigma^{(v_1, v_2)} = \frac{1}{n-1} \bar{U}^{(v_1)T} \bar{U}^{(v_2)}
\]

10
Figure 4.1: Architecture Overview: LSGMC uses one autoencoder per view \((f_1, f_2)\) to project the data from the respective view to an embedding space \((U^{(1)}, U^{(2)})\) and a corresponding decoder \((g_1, g_2)\) reconstructs the data representation. LSGMC further constructs local similarity graphs for each view \((G^{(1)}, G^{(2)})\) based on the input data. LSGMC merges local similarity graphs into a unified graph \((G_{unified})\) and optionally constructs a graph with common edges between view specific local similarity graphs \((G_{common})\). LSGMC trains the encoders and decoders to keep reconstruction error of each view small, maximize correlation between embeddings of different views to enforce view consistency, all while exploring first order proximity within each view and complementary information across views via \(G_{unified}\) and \(G_{common}\) to searching for better data representations.

the cross-covariance matrix between embeddings of view \(v_1\) and \(v_2\). We further denote

\[
T = \Sigma^{(v_1)-1/2}\Sigma^{(v_1,v_2)}\Sigma^{(v_2)-1/2}
\]

and using the singular values of \(T\) we define the correlation loss

\[
L_{corr} = -\sum_{i=1}^{p} \sigma_i
\]

(4.2)

where the \(\sigma_i\)'s are the top \(p\) singular values of \(T\), with \(p\) being the dimension of the latent embedding.

4.3 Utilizing Local Similarity Graphs

A commonly used idea in learning good data representations is that if two data points are close in the original space via some domain appropriate similarity function, their representations in the embedding space should also be close. Further, one might observe that if two data points are close to each other in the original space, they are more likely to be from the same cluster. We call a graph of pairs established via such a heuristic a local similarity graph and use it to guide the search for a better embedding since an edge in such a graph indicates a higher probability
that two incident nodes belong to the same cluster.

To further improve the quality of the graph, we use a mutual nearest neighbor approach, meaning that two nodes are only connected if they are both nearest neighbors of each other in a K nearest neighbor graph. Prior research [8] has shown that graphs constructed through mutual K nearest neighbors can better capture local similarities in clustering tasks. We construct one such local similarity graph $G^{(v)}$ for each view $v$ and in our experiments use cosine similarity between data points in the original space.

### 4.4 Using Information From Graphs Across Views

We observe that view specific local similarity graphs encode view specific information as they differ across views due to different data representations in the respective original spaces. Thus, view specific graphs can provide complementary information. An edge connected in $G^{(v_1)}$ might be disconnected in $G^{(v_2)}$ but is expected to still provide valuable information in learning the embedding for view $v_2$. Thus we consider unifying all view specific graphs into one common graph $G = \cup_{i=1}^V G^{(v)}$ and use the common graph to guide the learning of embeddings for all views.

To encourage first order proximity in the embedding space, we define the following proximity loss, $\forall v$,

$$L_{\text{prox}}^{(v)} = \sum_{(i,j) \in G} \|U_i^{(v)} - U_j^{(v)}\|_2^2 \quad (4.3)$$

It is important to note two potential drawbacks of the proximity loss in isolation. First, a trivial solution to minimizing the loss is to collapse all data representations in the embedding into a single cluster. The embedding thus fails to represent the actual relationship between data points. Second, due to relying on a heuristic such as K mutual nearest neighbors, view specific graphs will inevitably contain false connections. $L_{\text{prox}}^{(v)}$ by itself provides no mechanism to account for such mistakes. However, the reconstruction loss is able to counter such issues. By forcing a reconstructed sample to be close to its representation in the original space, the reconstruction loss encourages learning of a meaningful embedding and avoids collapse of all samples towards a single point.

### 4.5 Injecting Beliefs About Consensus

We may consider a complementary operation of unifying all view specific similarity graphs $G^{(v)}$ to find the common edges among all graphs. A natural belief derived from the multi-view consensus clustering assumption is that edges occurring in all view specific local similarity graphs are more likely to provide true information that two incident nodes belong to the same cluster. In order to avoid reinforcing false information, we consider uniformly sampling a subset of the common edges. Let $G' = \text{sample}(\cap_{v=1}^V G^{(v)})$. We can encourage proximity through a
subset of common edges as follows

\[ L_{\text{com}} = \sum_{(i,j) \in G'} |U_i^{(v)} - U_j^{(v)}|^2 \]  \hspace{1cm} (4.4)

every constant number of epochs, to reinforce the consensus belief.

### 4.6 Overall Objective and Optimization

Let \( \Theta = \{\theta_v, \phi_v\}_{v=1}^V \) denote the overall parameters to train. The joint objective including all previously mentioned losses is

\[ L_{\Theta} = \sum_{v=1}^V (L_{\text{rec}}^{(v)} + \lambda_1^{(v)} L_{\text{prox}}^{(v)} + \lambda_2 L_{\text{com}}) \]  \hspace{1cm} (4.5)

where \( \lambda_1 \) and \( \lambda_2 \) are trade-off hyperparameters to balance the losses. In our experiments, we set \( \lambda_1^{(v)} = \frac{|X^{(v)}|_F}{\sigma} \), where \( | \cdot |_F \) denotes the Frobenius norm and \( \sigma \) is the largest eigenvalue of the graph Laplacian based on the cosine distance matrix we used to construct \( G^{(v)} \). See [28] for details. We set \( \lambda_2 = 1 \). The overall LSGMC learning procedure is presented in **Algorithm 1**. Equation 4.5 can be minimized through standard backpropagation algorithms to update parameters in the encoder and the decoder for each view. In our implementation we use RMSprop to learn \( \Theta \) and apply K Means to obtain the final clustering based on the embeddings that were learned. We note that the correlation loss can be optimized efficiently only if the gradient is estimated using a sufficiently large minibatch [31].

### 4.7 Extension to Semi-supervised Clustering

In some applications one may have access to reliable pairwise information for a small portion of observations from ground truth or meta data. Such pairs usually come in the form of pairwise *must-link* and *cannot-link* constraints. LSGMC is able to incorporate such information by augmenting all local similarity graphs.
Chapter 5

Experiment Setup

5.1 Datasets

We use five widely used multi-view clustering benchmarks and real world datasets for LS-GMC evaluation. Figure 5.1 and Figure 5.2 shows examples from the Noisy MNIST dataset and the Digit/MNIST-USPS dataset respectively. Table 5.1 presents a summary of all datasets used in the experiments.

![Noisy MNIST data](image)

Figure 5.1: Examples of Noisy MNIST data.
Table 5.1: A summary of experiment datasets.

<table>
<thead>
<tr>
<th>Dataset</th>
<th># samples</th>
<th># features view 1</th>
<th># features view 2</th>
<th># clusters</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Noisy MNIST</td>
<td>4000</td>
<td>784</td>
<td>784</td>
<td>10</td>
<td>image</td>
</tr>
<tr>
<td>Digit/MNIST-USPS</td>
<td>4000</td>
<td>784</td>
<td>256</td>
<td>10</td>
<td>image</td>
</tr>
<tr>
<td>BBC+The Guardian</td>
<td>169</td>
<td>3560</td>
<td>3631</td>
<td>6</td>
<td>text</td>
</tr>
<tr>
<td>XRMB</td>
<td>20,000</td>
<td>273</td>
<td>112</td>
<td>20</td>
<td>acoustic-articulatory</td>
</tr>
</tbody>
</table>

5.1.1 Noisy MNIST

Following [31], we create two noisy views from MNIST handwritten digits [17]. We randomly select a small subset of 4000 samples out of the entire 70K samples, with 400 samples per class. The first view is a random rotation of the original digit with angles sampled uniformly within range $\left[ -\frac{\pi}{4}, \frac{\pi}{4} \right]$. The corresponding second view is randomly selected from the same class from the original dataset. The pixel values are scaled to $[0, 1]$, masked with i.i.d noise uniformly drawn from $[0, 1]$, and subsequently truncated to $[0, 1]$. Each view has a dimension of 784. The dataset has 10 classes.

5.1.2 Digit/MNIST-USPS

Following [25] we randomly select 4000 samples from MNIST and USPS, with 400 samples per class. We randomly match digits from MNIST with digits of the same class from USPS to

https://csc.lsu.edu/~saikat/n-mnist/
form a two view handwritten digits data. The first view (MNIST) has a dimension of 784 and the second view (USPS) has a dimension of 256. The dataset has 10 classes.

5.1.3 BBC+The Guardian

The 3 sources dataset is a multi-view text dataset consisting of news articles collected from three online news sources: BBC, The Guardian, and Reuters. All articles are represented as bag-of-words and each article is annotated with at least one of six topics. Following [7], we use 169 articles that are available in all three sources. We use one annotation for each article and use BBC and The Guardian as the first and second view. The first view (BBC) has a dimension of 3560 and the second view (the Guardian) has a dimension of 3631. The dataset has 6 classes.

5.1.4 XRMB

The Wisconsin X-ray Microbeam (XRMB) data consists of simultaneously recorded speech and articulatory measurements from 47 American English speakers. We use a processed version from [30]. The first view (acoustic measurements) has 39 features consisting of mel frequency cepstral coefficients (MFCCs) and their first and second derivatives, and the second view (articulatory measurements) has horizontal/vertical displacement of 8 pellets attached to different parts of the vocal tract. Both features are concatenated over a 7-frame window around each frame, resulting in a total of 273 features for the first view and 112 features for the second view. The task is to cluster phones based on two set of features. We randomly sample 20 phones from 35 speakers, with each phone 1000 samples, resulting in a total of 20,000 samples.

5.2 Methods for Comparison

We use LSGMC to denote our proposed approach without injecting beliefs about consensus among edges across local similarity graphs from different views. We use LSGMC+ to denote our proposed approach with injecting beliefs about consensus. In our experiments using LSGMC+, we uniformly sample 20% common edges and compute \( L_{com} \) every 5 epochs. We compare LSGMC and LSGMC+ against the following state-of-art multi-view clustering approaches:

5.2.1 Deep Canonical Correlation Analysis (DCCA)

Deep Canonical Correlation Analysis (DCCA) [5] is a multi-view clustering method based on canonical correlation analysis (CCA) for regularizing data from two views. DCCA improves CCA by using deep autoencoders to better extract nonlinear features from data. DCCA is shown to outperform CCA and kernel CCA on various datasets.

[https://github.com/mbrbic/Multi-view-LRSSC](https://github.com/mbrbic/Multi-view-LRSSC)
[https://ttic.uchicago.edu/~klivescu/XRMB_data/full/README](https://ttic.uchicago.edu/~klivescu/XRMB_data/full/README)
5.2.2 **Deep Canonically Correlated Autoencoders (DCCAE)**

Deep Canonically Correlated Autoencoders (DCCAE) [31] is an improved version of DCCA. DCCAE uses not only CCA parameterized by deep autoencoders but also uses decoders and reconstruction error to improve the learnt data embedding. DCCAE is able to outperform DCCA on several datasets.

5.2.3 **Deep Matrix Factorization (DMF)**

Deep Matrix Factorization (DMF) [39] explores complementary information across views by using semi-nonnegative matrix factorization to learn the hierarchical semantics of multi-view data in a layer-wise fashion, with graph regularizers to represent intrinsic geometric structure in each view data.

5.2.4 **Low-rank Sparse Subspace Clustering (LRSSC)**

Multi-view Low-rank Sparse Subspace Clustering (LRSSC) [7] learns a joint subspace representation by constructing affinity matrix shared among all views while encouraging sparsity and low-rankness solutions at the same time. Note that LRSSC has four different variations. We report the highest scores among the four variations in our experiments.

5.2.5 **Deep Multimodal Subspace Clustering (DMSC)**

Deep Multimodal Subspace Clustering (DMSC) [1] is a convolutional neural network based deep multimodal subspace clustering mainly for image datasets. DMSC uses encoders and decoders for learning a lower dimensional data representation and explores different affinity fusion techniques to regularize data across views. The DMSC implementation does not allow clustering of multi-view data with different dimensionality for each view. We are thus only able to report results of DMSC on dataset where the number of features in both views are the same.

5.3 **Implementation Details**

In order to enable a fair comparison of LSGMC against DCCA and DCCAE, we use the same set of hyperparameters and optimizer across the three approaches. In computing correlation loss, we use the top $p$ singular values, where $p$ is the same as the embedding dimension. We evaluate all clustering methods in an unsupervised clustering setting using K means clustering on the learned data representation for each view and report the best score among two views. For our approach, to construct the local similarity graph for each view, we fix the number of nearest neighbors to 10 to create a mutual K nearest neighbors graph for each dataset. [20] suggests a lower bound on the number of neighbors in a mutual K nearest neighbors graph to successfully identify clusters as $K \propto \log(n)$ where $n$ is the number of samples. In our experiments, our samples range from about 200 to 4000, meaning the lower bound lies in $[7.5, 12]$. 
We use the RMSprop optimizer with a weight decay of $1e^{-5}$ and a learning rate of $1e^{-3}$. We fix the dimension of the embedding space to be 10, which is the same as the original setting used in both DCCA and DCCAE. The autoencoder for each view has three hidden layers and each layer has 1024 units. We run the experiment for at most 100 epochs for Noisy MNIST, BBC+The Guardian and XRMB dataset, at most 300 epochs for Digit/MNIST-USPS dataset. We use a batch size of 800 in each minibatch update. We use random initialization of all parameters in the encoders and decoders without pre-training. We did not find a significant improvement of the clustering performance using pre-training.

5.4 Evaluation Metrics

Consistent with relevant literature [4, 22], we use four extrinsic metrics, which compares the output of the clustering algorithm and a ground truth (true categories/labels), for evaluating the clustering performance: Normalized Mutual Information (NMI), Adjusted RAND Index (ARI), F1 score and Purity for evaluating clustering performance. Those four widely applied metrics in clustering literature captures different aspects of the clustering algorithms. NMI is a metric based on entropy. The entropy of a predicted cluster reflects how the members of different true categories are distributed within each the cluster. ARI is a metric which considers statistics over pairs of items. Purity and F1 score are metrics based on set matching. Both assume a one to one mapping between predicted clusters and true categories, and used precision and recall for comparison. We give detailed description of each metric below.

Since we are comparing the quality of learnt data embedding, we assume the true number of clusters, $k$, is known a priori during evaluation. Let $Y = \{y_1, \ldots, y_k\}$ be the set of true categories, $C = \{c_1, \ldots, c_k\}$ be the set of predicted clusters and $N$ be the number of samples.

5.4.1 Normalized Mutual Information (NMI)

NMI is an information theory based metric. NMI is defined as

$$\text{NMI}(Y, C) = \frac{2 \times I(Y; C)}{H(Y) + H(C)}$$

$$I(Y; C) = \sum_i \sum_j \Pr[y_i \cap c_j] \log \frac{\Pr[y_i \cap c_j]}{\Pr[y_i] \Pr[c_j]}$$

$$= \sum_i \sum_j \frac{|y_i \cap c_j|}{N} \log \frac{N|y_i \cap c_j|}{|y_i||c_j|}$$

$$H(Y) = -\sum_i \Pr[y_i] \log \Pr[y_i]$$

$$= -\sum_i \frac{|y_i|}{N} \log \frac{|y_i|}{N}$$

where $\Pr[y_i]$, $\Pr[c_j]$ and $\Pr[y_i \cap c_j]$ denote the probability of a data sample being in category $y_i$, predicted cluster $c_j$ and in the intersection of $y_i$ and $c_j$, respectively. $I(\cdot; \cdot)$ is the mutual information and $H(\cdot)$ is the entropy. $H(C)$ is defined similarly as $H(Y)$. 19
Mutual information $I(Y; C)$ measures the amount of information by which our knowledge about the true categories increases when we are told what the predicted clusters are. The minimum of $I(Y; C)$ is 0 if the clustering is random. The maximum of $I(Y; C)$ is when $C$ matches $Y$ exactly. A larger number of clusters will result in a larger $I(Y; C)$. To fix the problem, people usually normalize the mutual information by $H(Y) + H(C)$ since a larger number of clusters results in an increase in the entropy. $H(Y) + H(C)$ is a tight upper bound of $I(Y; C)$ and thus $\text{NMI}(Y; C) \in [0, 1]$.

### 5.4.2 Adjusted RAND Index (ARI)

ARI measures the similarity between clustering through considering $\{N\text{choose}2\}$ pairs of samples. A True Positive (TP) denotes that two samples from the same true category are assigned to the same cluster. A True Negative (TN) denotes that two samples from different category are assigned to different cluster. A False Positive (FP) denotes that samples from two different category are assigned to the same cluster. A False Negative (FN) denotes that samples from the same category are assigned to different clusters. RAND index (RI) is defined as

$$ RI = \frac{TP + TN}{TP + FP + FN + TN} $$

Adjusted RAND Index (ARI) is an “adjusted for chance” under the permutation model. This means ARI gets an expected score of 0 on random clustering assignments, which makes ARI more interpretable than RI. Consider the contingency matrix $T \in \mathbb{R}^{k \times k}$, where $T_{ij} = |y_i \cap c_j|$. Let $a_i = \sum_j T_{ij}$ (row-wise sum) and $b_j = \sum_i T_{ij}$ (column-wise sum).

$$ \text{ARI} = \frac{\sum_i \sum_j \left(\frac{T_{ij}}{2}\right)^2 - \left[\sum_i \left(\frac{a_i}{2}\right)^2 \sum_j \left(\frac{b_j}{2}\right)^2\right] / \left(\binom{N}{2}\right)}{\frac{1}{2} \left[\sum_i \left(\frac{a_i}{2}\right)^2 + \sum_j \left(\frac{b_j}{2}\right)^2\right] - \left[\sum_i \left(\frac{a_i}{2}\right)^2 \sum_j \left(\frac{b_j}{2}\right)^2\right] / \left(\binom{N}{2}\right)} $$

A bad or random clustering will get an expected ARI value 0 and an exact matching between $Y$ and $C$ will get ARI value 1.

### 5.4.3 F1 score

F1 score is a trade-off between clustering correctly all samples from the same true category into the same predicted cluster and making sure that each predicted cluster contains points from only one category. Let Precision and Recall for a true category $y_i$ and a predicted cluster $c_j$ defined as follow

$$ \text{Precision}(y_i, c_j) = \frac{|c_j \cap y_i|}{|c_j|} $$

$$ \text{Recall}(y_i, c_j) = \frac{|c_j \cap y_i|}{|y_i|} $$
Let $F$ denotes the harmonic mean of Precision and Recall. We have the $F1$ score as

$$F(y_i, c_j) = \frac{2 \times \text{Recall}(y_i, c_j) \text{Precision}(y_i, c_j)}{\text{Recall}(y_i, c_j) + \text{Precision}(y_i, c_j)}$$

$$F1 \text{ score}(Y, C) = \frac{1}{N} \sum_i \frac{|y_i|}{\max_j F(y_i, c_j)}$$

A bad clustering will have a low $F1$ score, while an exact match between $Y$ and $C$ will get a $F1$ score of 1.

### 5.4.4 Purity

Purity is a simple and straight-forward evaluation metric based on set matching. When calculating purity, each predicted cluster $c_j$ is assigned to the category $y_i$ which is most frequent in the cluster and then the accuracy of this assignment is measured by counting the number of correctly assigned samples and dividing by $N$. Specifically, Purity is defined as

$$\text{Purity}(Y, C) = \frac{1}{N} \sum_j \max_i |y_i \cap c_j|$$

A bad clustering will get purity 0 and an exact matching between $Y$ and $C$ will get purity 1. However, high purity can also be achieved when the number of cluster is large. Thus we might also need to consider alternative evaluation metrics when assessing the clustering performance.
Chapter 6

Results

6.1 Performance Comparison

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>NMI</th>
<th>ARI</th>
<th>F1 score</th>
<th>Purity</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCCA</td>
<td>60.56 (1.62)</td>
<td>49.80 (2.58)</td>
<td>54.88 (2.32)</td>
<td>70.79 (2.76)</td>
</tr>
<tr>
<td>DCCAE</td>
<td>62.53 (0.31)</td>
<td>52.88 (0.88)</td>
<td>57.72 (0.81)</td>
<td>70.65 (0.89)</td>
</tr>
<tr>
<td>DMF</td>
<td>15.16 (0.39)</td>
<td>5.85 (0.12)</td>
<td>15.30 (0.12)</td>
<td>18.47 (0.42)</td>
</tr>
<tr>
<td>LRSSC</td>
<td>47.73 (0.03)</td>
<td>34.61 (0.03)</td>
<td>41.34 (0.02)</td>
<td>57.25 (0.65)</td>
</tr>
<tr>
<td>DMSC</td>
<td>48.27 (1.66)</td>
<td>28.98 (0.93)</td>
<td>37.52 (0.59)</td>
<td>56.43 (3.28)</td>
</tr>
<tr>
<td>LSGMC</td>
<td>74.22 (0.72)</td>
<td>61.94 (2.66)</td>
<td>65.90 (2.28)</td>
<td>79.59 (3.08)</td>
</tr>
<tr>
<td>LSGMC+</td>
<td>74.79 (0.76)</td>
<td>63.39 (1.72)</td>
<td>67.16 (1.52)</td>
<td>80.14 (2.50)</td>
</tr>
</tbody>
</table>

Table 6.1: Performance on Noisy MNIST dataset.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>NMI</th>
<th>ARI</th>
<th>F1 score</th>
<th>Purity</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCCA</td>
<td>96.53 (0.49)</td>
<td>96.96 (0.45)</td>
<td>97.26 (0.41)</td>
<td>98.62 (0.21)</td>
</tr>
<tr>
<td>DCCAE</td>
<td>95.20 (0.06)</td>
<td>95.66 (0.11)</td>
<td>96.09 (0.10)</td>
<td>98.02 (0.05)</td>
</tr>
<tr>
<td>DMF</td>
<td>53.44 (0.23)</td>
<td>41.29 (0.22)</td>
<td>47.52 (0.20)</td>
<td>63.19 (0.15)</td>
</tr>
<tr>
<td>LRSSC</td>
<td>46.07 (0.06)</td>
<td>35.28 (0.06)</td>
<td>42.01 (0.06)</td>
<td>56.98 (0.05)</td>
</tr>
<tr>
<td>DMSC</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>LSGMC</td>
<td>97.76 (0.17)</td>
<td>98.10 (0.16)</td>
<td>98.29 (0.14)</td>
<td>99.14 (0.07)</td>
</tr>
<tr>
<td>LSGMC+</td>
<td>97.72 (0.25)</td>
<td>98.02 (0.26)</td>
<td>98.22 (0.23)</td>
<td>99.11 (0.12)</td>
</tr>
</tbody>
</table>

Table 6.2: Performance on Digit/MNIST-USPS dataset.

We conduct 5 random runs of each experiment and report mean and standard deviation of values of the best under each metric from each run. Table 6.1, Table 6.2, Table 6.3 and Table 6.4 provide comparisons of the clustering performance of LSGMC against other state-of-art approaches on each respective dataset. We first compare LSGMC against related methods and then compare LSGMC against LSGMC+.
<table>
<thead>
<tr>
<th>Algorithm</th>
<th>NMI</th>
<th>ARI</th>
<th>F1 score</th>
<th>Purity</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCCA</td>
<td>44.83 (7.54)</td>
<td>35.41 (9.53)</td>
<td>50.05 (6.92)</td>
<td>66.63 (4.97)</td>
</tr>
<tr>
<td>DCCAE</td>
<td>45.08 (1.36)</td>
<td>38.49 (2.67)</td>
<td>51.86 (1.08)</td>
<td>65.68 (1.74)</td>
</tr>
<tr>
<td>DMF</td>
<td>33.78 (0.97)</td>
<td>20.82 (1.91)</td>
<td>36.97 (1.87)</td>
<td>59.41 (1.43)</td>
</tr>
<tr>
<td>LRSSC</td>
<td>56.36 (0.56)</td>
<td>45.97 (1.34)</td>
<td>57.45 (1.32)</td>
<td>77.32 (0.56)</td>
</tr>
<tr>
<td>DMSC</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>LSGMC</td>
<td>73.95 (1.87)</td>
<td>72.21 (1.67)</td>
<td>78.52 (1.17)</td>
<td>84.33 (0.44)</td>
</tr>
<tr>
<td>LSGMC+</td>
<td>71.16 (1.41)</td>
<td>67.48 (2.92)</td>
<td>75.19 (2.06)</td>
<td>83.20 (0.60)</td>
</tr>
</tbody>
</table>

Table 6.3: Performance on **BBC+The Guardian** dataset.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>NMI</th>
<th>ARI</th>
<th>F1 score</th>
<th>Purity</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCCA</td>
<td>34.72 (0.58)</td>
<td>14.88 (0.95)</td>
<td>19.70 (0.78)</td>
<td>31.89 (1.14)</td>
</tr>
<tr>
<td>DCCAE</td>
<td>24.18 (0.14)</td>
<td>2.16 (0.08)</td>
<td>10.85 (0.04)</td>
<td>20.25 (0.12)</td>
</tr>
<tr>
<td>DMF</td>
<td>3.30 (0.11)</td>
<td>0.96 (0.09)</td>
<td>8.55 (0.55)</td>
<td>9.00 (0.15)</td>
</tr>
<tr>
<td>LRSSC</td>
<td>28.95 (0.04)</td>
<td>14.45 (0.13)</td>
<td>18.81 (0.11)</td>
<td>31.17 (0.70)</td>
</tr>
<tr>
<td>DMSC</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>LSGMC</td>
<td>36.96 (0.33)</td>
<td>16.78 (0.61)</td>
<td>21.56 (0.52)</td>
<td>35.30 (0.60)</td>
</tr>
<tr>
<td>LSGMC+</td>
<td>36.69 (0.31)</td>
<td>17.07 (0.35)</td>
<td>21.75 (0.34)</td>
<td>35.40 (0.32)</td>
</tr>
</tbody>
</table>

Table 6.4: Performance on **XRMB** dataset.

LSGMC, DCCA and DCCAE perform better than the other clustering methods on the image datasets. On Noisy MNIST, LSGMC significantly outperforms DCCA and DCCAE, across all performance metrics we compute, see Table 6.1. On Digit/MNIST-USPS, LSGMC also outperforms all related approaches, but the gap in performance is less pronounced as the methods achieve high quality clustering results, see Table 6.2.

On the text data, LSGMC and LRSSC perform better than the related methods. LSGMC significantly outperforms LRSSC as can be seen in Table 6.3. On the acoustic-articulatory data, LSGMC and DCCA perform better than the related methods. LSGMC slightly outperforms DCCA as can be seen in Table 6.4.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>View 1</th>
<th>View 2</th>
<th>Unified</th>
<th>Common</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>%Total</td>
<td>#True</td>
<td>%Total</td>
<td>#True</td>
</tr>
<tr>
<td>Noisy MNIST</td>
<td>0.14</td>
<td>9765</td>
<td>88.62</td>
<td>0.07</td>
</tr>
<tr>
<td>Digit</td>
<td>0.14</td>
<td>10669</td>
<td>95.18</td>
<td>0.13</td>
</tr>
<tr>
<td>BBC+Gua</td>
<td>4.09</td>
<td>469</td>
<td>80.72</td>
<td>3.68</td>
</tr>
<tr>
<td>XRMB</td>
<td>0.027</td>
<td>31012</td>
<td>57.34</td>
<td>0.026</td>
</tr>
</tbody>
</table>

Table 6.5: Characterization of local similarity graphs from view 1, view 2, unified graph and graph with common edges on different dataset. % Total, #True and % denote the percentage of the number of edges in the local similarity graph among total edges in a complete graph, the total number of correct edges and the percentage of correct edges in each local similarity graph.
The results comparing our LSGMC and LSGMC+ methods are inconclusive. LSGMC+, which takes into account additional consensus information, slightly outperforms LSGMC on the Noisy MNIST data, but not significantly. LSGMC+ has the same performance as LSGMC on Digit/MNIST-USPS and XRMB dataset, and performs slightly worse on the small BBC+The Guardian dataset.

6.2 Visualization of Local Similarity Graphs

To better understand the local similarity graphs constructed by Mutual K nearest neighbors (MKNN), in Figure 6.1, we plot the view specific MKNN graphs, on two components of TSNE embeddings of the original data, on the Noisy MNIST and BBC+The Guardian dataset. Each dot in the plot represents a data point in the original space, and each red dashed line represents an edge in the MKNN graph. We observe that the density of MKNN edges provide some information about the clusters.

6.3 Visualization of Embeddings

In Figure 6.2 and Figure 6.3 we visualize and compare TSNE plots of the low dimensional embeddings learned on the Noisy MNIST and BBC+The Guardian dataset, where LSGMC greatly outperforms the other multi-view clustering methods. The plots strongly suggest that the representation learned by our proposed LSGMC is more separable. The TSNE plots provide visual evidence suggesting that local connectivity graph can be a very useful signal in unsupervised learning on multi-view data.

6.4 Discussion

Our experiments suggest that the proposed algorithm LSGMC outperforms state-of-art alternatives consistently on popular image and text benchmark datasets as well as a real world acoustic-articulatory dataset. While correlations between data from different views can be powerful in regularizing view consistency, the guidance of local similarity graphs can significantly improve the performance of a clustering algorithm. This is evidenced by the performance of LSGMC over related correlation based multi-view clustering methods, DCCA and DCCAE.

We show characteristics of the local similarity graphs on each dataset in Table 6.5. We characterize the percentage of edges in a complete graph, the total number of correct edges (i.e. edges whose incident nodes belong to the same cluster) and the percentage of correct edges in each local similarity graph from view 1, view 2, unified graph and graph with common edges. We observe that the percentage of correct edges of unified graphs is \( \geq 80\% \) on each dataset except XRMB on which the percentage of correct edges is only 54\%, while covering only a small portion of all possible edges. The results demonstrate that a small amount of information from the local similarity graphs with a reasonable quality–i.e. a sufficiently large portion of the edges
Figure 6.1: Visualization of edges constructed by Mutual K nearest graphs in each view on the two components of TSNE embeddings of the original data.
Figure 6.2: TSNE plots of the learned data representation (embedding) on 4000 samples Noisy MNIST dataset (10 classes) by different multi-view clustering approaches. The embedding of view 1 is on the left and view 2 on the right. Note that DMSC applies fusion algorithms on the learned data representation from different views to gain a unified representation. We only plot the final data representation learnt by DMSC.
Figure 6.3: TSNE plots of the learned data representation (embedding) on 169 samples BBC+The Guardian dataset (6 classes) by different multi-view clustering approaches. The embedding of view 1 is on the left and view 2 on the right.
in the graph is indicating the correct relationship between data points–can be used as a powerful training signal to guide the search for a better embedding space. Our experiments and results demonstrate that this is indeed the case for several different data types and common multi-view benchmark datasets. We did not tune the number of neighbors for each dataset to improve the quality of the graphs but rather chose a fixed value in a range suggested by the analysis presented in [8].

Additionally, the characteristics of local similarity graphs provide insights into the performance difference between LSGMC and LSGMC+ on different datasets.

On Noisy MNIST, the quality of view 2’s local similarity is low, with only 67.29% correct edges, compared to view 1’s local similarity graph, with 88.62% correct edges. As a result, the unified graph has only 81.39% correct edges, while there are 95.24% correct edges among the common edges between view 1 and view 2. On XRMB, the quality of both view 1 and view 2’s local similarity is low, with only 57.34% and 53.42% correct edges respectively. The unified graph has only 54.21% correct edges while there are 92.50% correct common edges. Since LSGMC uses the unified graph to learn the embedding for both views, the common edge graph with high correctness could provide valuable additional information to the learning process. This explains why we observe > 1% increase in ARI and F1 score in LSGMC+ on Noisy MNIST, and a slight increase in ARI, F1 score and Purity in LSGMC+ on XRMB, even if we only use the information of 20% common edges selected uniformly randomly every 5 epochs in training. This case corresponds to the general belief on consensus that common edges could provide more reliable training signals.

On the other two datasets, however, the difference between the percentage of correct edges in the unified graph and the graph with common edges is small, 5.83% and 5.28% on Digit/MNIST-USPS and BBC+The Guardian respectively. Common edges do not provide much more information than edges in the unified graph. The slight drop in performance of LSGMC+ on the two datasets may be explained by the fact that the algorithm is biased towards a smaller number common edges despite them not being much more accurate than other edges. This case is contrary to the belief that common edges could provide more reliable training signals. In conclusion, whether a graph with common edges across views should be used as an additional training signal to the unified graph, depends on the inherent local structure of the data.

6.5 Extension to Semi-supervised Clustering

We demonstrate the use of LSGMC in the semi-supervised clustering setting and demonstrate the flexibility it offers in incorporating pairwise must-link and cannot-link constraints, on the Noisy MNIST dataset. We first split the dataset into training and testing sets. We experiment with 50% training - 50% testing split, 60% training - 40% testing split and 70% training - 30% testing split. We construct the unified local similarity graphs based on the entire dataset. For pairwise must-link constraints, we randomly select 500, 1000 and 5000 and 10000 pairs of
samples which belong to the same cluster and which do not appear in the unified local similarity graph, based on the training set only. For pairwise cannot-link constraints, we randomly remove 100, 500 and 1000 pairs which belong to different clusters in the unified similarity graph across data in the training set only. We evaluate the performance on the testing set.

The results for incorporating pairwise must-link constraints are presented in Table 6.8, 6.6 and 6.7. The results for removing pairwise cannot-link constraints are presented in Table 6.9, 6.10 and 6.11. We observe an increase across all performance metrics, as the number of must-link or cannot-link constraints increase. This shows LSGMC is able to make use of pairwise information about the data, when such information is available.
### Table 6.8: Performance on Noisy MNIST dataset, 70% training data, 30% testing data.

<table>
<thead>
<tr>
<th># must-link</th>
<th>NMI</th>
<th>ARI</th>
<th>F1 score</th>
<th>Purity</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (baseline)</td>
<td>74.22 (0.72)</td>
<td>61.94 (2.66)</td>
<td>65.90 (2.28)</td>
<td>79.59 (3.08)</td>
</tr>
<tr>
<td>500</td>
<td>75.85 (1.62)</td>
<td>62.51 (1.71)</td>
<td>66.51 (1.60)</td>
<td>79.93 (0.69)</td>
</tr>
<tr>
<td>1000</td>
<td>76.23 (2.44)</td>
<td>64.41 (4.59)</td>
<td>68.19 (4.10)</td>
<td>80.57 (3.40)</td>
</tr>
<tr>
<td>5000</td>
<td>75.14 (2.03)</td>
<td>63.91 (0.69)</td>
<td>67.68 (0.64)</td>
<td>81.07 (0.70)</td>
</tr>
<tr>
<td>10000</td>
<td>76.09 (1.47)</td>
<td>64.38 (2.56)</td>
<td>68.11 (2.24)</td>
<td>81.67 (1.59)</td>
</tr>
</tbody>
</table>

### Table 6.9: Performance on Noisy MNIST dataset, 50% training data, 50% testing data.

<table>
<thead>
<tr>
<th># cannot-link</th>
<th>NMI</th>
<th>ARI</th>
<th>F1 score</th>
<th>Purity</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (baseline)</td>
<td>74.22 (0.72)</td>
<td>61.94 (2.66)</td>
<td>65.90 (2.28)</td>
<td>79.59 (3.08)</td>
</tr>
<tr>
<td>100</td>
<td>75.00 (0.85)</td>
<td>61.84 (3.24)</td>
<td>65.93 (2.83)</td>
<td>77.82 (2.41)</td>
</tr>
<tr>
<td>300</td>
<td>75.16 (1.49)</td>
<td>63.30 (3.08)</td>
<td>67.05 (2.75)</td>
<td>80.68 (2.42)</td>
</tr>
<tr>
<td>500</td>
<td>75.91 (1.05)</td>
<td>62.81 (2.96)</td>
<td>66.86 (2.59)</td>
<td>79.95 (1.78)</td>
</tr>
</tbody>
</table>

### Table 6.10: Performance on Noisy MNIST dataset, 60% training data, 40% testing data.

<table>
<thead>
<tr>
<th># cannot-link</th>
<th>NMI</th>
<th>ARI</th>
<th>F1 score</th>
<th>Purity</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (baseline)</td>
<td>74.22 (0.72)</td>
<td>61.94 (2.66)</td>
<td>65.90 (2.28)</td>
<td>79.59 (3.08)</td>
</tr>
<tr>
<td>100</td>
<td>74.77 (0.69)</td>
<td>60.13 (1.44)</td>
<td>64.40 (1.25)</td>
<td>78.76 (1.61)</td>
</tr>
<tr>
<td>300</td>
<td>76.11 (1.40)</td>
<td>62.73 (2.69)</td>
<td>66.62 (2.48)</td>
<td>80.46 (2.38)</td>
</tr>
<tr>
<td>500</td>
<td>75.55 (0.91)</td>
<td>62.66 (1.59)</td>
<td>66.53 (1.40)</td>
<td>80.31 (2.21)</td>
</tr>
</tbody>
</table>

### Table 6.11: Performance on Noisy MNIST dataset, 70% training data, 30% testing data.

<table>
<thead>
<tr>
<th># cannot-link</th>
<th>NMI</th>
<th>ARI</th>
<th>F1 score</th>
<th>Purity</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (baseline)</td>
<td>74.22 (0.72)</td>
<td>61.94 (2.66)</td>
<td>65.90 (2.28)</td>
<td>79.59 (3.08)</td>
</tr>
<tr>
<td>100</td>
<td>75.20 (0.60)</td>
<td>63.86 (2.31)</td>
<td>67.62 (2.02)</td>
<td>81.28 (2.33)</td>
</tr>
<tr>
<td>300</td>
<td>75.04 (1.48)</td>
<td>62.04 (2.66)</td>
<td>66.05 (2.32)</td>
<td>80.10 (2.32)</td>
</tr>
<tr>
<td>500</td>
<td>75.95 (1.12)</td>
<td>62.74 (3.05)</td>
<td>66.63 (2.71)</td>
<td>80.30 (2.79)</td>
</tr>
</tbody>
</table>

Table 6.8: Performance on Noisy MNIST dataset, 70% training data, 30% testing data.

Table 6.9: Performance on Noisy MNIST dataset, 50% training data, 50% testing data.

Table 6.10: Performance on Noisy MNIST dataset, 60% training data, 40% testing data.

Table 6.11: Performance on Noisy MNIST dataset, 70% training data, 30% testing data.
Chapter 7

Conclusion

In this work, we present LSGMC, an improved correlation based multi-view clustering method which explores first order proximity within each view and complementary information across views. The proposed approach does so by using a unified graph based on local similarity graphs from each individual view. The use of informative local similarities as a powerful learning signal in unsupervised methods is under-explored in multi-view clustering, and LSGMC unifies this information with correlation-based representation learning. LSGMC also allows for flexibility in incorporating pairwise constraints and can thus be easily extended to semi-supervised clustering. Results from experiments presented in this paper suggest that LSGMC outperforms a large number of existing state-of-art multi-view clustering approaches on both image, text and acoustic-articulatory datasets. Results from experiments further show that LSGMC is able to incorporate pairwise information to improve the learnt data representation when such information is available.

One direction for future works is that LSGMC is currently unable to handle data with more than two views because CCA can only deal with two views. We might want to try out different regularization techniques between views and see how LSGMC performs on data with more than two views. Another direction for future works is that LSGMC cannot handle cannot-link constraints if the links are not in local similarity graphs in semi-supervised learning. We might want to add additional regularization to force margins between cannot-link constraints.
Bibliography


[10] Guoqing Chao, Shiliang Sun, and Jinbo Bi. A survey on multi-view clustering. 2018. 1.1, 2.2

[11] Sharon Fogel, Hadar Averbuch-Elor, Jacob Goldberger, and Daniel Cohen-Or. Clustering-driven deep embedding with pairwise constraints. 2018. 1.1, 2.1, 2.4, 3.4, 4.1

[12] Hongchang Gao, Feiping Nie, Xuelong Li, and Heng Huang. Multi-view subspace cluster-
ing. In 2015 IEEE International Conference on Computer Vision (ICCV), pages 4238–4246, Dec 2015.  2.3


[18] Jialu Liu, Chi Wang, Jing Gao, and Jiawei Han. Multi-view clustering via joint nonnegative matrix factorization. In the 2013 SIAM International Conference on Data Mining, 2013.  2.3


[21] Lucińśka Malgorzata and Slawomir Wierzchoń. Spectral clustering based on k-nearest neighbor graph. 11th International Conference on Computer Information Systems and Industrial Management (CISIM), Sep 2012, Venice, Italy., pages 254–265, 2012.  1.1  2.4


