ABSTRACT

Clustering is a task that divides objects into groups based on the similarity between objects. It is usually used as a tool for exploratory knowledge discovery, i.e., it is used to extract potentially useful and previously unknown knowledge from data, before experts have any insight. Because of the exploratory nature of clustering tasks, it is usually not adequate to simply provide clustering results that separate samples into groups. The domain scientists or data analysts in general also want to gain insight into the data. Therefore, it is desired to develop interpretable clustering models, which help the experts to attain deeper knowledge, by understanding what characterizes a cluster and how a cluster is distinguished from others.

This dissertation focuses on improving interpretability of clustering algorithms by targeting the following three aspects:

1) Clustering with interpretable rules. One possible strategy to improve interpretability is to describe the clusters using interpretable rules. This dissertation first introduces a model that defines each cluster using rectangular decision rules with all features. Based on this model, a generative model and a discriminative model are developed to incorporate feature selection, which use a subset of features to define each cluster.

2) Interpretable clustering with similarity matrices. Similarity-matrix-based clustering methods are usually less interpretable. This dissertation introduces a clustering model that improves interpretability of similarity-matrix-based clustering methods. This model generates a set of interpretable rules for each cluster, using a subset of selected features in a feature matrix; and at the same time, it forces the clustering results to be consistent with the observed similarity matrix.

3) Interpretable crowdclustering with partition labels. Most existing crowdclustering methods analyze pairwise similarity labels provided by different experts, without explaining how different expert solutions are related. This dissertation presents a crowdclustering method that analyzes the partition labels from experts. This model explicitly learns the relationship between the latent consensus cluster solution and each expert solution, revealing the agreements and disagreements across different experts.

The methods introduced in this dissertation are applied to discover subtypes of a heterogeneous lung disease, called Chronic Obstructive Pulmonary Disease (COPD).
ACKNOWLEDGMENTS

I would like to thank my PhD advisor Professor Jennifer Dy. I appreciate her support, guidance, advice, effort, patience, and encouragement. I was enlightened a lot by her wisdom and the ways she solves problems.

I appreciate my PhD committee members Professor Dana Brooks, and Professor Stratis Ioannidis, for their valuable suggestions and generous support.

I am thankful for my collaborators from the Brigham and Women’s Hospital, Dr. Peter Castaldi, Dr. Michael Cho, Dr. Brian Hobbs, Dr. James Ross, and Dr. Edwin Silverman. They deepened my understanding about the COPD clinical data and assisted me in developing machine learning models for the COPD project.

I would like to express my gratitude to members in the Machine Learning group at Northeastern University, including Sindhu Ghanta, Jamshid Sourati, Sarah Brown, Yale Chang, Yumin Liu, Zulqarnain Khan, Alican Bozkurt, Chieh Wu, Aria Masoomi and Bahar Azari. I have learned a lot through discussing my research with them.

I wish to thank my parents for their love and encouragement, without whom I would never have enjoyed so many opportunities.

My PhD studies are partially supported by the NIH grant NIH/NHLBI RO1HL089856, RO1HL089857 and NSF/IIS-1546428.
# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ABSTRACT</strong></td>
<td>i</td>
</tr>
<tr>
<td><strong>List of Figures</strong></td>
<td>vi</td>
</tr>
<tr>
<td><strong>List of Tables</strong></td>
<td>viii</td>
</tr>
</tbody>
</table>

### 1. Introduction
1.1 Clustering with Interpretable Rules 2
1.2 Interpretable Clustering with Similarity Matrices 5
1.3 Interpretable Crowdclustering 6
1.4 Outline of the Dissertation 6

### 2. Related Work
2.1 Interpretable Clustering 8
2.2 Unsupervised Feature Selection 9
2.3 Similarity-Matrix-Based Clustering 10
2.4 Crowdclustering 11

### 3. Discriminative Rectangle Mixture Model (DReaM)
3.1 Rule-Generating Features and Cluster-Preserving Features 12
3.2 The DReaM Model 13
3.2.1 Decision Boundaries with Prior Knowledge 14
3.2.2 Soft Clustering Indicator 15
3.2.3 Clustering with Soft Decision Rectangles 16
3.2.4 Generating the Cluster-Preserving Features 17
3.2.5 The Overall Model of DReaM 17
3.3 Training DReaM with Expectation Maximization 18
3.4 Experiments with DReaM 22
3.4.1 Testing DReaM with Synthetic Data 22
3.4.2 Testing DReaM with Benchmark Data 24
3.4.3 Applying DReaM on COPD Subtyping Problem 25

### 4. Interpretable Clustering with Feature Selection
4.1 A Generative Interpretable Clustering Model with Feature Selection (GICMFS) 30
4.1.1 The GICMFS model 30
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1.1.1</td>
<td>Clustering with Generative Rules</td>
<td>31</td>
</tr>
<tr>
<td>4.1.1.2</td>
<td>Axis Parallel Cuts by Partitioning on Features</td>
<td>34</td>
</tr>
<tr>
<td>4.1.1.3</td>
<td>The Overall Model of GICMFS</td>
<td>36</td>
</tr>
<tr>
<td>4.1.2</td>
<td>Training GICMFS via Variational Inference</td>
<td>36</td>
</tr>
<tr>
<td>4.1.3</td>
<td>Prediction with Generative Rules</td>
<td>40</td>
</tr>
<tr>
<td>4.1.3.1</td>
<td>Generating the Rules</td>
<td>40</td>
</tr>
<tr>
<td>4.1.3.2</td>
<td>Clustering with Generated Rules</td>
<td>41</td>
</tr>
<tr>
<td>4.1.4</td>
<td>Experiments with GICMFS</td>
<td>42</td>
</tr>
<tr>
<td>4.1.4.1</td>
<td>Testing GICMFS with Synthetic Data</td>
<td>42</td>
</tr>
<tr>
<td>4.1.4.2</td>
<td>Testing GICMFS with Benchmark Data</td>
<td>43</td>
</tr>
<tr>
<td>4.2</td>
<td>DReaM Model with Feature Selection (DReaM-FS)</td>
<td>49</td>
</tr>
<tr>
<td>4.2.1</td>
<td>The DReaM-FS Model</td>
<td>49</td>
</tr>
<tr>
<td>4.2.1.1</td>
<td>Defining the Discriminative Rules</td>
<td>50</td>
</tr>
<tr>
<td>4.2.1.2</td>
<td>Clustering and Selecting Features with Discriminative Rules</td>
<td>51</td>
</tr>
<tr>
<td>4.2.1.3</td>
<td>Determining Whether a Rule is Satisfied</td>
<td>53</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Training DReaM-FS with Expectation Maximization</td>
<td>54</td>
</tr>
<tr>
<td>4.2.3</td>
<td>Clustering with Discriminative Rules</td>
<td>57</td>
</tr>
<tr>
<td>4.2.4</td>
<td>Experiments with DReaM-FS</td>
<td>59</td>
</tr>
<tr>
<td>4.2.4.1</td>
<td>Testing DReaM-FS with Benchmark Data</td>
<td>59</td>
</tr>
<tr>
<td>4.2.4.2</td>
<td>Applying DReaM-FS on COPD Subtyping Problem</td>
<td>61</td>
</tr>
<tr>
<td>4.2.4.3</td>
<td>Sensitivity Test for DReaM-FS</td>
<td>65</td>
</tr>
<tr>
<td>5</td>
<td>Interpretable Clustering with Similarity Matrices (ICSM)</td>
<td>67</td>
</tr>
<tr>
<td>5.1</td>
<td>The ICSM model</td>
<td>68</td>
</tr>
<tr>
<td>5.1.1</td>
<td>A Factor Graph Model</td>
<td>68</td>
</tr>
<tr>
<td>5.1.2</td>
<td>Defining Rules with the Feature Matrix</td>
<td>68</td>
</tr>
<tr>
<td>5.1.3</td>
<td>Determining Whether a Rule is Satisfied via Soft Thresholding</td>
<td>69</td>
</tr>
<tr>
<td>5.1.4</td>
<td>Clustering and Feature Selection with Rules</td>
<td>71</td>
</tr>
<tr>
<td>5.1.5</td>
<td>Clustering with Similarity Matrices</td>
<td>72</td>
</tr>
<tr>
<td>5.2</td>
<td>Training ICSM with Expectation Maximization</td>
<td>73</td>
</tr>
<tr>
<td>5.3</td>
<td>Experiments with ICSM</td>
<td>75</td>
</tr>
<tr>
<td>5.3.1</td>
<td>Testing ICSM with Benchmark Data</td>
<td>75</td>
</tr>
<tr>
<td>5.3.2</td>
<td>Applying ICSM on COPD Subtyping Problem</td>
<td>79</td>
</tr>
<tr>
<td>5.3.3</td>
<td>Sensitivity Test for ICSM</td>
<td>82</td>
</tr>
<tr>
<td>6</td>
<td>An Interpretable Crowdclustering Model with Partition Labels (ICMPL)</td>
<td>84</td>
</tr>
<tr>
<td>6.1</td>
<td>The ICMPL Model</td>
<td>84</td>
</tr>
<tr>
<td>6.1.1</td>
<td>Modified Multinomial Logistic Regression</td>
<td>85</td>
</tr>
<tr>
<td>6.1.2</td>
<td>Determining the Number of Clusters</td>
<td>87</td>
</tr>
<tr>
<td>6.1.3</td>
<td>Generating Expert Labels</td>
<td>88</td>
</tr>
<tr>
<td>6.1.4</td>
<td>The Overall Model of ICMPL</td>
<td>89</td>
</tr>
<tr>
<td>6.2</td>
<td>Training ICMLP via Variational Inference</td>
<td>90</td>
</tr>
<tr>
<td>6.3</td>
<td>Experiments with ICMPL</td>
<td>92</td>
</tr>
<tr>
<td>6.3.1</td>
<td>Testing ICMPL with Benchmark Data</td>
<td>93</td>
</tr>
<tr>
<td>6.3.2</td>
<td>Applying ICMPL on COPD Subtyping Problem</td>
<td>96</td>
</tr>
</tbody>
</table>
## Contents

7 Conclusions 100

A Variational Inference 102

B Benchmark Datasets 105

C Evaluation of Clustering 106
   C.1 Normalized Mutual Information (NMI) 106
   C.2 Adjusted Rand Index (ARI) 107

Bibliography 108
List of Figures

1.1 Gaussian Mixture Model (GMM) defines clusters using mean and covariance parameters. The clustering result is less interpretable because GMM generates complicated quadratic decision boundaries. In practice, domain experts separate subjects into categories based on rectangular rules. . . . 3

3.1 I plot the step function $f(t)$ and the logistic function $g(t)$ with different steepness parameter $a$. As $a \to \infty$, the soft-thresholding function $g(t)$ becomes closer to the step function $f(t)$. ........................................... 16

3.2 The graphical model for DReaM. The dots represent the hyper-parameters. The regular circle represent latent random variables. The shaded circles represent observed random variables. The arrows represent the dependency between hyper-parameters and random variables. Each plate denotes that the structure inside the plate is repeated. .................................................. 18

3.3 Clustering the synthetic data into two clusters using DReaM. .......... 23

3.4 Plots of the clustering result in the iris dataset. The rectangles represent the rules discovered by DReaM. ................................. 25

3.5 Applying GOLD criterion and the rules discovered to the COPD test data. The rectangles represent the decision rules. .................. 26

4.1 Consider an arbitrary 3-component one-dimensional GMM. I plot the pdf of the 3 components above the axis. The GMM partitions the axis into 3 parts. The decision boundaries of the partition are given by the intersection points of the functions $\{\omega_l \cdot N(x^n; \mu_l, \sigma^2_l)\}_{l=1}^3$. ................................. 35

4.2 The graphical model of the Generative Interpretable Clustering Model with Feature Selection (GICMFS). ................................. 36

4.3 Plots of clustering results of GICMFS on synthetic data. Different clusters are plotted using different color and shape. The rules are represented using lines with different colors, where the upper bounds are plotted using dash lines and the lower bounds are plotted using solid lines. ................................. 43

4.4 Plots of rules with the feature 7 and 10 for the wine data set. The rules are represented using lines with different colors, where the upper bounds are plotted using dash lines and the lower bounds are plotted using solid lines. ................................. 46

4.5 The features selected by GICMFS and the LFS. The selected features are plotted in black. ................................................................. 47

4.6 Plots for the soft-thresholding function $g(x_{nd}, t_{kd})$ with different parameter $a$, where I let $t_{kd} = [-3, 3]$. As $a \to \infty$, $g(x_{nd}, t_{kd}) \to f(x_{nd}, t_{kd})$. . . 53

4.7 Rules discovered by DReaM-FS for the wine data set. ................................. 61

4.8 Plots of rules discovered by our method on COPD subtyping problem. . 65
4.9 Sensitivity test on the wine data set with 5-fold cross validation. The square represents the mean and the error bar represents the standard deviation.

5.1 The factor graph of the ICSM. The square represents the factors, the normal circles represent the latent variables and the shaded circles represent the observed variables. A factor and a variable are connected, only if this factor is a function of this variable.

5.2 Rules discovered by ICSM for the wine data set.

5.3 Rules discovered by ICSM for COPD Subtyping Problem.

5.4 Sensitivity test for ICSM on the wine data set with 5-fold cross validation. The square represents the mean and the error bar represents the standard deviation.

6.1 The problem setup. ICMPL analyzes both features and partition cluster labels from multiple experts. ICMPL simultaneously determines the number of clusters, partitions samples into clusters, and learns the mapping between the learned clusters and the expert labels.

6.2 The graphical model for interpretable crowdclustering.

6.3 Normalized Mutual Information (NMI) in training and validation sets.
### List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>The GOLD Spirometry Categories [1, 2]</td>
<td>3</td>
</tr>
<tr>
<td>3.1</td>
<td>Synthetic Features</td>
<td>23</td>
</tr>
<tr>
<td>3.2</td>
<td>DReaM Results on Benchmark Data</td>
<td>25</td>
</tr>
<tr>
<td>3.3</td>
<td>Rules Discovered by DReaM for COPD Subtyping Problem</td>
<td>26</td>
</tr>
<tr>
<td>3.4</td>
<td>Results on COPD Subtyping Problem</td>
<td>27</td>
</tr>
<tr>
<td>4.1</td>
<td>Clustering Performance of GICMFS on Benchmark Data</td>
<td>45</td>
</tr>
<tr>
<td>4.2</td>
<td>Number of Features Selected by GICMFS for Each Cluster</td>
<td>46</td>
</tr>
<tr>
<td>4.3</td>
<td>Rules Discovered by GICMFS for the Wine Dataset</td>
<td>46</td>
</tr>
<tr>
<td>4.4</td>
<td>Clustering Performance of DReaM-FS on Benchmark Data</td>
<td>60</td>
</tr>
<tr>
<td>4.5</td>
<td>Rules Discovered by DReaM-FS for the Wine Dataset</td>
<td>60</td>
</tr>
<tr>
<td>4.6</td>
<td>Rules Discovered by DReaM-FS for COPD Subtyping Problem</td>
<td>62</td>
</tr>
<tr>
<td>4.7</td>
<td>Cluster-Preserving Features for Different Clusters</td>
<td>64</td>
</tr>
<tr>
<td>4.8</td>
<td>p-values in Kruskal-Wallis Test</td>
<td>64</td>
</tr>
<tr>
<td>5.1</td>
<td>Important Notations</td>
<td>69</td>
</tr>
<tr>
<td>5.2</td>
<td>Rules Discovered by ICSM for the Wine Dataset</td>
<td>78</td>
</tr>
<tr>
<td>5.3</td>
<td>Performance of ICSM on Benchmark Data</td>
<td>78</td>
</tr>
<tr>
<td>5.4</td>
<td>Rules discovered by ICSM for COPD Subtyping Problem</td>
<td>80</td>
</tr>
<tr>
<td>5.5</td>
<td>p-values in Kruskal-Wallis Test</td>
<td>81</td>
</tr>
<tr>
<td>6.1</td>
<td>Number of Clusters Discovered by ICMPL</td>
<td>96</td>
</tr>
<tr>
<td>6.2</td>
<td>p-values in Kruskal-Wallis test</td>
<td>98</td>
</tr>
<tr>
<td>6.3</td>
<td>Clustering Results of ICMPL on COPD Subtyping Problem</td>
<td>98</td>
</tr>
<tr>
<td>6.4</td>
<td>Mapping between Clusters and Expert Labels</td>
<td>98</td>
</tr>
<tr>
<td>B.1</td>
<td>Statistics of Benchmark Datasets</td>
<td>105</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

Clustering is a task that divides objects into groups based on the similarity between objects. It is usually used to extract potentially useful and previously unknown knowledge from data, before experts have any insight. Because of its exploratory nature, it is usually not adequate to simply provide clustering results that separate samples into groups. The domain scientists or data analysts in general want to gain insight into the data. Observing these requirements, this dissertation focuses on introducing interpretable clustering models, which help the experts to attain deeper knowledge, by understanding what characterizes a cluster and how a cluster is distinguished from others.

The models introduced in this dissertation are motivated by a real-world cluster discovery problem, i.e., Chronic Obstructive Pulmonary Disease (COPD) subtyping problem. COPD is characterized by chronic, progressive, and irreversible lung airflow obstruction. It is predicted to be the third leading cause of death worldwide by the year of 2020 [1]. COPD is a clinically heterogeneous disease that may be partitioned into multiple subtypes (clusters) relevant to disease prognosis and treatment. The purpose of the project is to stratify COPD patients into several subtypes by analyzing patient data. Meaningful analysis of the data has the potential to lead to a better understanding of the disease and therefore, better treatment for patients. The data available in this project includes spirometry, symptom, CT imaging, RNA sequencing, and Genome-Wide Association Study (GWAS) data for heavy smokers collected during multiple visits, as well as clustering labels provided by several clinicians, radiologists, and data scientists. This
dissertation introduces clustering models that generate interpretable clustering rules for the COPD dataset under several problem settings and develops an interpretable crowd-clustering model that helps to find and understand consensus clustering solutions based on partition labels provided by multiple domain experts.

This dissertation focuses on improving interpretability for clustering models by targeting the following three aspects: (1) Clustering with interpretable rules. (2) Interpretable clustering with similarity matrices. (3) Interpretable crowd-clustering with partition labels.

1.1 Clustering with Interpretable Rules

As mentioned earlier, interpretability of clustering models helps domain scientists or data analysts gain insight into the data. However, traditional methods such as k-means [3] and Gaussian Mixture Model (GMM) [4] are not sufficiently interpretable, because they define clusters using parameters such as means and covariances. As shown in Figure 1.1a, with these parameters, it is difficult to determine which cluster a given test sample belongs to by intuition. One has to compare either the Euclidean or Mahalanobis distances to the centroid of each cluster. It is difficult to understand how the decision was made and what are the relevant features in the decision making, especially when dimensionality is high.

Rather than representing clusters with parameters, domain experts usually separate subjects into categories based on rules in practice. Take the COPD staging criterion as an example. As shown in Figure 1.1b and Table 1.1, the Global initiative for Chronic obstructive Lung Disease (GOLD) spirometry staging criterion[1, 2] separates COPD patients into stages, based on two spirometry measures: (1) FEV1 (forced expiratory volume in 1 second) % of predicted normal (based on age, sex, height, and race), and (2) FEV1 to forced vital capacity ratio (FEV1/FVC). The decision rules associated with these variables are clearly stated in Table 1.1. We consider such guideline more interpretable because the doctor can quickly classify patients into categories by simply observing the relevant features and the corresponding decision boundaries, without complex computation. At the same time, it clearly defines what characterizes one cluster and how one cluster is distinguished from others.
Introduction

(a) The decision boundaries generated by GMM.

(b) The GOLD Criterion [1, 2] that is used to classify COPD patients in practice.

Figure 1.1 Gaussian Mixture Model (GMM) defines clusters using mean and covariance parameters. The clustering result is less interpretable because GMM generates complicated quadratic decision boundaries. In practice, domain experts separate subjects into categories based on rectangular rules.

Table 1.1: The GOLD Spirometry Categories [1, 2]

<table>
<thead>
<tr>
<th>GOLD 0</th>
<th>Healthy</th>
<th>FEV₁/FVC ≥ 0.7 &amp; FEV₁ ≥ 80% predicted</th>
</tr>
</thead>
<tbody>
<tr>
<td>GOLD 1</td>
<td>Mild</td>
<td>FEV₁/FVC &lt; 0.7 &amp; FEV₁ ≥ 80% predicted</td>
</tr>
<tr>
<td>GOLD 2</td>
<td>Moderate</td>
<td>FEV₁/FVC &lt; 0.7 &amp; 50% &lt; FEV₁ &lt; 80% predicted</td>
</tr>
<tr>
<td>GOLD 3</td>
<td>Severe</td>
<td>FEV₁/FVC &lt; 0.7 &amp; 30% &lt; FEV₁ &lt; 50% predicted</td>
</tr>
<tr>
<td>GOLD 4</td>
<td>Very severe</td>
<td>FEV₁/FVC &lt; 0.7 &amp; FEV₁ &lt; 30% predicted</td>
</tr>
</tbody>
</table>

The challenge is to develop interpretable clustering models, which is able to generate a list of rules that is similar to Figure 1.1b and Table 1.1. We want the model to automatically select the relevant features, build the axis-parallel cuts for the selected features, and generate a list of rules for each of the clusters based on a conjunction of these axis-parallel cuts. This dissertation presents three solutions to this challenge in Chapter 3, 4.1, and 4.2, respectively.

In Chapter 3, I first present Discriminative Rectangle Mixture (DReaM) model. Motivated by the COPD dataset, I separate the features into two sets: a set of features that are used to construct rules, I call rule-generating features, and a set of features that present preserve interesting clustering structures, I call cluster-preserving features. In CDPD subtyping problem, we want the patients to be assigned into different groups, based on sign and symptoms (e.g., spirometry measures and CT images), but we may be interested in finding a clustering structure involving expected future health decline of the patients. We do not want the clustering rules to include the future health decline, because although it is available in the dataset, it is not-yet available when the diagnosis
is made. DReaM constructs interpretable rules using the rule-generating features by applying logistic-function-based soft thresholds to these feature and preserves clustering structures by assuming that the cluster-preserving features are generated from a mixture of distributions whose components belong to the exponential family.

Although DReaM is able to generate interpretable rules that describe each cluster, it does not automatically select features to define the rules and always uses all rule-generating features. However, in real applications, some features might be irrelevant and redundant. Removing irrelevant and redundant features via feature selection may improve the performance of clustering algorithm and it also makes the results more interpretable, because the complexity of the results is reduced.

Observing the limitations of DReaM, I introduce a Generative Interpretable Clustering Model with Feature Selection (GICMFS) in Chapter 4.1. This model automatically generates a list of rules that might involve different sets of features across different clusters. If a sample belongs to one of the clusters, then the rules in the corresponding list tend to be satisfied. The axis-parallel cut rules are based on partitioning features into several parts via one-dimensional Gaussian mixture models. Whether one feature is selected in the rule list for one cluster is determined based on whether it characterizes the cluster. Unlike DReaM, this model does not separate features into rule-generating features and cluster-preserving features.

GICMFS incorporates interpretability and feature selection. However, it is not straightforward to apply this model to the COPD dataset, because it does not incorporate the flexibility that rule-generating features can be different from the cluster-preserving features. Therefore, I develop a discriminative interpretable clustering model with feature selection in Chapter 4.2. I call this model DReaM-FS because it is a Feature Selection (FS) version of DReaM that selects a subset of cluster-preserving features to define each cluster. It allows the features selected to be different across clusters. In the model, if a sample satisfies more rules defined for one certain cluster compared to other clusters, then it is more likely that this sample belongs to this cluster. If a rule associated with a certain feature better distinguishes one certain cluster from the rest than random, then it is more likely that this feature is selected for this cluster.
1.2 Interpretable Clustering with Similarity Matrices

In Chapter 1.1, I focus on discussing the case that the data is provided in the form of a feature matrix. However, in some applications, a similarity matrix, which describes the data using an assessment of relative similarity between each pair of samples, is provided instead. Spectral clustering [5–7] and kernel k-means [8] are the most well-known similarity-matrix-based methods, which analyze similarity matrices, and give clustering results. Unlike methods such as k-means [3] and Gaussian Mixture Model (GMM) [9] that are able to describe each cluster with parameters such as mean and covariance, limited information that characterizes each cluster is provided by similarity-matrix-based methods. Therefore, it is usually more difficult to interpret the clustering result.

This dissertation presents a model that achieves Interpretable Clustering with Similarity Matrices (ICSM). The model is motivated by the COPD subtyping dataset that includes a feature matrix, which describes the phenotypes, and genetic variants, which can be used to generate a Genetic Relationship Matrix (GRM) that describes the genetic similarity between subjects. The challenge is to divide the subjects into subtypes (clusters), that can take into account multi-model information of both the feature matrix (phenotypes) and the similarity matrix (GRM). In particular, as an exploratory data analysis problem, we want the results to be interpretable, such that we are able to understand the clustering results.

The ICSM model assumes that both a feature matrix and a similarity matrix are simultaneously provided as the input. This model generates a set of interpretable rules for each cluster. Similar to GICMFS in Chapter 4.2, these rules are represented using lower and upper bounds associated with a subset of selected features in the feature matrix. ICSM determines which cluster a sample belongs to, by observing whether rules for each cluster are satisfied or not through soft thresholding. At the same time, it forces the clustering results to be consistent to the observed similarity.
1.3 Interpretable Crowdclustering

In Chapter 1.1 and 1.2, this dissertation focuses on clustering methods that generate interpretable rules, where a feature matrix and/or a similarity matrix is provided. As mentioned earlier, the COPD dataset also contains clustering labels provided by multiple domain experts. However, these domain experts do not completely agree with each other. Therefore, it is desired to combine the opinions from these experts into one consensus clustering solution. Methods that find such consensus is known as crowdclustering [10–14].

Although existing crowdclustering methods are able to generate a consensus clustering solution, most of them are based on pairwise similarity labels, indicating whether a pair of samples are similar and should belong to the same cluster or not. However, in the case of COPD dataset, partition clustering labels are available. If we convert partition labels into pairwise similarity lables, then it would be difficult to understand the agreement and disagreement between experts, and how the consensus clusters learned are related to the observed expert labels. Since COPD subtyping is an exploratory discovery task, it is not only desired to find the consensus between the experts, it is also important to understand the relationship between the consensus result and the clustering solution provided by each expert.

Observing the limitations of existing methods, this dissertation introduces an Interpretable Crowdclustering Model with Partition Labels (ICMPL) in Chapter 6. ICMPL analyzes the partition labels provided by the experts, rather than pairwise similarity labels. It adopts a novel approach based on a modified multinomial logistic regression model, which simultaneously learns the number of clusters and determines hyper-planes that partition samples into clusters. It explicitly learns a mapping from the latent consensus clusters to each expert label, revealing the agreements and disagreements between labels from different experts.

1.4 Outline of the Dissertation

The rest of this dissertation is organized as follows. In Chapter 2, I first introduce the related work for the models introduced. The related topics include interpretable clustering,
unsupervised feature selection, similarity-matrix-based clustering and crowdclustering. In Chapter 3, I present Discriminative Rectangle Mixture (DReaM) model. In Chapter 4, I introduce interpretable clustering models with feature selection. Two models are introduced in this chapter. In Chapter 4.1, I first introduced a Generative Interpretable Clustering Model with Feature Selection (GICMFS). Then I introduced DReaM Model with Feature Selection (DReaM-FS) in Chapter 4.2. In Chapter 5, I introduce a model for Interpretable Clustering with Similarity Matrices (ICSM). In Chapter 6, I introduce an Interpretable Crowdclustering Model with Partition Labels (ICMPL). I conclude this dissertation in Chapter 7.
Chapter 2

Related Work

In this section, I review the literature related to the dissertation. The dissertation involves the topics of interpretable clustering, unsupervised feature selection, similarity-matrix-based clustering and crowdclustering. The related work with each of these topics is introduced in the following sections.

2.1 Interpretable Clustering

Interpretable clustering models refer to clustering models that explain what characterizes a cluster and how a cluster is distinguished from others.

Some effort have been made to develop interpretable methods for supervised problems. Classification and Regression Trees (CART) [15] was one of the earliest interpretable methods to solve classification and regression problems. In this model, a decision tree is induced by recursively finding one optimal feature and the corresponding threshold that maximizes the information gain or other criteria in a greedy way. Goh and Rudin [16] proposed a box drawing learning method for imbalanced classification problems, by solving a mixed integer programming. Letham et al. [17] and [18] proposed a Bayesian model that finds an interpretable rule list. All these models are interpretable supervised methods; in contrast, this dissertation focuses on solving an unsupervised clustering problem.

There have been interpretable methods that solve the clustering problem by constructing decision trees. Liu et al. [19] proposed to construct decision trees that partition the data
space into clusters and sparse regions by adding artificial samples that are uniformly distributed in space. Fraiman et al. [20] developed a method that firstly separates the sample space using a decision tree by greedily maximizing a heterogeneity measure. Then the nodes in the decision tree are pruned and joined according to a dissimilarity measure. These models construct decision trees in a greedy manner. This dissertation introduces probabilistic models that find decision rules for each cluster.

Pelleg and Moore [21] proposed a probabilistic generative model that assumes a mixture of tailed rectangular distributions. This model does not involve feature selection. GICMFS introduced in Chapter 4.1 is also a generative model, but incorporates feature selection. DReaM in Chapter 3 and DReaM-FS in Chapter 4.2 are probabilistic discriminative model. These two models introduce additional flexibility that the rule-generating features might be different from the cluster-preserving features. Such flexibility facilitates us to apply them to solve the COPD subtyping problem.

Kim et al. [22] proposed an interpretable clustering model that describes each cluster using logical formula. This model focuses on binary-valued data but can not automatically determine the threshold for continuous features. This dissertation focuses on clustering continuous data and the methods introduced are able to automatically find thresholds for continuous features.

Gibert and Cortés [23] proposed a method that makes use of expert knowledge to discover the rules for clustering. In [23], the expert interactively updates the rules in each iterative step, which are used as constraints for hierarchical clustering. In this method, the rules are generated and updated by an expert, and the clustering method is used as a tool to validate and confirm the rules discovered. In contrast, methods introduced in this dissertation automatically learns the rules without expert’s interaction.

2.2 Unsupervised Feature Selection

Feature selection is the task of removing irrelevant and redundant features, where irrelevant features are features that do not reveal cluster structures and redundant features are features that do not add additional clustering information to a certain set of selected features. Feature selection may improve the performance of the clustering algorithm
in high-dimensional data, because it enables removing noisy features and it also makes results more interpretable, because the complexity of the results is reduced.

In general, unsupervised feature selection algorithms can be divided into three categories: filter, wrapper, and embedded methods. Filter methods utilize some properties of the features to decide which features are kept, without running any clustering algorithm. Examples of filter methods include [24–26]. In contrast, wrapper methods wrap feature search around the clustering algorithm. [27–31] are examples of wrapper methods. Embedded methods incorporate feature selection and clustering in a unified algorithm. [32–35] are some examples of embedded methods. In this dissertation, GICMFS, DReaM-FS, and ICSM, described in Chapter 4.1, 4.2 and 5, respectively, are embedded feature selection algorithms. These methods differ from previous embedded methods in that they generate interpretable clustering results, described by hyper-rectangular rules.

2.3 Similarity-Matrix-Based Clustering

Similarity-matrix-based clustering models analyze a similarity matrix, which describes the data using an assessment of relative similarity between each pair of samples.

Spectral clustering [5–7] and kernel k-means [8] are the most well-known similarity-matrix-based clustering models. Because limited information that characterizes each cluster is provided by these methods, it is usually difficult to interpret the clustering results generated by these methods. ICSM described in Chapter 5 focuses on improving the interpretability of similarity-matrix-based clustering models.

One challenge for Similarity-Matrix-Based Clustering algorithms is that it is not straightforward to extend the clustering results to out-of-sample data. Bengio et al. [36] introduces eigenfunctions to generalize the results for out-of-sample data. Alzate and Suykens [37] derives the Least-Squares Support Vector Machine (LS-SVM) formulations to achieve out-of-sample extension. It is not straightforward to understand what characterises one cluster and how the clusters discovered differ from each other, based on these out-of-sample extension methods. In contrast, ICSM gives interpretable clustering results by defining each cluster with rules.
2.4 Crowdclustering

Crowdclustering methods refer to the clustering methods that find a consensus clustering solution by summarizing several clustering solutions provided by multiple experts.

Several crowdclustering methods have been proposed in recent years. [10] and [11] analyse pairwise similarity labels given by the experts without accessing the features of the samples, but simply find a consensus among the experts. [12–14] make use of both the pairwise similarity labels and the sample features to generate a clustering solution. Unlike these methods that analyze the pairwise similarity labels, ICMPL introduced in Chapter 5 directly observes the partition cluster labels. Since ICMPL learns a mapping between the clusters and the expert labels, it reveals the agreement and disagreement between experts.

Clustering ensemble [38–43] methods are also closely related to crowdclustering, since they involve generating a consensus clustering result from several clustering solutions. Unlike crowdclustering methods, where clustering solutions are provided by domain experts, clustering ensemble methods analyze the clustering solutions generated by basic clustering models such as k-means. Although clustering ensemble techniques can also be used to generate consensus results based on the expert labels; these methods have difficulties in out-of-sample prediction, because the expert labels are often not available for test samples. ICMPL learns a linear discriminative model and can be used to predict cluster memberships on test samples based on the features.
Chapter 3

Discriminative Rectangle Mixture Model (DReaM)

As shown in Table 1.1 and Figure 1.1b, the clinical guideline that is used in practice is usually described using rectangular rules. Such guideline is considered more interpretable because the doctors can quickly classify patients into categories by simply observing the relevant features and the corresponding decision boundaries, without complex computation such as computing Euclidean or Mahalanobis distances. The rectangular rules clearly define what characterizes one cluster, and how one cluster is distinguished from others.

The challenge is how to design a clustering method that automatically generates such interpretable rectangular rules. In this section, I introduce the Discriminative Rectangle Mixture (DReaM) model to achieve this purpose.

3.1 Rule-Generating Features and Cluster-Preserving Features

As mentioned above, we want DReaM to describe each cluster using interpretable rectangular rules. The challenge for the COPD subtyping problem is that we want the rules to be constructed based on a subset of features (spirometry measures), but we are interested in the clustering structure according to the other subset of features (future
health decline). We do not want the cluster to be generated using rules features such as future health decline, because such features are not available when the clinical decisions are made.

To solve this problem, I separate the features into two sets: a set of features that are used to construct rules (spirometry measures), called rule-generating features, and a set of features that present preserve interesting clustering structures (future health decline), called cluster-preserving features. DReaM constructs interpretable rules using the rule-generating features by applying logistic-function-based soft thresholds to these feature, and preserves clustering structures by assuming that the cluster-preserving features are generated from a mixture of distributions whose components belong to the exponential family.

Note that unlike supervised methods whose purpose is to predict labels or regression targets; under this setting, we still try to separate samples into groups, as a clustering method does. With DReaM, we let these two sets of features to be identical by default, which is the regular clustering setting (as in k-means and GMM). Separating features into two sets allows the additional flexibility for DReaM that these two sets of features are disjoint or with partially overlapping features.

3.2 The DReaM Model

In this section, I introduce the DReaM model. To begin with, I introduce some notations. I let $N$ represent the number of samples. It is assumed that the features can be divided into subsets of rule-generating features and cluster-preserving features. For the $n$-th sample, I let $x_n \in \mathbb{R}^D$ represent the rule-generating features for sample $n$, and $y_n \in \mathbb{R}^L$ represent the cluster-preserving features for the same sample. I standardize all features such that each feature has zero mean and unit variance.

I let $K$ be the pre-defined integer that represents the number of clusters. I define $z_n$ to be the $K$-dimensional cluster indicator for the $n$-th sample, such that $z_{nk} = 1$ if and only if the $n$-th sample belongs to the cluster $k$. I let the prior distribution of $z_n$ be a uniform categorical distribution.
3.2.1 Decision Boundaries with Prior Knowledge

We would like DReaM to represent each cluster using interpretable rectangular results. Thus, I first introduce how to define the rectangular rules. To achieve this, I use a vector $t_{kd} = [t_{kd}^{-}, t_{kd}^{+}]^T$ to represent the rectangular decision boundaries for cluster $k$ in the $d$-th dimension. I assume that all samples $x_n$ that belong to cluster $k$ are inside the corresponding decision rectangle, satisfying $t_{kd}^{-} \leq x_{nd} \leq t_{kd}^{+}$ for all dimensions $d$.

In particular, as shown in Table 1.1, there is already a set of rules, known as GOLD criteria, defined for the COPD. We might want to make use of GOLD criteria as prior knowledge. To achieve this, I apply a prior distribution to each decision boundary $t_{kd}$, such that

$$p(t_{kd}) \propto \exp\{-\frac{1}{2} \alpha_t (t_{kd}^{-} - \mu_{t_{kd}^{-}})^2 - \frac{1}{2} \alpha_t (t_{kd}^{+} - \mu_{t_{kd}^{+}})^2 - \frac{1}{2} \beta_t (t_{kd}^{+} - t_{kd}^{-})^2\}.$$  \hspace{1cm} (3.1)

In this equation, $\mu_{t_{kd}^{-}}$ and $\mu_{t_{kd}^{+}}$ represent the location where decision boundaries of the prior rules are. When such knowledge is not available, I set $\mu_{t_{kd}^{-}} = \mu_{t_{kd}^{+}} = 0$. The first two exponential terms in the formula enforce the boundaries to be close to the prior domain knowledge. The last exponential term penalizes for large rectangles. $\alpha_t$ and $\beta_t$ are positive parameters that control the tradeoff between these terms.

Note that Equation (3.1) is equivalent to

$$p(t_{kd}) \propto \exp\left(-\frac{1}{2} (t_{kd} - \tilde{\mu}_{t_{kd}})^T \tilde{\Sigma}_{t_{kd}}^{-1} (t_{kd} - \tilde{\mu}_{t_{kd}})\right),$$  \hspace{1cm} (3.2)

where

$$\tilde{\mu}_{t_{kd}} = \begin{bmatrix} (\alpha_t + \beta_t) \mu_{t_{kd}^{-}} + \beta_t \mu_{t_{kd}^{+}} & \beta_t \mu_{t_{kd}^{-}} + (\alpha_t + \beta_t) \mu_{t_{kd}^{+}} \end{bmatrix}^T,$$

$$\tilde{\Sigma}_{t_{kd}}^{-1} = \begin{bmatrix} \alpha_t + \beta_t & -\beta_t \\ -\beta_t & \alpha_t + \beta_t \end{bmatrix}.$$  \hspace{1cm} (3.3)

Therefore, we can simplify and assume that $t_{kd}$ is a sample from a multivariate Gaussian distribution with mean $\tilde{\mu}_{t_{kd}}$, and covariance $\tilde{\Sigma}_{t_{kd}}$, such that $t_{kd} \sim \mathcal{N}(\tilde{\mu}_{t_{kd}}, \tilde{\Sigma}_{t_{kd}}).$
3.2.2 Soft Clustering Indicator

Given the decision rectangles \( \{t_{kd}^-, t_{kd}^+\}_{d=1}^D \), one possible way to represent the cluster indicator \( z_n \) is

\[
z_{nk} = \prod_{d=1}^D f(x_{nd} - t_{kd}^-) f(t_{kd}^+ - x_{nd}),
\]

(3.4)

where \( f(t) \) is the step function defined as

\[
f(t) = \begin{cases} 
1, & \text{if } t \geq 0 \\
0, & \text{otherwise.}
\end{cases}
\]

(3.5)

In Equation (3.4), the step function \( f(t) \) can be considered as an indicator function on whether the inequality \( t \geq 0 \) is satisfied, and the product can be considered as the logic AND operator. Therefore, we have \( z_{nk} = 1 \) if the sample \( x_n \) is located inside the decision rectangle; and \( z_{nk} = 0 \), otherwise.

However, because the step function in Equation (3.5) is not differentiable, it would be difficult to optimize with respect to the decision rectangles \( t_{kd} \). Thus, I propose to approximate the strict step function \( f(t) \) using a differentiable soft-thresholding function \( g(t) \) that is defined as

\[
g(t) = \frac{1}{1 + \exp(-at)},
\]

(3.6)

where \( a \) is a positive parameter that defines the “steepness” of the soft step function, such that a larger \( a \) implies that \( g(t) \) is closer to \( f(t) \). I plot \( g(t) \) with different \( a \) and \( f(t) \) in Fig. 3.1. Note that as \( a \to \infty \), the soft-thresholding function \( g(t) \) becomes closer to the step function \( f(t) \).

Now I replace \( f(\cdot) \) in the right hand side in Equation (3.4) with \( g(\cdot) \), and define a new variable \( \gamma_{nk} \), for each sample \( n \in \{1 \ldots N\} \) and each cluster \( k \in \{1 \ldots K\} \) such that

\[
\gamma_{nk} = \prod_{d=1}^D g(x_{nd} - t_{kd}^-) g(t_{kd}^+ - x_{nd}).
\]

(3.7)

We have \( 0 < \gamma_{nk} < 1 \) because \( 0 < g(t) < 1 \) for all real values \( t \). If the sample \( x_n \) is located inside the decision rectangle for cluster \( k \), we have \( \gamma_{nk} \approx 1 \); while we have \( \gamma_{nk} \approx 0 \), if \( x_n \) is outside the rectangle. \( \gamma_{nk} \) can be considered as a soft cluster indicator. However a cluster indicator \( z_{nk} \) is binary and \( \sum_k z_{nk} = 1 \). Therefore, we need to find a
I plot the step function $f(t)$ and the logistic function $g(t)$ with different steepness parameter $a$. As $a \to \infty$, the soft-thresholding function $g(t)$ becomes closer to the step function $f(t)$.

method to derive the cluster indicator $z_n$ from the soft indicator $\gamma_n = \{\gamma_{nk}\}_{k=1}^K$. This is introduced in the next section.

### 3.2.3 Clustering with Soft Decision Rectangles

Now I introduce how I generate cluster indicator $z_n$, based on the soft-cluster indicator $\gamma_{nk}$. In particular, if the $n$-th sample belongs to cluster $k$, i.e., $z_{nk} = 1$ and $z_{nj} = 0$ for all $j \neq k$, then the following two conditions for $\gamma_n$ need to be satisfied:

1. $x_n$ is located inside the decision rectangle for the $k$-th cluster, i.e., $\gamma_{nk} \approx 1$.
2. $x_n$ is located outside all other decision rectangles, i.e., $\gamma_{nj} \approx 0$, for all $j \neq k$.

I achieve this by introducing additional random variables $\Phi = \{\phi_k\}_{n=1}^N$, such that if these two conditions are satisfied for all $k \in \{1, \ldots, K\}$ with certain $n$, then $\phi_n = 1$; and $\phi_n = 0$, otherwise. I assume that $\phi_n$ follows a Bernoulli distribution such that

$$p(\phi_n = 1|\gamma_n, z_n) = \prod_{j=1}^K \gamma_{nj} z_{nj} (1 - \gamma_{nj})^{1 - z_{nj}}$$

$$p(\phi_n = 0|\gamma_n, z_n) = 1 - p(\phi_n = 1|\gamma_n, z_n).$$

Note that $z_{nk}$ is binary, and $0 < \gamma_{nk} < 1$, for all $k$. Therefore, we have $0 < p(\phi_n = 1|\gamma_n, z_n) < 1$. This implies that Equation (3.8) defines a valid binary distribution.
Since we always want the conditions at the beginning of Chapter 3.2.3 to be satisfied, I let $\phi_{nk}$ always observe to be 1 for all $n \in \{1, \ldots, N\}$ and $k \in \{1, \ldots, K\}$. After introducing this variable, if the $n$-th sample belongs to cluster $k$, i.e. $z_{nk} = 1$ and $z_{nj} = 0$ for all $j \neq k$, then by Bayes’ theorem

$$\log p(\gamma_n | \phi_{nk} = 1, z_{nk} = 1) = \log p(\gamma_n) + \log \gamma_{nk} + \sum_{j \neq k} \log(1 - \gamma_{nj}) + \text{const}, \quad (3.9)$$

where $p(\gamma_n)$ is a prior distribution for $\gamma_n$. Note that $\log \gamma_{nk} \to -\infty$ if $\gamma_{nk}$ is close to 0; and $\log(1 - \gamma_{nj}) \to -\infty$ if $\gamma_{nj}$ is close to 1. This implies that given the same prior distribution $p(\gamma_n)$, the posterior distribution of $\gamma_n$ after observing $\phi_{nk}$ tend to satisfy the conditions introduced in the beginning of Chapter 3.2.3, because $p(\gamma_n | \phi_{nk} = 1, z_{nk} = 1) \approx 0$ if $\gamma_{nk} \approx 0$ or $\gamma_{nj} \approx 1$ for any $j \neq k$.

### 3.2.4 Generating the Cluster-Preserving Features

I assume that $y_n$ contains a clustering structure. Let $y_n$ follow a mixture of distributions whose components belong to the exponential family, where we can choose any proper distributions according to the form of $y_n$ (e.g., Bernoulli distribution if $y_n$ is binary). In the COPD application, we are interested in the case that $y_n$ is continuous. Therefore, I let $y_n$ follow a mixture of multivariate Gaussian distributions, such that

$$y_n | z_n \sim \prod_{k=1}^{K} \mathcal{N}(\mu_k, \Sigma_k)^{z_{nk}}, \quad (3.10)$$

where $\mu_k$ and $\Sigma_k$ are the mean vector and the covariance matrix for the $k$-th component, respectively.

### 3.2.5 The Overall Model of DReaM

I summarize DReaM with a graphical model in Fig. 3.2. The joint probability of DReaM is

$$p(T, Y, Z, \Phi | X, \Theta) = \prod_{k=1}^{K} \prod_{d=1}^{D} p(t_{kd}) \prod_{n=1}^{N} p(z_n) \prod_{n=1}^{N} p(\phi_{n} = 1 | z_n, T, x_n) \prod_{n=1}^{N} p(y_n | z_n, \Theta), \quad (3.11)$$

where I use $T = \{t_k\}_{k=1}^{K}$ to represent all rectangular decision boundaries, and $\Theta = \{\mu_k, \Sigma_k\}_{k=1}^{K}$ to represent all parameters.
### 3.3 Training DReaM with Expectation Maximization

In the section, I introduce an Expectation Maximization (EM) algorithm [32] that finds a Maximum a Posteriori (MAP) estimator. In the EM algorithm, I treat $Z$ as the latent variables and find a MAP estimator for the parameters $T$ and $\Theta$. The objective function is given by

$$\hat{T}, \hat{\Theta} = \arg \max_{T, \Theta} \log p(T, Y, \Phi | X, \Theta)$$

$$= \arg \max_{T, \Theta} \log \sum_Z p(T, Y, Z, \Phi | X, \Theta)$$

$$= \arg \max_{T, \Theta} \log \sum_Z \left\{ \left( \prod_{k=1}^{K} \prod_{d=1}^{D} p(t_{kd} | \alpha_t, \beta_t, \mu_{t_k^d}, \mu_{t_k^d+}) \right) \right\}$$

In the EM algorithm, I iteratively apply the Expectation step (E Step) and Maximization Step (M Step) until convergence.
Expectation Step

In the E Step, I first compute the posterior distribution of the latent variable $Z$, given the parameters $\{T, \Theta\}$ and the observed variables $\{\Phi, X, Y\}$. The log posterior distribution is given by

$$
\log p(Z|T, \Theta, \Phi, X, Y) = \sum_{n=1}^{N} \left\{ \log p(z_n) + \log p(\phi_n|z_n, x_n, T) + \log p(y_n|z_n, \Theta) \right\} + \text{const}
$$

$$
= \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} \left\{ \sum_{d=1}^{D} \log g(x_{nd} - t_{kd}^-) + \log g(t_{kd}^+ - x_{nd}) - \log \left(1 - \prod_{d=1}^{D} g(x_{nd} - t_{kd}^-)g(t_{kd}^+ - x_{nd})\right) - \frac{1}{2} \log |\Sigma_k| - \frac{1}{2} (y_n - \mu_k)^T \Sigma_k^{-1} (y_n - \mu_k) \right\} + \text{const}
$$

(3.13)

where $\text{const}$ represents constants that are not functions of $Z$ and normalize this equation such that it defines a valid probability. By observing this Equation, we can conclude that the posterior distribution for each $z_n$ with $n \in \{1 \ldots N\}$ is conditional independent with each other, and is given as

$$
z_n|x_n, \phi_n, y_n, T, \Theta \sim \text{Categorical}(\pi_n)
$$

(3.14)

where $\pi_n$ is a $K$-dimensional vector, each of whose element is defined by

$$
\pi_{nk} \propto \exp \left\{ \sum_{d=1}^{D} \log g(x_{nd} - t_{kd}^-) + \log g(t_{kd}^+ - x_{nd}) - \log \left(1 - \prod_{d=1}^{D} g(x_{nd} - t_{kd}^-)g(t_{kd}^+ - x_{nd})\right) - \frac{1}{2} \log |\Sigma_k| - \frac{1}{2} (y_n - \mu_k)^T \Sigma_k^{-1} (y_n - \mu_k) \right\}
$$

(3.15)

and $\pi_n$ is normalized such that $\sum_{k=1}^{K} \pi_{nk} = 1$. The expected value of $z_{nk}$ with respect to the posterior distribution is given as $E[z_{nk}] = \pi_{nk}$. 

Given the posterior distribution of \( \mathbf{Z} \), I compute the expected value of the log joint distribution, denoted by \( Q(\mathbf{T}, \Theta) \) such that

\[
Q(\mathbf{T}, \Theta) = \mathbb{E}[\log p(\mathbf{T}, \mathbf{Y}, \mathbf{Z}, \Phi|\mathbf{X}, \Theta)]
\]

\[
= -\frac{1}{2} \alpha_t \sum_{k=1}^{K} \sum_{d=1}^{D} (t_{kd} - \mu_{t_{kd}})^2 - \frac{1}{2} \alpha_t \sum_{k=1}^{K} \sum_{d=1}^{D} (t_{kd} - \mu_{t_{kd}}^+)^2 - \frac{1}{2} \beta_t \sum_{k=1}^{K} \sum_{d=1}^{D} (t_{kd}^+ - t_{kd}^-)^2
+ \sum_{n=1}^{N} \sum_{k=1}^{K} \pi_{nk} \sum_{d=1}^{D} \left( \log g(x_{nd} - t_{kd}) + \log g(t_{kd}^+ - x_{nd}) \right)
+ \sum_{n=1}^{N} \sum_{k=1}^{K} (1 - \pi_{nk}) \log \left( 1 - \prod_{d=1}^{D} g(x_{nd} - t_{kd}) g(t_{kd}^+ - x_{nd}) \right)
+ \sum_{n=1}^{N} \sum_{k=1}^{K} \pi_{nk} \left( -\frac{1}{2} \log |\Sigma_k| - \frac{1}{2} (\mathbf{y}_n - \mu_k^T) \Sigma_k^{-1} (\mathbf{y}_n - \mu_k) \right).
\]

(3.16)

In the computation, I make use of the fact that \( \mathbb{E}[z_{nk}] = \pi_{nk} \).

**Maximization Step**

After computing the expected value \( Q(\mathbf{T}, \Theta) \) in Equation (3.16), I find the optimal \( \mathbf{T} \) and \( \Theta \) the maximizes \( Q(\mathbf{T}, \Theta) \) in the M step, such that

\[
\hat{\mathbf{T}}, \hat{\Theta} = \arg\max_{\mathbf{T}, \Theta} Q(\mathbf{T}, \Theta)
\]

(3.17)

By observing Equation (3.16), we can conclude that we can maximize \( \{t_{kd}\}_{d=1}^{D} \) and \( \{\mu_k, \Sigma_k\} \) independently for each \( k \in \{1, \ldots, K\} \).

The optimal \( \{t_{kd}\}_{d=1}^{D} \) is given by

\[
\left\{ \hat{t}_{kd} \right\}_{d=1}^{D} = \arg\max_{\{t_{kd}\}_{d=1}^{D}} -\frac{1}{2} \alpha_t \sum_{d=1}^{D} (t_{kd} - \mu_{t_{kd}})^2 - \frac{1}{2} \alpha_t \sum_{d=1}^{D} (t_{kd} - \mu_{t_{kd}}^+)^2 - \frac{1}{2} \beta_t \sum_{d=1}^{D} (t_{kd}^+ - t_{kd}^-)^2
+ \sum_{n=1}^{N} \pi_{nk} \sum_{d=1}^{D} \left( \log g(x_{nd} - t_{kd}) + \log g(t_{kd}^+ - x_{nd}) \right)
+ \sum_{n=1}^{N} (1 - \pi_{nk}) \log \left( 1 - \prod_{d=1}^{D} g(x_{nd} - t_{kd}) g(t_{kd}^+ - x_{nd}) \right)
\]

I solve this maximization problem with the conjugate gradient method [44]. With this algorithm, I need to make use of the gradient of Equation (3.18). Therefore, I compute
Algorithm 1 The EM Algorithm for DReaM

repeat

E step:
for $n \leftarrow 1$ to $N$ do
   Update the posterior distributions of $z_n$ according to Equation (3.14) and (3.15).
end for

M step:
for $k \leftarrow 1$ to $K$ do
   Update $\{t_{kd}\}_{d=1}^{D}$ according to Equation (3.18) using the BFGS algorithm.
   Update $\mu_k$ and $\Sigma_k$ according to Equation (3.21).
end for

until Convergence

the partial derivatives of Equation (3.18) with respect to $t_{kd}^+$ and $t_{kd}^-$, respectively, as follows:

\[
\frac{\partial Q(T, \Theta)}{\partial t_{kd}^+} = -\alpha t_{kd}^+ + \alpha t_{kd}^+ + \beta t_{kd}^- + \beta t_{kd}^- + \sum_{n=1}^{N} \frac{a\pi_{nk}^+}{1 + \exp(a(t_{kd}^+ - x_{nd}))} - \sum_{n=1}^{N} (1 - \pi_{nk}) \frac{a\exp(-a(t_{kd}^+ - x_{nd}))}{\exp(-a(t_{kd}^- - x_{nd})) + 1} \frac{\prod_{d=1}^{D} g(x_{nd} - t_{kd})g(t_{kd}^+ - x_{nd})}{1 - \prod_{d=1}^{D} g(x_{nd} - t_{kd})g(t_{kd}^- - x_{nd})}
\]

(3.19)

\[
\frac{\partial Q(T, \Theta)}{\partial t_{kd}^-} = -\alpha t_{kd}^- + \alpha t_{kd}^- + \beta t_{kd}^- + \beta t_{kd}^- - \sum_{n=1}^{N} \frac{a\pi_{nk}^-}{1 + \exp(a(x_{nd} - t_{kd}))} + \sum_{n=1}^{N} (1 - \pi_{nk}) \frac{a\exp(-a(x_{nd} - t_{kd}^-))}{\exp(-a(x_{nd} - t_{kd}^-)) + 1} \frac{\prod_{d=1}^{D} g(x_{nd} - t_{kd})g(t_{kd}^+ - x_{nd})}{1 - \prod_{d=1}^{D} g(x_{nd} - t_{kd})g(t_{kd}^- - x_{nd})}
\]

(3.20)

I compute $\{\mu_k, \Sigma_k\}$ in closed form as follows:

\[
\mu_k = \frac{\sum_{n=1}^{N} \pi_{nk} y_n}{\sum_{n=1}^{N} \pi_{nk}}
\]

\[
\Sigma_k = \frac{\sum_{n=1}^{N} \pi_{nk} (y_n - \mu_k)(y_n - \mu_k)^T}{\sum_{n=1}^{N} \pi_{nk}}
\]

(3.21)

I repeat the E step and M step until the objective function defined in Equation (3.12) converges. The EM algorithm is summarized in Algorithm 1.
3.4 Experiments with DReaM

In this section, I present the experimental results on both synthetic and real data. In all experiments, I set the hyper-parameters $\alpha_t = \beta_t = 1$ and the steepness parameter $a = 10$.

With DReaM, I make cluster prediction on most of the test samples by simply observing which decision rectangle these samples are located in. Note that, some samples might be located outside all decision rectangles or inside more than one decision rectangle. In this case, I assign the $n$-th sample to the cluster $k$ by observing the expected value of the soft cluster indicator, such that

$$k = \arg \max_{j \in \{1,\ldots,K\}} \left\{ \log \gamma_{nk} - \log(1 - \gamma_{nk}) \right\}$$

(3.22)

where $\gamma_{nj}$ is defined in Equation (3.7), and $E$ represents that an expected value is taken with respect to the variational distributions $q$. Note that, $\gamma_{nk}$ is a function of $x_n$ but not $y_n$. Note that, when we cluster test samples, we use only the rule-generating features $X$, but not the cluster-preserving features $Y$.

3.4.1 Testing DReaM with Synthetic Data

In this section, I demonstrate how DReaM can adjust the expert’s prior guess of the boundary based on data. I also show that given two possible reasonable clustering solutions, how the expert’s prior rule can guide choosing the clustering solution closer to the prior rule. I generate both rule-generating features $X$ and cluster-preserving features $Y$ for 1,000 synthetic samples. I first generate the $X$ features as a 2-dimensional independent Gaussian distribution from $\mathcal{N}(0, I)$. I then divide the samples into four parts according to the values of $X$ and generate the 2-dimensional cluster-preserving features $Y$, conditioned on $X$ as shown in Table 3.1.

I use DReaM to cluster the synthetic data into two clusters. I first apply DReaM without using an informative prior distribution. I plot the results in Fig. 3.3a and observe that the model is able to find the $X_1 = 0$ boundary. I plot the corresponding $Y$ in Fig. 3.3b, where we can observe that the clustering boundaries in $X$ correspond to the clustering results that optimally separate $Y$ into two clusters.
Table 3.1: Synthetic Features

<table>
<thead>
<tr>
<th>Condition that satisfied</th>
<th>How $Y$ is generated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Part 1 $x_1 &gt; 0$ and $x_2 &gt; 0$</td>
<td>$y_1 \sim \mathcal{N}(5,1), y_2 \sim \mathcal{N}(3,1)$</td>
</tr>
<tr>
<td>Part 2 $x_1 &gt; 0$ and $x_2 &lt; 0$</td>
<td>$y_1 \sim \mathcal{N}(5,1), y_2 \sim \mathcal{N}(-3,1)$</td>
</tr>
<tr>
<td>Part 3 $x_1 &lt; 0$ and $x_2 &gt; 0$</td>
<td>$y_1 \sim \mathcal{N}(-5,1), y_2 \sim \mathcal{N}(3,1)$</td>
</tr>
<tr>
<td>Part 4 $x_1 &lt; 0$ and $x_2 &lt; 0$</td>
<td>$y_1 \sim \mathcal{N}(-5,1), y_2 \sim \mathcal{N}(-3,1)$</td>
</tr>
</tbody>
</table>

Figure 3.3 Clustering the synthetic data into two clusters using DReaM.

Now let us assume experts have prior rules of $X_2 < -1$ as one group and $X_2 > -1$ as the other as shown with the dashed lines in Fig. 3.3c. The decision boundary discovered by DReaM is $X_2 = 0$, which is plotted with the solid lines. The corresponding $Y$ are plotted in Fig. 3.3d. We can observe that the clustering rule discovered corresponds to a sub-optimal clustering result with respect to $Y$, but the decision boundaries with $X$ are kept as consistent to the prior rules as possible. This experiment simulates a scenario that the domain experts have an inaccurate guess about where the decision boundaries are. DReaM is able to take advantage of the prior information on $X$, and adjust the
boundaries by making use of the cluster-preserving features $Y$.

### 3.4.2 Testing DReaM with Benchmark Data

In this section, I test DReaM using traditional clustering settings (the same setting as k-means and GMM); thus, I set $X = Y$. I compare DReaM with k-means and GMM. In addition, I also include a simple rule-generating clustering method, called $GMM-DT$, where I use the GMM clustering results as the labels to train a decision tree. Here, I show that DReaM provides interpretable clustering rules and yet can generate reasonable clustering results.

I test DReaM using several UCI datasets [45]. The details about these datasets are provided in Appendix B. I compare the methods by performing 5-fold cross validation. I estimate the performance of the algorithms using Normalized Mutual Information (NMI) and Adjusted Rand Index (ARI) of the clustering results with respect to the ground-truth labels. The definitions of NMI and ARI are given in Appendix C.

In Table 3.2, I report the mean value and standard deviation in parenthesis of the 5-fold cross-validation NMI and ARI results on UCI data. The values in bold are the top-two mean NMI or ARI value for each task. In the table, we observed that GMM performed best in most cases followed closely by DReaM, except for wdbc where DReaM tie with k-means and wine where DReaM tie with GMM. Moreover, in all cases, DReaM performs better than $GMM-DT$.

DReaM has comparable performance to GMM and/or k-means depending on the data and has an advantage that the clustering result is more interpretable, because we can explicitly characterize each cluster, and find out how a cluster is distinguished from others. I plot the samples, clustering results and the decision rectangles of the iris dataset as an example in Fig. 3.4. For example, we can explicitly describe that the cluster 1 of this data set is given by $4.13 < X_1 < 5.96, 2.30 < X_2 < 4.40, 0.36 < X_3 < 2.59$ and $-0.24 < X_4 < 0.81$, where $\{X_d\}_{d=1}^{4}$ represents each of the features. To get a similar rule for cluster $k$ from GMM results, we need to solve a system of quadratic inequalities that $\pi_k \mathcal{N}(x|\mu_k, \Sigma_k) > \pi_j \mathcal{N}(x|\mu_j, \Sigma_j)$ for all $j \neq k$, where $\pi_k$ represents the posterior probability that an arbitrary sample belongs to cluster $k$ and $\mathcal{N}$ represents the PDF of multivariate Gaussian distribution. The solution to these inequalities is usually complex,
Table 3.2: DReaM Results on Benchmark Data

<table>
<thead>
<tr>
<th></th>
<th>DReaM</th>
<th>GMM</th>
<th>K-means</th>
<th>GMM-DT</th>
</tr>
</thead>
<tbody>
<tr>
<td>iris</td>
<td>NMI</td>
<td>.861(.093)</td>
<td>.894(.108)</td>
<td>.653(.102)</td>
</tr>
<tr>
<td></td>
<td>ARI</td>
<td>.828(.126)</td>
<td>.879(.126)</td>
<td>.596(.191)</td>
</tr>
<tr>
<td>wine</td>
<td>NMI</td>
<td>.846(.129)</td>
<td>.846(.129)</td>
<td>.839(.128)</td>
</tr>
<tr>
<td></td>
<td>ARI</td>
<td>.829(.131)</td>
<td>.842(.154)</td>
<td>.829(.153)</td>
</tr>
<tr>
<td>seeds</td>
<td>NMI</td>
<td>.746(.057)</td>
<td>.873(.081)</td>
<td>.703(.097)</td>
</tr>
<tr>
<td></td>
<td>ARI</td>
<td>.720(.051)</td>
<td>.872(.092)</td>
<td>.710(.089)</td>
</tr>
<tr>
<td>breast tissue</td>
<td>NMI</td>
<td>.636(.071)</td>
<td>.639(.073)</td>
<td>.623(.066)</td>
</tr>
<tr>
<td></td>
<td>ARI</td>
<td>.334(.119)</td>
<td>.337(.118)</td>
<td>.317(.048)</td>
</tr>
<tr>
<td>wdbc</td>
<td>NMI</td>
<td>.578(.080)</td>
<td>.478(.062)</td>
<td>.586(.095)</td>
</tr>
<tr>
<td></td>
<td>ARI</td>
<td>.693(.064)</td>
<td>.596(.071)</td>
<td>.692(.077)</td>
</tr>
<tr>
<td>page blocks</td>
<td>NMI</td>
<td>.416(.034)</td>
<td>.433(.027)</td>
<td>.327(.139)</td>
</tr>
<tr>
<td></td>
<td>ARI</td>
<td>.254(.018)</td>
<td>.369(.030)</td>
<td>.264(.180)</td>
</tr>
</tbody>
</table>

![Figure 3.4](image)

Figure 3.4 Plots of the clustering result in the iris dataset. The rectangles represent the rules discovered by DReaM.

and it would be difficult for the domain experts to interpret such solution. Although GMM-DT is also able to find similarly interpretable rules as DReaM, we observe that its performance is usually worse.

3.4.3 Applying DReaM on COPD Subtyping Problem

Now, I illustrate the performance of DReaM on a real-world COPD subtyping problem. As shown in Table 1.1 and Fig. 1.1b, there already exists a rule-based guideline for COPD. This guideline is established for clinical ease and simplicity [1]. However, a good clinical rule should not only be simple (such as involving only two features), but should also reveal as much clinically relevant information as possible. This motivates us to apply DReaM to adjust the decision boundaries with the same $FEV_1 \%$ predicted and $FEV_1/FVC$ features (i.e., these are the rule-generating features $X$), based on the future
Discriminative Rectangle Mixture Model (DReaM)

Figure 3.5 Applying GOLD criterion and the rules discovered to the COPD test data. The rectangles represent the decision rules.

Table 3.3: Rules Discovered by DReaM for COPD Subtyping Problem

<table>
<thead>
<tr>
<th>Cluster</th>
<th>Rule Description</th>
<th>Predicted FEV1%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cluster 1</td>
<td>(0.732 &lt; \text{FEV}_1/FVC &lt; 0.940) and (83.2% &lt; \text{FEV}_1 &lt; 125.0% \text{ predicted})</td>
<td></td>
</tr>
<tr>
<td>Cluster 2</td>
<td>(0.668 &lt; \text{FEV}_1/FVC &lt; 0.890) and (48.8% &lt; \text{FEV}_1 &lt; 83.3% \text{ predicted})</td>
<td></td>
</tr>
<tr>
<td>Cluster 3</td>
<td>(0.549 &lt; \text{FEV}_1/FVC &lt; 0.738) and (83.9% &lt; \text{FEV}_1 &lt; 131.4% \text{ predicted})</td>
<td></td>
</tr>
<tr>
<td>Cluster 4</td>
<td>(357 &lt; \text{FEV}_1/FVC &lt; 0.666) and (49.7% &lt; \text{FEV}_1 &lt; 83.9% \text{ predicted})</td>
<td></td>
</tr>
<tr>
<td>Cluster 5</td>
<td>(0.311 &lt; \text{FEV}_1/FVC &lt; 0.653) and (30.6% &lt; \text{FEV}_1 &lt; 50.2% \text{ predicted})</td>
<td></td>
</tr>
<tr>
<td>Cluster 6</td>
<td>(0.221 &lt; \text{FEV}_1/FVC &lt; 0.401) and (11.1% &lt; \text{FEV}_1 &lt; 32.3% \text{ predicted})</td>
<td></td>
</tr>
</tbody>
</table>

Development or worsening of the disease. In COPD diagnosis, it is desirable to identify groups that lose more lung function over time. Therefore, I choose the change in \(\text{FEV}_1\) (denoted as \(\Delta \text{FEV}_1\)) and the change in \(\text{FEV}_1\% \text{ predicted}\) (denoted as \(\Delta \text{FEV}_1\% \text{ predicted}\)) after 5 years as the cluster-preserving features \(Y\). Note that these \(Y\) features are not available when the diagnosis is made, since it takes 5 years to obtain. Therefore, we do not want the rules to involve these features.

In the database, we have 2,000 subject data available. The data is randomly divided into a training and test set of equal size. I train DReaM with the training set using the GOLD criterion described in Table 1.1 as the prior knowledge. The rules discovered by DReaM are summarized in Table 3.3. These rules can be easily understood and interpreted by doctors who have less mathematical background. I apply both the GOLD criterion and the rules discovered to the test data. The results are plotted in Fig. 3.5. By comparing the rules and the figures, we can conclude that DReaM is able to adjust the decision boundaries based on the given prior knowledge.

I compare the results of DReaM with traditional clustering method k-means and GMM. To be fair, I train k-means and GMM using the concatenated data matrix \([X, Y]\); while during test the cluster assignments are generated based on only \(X\), assuming \(Y\)
is unobserved. Since the setting of this problem has some similarity to a supervised problem, I also compare DReaM with a decision tree regressor. Because a regressor usually takes only one target value, I train two decision trees, denoted as DT1 and DT2. Both decision trees use \( X \) as the features, but DT1 is trained using \( \Delta \text{FEV}_1 \) as the target, and DT2 is trained using \( \Delta \text{FEV}_1 \% \text{predicted} \) as the target. To make sure of a fair comparison to the clustering methods, I limit the number of leaves of the decision tree to be 6. In the experiments, I do not train regular supervised models, such as a linear regressor, because these models do not separate objects into categories.

In the first column of Table 3.4, I report the within-cluster sum of squares (WCSS) for training data \([X, Y]\), as one possible measure of cluster quality. A smaller WCSS implies a better clustering result. K-means gives the lowest value, as expected, because this is the objective function for k-means. DReaM does not directly optimize the cluster quality measure with respect to \( X \), and thus gives a higher WCSS value.

I report the NMI and ARI in the second and third column of the table, that measures the similarity between the clustering result with respect to the GOLD criterion. In real application, we want the clustering results be close to the existing GOLD criterion, because this criterion has been widely used by the doctors. A similar clustering result will help the doctors to make use of their existing knowledge of the disease. Since DReaM uses GOLD as prior knowledge, it gives clustering results that are most similar to the GOLD criterion and yet adjust the rules to be consistent with data.

As mentioned earlier, we want the clustering results to be relevant to the future change of lung function, measured by \( \Delta \text{FEV}_1 \) and \( \Delta \text{FEV}_1 \% \text{predicted} \). I report the within-cluster sum of squares (WCSS) with respect to these two values for test data in the 4-th column of Table 3.4. DReaM gives the lowest value, which implies that the clustering results given by DReaM are the most predictive with respect to these values.
In the COPD subtype discovery application, we want each cluster to differ in terms of $\Delta FEV_1$ and $\Delta FEV_1\%$ predicted, such that we can identify patients who lose more lung function over time and those who are more resilient. Therefore, I conduct a Kruskal-Wallis one-way analysis of variance [46], which is a non-parametric method for testing whether samples originate from the same distribution. The p-values for each of these variables are summarized in the 5-th and 6-th columns, where a smaller p-value implies a more statistically significant result. DReaM gives the smallest p-value in these tests. This implies that the rules generated by DReaM can better separate patients into groups according to the change of lung function over time.
Chapter 4

Interpretable Clustering with Feature Selection

In Chapter 3, I present an interpretable clustering model, DReaM. However, one limitation of DreaM model is that it does not incorporate features selection when generating the rules, and always uses all rule-generating features. However, in practice, some of these features might be irrelevant. Irrelevant features may potentially harm the performance if they are used to define the rules.

To overcome this limitation, I propose to incorporate feature selection for the interpretable clustering models. Feature selection improves the performance of clustering algorithm because it enables removing noisy and irrelevant features in the data and it also makes the results more interpretable, because the complexity of the rules is reduced.

This dissertation introduces two strategies to incorporate feature selection for interpretable clustering models. In Chapter 4.1, I introduce a Generative Interpretable Clustering Model with Feature Selection (GICMFS). Then in Chapter 4.2, I present DReaM Model with Feature Selection (DReaM-FS).
4.1 A Generative Interpretable Clustering Model with Feature Selection (GICMFS)

In this section, I propose a Generative Interpretable Clustering Model with Feature Selection (GICMFS). GICMFS automatically generates a list of rules that might involve different sets of features across different clusters. It assumes that if a sample belongs to one of the clusters, then the rules in the corresponding list tend to be satisfied. The axis-parallel-cut rules are based on partitioning features into several parts via one-dimensional Gaussian mixture models. Whether one feature is selected in the rule list for one cluster is determined based on whether it characterizes the cluster. GICMFS jointly learns the selected features, axis-parallel cut for each selected feature and cluster assignments. After training the model, we can derive a list of rules for each of the cluster by analyzing the posterior probability distributions of the Gaussian components to give us a conjunction of axis-parallel cut rules for each cluster.

4.1.1 The GICMFS model

In this section, I introduce the GICMFS model. I start by introducing some notations. GICMFS is a clustering model that separates the given samples into different clusters. I let \( K \) be a pre-defined integer that define the number of clusters. It is possible to extend GICMFS with a Dirichlet process prior \([47]\) such that \( K \) can be automatically learned. Since this is not the major focus of this model, I introduce a simpler version where \( K \) is pre-defined.

I assume that the data contains \( N \) samples and \( D \) features. I let \( \mathbf{x}_n \in \mathbb{R}^D \) with \( n = 1, \ldots, N \) be the vector representing the features for sample \( n \). I standardize all features such that each feature has zero mean and unit variance. I define \( \mathbf{z}_n \) to be the \( K \)-dimensional cluster indicator for sample \( n \), such that \( z_{nk} = 1 \) if and only if the \( n \)-th sample belongs to cluster \( k \); and \( z_{nk} = 0 \), otherwise. I let the prior probability that the \( n \)-th sample belongs to each cluster be equal, i.e.,

\[
\mathbf{z}_n \sim \text{Categorical}(\pi),
\]  

(4.1)
where \( \pi \) is a \( K \)-dimensional vector such that
\[
\pi = \left[ \frac{1}{K}, \ldots, \frac{1}{K} \right].
\] (4.2)

### 4.1.1.1 Clustering with Generative Rules

As mentioned earlier, we want to develop an interpretable clustering algorithm. To achieve interpretability, I generate a list of rules for each cluster \( k \) using a subset of features. I assume that if a sample belongs to cluster \( k \), then all rules defined for cluster \( k \) tends to be satisfied. I define a \( K \)-dimension vector \( \zeta^{(kd)} \in (\mathbb{R}^+)^K \), s.t., \( \sum_{j=1}^{K} \zeta_j^{(kd)} = 1 \), where \( \zeta_j^{(kd)} \) defines the probability that the rule defined for cluster \( j \) with feature \( d \) is satisfied by a sample, assuming that this sample belongs to cluster \( k \). We have \( \sum_{j=1}^{K} \zeta_j^{(kd)} = 1 \), because we want the rule defined with each feature to partition the feature into non-overlapping intervals, such that each sample satisfies exactly one rule defined with each feature.

Given that a sample belongs to cluster \( k \), we want to make sure that all rules for cluster \( k \) are more likely to be satisfied, while the rules defined for other clusters \( j \neq k \) are less likely to be satisfied. Therefore, I define a prior distribution for \( \zeta^{(kd)} \) such that
\[
\zeta^{(kd)} \sim \text{Dirichlet} \left( \psi^{(k)} \right),
\] (4.3)
where \( \psi^{(k)} \in \mathbb{R}^K \) is a \( K \)-element vector whose component is defined as
\[
\psi_j^{(k)} = \begin{cases} 
\alpha \zeta_0, & \text{if } j = k \\
1, & \text{if } j \neq k,
\end{cases}
\] (4.4)

and \( \alpha \zeta_0 \) is a predefined hyper-parameter satisfying \( \alpha \zeta_0 \gg 1 \). Due to the properties of Dirichlet distribution, the expected value of \( \zeta_j^{(kd)} \) is given by
\[
\mathbb{E}[\zeta_j^{(kd)}] = \begin{cases} 
\frac{\alpha \zeta_0}{\alpha \zeta_0 + K - 1} \approx 1, & \text{if } j = k \\
\frac{1}{\alpha \zeta_0 + K - 1} \approx 0, & \text{if } j \neq k.
\end{cases}
\] (4.5)

This guarantees that if a sample belongs to cluster \( k \), then rules for cluster \( k \) are more likely to be satisfied, while rules for clusters \( j \neq k \) are more likely to be dissatisfied. In
the experiment, I choose $\alpha_{kd} = 4(K - 1)$, such that $E[\zeta_{k}^{(kd)}] = 0.8$. Note that I define the prior distribution for $\zeta^{(kd)}$ here, and the posterior distribution will be fit to the data in the training process.

I assume that the rules defined for each cluster $k$ only involves a subset of features, while the remaining features are irrelevant. Therefore, I define an binary indicator $s_{kd} \in \{0, 1\}$ for each cluster $k$ and feature $d$. If feature $d$ is used to define the rules for cluster $k$, I let $s_{kd} = 1$; and $s_{kd} = 0$, otherwise. I assign a non-informative prior for each $s_{kd}$ such that

$$s_{kd} \sim \text{Bernoulli} \left( \frac{1}{2} \right). \quad (4.6)$$

Now I define a $K$ dimensional binary vector $r^{(nd)} \in \{0, 1\}^K$, s.t., $\sum_{k=1}^{K} r_{k}^{(nd)} = 1$, where $r_{k}^{(nd)}$ is the indicator that sample $n$ satisfies the rule for cluster $k$ defined with feature $d$. I let

$$r^{(nd)} \sim \prod_{k=1}^{K} \left\{ \text{Categorical}(\zeta^{(kd)}) \right\}^{z_{nk}s_{kd}} \left\{ \text{Categorical}(\eta) \right\}^{z_{nk}(1-s_{kd})}. \quad (4.7)$$

where $\eta$ is a $K$-dimensional vector defined as $\eta = \left[ \frac{1}{K}, \ldots, \frac{1}{K} \right]$.

With the first part in the r.h.s of Equation (4.7), I assume that if sample $n$ belongs to cluster $k$ (i.e., $z_{nk} = 1$) and feature $d$ is used to define the rules for cluster $k$ (i.e., $s_{kd} = 1$) then $r_{nd}$ follows $\text{Categorical}(\zeta^{(kd)})$. Because of the prior distribution of $\zeta^{(kd)}$ defined in Equation (4.3), we have $\zeta_{k}^{(kd)} > \zeta_{j}^{(kd)}$, for $j \neq k$. This implies that it is more likely that $r_{k}^{(nd)} = 1$; i.e., if sample $n$ belongs to cluster $k$, then sample $n$ tends to satisfy the rule for cluster $k$ defined with feature $d$.

In the second part in the r.h.s of Equation (4.7), I assume that if $z_{nk} = 1$ and $s_{kd} = 0$, then $r^{(nd)}$ follows a uniform categorical distribution. This indicates that if sample $n$ belongs to cluster $k$ (i.e., $z_{nk} = 1$), but feature $d$ is not used to define the rules for cluster $k$ (i.e., $s_{kd} = 0$ ), we do not care the value of $r^{(nd)}$ and therefore I let $r^{(nd)}$ be random. Note that even if feature $d$ is not used to define the rules for cluster $k$ (i.e., $s_{kd} = 0$), it is still possible that $r_{k}^{(nd)} = 1$, due to the uniform distribution. In this case, since $\sum_{j=1}^{K} r_{j}^{(nd)} = 1$ and $r_{k}^{(nd)}$ are binary, we have $r_{j}^{(nd)} = 0$ for all $j \neq k$. Therefore, when $s_{kd} = 0$, $r_{k}^{(nd)} = 1$ in fact indicates that the rules for all other clusters $j \neq k$ defined with feature $d$ are not satisfied.
To better understand the clustering mechanism of GICMFS, I derive the log posterior distribution for the cluster indicator $z_n$ as follows

$$\log p \left( z_{nk} = 1 \left\{ r^{(nd)}, s_{kd}, \zeta^{(kd)} \right\}_{d=1}^D \right) = \sum_{d=1}^D s_{kd} \sum_{j=1}^K r_{j}^{(nd)} \log \zeta_{j}^{(kd)} + \text{const}. \quad (4.8)$$

where “const” represents the normalization constant that guarantees $\sum_{k=1}^K z_{nk} = 1$. The equation does not involve $\eta$, because $1/K$ is a constant. Because of Equation (4.5), we have $\log \zeta_{j}^{(kd)} < \log \zeta_{k}^{(kd)}$ for all $j \neq k$. Therefore, the posterior probability that $z_{nk} = 1$ is greater, only if $r_{k}^{(nd)} = 1$ for more selected feature $d$ (since the summation is taken over $s_{kd}$); i.e., it is more likely that the sample $n$ belongs to cluster $k$ (s.t., $z_{nk} = 1$), if the sample $n$ satisfies more rules (s.t., $r_{k}^{(nd)} = 1$) defined for cluster $k$ (i.e., for all $d$ s.t., $s_{kd} = 1$). In this equation, the values $\{ \log \zeta_{j}^{(kd)} \}_{j=1}^K$, can be considered as measures for the quality of each rule, defining the penalty if a rule is satisfied or not. We also want to highlight that when deriving the posterior distribution of $z_n$, I assume that all parameters $\{ y^{(nd)}, s_{kd}, \zeta^{(kd)} \}_{d=1}^D$ follow the posterior distributions given the observed data $\{ x_n \}_{n=1}^N$ rather than their prior distributions.

I also derive the posterior distribution for $s_{kd}$, given the random variables $\zeta^{(kd)}$ and $\{ y^{(nd)}, z_{nk} \}_{n=1}^N$, to understand how features are selected by GICMFS. The posterior of $s_{kd}$ is given by

$$p \left( s_{kd} = 1 \left\{ \zeta^{(kd)}, r^{(nd)}, z_{nk} \right\}_{n=1}^N \right) = \frac{\exp(a_{s_{kd}})}{\exp(a_{s_{kd}}) + 1}. \quad (4.9)$$

where $a_{s_{kd}}$ is defined as

$$a_{s_{kd}} = \sum_{n=1}^N \sum_{j=1}^K z_{nk} r_{j}^{(nd)} \log \zeta_{j}^{(kd)} - \sum_{n=1}^N z_{nk} \log \frac{1}{K}. \quad (4.10)$$

In Equation (4.10), the first term is a measure for the quality of the rule for cluster $k$ defined with feature $d$. Because of the prior distribution in Equation (4.3), we have $\log \zeta_{j}^{(kd)} < \log(1/K) < \log \zeta_{k}^{(kd)}$ for all $j \neq k$; therefore, this term gives larger values if the rule defined for cluster $k$ with feature $d$ is satisfied by more samples $n$ (i.e., $r_{k}^{(nd)} = 1$) that belong to cluster $k$ (since the summation is taken over $z_{nk}$). The second term is a baseline that simply assumes whether the rule is satisfied (i.e., the value of $y_{k}^{(nd)}$) is completely random, for all samples belonging to cluster $k$ (the summation is taken over
I take the difference between these two terms, indicating that for all samples in cluster $k$, if the rule defined with feature $d$ is more likely to be satisfied than random, then it is more likely that feature $d$ is selected to define a rule for cluster $k$.

### 4.1.1.2 Axis Parallel Cuts by Partitioning on Features

In Equations (4.8) and (4.9), I show how GICMFS achieves clustering and feature selection, assuming that $r^{(nd)}$ is given. In this section, I demonstrate how GICMFS learns $r^{(nd)}$ from the observed data, by making use of the property of one-dimensional Gaussian Mixture Model (GMM).

Consider an $L$-component one-dimensional GMM defined as

$$x \sim \prod_{l=1}^{L} \left\{ \mathcal{N}(x; \mu_l, \sigma_l^2) \right\}^{t_l}, \quad (4.11)$$

where $t \in \{0, 1\}^L$, s.t., $\sum_{l=1}^{L} t_l = 1$, is an $L$-dimensional binary cluster indicator, such that $t_l = 1$, if $x$ belongs to cluster $l$; and $t_l = 0$, otherwise. $\mathcal{N}(x; \mu_l, \sigma_l^2)$ is a probabilistic density function (PDF) of a Gaussian distribution with random variable $x$, that is parameterized by mean $\mu_l$ and variance $\sigma_l^2$. I further assume a prior distribution for $t$ such that

$$t \sim \text{Categorical}(\omega), \quad (4.12)$$

where $\omega \in \{\mathbb{R}^+\}^L$, s.t., $\sum_{l=1}^{L} \omega_l = 1$ is an $L$-dimensional non-negative vector, each of whose component $\omega_l$ represents the probability that $x$ belongs to cluster $l$, i.e.,

$$p(t_l = 1) = \omega_l.$$

Now consider a test sample $x^* \in \mathbb{R}$. The maximum a posteriori (MAP) estimator $\hat{l}^* \in \{1, \ldots, K\}$ that estimates which component $x^*$ belongs to is given by

$$\hat{l}^* = \arg \max_{l \in \{1, \ldots, L\}} p(t_l = 1 | x^*). \quad (4.13)$$

By Bayes’ theorem, we have

$$p(t_l = 1 | x^*) \propto \omega_l \mathcal{N}(x^*; \mu_l, \sigma_l^2). \quad (4.14)$$
Consider an arbitrary 3-component one-dimensional GMM. I plot the pdf of the 3 components above the axis. The GMM partitions the axis into 3 parts. The decision boundaries of the partition are given by the intersection points of the functions $\{\omega_l \cdot \mathcal{N}(x^*; \mu_l, \sigma^2_l)\}_{l=1}^3$.

Therefore, given the parameters $\omega$ and $\{\mu_l, \sigma^2_l\}_{l=1}^L$, the one-dimensional GMM in fact partitions the real axis into $L$ consecutive or intermittent intervals, where the decision boundaries are given by the intersections of the functions defined in Equation (4.14). Therefore, the estimator $\hat{l}^*$ can be considered as the index of the partition that $x^*$ belongs to. This partition is defined by the system of inequalities

$$p(t_{\hat{l}^*} = 1|x^*) \geq p(t_l = 1|x^*), \quad \text{for all } l \neq \hat{l}^*.$$ (4.15)

By solving the system of inequalities defined above, we can generate the rules for each component given the parameters, as shown in Figure 4.1.

GICMFS is motivated by this property. In GICMFS, I assume each observed feature $d$ of sample $n$ follows a one-dimensional GMM, such that

$$x_{nd}|r^{(nd)} \sim \prod_{k=1}^K \{\mathcal{N}(\mu_{kd}, \sigma^2_{kd})\}^{r^{(nd)}_k},$$ (4.16)

where $\mu_{kd}$ and $\sigma^2_{kd}$ are the mean and variance parameters for the Gaussian distribution in the $d$-th dimension for the $k$-th cluster. With the distribution, we can derive the rules after we learn the posterior distribution, by solving similar inequalities defined in Equation (4.15). More details are given in Chapter 4.1.3.
I assign a normal-inverse-gamma prior for the Gaussian parameters \( \Theta_{kd} = \{\mu_{kd}, \sigma^2_{kd}\} \) such that

\[
(\mu_{kd}, \sigma^2_{kd}) \sim NIG(\mu_{\theta_0}, \lambda_{\theta_0}, \alpha_{\theta_0}, \beta_{\theta_0}),
\]

where \( \mu_{\theta_0}, \lambda_{\theta_0}, \alpha_{\theta_0}, \beta_{\theta_0} \) are the parameters for the normal-inverse-gamma distribution. In the experiments, I choose \( \mu_{\theta_0} = 0, \lambda_{\theta_0} = 1, \alpha_{\theta_0} = 1, \) and \( \beta_{\theta_0} = 1. \)

### 4.1.1.3 The Overall Model of GICMFS

Above, I have described the GICMFS model. The joint distribution of GICMFS is given by

\[
p(\{x_n\}_{n=1}^N, \Phi | \Gamma) = \prod_{n=1}^N p(z_n) \prod_{k=1}^K \prod_{d=1}^D p(\zeta^{(kd)} | \alpha_{\zeta0}) \prod_{k=1}^K \prod_{d=1}^D p(s_{kd}) \prod_{k=1}^K \prod_{d=1}^D p(\Theta_{kd}) \prod_{n=1}^N \prod_{d=1}^D p(x_{nd} | r_{nd}, \{\Theta_{kd}\}_{k=1}^K),
\]

where I use \( \Gamma \) to represent all hyper-parameters and \( \Phi \) to represent all latent variables.

The generative process can be summarized in Algorithm 2. The model can be described using a directed graphical model in Figure 4.2.

### 4.1.2 Training GICMFS via Variational Inference

Given the observed data \( \{x_n\}_{n=1}^N \) and all hyper-parameters \( \Gamma \), we are interested in learning the posterior distribution for all the latent variables \( \Phi \). However, it is computationally intractable to derive this posterior distribution. Therefore, I use a variational
Algorithm 2 Generative Process

\begin{algorithm}
\begin{algorithmic}
\FOR{$n \leftarrow 1$ \TO $N$}
\STATE Generate $z_n$ according to Equation (4.1).
\ENDFOR
\FOR{$k \leftarrow 1$ \TO $K$}
\FOR{$d \leftarrow 1$ \TO $D$}
\STATE Generate $\zeta_{kd}$ according to Equation (4.3).
\STATE Generate $s_{kd}$ according to Equation (4.6).
\STATE Generate $\Theta_{kd}$ according to Equation (4.17).
\ENDFOR
\ENDFOR
\FOR{$n \leftarrow 1$ \TO $N$}
\FOR{$d \leftarrow 1$ \TO $D$}
\STATE Generate $r_{nd}$ according to Equation (4.7).
\STATE Generate $x_{nd}$ according to Equation (4.16).
\ENDFOR
\ENDFOR
\end{algorithmic}
\end{algorithm}

distribution $q(\Phi)$ to approximate the posterior distribution such that

$$q(\Phi) \approx p(\Phi|\{x_n\}_{n=1}^N, \Gamma) \quad (4.19)$$

To make sure we can derive a tractable $q(\Phi)$, I utilize the mean-field assumption that it factorizes such that

$$q(\Phi) = \prod_{n=1}^N q(z_n) \prod_{k=1}^K \prod_{d=1}^D q(\zeta_{kd}) \prod_{k=1}^K \prod_{d=1}^D q(s_{kd}) \prod_{k=1}^K \prod_{d=1}^D q(\mu_{kd}, \sigma_{kd}^2) \prod_{n=1}^N \prod_{d=1}^D q(r_{nd}). \quad (4.20)$$

In the inference, I derive an optimal variational distribution $q(\Phi)$ that minimizes the KL divergence $KL(q||p)$. Minimizing $KL(q||p)$ is equivalent to maximizing the lower-bound $L$, defined as

$$L = \sum_{n=1}^N \mathbb{E}_q[\log p(z_n)] + \sum_{k=1}^K \sum_{d=1}^D \mathbb{E}_q[\log p(\zeta_{kd}) \alpha_{kd}] + \sum_{k=1}^K \sum_{d=1}^D \mathbb{E}_q[\log p(s_{kd})]$$

$$+ \sum_{k=1}^K \sum_{d=1}^D \mathbb{E}_q[\log p(\Theta_{kd})] + \sum_{n=1}^N \sum_{d=1}^D \mathbb{E}_q[\log p(r_{nd}) | z_n, \{\zeta_{kd}\}_{k=1}^K]$$

$$+ \sum_{n=1}^N \sum_{d=1}^D \mathbb{E}_q[\log p(x_{nd}) | r_{nd}, \{\Theta_{kd}\}_{k=1}^K] + \sum_{n=1}^N \mathcal{H}(q(z_n)) + \sum_{k=1}^K \sum_{d=1}^D \mathcal{H}(q(\zeta_{kd}))$$

$$+ \sum_{k=1}^K \sum_{d=1}^D \mathcal{H}(q(s_{kd})) + \sum_{k=1}^K \sum_{d=1}^D \mathcal{H}(q(\Theta_{kd})) + \sum_{n=1}^N \sum_{d=1}^D \mathcal{H}(q(r_{nd})). \quad (4.21)$$
In this equation, $E_q$ represents that the expected value is taken with respect to the variational distribution $q(\Phi)$ and $\mathcal{H}(q(\cdot))$ represents the entropy of the variational distribution $q(\cdot)$.

Since I develop the model using conjugate priors, it is straightforward to derive the update of the variational distributions $q$, as described in Appendix A, by applying variational calculus. I omit the details of derivation, and list the update equations as follows:

$$q(z_n) \sim \text{Bernoulli}(\pi_{z_n})$$ (4.22)

where

$$\pi_{zn} \propto \exp \left( \sum_{d=1}^{D} \mathbb{E}_q[s_{kd}] \sum_{j=1}^{K} \mathbb{E}_q[r_{jd}(nd)] \mathbb{E}_q[\log \zeta_{j}^{(kd)}] \right)$$ (4.23)

and $\pi_{zn}$ is normalized, such that $\sum_{k=1}^{K} \pi_{zn} = 1$

$$q(\zeta_{kd}) \sim \text{Dirichlet}(\alpha_{\zeta_{kd}})$$ (4.24)

where $\alpha_{\zeta_{kd}}$ is a $K$-element vector, each of the components is given by

$$\alpha_{\zeta_{j}^{(kd)}} = \begin{cases} \alpha_{\zeta} + \sum_{n=1}^{N} \mathbb{E}_q[z_{nk}] \mathbb{E}_q[s_{kd}] \mathbb{E}_q[r_{jd}(nd)] & \text{for } j = k \\ 1 + \sum_{n=1}^{N} \mathbb{E}_q[z_{nk}] \mathbb{E}_q[s_{kd}] \mathbb{E}_q[r_{jd}(nd)] & \text{for } j \neq k \end{cases}$$ (4.25)

$$q(s_{kd}) \sim \text{Bernoulli}(\rho_{skd})$$ (4.26)

where

$$\rho_{skd} = \frac{\exp(a_{skd})}{\exp(a_{skd}) + 1}$$ (4.27)

and $a_{skd}$ is defined as

$$a_{skd} = \sum_{n=1}^{N} \mathbb{E}_q[z_{nk}] \left( \sum_{j=1}^{K} \mathbb{E}_q[r_{jd}(nd)] \mathbb{E}_q[\log \zeta_{j}^{(kd)}] - \log \frac{1}{K} \right)$$ (4.28)

$$q(\mu_{kd}, \sigma_{kd}^2) \sim \mathcal{NIG}(\lambda_{\theta_{kd}}, m_{\theta_{kd}}, \alpha_{\theta_{kd}}, \beta_{\theta_{kd}})$$ (4.29)
where

\[ \lambda_{\theta_{kd}} = \lambda_{\theta_0} + \sum_{n=1}^{N} \mathbb{E}_q[r_{kn}^{(nd)}] \]

\[ m_{\theta_{kd}} = \frac{\lambda_{\theta_0} m_{\theta_0} + \sum_{n=1}^{N} \mathbb{E}_q[r_{kn}^{(nd)}] x_{nd}}{\lambda_0 + \sum_{n=1}^{N} \mathbb{E}_q[r_{kn}^{(nd)}]} \]

\[ \alpha_{\theta_{kd}} = \alpha_{\theta_0} + \frac{1}{2} \sum_{n=1}^{N} \mathbb{E}_q[r_{kn}^{(nd)}] \]

(4.30)

\[ \beta_{\theta_{kd}} = \beta_{\theta_0} + \frac{1}{2} \sum_{n=1}^{N} \mathbb{E}_q[r_{kn}^{(nd)}] x_{nd}^2 + \frac{1}{2} \lambda_{\theta_0} m_{\theta_0}^2 \]

\[ - \frac{1}{2} \left( \lambda_{\theta_0} + \sum_{n=1}^{N} \mathbb{E}_q[s_{kd}] \mathbb{E}_q[r_{kn}^{(nd)}] \right) m_{\theta_{kd}}^2 \]

\[ q(r^{(nd)}) \sim \text{Categorical}(\pi_{r^{(nd)}}) \] (4.31)

where \( \pi_{r^{(nd)}} \) is a \( K \)-element vector, each of the components is given by

\[ \pi_{r_{nk}} \propto \exp \left( \sum_{j=1}^{K} \mathbb{E}_q[z_{nj}] \left( \mathbb{E}_q[s_{jd}] \mathbb{E}_q[\log \zeta^{(j)}] + (1 - \mathbb{E}_q[s_{jd}]) \log \frac{1}{K} \right) \right. \]

\[ \left. - \frac{1}{2} \mathbb{E}_q[\log \sigma^{2}_{kd}] - \frac{1}{2} \mathbb{E}_q \left[ \frac{1}{\sigma^{2}_{kd}} (x_{nd} - \mu_{kd})^2 \right] \right) \] (4.32)

and \( \pi_{r^{(nd)}} \) is normalized such that, \( \sum_{k=1}^{K} \pi_{r_{nk}} = 1 \).

The expected values that are involved in the update equations are given as follows:

\[ \mathbb{E}_q[z_{nk}] = \pi_{z_{nk}} \]

\[ \mathbb{E}_q[s_{kd}] = \rho_{s_{kd}} \]

\[ \mathbb{E}_q[r_{kn}^{(nd)}] = \pi_{r_{kn}^{(nd)}} \]

\[ \mathbb{E}_q[\log \zeta^{(j)}] = \psi \left( \alpha_{\zeta^{(j)}} \right) - \psi \left( \sum_{l=1}^{K} \alpha_{\zeta^{(l)}} \right) \] (4.33)

\[ \mathbb{E}_q[\log \sigma^{2}_{kd}] = \log \beta_{kd} - \psi(\alpha_{kd}) \]

\[ \mathbb{E}_q \left[ \frac{1}{\sigma^{2}_{kd}} (x_{nd} - \mu_{kd})^2 \right] = \frac{1}{\lambda_{kd}} + \frac{\alpha_{kd}}{\beta_{kd}} (x_{nd} - m_{kd})^2 \]

where \( \psi \) represents the digamma function, i.e., the logarithmic derivative of the gamma function.
Algorithm 3 Variational Inference for GICMFS

Initialize all factorized variational distributions $q(\Phi)$ as shown in Equation (4.20).

repeat
  for $n \leftarrow 1$ to $N$ do
    update $q(z_n)$ according to Equation (4.22).
  end for
  for $k \leftarrow 1$ to $K$ do
    for $d \leftarrow 1$ to $D$ do
      update $q(\zeta^{(kd)})$ according to Equation (4.24).
      update $q(s_{kd})$ according to Equation (4.26).
      update $q(\Theta_{kd})$ according to Equation (4.29).
    end for
  end for
  for $n \leftarrow 1$ to $N$ do
    for $d \leftarrow 1$ to $D$ do
      update $q(r^{(nd)})$ according to Equation (4.31).
    end for
  end for
until Convergence

I update each variational distribution $q$ iteratively until convergence. The optimization process is summarized in Algorithm 3. When applying each of the update equations, the value of the lower-bound $\mathcal{L}$ is always non-decreasing. Therefore, it is guaranteed that this optimization process converges.

4.1.3 Prediction with Generative Rules

In this section, I present how to generate the clustering rules and how to predict which cluster a test sample $x^*$ belongs to, using the rules generated.

4.1.3.1 Generating the Rules

To generate rules for each cluster, I first identify the subset of relevant features. I simply select all features $d$ for cluster $k$ satisfying $q(s_{kd} = 1) > 0.5$. I denote all features selected for cluster $k$ as a set $S_k$, such that

$$S_k = \{d \in \{1, \ldots, D\} \mid q(s_{kd} = 1) > 0.5\}$$  \hspace{1cm} (4.34)

Note that the number of features that are used to define cluster $k$, denoted by $|S_k|$, might be different across different $k \in \{1, \ldots, K\}$. 
Now I learn a rule for each selected feature \( d \in S_k \). I first need to estimate the prior probability that a rule is satisfied, i.e., \( \omega \) in Equation (4.14). I denote this prior probability for feature \( d \) and cluster \( k \) as \( \omega_{kd} \), and estimate it using a maximum likelihood estimator,

\[
\omega_{kd} = \frac{\sum_{n=1}^{N} E_q[y^{(nd)}_k]}{N}.
\]

Then I generate the rule for each \( d \in S_k \). By making use of Equation (4.15), we can learn each rule denoted by a set \( r_{kd} \) as

\[
r_{kd} = \{ x^*_d \in \mathbb{R} | \omega_{kd} E_q[\mathcal{N}(x^*_d | \mu_{kd}, \sigma_{kd}^2)] > \omega_{jd} E_q[\mathcal{N}(x^*_d | \mu_{jd}, \sigma_{jd}^2)] \text{ for all } j \neq k \},
\]

where \( E_q[\cdot] \) denotes that the expected values are taken with respect to the variational distribution \( q(\Phi) \). As shown in Figure 4.1, \( \{r_{kd}\}_{k=1}^{K} \) partitions the feature \( d \) into several consecutive or intermittent intervals. We can easily check whether the rule is satisfied by observing whether \( x^*_d \) falls into these intervals.

I assume that if a sample belongs to cluster \( k \), then all rules defined for cluster \( k \) needs to be satisfied. Therefore, the rules for cluster \( k \), denoted by \( R_k \), is defined using the conjunction of all rules that are defined with relevant features, such that

\[
R_k = \{ x^* \in \mathbb{R}^D | x^*_d \in r_{kd}, \text{ for all } d \in S_k \}.
\]

### 4.1.3.2 Clustering with Generated Rules

After the rules are generated, I assign the test sample \( x^* \) by observing whether it satisfies all the rules for a certain cluster \( k \), such that \( x^* \in R_k \). However, some test samples \( x^* \) might not satisfy all rules for any cluster, such that \( x^* \in \left( \mathbb{R}^D - \bigcup_{k=1}^{K} R_k \right) \). I consider these samples as noisy samples; but to fairly compare with other clustering methods, we want to also be able to predict which cluster these samples belong to.

I introduce a heuristic such that we can easily determine which cluster \( x^* \) belongs by computing the ratio of the rules that are satisfied, i.e.

\[
\hat{k}^* = \arg \max_{k} \left[ \frac{\left| \{ r_{kd} \in R_k | x^*_d \in r_{kd} \} \right|}{\left| S_k \right|} \right]
\]

(4.38)
where the numerator is the number of rules for cluster $k$ that are satisfied by $x^*$, and the denominator is the total number of rules for cluster $k$. Note that since $|S_k|$ may differ across different $k \in \{1, \ldots, K\}$, simply counting the number of rules satisfied or not might cause bias. Therefore, we predict clusters based on the ratio.

4.1.4 Experiments with GICMFS

In this section, I present the experimental results on both synthetic and real data.

4.1.4.1 Testing GICMFS with Synthetic Data

To illustrate the rule generation mechanism of GICMFS, I generate 300 10-dimensional synthetic samples. I let the first 2 features in the data follow a mixture of Gaussian distributions with unit variance, and with means $(0, 1)$, $(-4, -1.5)$ and $(4, -1.5)$, respectively. I let the remaining 8 features follow a Gaussian distribution with zero mean and unit variance.

I run GICMFS on this synthetic data set, and check which features are selected. We can observe that all the last 8 features are not selected for any cluster by the model. This is expected because these features are not distributed differently across different clusters, and we can not generate good rules with these features. For the first 2 features, cluster 2 uses both features to generate rules, but clusters 1 and 3 select feature 1 only. To illustrate why this happens, I plot the samples and the rules generated in Figure 4.3a.

Be reminded that I have demonstrated using Equation (4.9) that GICMFS selects features based on the quality of each rule, given the clustering solution. If a feature $d$ is selected for cluster $k$, then it must be true that most samples belong to cluster $k$ satisfy the rule defined with feature $d$. In Figure 4.3a, it means most samples of each cluster should be located inside the corresponding decision boundaries. We can observe in the figure that this condition is satisfied by the selected features for all the rules. For example, most samples of cluster 2 satisfies both $-1.8 < f_1 < 1.8$ and $f_2 > -0.8$. I want to highlight that some samples of clusters 1 and 3 also satisfy the rule $f_2 > -0.8$. However, according to Equation (4.9), this does not prevent $f_2$ from being selected for cluster 2, but it causes $f_2$ to be not selected for cluster 1 and 3.
Most existing feature selection methods, such as [35], select $f_2$ for clusters 1 and 3; it is distributed differently from cluster 2, and therefore reveals clustering structure. However, GICMFS does not select this feature. To better illustrate why, I assume $f_2$ is selected for both clusters 1 and 3, and plot the resulting rules in Figure 4.3b. Since we do not allow the rules for each feature to overlap, a large number of samples that belongs to clusters 1 and 3 are located outside the corresponding decision boundaries. These rules are not able to best describe the characteristic of clusters 1 and 3, thus the model does not select this feature.

We also want to highlight that there are samples in Figure 4.3a that are located outside all boxes. Those samples do not satisfy all rules for any of the clusters. According to Equation (4.38), the samples located at the lower part in the middle will be assigned to cluster 2, because it satisfies 1 out of 2 rules for cluster 2 (the ratio is $1/2 = 0.5$, but satisfies 0 out of 1 rules for cluster 1 and 3 (the ratio is $0/1=0$).

### 4.1.4.2 Testing GICMFS with Benchmark Data

In this section, I test GICMFS using 4 UCI data sets [45]: iris, wine, seeds and wdbc; 2 gene expression data sets: lung[48] and carcinomas[49]; and the following 3 image data sets: orlraws10P, CMU faces and semeion [45]. The details about these datasets are provided in Appendix B. I.e repeat the experiments using these image data sets for different tasks. In each of the tasks, I use a subset of samples in each data set. For
example, I use orlraws10P(5) to denote that the experiment is conducted only using the images of the first 5 people for both training and validation.

GICMFS is a probabilistic model. Therefore, I focus on comparing GICMFS with other probabilistic clustering methods in the experiments. I choose the Gaussian Mixture Model (GMM) [4] as the baseline. Since GICMFS is a local feature selection method, i.e., we allow the features selected to differ across different clusters, we include the local feature selection version of [35], denoted as LFS. LFS selects features that reveal a clustering structure but cannot generate interpretable rules. I compare GICMFS with DReaM, a probabilistic discriminative model that is introduced in Chapter 3. DReaM generates interpretable rules, but does not incorporate feature selection and always uses all features. I also include a simple rule generating clustering method I call GMM-DT, where I use the GMM clustering results for the training data as labels to train a decision tree.

Since GICMFS generates rules, we want to know how accurately these rules will perform in a predictive task. Therefore, I conduct a 5-fold cross validation and compare the predictive cluster labels in the validation sets with the ground-truth labels. I measure the performance using Normalized Mutual Information (NMI) and Adjusted Rand Index (ARI), where NMI is the normalized version of mutual information and ARI is the corrected-for-chance version of the Rand index. The definitions of NMI and ARI are given in Appendix C.

In Table 4.1, I report the mean value and standard deviation of the 5-fold cross-validation NMI and ARI results on each clustering task. The values in bold are the highest mean NMI or ARI value for each task. Since DReaM is not scalable, I cannot run this method on data sets with more than 1,000 features. In Table 4.2, I report the mean value and standard deviation of the number of features selected for each cluster by GICMFS and LFS. The mean value and standard deviation are estimated by simply treating the number of selected features for all clusters in all folds as independent samples.

To begin with, I analyze the clustering performance on the 4 UCI data sets and the 2 gene expression data sets. We can observe from Table 4.1 that GICMFS and LFS are the two best performers. The performance of these two methods is comparable, and are better than the rest of the methods. This demonstrates that feature selection helps to improve the performance in these clustering tasks. Both GMM and DReaM
Table 4.1: Clustering Performance of GICMFS on Benchmark Data

<table>
<thead>
<tr>
<th></th>
<th>GICMFS</th>
<th>GMM</th>
<th>GMM-FS</th>
<th>DReaM</th>
<th>GMM-DT</th>
</tr>
</thead>
<tbody>
<tr>
<td>iris</td>
<td>NMI</td>
<td>.85 (.09)</td>
<td>.81 (.08)</td>
<td>.81 (.08)</td>
<td>.83 (.09)</td>
</tr>
<tr>
<td></td>
<td>ARI</td>
<td>.76 (.17)</td>
<td>.76 (.17)</td>
<td>.84 (.10)</td>
<td>.79 (.13)</td>
</tr>
<tr>
<td>wine</td>
<td>NMI</td>
<td>.87 (.05)</td>
<td>.85 (.09)</td>
<td>.87 (.07)</td>
<td>.85 (.13)</td>
</tr>
<tr>
<td></td>
<td>ARI</td>
<td>.87 (.06)</td>
<td>.85 (.10)</td>
<td>.87 (.07)</td>
<td>.83 (.13)</td>
</tr>
<tr>
<td>seeds</td>
<td>NMI</td>
<td>.71 (.15)</td>
<td>.70 (.15)</td>
<td>.72 (.14)</td>
<td>.70 (.06)</td>
</tr>
<tr>
<td></td>
<td>ARI</td>
<td>.65 (.18)</td>
<td>.66 (.17)</td>
<td>.67 (.16)</td>
<td>.65 (.05)</td>
</tr>
<tr>
<td>wdbc</td>
<td>NMI</td>
<td>.61 (.10)</td>
<td>.56 (.08)</td>
<td>.58 (.09)</td>
<td>.57 (.11)</td>
</tr>
<tr>
<td></td>
<td>ARI</td>
<td>.70 (.09)</td>
<td>.67 (.07)</td>
<td>.67 (.08)</td>
<td>.67 (.09)</td>
</tr>
<tr>
<td>lung</td>
<td>NMI</td>
<td>.59 (.15)</td>
<td>.44 (.15)</td>
<td>.58 (.17)</td>
<td>NA</td>
</tr>
<tr>
<td></td>
<td>ARI</td>
<td>.48 (.24)</td>
<td>.28 (.15)</td>
<td>.46 (.27)</td>
<td>NA</td>
</tr>
<tr>
<td>carcinomas</td>
<td>NMI</td>
<td>.70 (.08)</td>
<td>.60 (.02)</td>
<td>.71 (.06)</td>
<td>NA</td>
</tr>
<tr>
<td></td>
<td>ARI</td>
<td>.44 (.14)</td>
<td>.39 (.05)</td>
<td>.41 (.12)</td>
<td>NA</td>
</tr>
<tr>
<td>orlraws10P(5)</td>
<td>NMI</td>
<td>.85 (.08)</td>
<td>.82 (.12)</td>
<td>.81 (.07)</td>
<td>NA</td>
</tr>
<tr>
<td></td>
<td>ARI</td>
<td>.62 (.19)</td>
<td>.60 (.24)</td>
<td>.52 (.07)</td>
<td>NA</td>
</tr>
<tr>
<td>orlraws10P(10)</td>
<td>NMI</td>
<td>.88 (.03)</td>
<td>.81 (.07)</td>
<td>.87 (.03)</td>
<td>NA</td>
</tr>
<tr>
<td></td>
<td>ARI</td>
<td>.63 (.09)</td>
<td>.45 (.17)</td>
<td>.60 (.09)</td>
<td>NA</td>
</tr>
<tr>
<td>CMU faces (5)</td>
<td>NMI</td>
<td>.87 (.03)</td>
<td>.87 (.05)</td>
<td>.87 (.03)</td>
<td>NA</td>
</tr>
<tr>
<td></td>
<td>ARI</td>
<td>.71 (.08)</td>
<td>.71 (.11)</td>
<td>.71 (.08)</td>
<td>NA</td>
</tr>
<tr>
<td>CMU faces (10)</td>
<td>NMI</td>
<td>.90 (.02)</td>
<td>.89 (.03)</td>
<td>.89 (.02)</td>
<td>NA</td>
</tr>
<tr>
<td></td>
<td>ARI</td>
<td>.78 (.06)</td>
<td>.76 (.08)</td>
<td>.76 (.05)</td>
<td>NA</td>
</tr>
<tr>
<td>CMU faces (20)</td>
<td>NMI</td>
<td>.91 (.03)</td>
<td>.89 (.03)</td>
<td>.94 (.01)</td>
<td>NA</td>
</tr>
<tr>
<td></td>
<td>ARI</td>
<td>.75 (.08)</td>
<td>.67 (.08)</td>
<td>.83 (.03)</td>
<td>NA</td>
</tr>
<tr>
<td>semeion(2)</td>
<td>NMI</td>
<td>.98 (.04)</td>
<td>.82 (.07)</td>
<td>.85 (.09)</td>
<td>.89 (.06)</td>
</tr>
<tr>
<td></td>
<td>ARI</td>
<td>.99 (.02)</td>
<td>.87 (.06)</td>
<td>.89 (.07)</td>
<td>.93 (.05)</td>
</tr>
<tr>
<td>semeion(5)</td>
<td>NMI</td>
<td>.58 (.11)</td>
<td>.59 (.09)</td>
<td>.61 (.10)</td>
<td>.58 (.05)</td>
</tr>
<tr>
<td></td>
<td>ARI</td>
<td>.50 (.10)</td>
<td>.51 (.11)</td>
<td>.54 (.12)</td>
<td>.49 (.05)</td>
</tr>
<tr>
<td>semeion(10)</td>
<td>NMI</td>
<td>.38 (.05)</td>
<td>.47 (.03)</td>
<td>.50 (.02)</td>
<td>.36 (.16)</td>
</tr>
<tr>
<td></td>
<td>ARI</td>
<td>.20 (.04)</td>
<td>.30 (.03)</td>
<td>.33 (.03)</td>
<td>.16 (.05)</td>
</tr>
</tbody>
</table>

use all features to do clustering, which perform worse. Although decision tree is able to achieve feature selection, the training labels of decision tree is given by GMM that uses all features; therefore, the performance of GMM-DT is also not as good.

I compare the number of features selected for each cluster by GICMFS and LFS in Table 4.2. We can observe that GICMFS selects much less features compared to LFS. This has been discussed in Chapter 3.4.1 with synthetic data that LFS will select feature $f_2$ for clusters 1 and 3 because this feature reveals certain clustering structure. However, GICMFS will not select this feature because $f_2$ does not generate good rules for clusters 1 and 3. GICMFS tends to select less features, but still gives comparable clustering performance, as shown in Table 4.1.

Note that although GICMFS selects less features, the number of features might still
Table 4.2: Number of Features Selected by GICMFS for Each Cluster

<table>
<thead>
<tr>
<th></th>
<th>GICMFS</th>
<th>LFS</th>
</tr>
</thead>
<tbody>
<tr>
<td>iris</td>
<td>3.55 (0.48)</td>
<td>4.0 (0.0)</td>
</tr>
<tr>
<td>wine</td>
<td>4.78 (2.01)</td>
<td>11.8 (1.6)</td>
</tr>
<tr>
<td>seeds</td>
<td>4.60 (0.49)</td>
<td>6.93 (0.25)</td>
</tr>
<tr>
<td>wdbc</td>
<td>13.30 (1.42)</td>
<td>29.08 (0.68)</td>
</tr>
<tr>
<td>lung</td>
<td>756.2 (558.1)</td>
<td>2,218.7 (661.4)</td>
</tr>
<tr>
<td>carcinomas</td>
<td>1,645.3 (1364.3)</td>
<td>5,851.1 (1291.3)</td>
</tr>
<tr>
<td>orlraws10P (5)</td>
<td>4,315.2 (1416.7)</td>
<td>7,235.0 (2121.7)</td>
</tr>
<tr>
<td>orlraws10P (10)</td>
<td>3,224.0 (1412.1)</td>
<td>7,712.70 (1785.7)</td>
</tr>
<tr>
<td>CMU faces (5)</td>
<td>2,405.2 (210.8)</td>
<td>3,549.44 (57.5)</td>
</tr>
<tr>
<td>CMU faces (10)</td>
<td>1,645.7 (281.7)</td>
<td>3,528.8 (119.0)</td>
</tr>
<tr>
<td>CMU faces (20)</td>
<td>276.3 (306.4)</td>
<td>3,524.2 (145.5)</td>
</tr>
<tr>
<td>semeion (2)</td>
<td>121.2 (17.6)</td>
<td>216.7 (1.9)</td>
</tr>
<tr>
<td>semeion (5)</td>
<td>41.4 (25.6)</td>
<td>200.1 (25.3)</td>
</tr>
<tr>
<td>semeion (10)</td>
<td>12.5 (9.9)</td>
<td>189.6 (24.1)</td>
</tr>
</tbody>
</table>

Table 4.3: Rules Discovered by GICMFS for the Wine Dataset

<table>
<thead>
<tr>
<th>Clusters</th>
<th>The Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$f_2 &gt; 2.01$</td>
</tr>
<tr>
<td></td>
<td>$f_6 &lt; 3.61$</td>
</tr>
<tr>
<td></td>
<td>$f_7 &lt; 1.56$, $f_{11} &lt; 4.09$, $f_{12} &lt; 3.45$</td>
</tr>
<tr>
<td>2</td>
<td>$f_1 &lt; 16.16$</td>
</tr>
<tr>
<td></td>
<td>$f_{10} &lt; 1.96$</td>
</tr>
<tr>
<td>3</td>
<td>$f_7 &gt; 2.26$</td>
</tr>
<tr>
<td></td>
<td>$f_{13} &gt; 2.33$</td>
</tr>
</tbody>
</table>

Figure 4.4 Plots of rules with the feature 7 and 10 for the wine data set. The rules are represented using lines with different colors, where the upper bounds are plotted using dash lines and the lower bounds are plotted using solid lines.

be large (for example, exceeding 1,000 in some of the tasks). It might still be difficult to analyze all rules, making the result less interpretable. In this case, we can analyze several top rules that are ranked according to $E_q[\zeta^{(kd)}_k]$, which can be considered as a measure of the quality of each rule.
One of the advantages of GICMFS is that it can select features and generate rules. I describe the rules generated for the wine data set in Table 4.3. Although the data set contains 13 features, the list of rules for each cluster contains either 2 or 5 features. The rules in the table clearly define what characterizes one cluster and how one cluster is distinguished from others. To check how good the rule is, I plot features 7 and 10 in Figure 4.4. We can observe in the figure that the generated rules are satisfied by most samples in that cluster.

Analyzing whether the selected features make sense for the UCI and gene expression data are difficult and require domain knowledge. But we can compare the features selected for image data by plotting the selected features. I plot the features selected for two of the clusters that are identified correctly by both GICMFS and LFS in the orlraws10P (10) task. The plot is shown in Figure 4.5, where the selected features are plotted in black.

From Figure 4.5b and 4.5e, we can easily understand how GICMFS distinguishes these two clusters from the rest. The person in cluster 1 is separated from others because of...
her hair, forehead, eyes, cheek and chin. The person in cluster 2 is distinguished from others because of his hair, eyes and mouth. Note that the rules similar to Table 4.3 are learned for all these selected features, making it straightforward to identifying each person by simply observing the color of the selected region; for example, by checking the color of the hair. As shown in Figure 4.5c and 4.5f, the LFS method selects much more features, since most of the features contain some clustering structure. It is not clear how the clustering decision is made with LFS. One has to estimate the likelihood function before predicting which cluster a test sample belongs to. By comparing the selected features, I believe that GICMFS gives more interpretable clustering results, because these selected features help us to understand each cluster.

As shown in Table 4.2, when I apply GICMFS to separate more clusters in the orlraws10P and faces tasks, the number of selected features is reduced significantly. This is due to the feature selection mechanism of GICMFS. As the number of clusters increases, it is more difficult to find good rules that are able to characterize one cluster; i.e., more features will be similar to $f_2$ as shown in the synthetic example, and are not selected. The LFS method always select similar number of features for each cluster, despite the change of the number of clusters.

We can observe in Table 4.1 that when I apply the GICMFS to Semion (10), i.e., to separate all 10 hand-written digits from each other, the GICMFS is outperformed by GMM and GMM-FS. I believe that this is due to the characteristic of this clustering task. In the face recognition task as demonstrated in Figure 4.5, there exists certain features such as the color of hair and eyes that can be used by itself to distinguish one person from all others, without observing any other features. Rule-based methods are able to identify such features and generate good rules that predict well in the test set. However, in hand-written digit recognition task, such features might not exist, because the color of one certain pixel by itself usually does not imply which digit it is and one needs to observe the rest of the image to identify the digit. In fact, by checking the feature selection results given by the GICMFS for this task, we can observe that no features are selected for several clusters, and these clusters contain no samples. This implies that the GICMFS can not find rules to identify some clusters in the Semion (10) task.
4.2 DReaM Model with Feature Selection (DReaM-FS)

The GICMFS model introduced in Chapter 4.1 is able to generate a list of interpretable clustering rules that might involve different sets of features across different clusters. However, this model might not be best for the COPD subtyping problem. As mentioned in Chapter 3.1, the special structure of the COPD subtyping problem requires the flexibility that separates the features into rule generating subset and cluster-preserving subset. The GICMFS does not allow such flexibility. Therefore, I develop a Discriminative Interpretable Clustering Model with Feature Selection (DReaM-FS).

DReaM-FS introduces additional latent variables that represent whether features are relevant, to extend the DReaM model as introduced in Chapter 3. This allows the DReaM-FS model to automatically select the relevant features, generate a list of interpretable rules that define each of the clusters with hyper-rectangles using the selected features, and separate the samples into clusters based on the cluster-preserving features. In the model, I define each rule by an upper and a lower bound, which is similar to rules in Table 1.1. DReaM-FS assumes that if a sample belongs to one of the clusters, then the rules defined with the selected rule-generating features tend to be satisfied. Whether a rule is satisfied or not is determined by a soft-thresholding function. Given the cluster indicators, DReaM-FS assumes that the cluster-preserving features are generated from a mixture of Gaussian distributions.

4.2.1 The DReaM-FS Model

In DReaM-FS, I assume that the features for each sample can be separated into rule-generating and cluster-preserving features. I use $N$ to denote the number of samples. For a given sample $n \in \{1, \ldots, N\}$, I let $x_n \in \mathbb{R}^D$ be the vector representing the $D$ rule-generating features and $y_n \in \mathbb{R}^L$ be the vector representing the $L$ cluster-preserving features. I use $X = [x_1, \ldots, x_N]$ to denote rule-generating features of all data, and $Y = [y_1, \ldots, y_N]$ to denote cluster-preserving features of all data. Note that, in the analysis of the COPDGene data, I let these two sets be mutually exclusive, but these two feature sets can be identical (as in a traditional clustering paradigm) or overlap depending on the detailed application.
Let $K$ be a pre-defined integer that determines the number of clusters. I define $z_n$ to be the $K$-dimensional binary cluster indicator for sample $n$, such that $z_{nk} = 1$ if and only if the $n$-th sample belongs to cluster $k$; and $z_{nk} = 0$, otherwise.

Given data $X$ and $Y$, the goal is to determine cluster memberships, i.e., $Z = \{z_n\}_{n=1}^N$. In addition, we want to learn a list of interpretable rules that define each cluster using hyper-rectangles based on a subset of selected rule-generating features $X$, we want the cluster membership $Z$ to reveal the clustering structure in the cluster-preserving features $Y$.

4.2.1.1 Defining the Discriminative Rules

As mentioned earlier, I assume that there exist a list of rules for each cluster $k$ using a subset of features. I assume that this list of rules might not use all the rule-generating features. Therefore, I introduce a binary indicator $s_{kd} \in \{0, 1\}$ to represent whether the $d$-th rule-generating feature is used to define the rules for cluster $k$.

Similar to the DReaM model, I represent the rules for cluster $k$ defined with each feature $d$ using a pair of upper and lower bound, such that

$$t_{kd} = [t_{kd}^-, t_{kd}^+]^T. \tag{4.39}$$

I assign a prior distribution for each rule such that

$$p(t_{kd}) \propto \exp \left\{ -\frac{1}{2} \alpha t_{kd}^+ \alpha - \frac{1}{2} \alpha t_{kd}^- \alpha - \frac{1}{2} \beta (t_{kd}^+ - t_{kd}^-)^2 \right\} \tag{4.40}$$

The first two exponential terms in this prior distribution penalize for large $t_{kd}^+ \alpha$ and $t_{kd}^- \alpha$, and the last exponential term penalizes for the difference between $t_{kd}^+$ and $t_{kd}^- \alpha$ (smaller rectangular width is better, providing more compact clusters). $\alpha$ and $\beta$ are the parameters that control the trade-off between these penalties. This prior distribution introduces regularization to the model to prevent over-fitting.

Note that in the DReaM-FS, I define a pair of lower and upper bound $t_{kd}$ for all $k$ and all dimensions $d$. However, only the selected subset of rules with $s_{kd} = 1$ are relevant in determining the cluster membership $Z$. 
4.2.1.2 Clustering and Selecting Features with Discriminative Rules

Given the rule defined for cluster $k$ with the $d$-th rule-generating feature, a sample $n$ may either satisfy the rule or not. I use a binary indicator $r_{nkd} \in \{0, 1\}$ to represent whether the rule defined for cluster $k$ with $d$-th rule-generating feature is satisfied by the sample $n$ or not.

I assume that if a sample $n$ belongs to cluster $k$, then all rules defined for cluster $k$ are likely to be satisfied; while if a sample does not belong to cluster $k$, then all rules defined for cluster $k$ are not likely to be satisfied. Therefore, I let $r_{nkd}$ follow a mixture of Bernoulli distributions such that

$$r_{nkd} \sim \left\{ \text{Bernoulli}(\zeta) \right\}^{z_{nk}s_{kd}} \left\{ \text{Bernoulli}(1 - \zeta) \right\}^{(1 - z_{nk})s_{kd}} \left\{ \text{Bernoulli}\left(\frac{1}{K}\right) \right\}^{1 - s_{kd}}$$

(4.41)

where $\zeta$ is a real parameter with $0 < \zeta < 1$.

In the first two components of Equation (4.41), $\zeta$ defines the conditional probability

$$\zeta = p(r_{nkd} = 1|s_{kd} = 1 \text{ and } z_{nk} = 1) = p(r_{nkd} = 0|s_{kd} = 1 \text{ and } z_{nk} = 0),$$

(4.42)

i.e., $\zeta$ is the probability that the rule satisfaction indicator $r_{nkd}$ is consistent with cluster indicator $z_{nk}$, given that the rule-generating feature $d$ is selected for cluster $k$ (i.e., $s_{kd} = 1$). In the model, I assume that given the rule-generating feature $d$ is selected for cluster $k$ (i.e., $s_{kd} = 1$), the corresponding rules are almost always satisfied (i.e., $r_{nkd} = 1$), if a sample $n$ belongs to cluster $k$ (i.e., $z_{nk} = 1$); and these rules are almost always not satisfied (i.e., $r_{nkd} = 0$), if a sample $n$ does not belong to cluster $k$ (i.e., $z_{nk} = 0$). Therefore, I let $\zeta \approx 1$. I do not enforce $\zeta$ to be exactly 1, because I assume the rules are imperfect and I allow errors. Unless otherwise stated, I choose $\zeta = 0.99$ in the experiments. A more detailed discussion about the choice of this parameter $\zeta$ is discussed in Chapter 4.2.4.3.

The third component of Equation (4.41) indicates that whether the rule is satisfied or not is completely random, if a rule-generating feature $d$ is not selected for cluster $k$ (i.e., $s_{kd} = 0$). In this equation, I assume that the probability that an arbitrary sample belongs to each cluster is equal, i.e. the probability is $1/K$. 
To understand how the model cluster samples according to the defined rules, I derive the posterior distribution of the cluster indicator \( z_n \) given the selected features \( S \) and the rule satisfaction indicator \( r_n \), as follows:

\[
p(z_{nk} = 1 | S, r_n) \propto \exp \left\{ \sum_{d=1}^{D} s_{kd} \left( 2r_{nkd} - 1 \right) \left( \log \zeta - \log(1 - \zeta) \right) \right\}
\]

and the probability is normalized, i.e., \( \sum_{k=1}^{K} p(z_{nk} = 1 | S, r_n) = 1 \).

Note that when \( \zeta \approx 1 \), we have \( \log \zeta - \log(1 - \zeta) > 0 \). Therefore, Equation (4.43) reveals the mechanism on how the model decides which cluster a sample belongs to. If sample \( n \) satisfies more rules defined for cluster \( k \) compared to other clusters (i.e., \( s_{kd} = 1 \) and \( r_{nkd} = 1 \)), then it is more likely that sample \( n \) belongs to cluster \( k \).

To understand how the model selects features, I derive the posterior distribution for \( s_{kd} \) given \( r_{kd} = \{ r_{nkd} \}_{n=1}^{N} \) and \( z_n \) as follows:

\[
p(s_{kd} = 1 | r_{kd}, z_n) = \frac{\exp(\psi_{kd})}{1 + \exp(\psi_{kd})}
\]

where \( \psi_{kd} \) is defined as

\[
\psi_{kd} = \sum_{n=1}^{N} \left( z_{nk} r_{nkd} + (1 - z_{nk})(1 - r_{nkd}) \right) \log \zeta \\
+ \sum_{n=1}^{N} \left( z_{nk}(1 - r_{nkd}) + (1 - z_{nk})r_{nkd} \right) \log(1 - \zeta) \\
- \sum_{n=1}^{N} r_{nkd} \log \frac{1}{K} - \sum_{n=1}^{N} (1 - r_{nkd}) \log \frac{K - 1}{K}
\]

Since \( \zeta \approx 1 \), we have \( \log \zeta > \log(1 - \zeta) \). Therefore, the first two terms in Equation (4.45) measure how well \( r_{nkd} \) predicts \( z_{nk} \), and the last two terms give the negative log-likelihood assuming \( r_{nkd} \) is random. This reveals the mechanism on how the model selects features: if the rule associated with a feature \( d \) better distinguishes the cluster \( k \) from the rest than random, then it is more likely that the feature \( d \) is selected for cluster \( k \) (i.e., it is more likely that \( s_{kd} = 1 \)).
Interpretable Clustering with Feature Selection

Figure 4.6 Plots for the soft-thresholding function \( g(x_{nd}, t_{kd}) \) with different parameter \( a \), where I let \( t_{kd} = [-3, 3] \). As \( a \to \infty \), \( g(x_{nd}, t_{kd}) \to f(x_{nd}, t_{kd}) \).

4.2.1.3 Determining Whether a Rule is Satisfied

In the previous section, I have defined the binary rule satisfaction indicators \( r_{nktd} \), and the rules \( t_{kd} \). Now I introduce how to determine \( r_{nktd} \) based on the observed data \( x_{nd} \) and the rule \( t_{kd} \).

A straightforward way to determine \( r_{nktd} \) based on \( x_{nd} \) and \( t_{kd} \) is to define a thresholding function \( f(x_{nd}, t_{kd}) \) such that

\[
\begin{align*}
    r_{nktd} &= f(x_{nd}, t_{kd}) = \\
    &\begin{cases} 
    1, & \text{if } t_{kd}^- \leq x_{nd} \leq t_{kd}^+ \\
    0, & \text{otherwise},
    \end{cases}
\end{align*}
\]  

(4.46)

However, this function is not differentiable, and therefore it would be difficult to optimize \( t_{kd} \) to learn the optimal rules.

To ameliorate this problem, I apply the same strategy as the DReaM as described in Chapter 3 that define a differentiable function \( g(x_{nd}, t_{kd}) \) to approximate \( f(x_{nd}, t_{kd}) \), defined as

\[
\begin{align*}
    g(x_{nd}, t_{kd}) &= \frac{1}{\left(1 + \exp\{-a(x - t^-)\}\right) \left(1 + \exp\{-a(t^+ - x)\}\right)}, \\
\end{align*}
\]  

(4.47)

where \( a \) is a positive parameter that controls the “smoothness” of this function. As shown in Figure 4.6, as \( a \to \infty \), \( g(x_{nd}, t_{kd}) \to f(x_{nd}, t_{kd}) \). Unless otherwise stated, I choose \( a = 100 \) in the experiments. A more detailed discussion about the choice of this parameter \( a \) is discussed in Chapter 4.2.4.3.
Now I need to connect $r_{nkd}$ with $g(x_{nd}, t_{kd})$. Similar to the DReaM model, I introduce a random variable $\phi_{nkd}$ that is always observed as 1. I assume that $\phi_{nkd}$ follows a Bernoulli distribution such that

$$\phi_{nkd} \sim \text{Bernoulli}(\eta_{nkd})$$

(4.48)

where $\eta_{nkd}$ is defined as

$$\eta_{nkd} = \left\{ g(x_n, t_{kd}) \right\}^{r_{nkd}} \frac{1}{1-r_{nkd}}$$

(4.49)

After introducing this variable, the log-posterior distribution is given as

$$\log p(r_{nkd}, x_n, t_{kd} | \phi_{nkd} = 1) = r_{nkd} \log g(x_n, t_{kd}) + \frac{1}{K-1} (1-r_{nkd}) \log (1-g(x_n, t_{kd}))$$

(4.50)

This equation enforces $r_{nkd}$ to be consistent with $g(x_n)$, because

$$p \left( r_{nkd} = 1 \text{ and } g(x_n, t_{kd}) \approx 0 \right| \phi_{nkd} = 1) \approx 0, \text{ and}$$

$$p \left( r_{nkd} = 0 \text{ and } g(x_n, t_{kd}) \approx 1 \right| \phi_{nkd} = 1) \approx 0$$

(4.51)

i.e., it is almost impossible that $r_{nkd}$ is inconsistent with $g(x_n)$.

In Equation (4.50), I introduce a factor $1/(K-1)$, because if each cluster is of equal size, then there are $(K-1)$ times as many samples with $r_{nkd} = 0$ as samples with $r_{nkd} = 1$. This factor balances the contribution of samples with $r_{nkd} = 0$ and $r_{nkd} = 1$.

### 4.2.2 Training DReaM-FS with Expectation Maximization

In Chapter 4.2.1, I have presented our model. In this section, I introduce how I train the model. In this paper, I learn the Maximum a Posteriori Probability (MAP) estimator for $T$ through Expectation Maximization (EM) [32].

**Expectation Step**

In the expectation step, I first take the expected value of the log joint probability as a function of $T$, with respect to the posterior distribution $p(\Theta, R, S, Z|\Phi, X, Y, \hat{T})$, where
\( \hat{T} \) is the current estimate of \( T \). This expected value can be written as

\[
\hat{h}(T) = \mathbb{E}[\log p(X, Y, \Phi, R, S, T, Z|\alpha_{t0}, \beta_{t0}, \zeta, \eta)]
\]

\[
= \sum_{k=1}^{K} \sum_{d=1}^{D} \left\{ \log p(t_{kd}|\alpha, \beta_{t0}) + \sum_{n=1}^{N} \mathbb{E}[\log p(\phi_{nk}|x_{nd}, t_{kd}, r_{nk})] \right\} + \text{const} \tag{4.52}
\]

where \( \text{const} \) represents all terms that are not functions of \( T \).

**Maximization Step**

Then, in the maximization step, I update the estimate of \( T \) by maximizing \( \hat{h}(T) \), i.e.,

\[
\hat{T} = \arg \max_T h(T).
\]

As shown in Equation (4.52), \( t_{kd} \) does not depend on other terms \( t_{ij} \) for all \( i \neq k \) and \( j \neq d \). Therefore, I can compute the optimal \( t_{kd} \) with each \( k \) and \( d \) separately, such that

\[
\hat{t}_{kd} = \arg \max_{t_{kd}} \left\{ -\frac{1}{2} \alpha_{t0} \left( t_{kd}^2 \right) - \frac{1}{2} \beta_{t0} \left( t_{kd}^2 - t_{kd} \right)^2 \right. \\
+ \left. \sum_{n=1}^{N} \mathbb{E}[r_{nk}] \log g(x_n, t_{kd}) \right. \\
+ \left. \frac{1}{K-1} \sum_{n=1}^{N} \left( 1 - \mathbb{E}[r_{nk}] \right) \log(1 - g(x_n, t_{kd})) \right\} \tag{4.53}
\]

I solve this optimization problem by the conjugate gradient method [44].

**Approximating the Posterior Distribution**

In the expectation step, I need to derive the posterior distribution \( p(\Theta, R, S, Z|\Phi, X, Y, \hat{T}) \). However, this distribution is computationally intractable. I use a variational distribution \( q(\Theta, R, S, Z) \) to approximate it such that

\[
q(\Theta, R, S, Z) \approx p(\Theta, R, S, Z|\Phi, X, Y, \hat{T}). \tag{4.54}
\]

I make \( q(\Theta, R, S, Z) \) tractable by applying mean-field approximation such that it is factorized as

\[
q(\Theta, R, S, Z) = \prod_{k=1}^{K} q(\theta_k) \prod_{n=1}^{N} \prod_{k=1}^{K} \prod_{d=1}^{D} q(r_{nk}) \prod_{k=1}^{K} \prod_{d=1}^{D} q(s_{kd}) \prod_{n=1}^{N} q(z_n) \tag{4.55}
\]

To make sure the variational distribution \( q \) is a good approximation for the posterior distribution, I update \( q \) such that the KL divergence between the variational distribution and the posterior distribution, denoted as \( KL(q||p) \), is minimized. As described in
Appendix A, I minimize $KL(q||p)$ by applying variational calculus. I omit the details of derivation, and list the update equations as follows.

The update equation for $q(\theta_k)$ is given as

$$q(\theta_k) \sim \prod_{d=1}^{D} \mathcal{NIG}(\lambda_{kd}, m_{kd}, \alpha_{kd}, \beta_{kd}), \quad (4.56)$$

where

$$\lambda_{kd} = \lambda_0 + \sum_{n=1}^{N} \mathbb{E}_q[z_{nk}]$$

$$m_{kd} = \frac{\lambda_0 m_0 + \sum_{n=1}^{N} \mathbb{E}_q[z_{nk}] y_{nd}}{\lambda_0 + \sum_{n=1}^{N} \mathbb{E}_q[z_{nk}]}$$

$$\alpha_{kd} = \alpha_0 + \frac{1}{2} \sum_{n=1}^{N} \mathbb{E}_q[z_{nk}]$$

$$\beta_{kd} = \beta_0 + \frac{1}{2} \sum_{n=1}^{N} \mathbb{E}_q[z_{nk}] y_{nd}^2 + \frac{1}{2} \lambda_0 m_0^2 - \frac{1}{2} (\lambda_0 + \sum_{n=1}^{N} \mathbb{E}_q[z_{nk}]) m_{kd}^2$$

The update equation for $q(r_{nk})$ is given by

$$q(r_{nk}) = \text{Categorical}(v_{nk})$$

$$v_{nk} = \frac{b_{nk}}{1 + b_{nk}}$$

$$b_{nk} = \exp \left\{ \mathbb{E}_q[s_{kd}](2 \mathbb{E}_q[z_{nk}] - 1)(\log \zeta - \log(1 - \zeta) - (1 - \mathbb{E}_q[s_{kd}]) \log (K - 1) \\
+ \log g(x_n, \hat{t}_{kd}) - \frac{1}{K-1} \log(1 - g(x_n, \hat{t}_{kd})) \right\}$$

The update equation for $q(s_{kd})$ is given by

$$q(s_{kd}) \sim \text{Bernoulli}(\rho_{kd})$$

where $\rho_{kd}$ is defined as

$$\rho_{kd} = \frac{c_{kd}}{1 + c_{kd}}$$
and \( c_{kd} \) is defined as

\[
c_{kd} = \exp \left\{ \sum_{n=1}^{N} \left( \mathbb{E}_q[z_{nk}]\mathbb{E}_q[r_{nkd}] + (1 - \mathbb{E}_q[z_{nk}])(1 - \mathbb{E}_q[r_{nkd}]) \right) \log \zeta \\
+ \sum_{n=1}^{N} \left( \mathbb{E}_q[z_{nk}] - \mathbb{E}_q[z_{nk}]\mathbb{E}_q[r_{nkd}] \right) \log (1 - \zeta) \right\} (4.63)
\]

The update equation for \( q(z_n) \) is given by

\[
q(z_n) \sim \text{Categorical}(\pi_n) \tag{4.64}
\]

where \( \pi_n \) is a \( K \)-dimensional vector, with each element defined as

\[
\pi_{nk} \propto \exp \left\{ \sum_{d=1}^{D} \mathbb{E}_q[s_{kd}] \left( 2\mathbb{E}_q[r_{nkd}] - 1 \right) \left( \log \zeta - \log (1 - \zeta) \right) \\
+ \sum_{d=1}^{D} \left( -\frac{1}{2} \mathbb{E}_q[\log \sigma_{kd}^2] - \frac{1}{2} \mathbb{E}_q \left[ \frac{(y_{nd} - \mu_{kd})^2}{\sigma_{kd}^2} \right] \right) \right\}, \tag{4.65}
\]

and \( \pi_{nk} \) is normalized such that \( \sum_{k=1}^{K} \pi_{nk} = 1 \).

In the update equations, \( \mathbb{E}_q \) represents that the expected value is taken with respect to the variational distribution \( q \). Since all the distributions \( q \) are common distributions in exponential family, it is straightforward to express these expected values using parameters of the corresponding distributions. The Expectation Maximization Algorithm is summarized in Algorithm 4.

### 4.2.3 Clustering with Discriminative Rules

With the algorithm described in Chapter 4.2.2, I learn the MAP estimator for the decision boundaries \( \hat{t}_{kd} \) and the variational distribution \( q \). I use a similar strategy as described in Chapter 4.1.3 to determine which cluster each test sample belongs to based on these results.

First, I identify the subset of features that are used to define each cluster. Since \( s_{kd} \) is the indicator whether a feature is selected, I simply observe whether the value \( \mathbb{E}_q[s_{kd}] \)
Algorithm 4 Expectation Maximization Algorithm for DReaM-FS

Randomly initialize the linear parameters \( \hat{T} \) and the factorized variational distributions \( q \).

repeat
  for \( k \leftarrow 1 \) to \( K \) do
    update \( q(\Theta_k) \) according to Equation (4.56).
  end for
  for \( n \leftarrow 1 \) to \( N \) do
    for \( k \leftarrow 1 \) to \( K \) do
      for \( d \leftarrow 1 \) to \( D \) do
        update \( q(r_{nkd}) \) according to Equation (4.58).
      end for
    end for
  end for
  for \( k \leftarrow 1 \) to \( K \) do
    for \( d \leftarrow 1 \) to \( D \) do
      update \( q(s_{kd}) \) according to Equation (4.61).
    end for
  end for
  for \( n \leftarrow 1 \) to \( N \) do
    update \( q(z_n) \) according to Equation (4.64).
  end for
  for \( k \leftarrow 1 \) to \( K \) do
    for \( d \leftarrow 1 \) to \( D \) do
      Update \( \hat{t}_{kd} \) based on the Equation (4.53) using conjugate gradient method.
    end for
  end for
until Converges

is greater than 0.5, where \( E_q \) represents the expected value is taken with respect to variational distribution \( q \); i.e., the subset of features selected for cluster \( k \), denoted by \( S_k \) is given by

\[
S_k = \left\{ d \in \{1, \ldots, D\} \left| E_q[s_{kd}] > 0.5 \right. \right\}. \tag{4.66}
\]

Then, I can determine which cluster \( k^* \) a test sample \( x^* \) belongs to by comparing the following ratio:

\[
k^* = \arg \max_{k \in \{1, \ldots, K\}} \frac{\sum_{d \in S_k} \mathbb{1}(\hat{t}_{kd}^+ < x^*_d < \hat{t}_{kd}^-)}{|S_k|}, \tag{4.67}
\]

where \( \mathbb{1}(\cdot) \) is a indicator function that returns 1 if the statement is true; and 0 otherwise. In this equation, the numerator is the number of rules for cluster \( k \) that are satisfied by \( x^* \). The denominator \( |S_k| \) is the number of rules that are used to defined cluster \( k \). Note that the value \( |S_k| \) might differ significantly across different clusters \( k \). Therefore,
simply counting the number of rules satisfied (i.e., the numerator) might cause bias. I predict with the ratio (normalization with $|S_k|$) to avoid such bias.

4.2.4 Experiments with DReaM-FS

4.2.4.1 Testing DReaM-FS with Benchmark Data

I first test DReaM-FS using benchmark data, including 4 UCI data sets [45]: iris, wine, breast, wdbc; 4 human faces data sets: CMU faces, orlraws10p, pixraw10p, Yale; and 1 hand-written digit data set: USPS. The details of these dataset are summarized in Appendix B.

I test DReaM-FS under traditional clustering settings (the same setting as k-means and GMM), i.e. I let the rule-generating features be identical as the cluster-preserving features, such that $X = Y$. I compare the performance of the DReaM-FS with existing methods. I choose the Gaussian Mixture Model (GMM) [9] and KMeans [3] as baselines. Since the DReaM-FS is a local feature selection method, i.e., we allow the features selected to differ across different clusters, I include the local feature selection version of [35], denoted as LFS. LFS selects features that reveal a clustering structure but cannot generate interpretable rules. I also compare the DReaM-FS with DReaM. I also include a simple rule generating clustering method I call GMM+DT, where I use the GMM clustering results as labels to train a decision tree.

Since the DReaM-FS generates rules, we want to know how accurately these rules will perform in a predictive task. Therefore, I conduct a 5-fold cross validation and compare the predictive cluster labels in the validation sets with the ground-truth labels. I measure the performance using Normalized Mutual Information (NMI) and Adjusted Rand Index (ARI), where NMI is the normalized version of mutual information and ARI is the corrected-for-chance version of the Rand index. The definitions of NMI and ARI are given in Appendix C. In Table 4.4, I report the mean value and standard deviation of the 5-fold cross-validation NMI and ARI results on each clustering task. The values in bold are the highest mean NMI or ARI value for each task.

To begin with, I analyze the clustering performance on the UCI data sets and the human faces data sets. We can observe from Table 4.4 that DReaM-FS and LFS are the two
Table 4.4: Clustering Performance of DReaM-FS on Benchmark Data

<table>
<thead>
<tr>
<th></th>
<th>DReaM-FS</th>
<th>GMM</th>
<th>KMeans</th>
<th>DReaM</th>
<th>LFS [35]</th>
<th>GMM+DT</th>
</tr>
</thead>
<tbody>
<tr>
<td>iris</td>
<td>NMI</td>
<td>0.896</td>
<td>0.810</td>
<td>0.772</td>
<td>0.789</td>
<td>0.797</td>
</tr>
<tr>
<td></td>
<td>(0.126)</td>
<td>(0.163)</td>
<td>(0.146)</td>
<td>(0.070)</td>
<td>(0.115)</td>
<td>(0.084)</td>
</tr>
<tr>
<td>ARI</td>
<td>0.854</td>
<td>0.754</td>
<td>0.720</td>
<td>0.718</td>
<td>0.777</td>
<td>0.715</td>
</tr>
<tr>
<td></td>
<td>(0.163)</td>
<td>(0.163)</td>
<td>(0.146)</td>
<td>(0.146)</td>
<td>(0.150)</td>
<td>(0.153)</td>
</tr>
<tr>
<td>wine</td>
<td>NMI</td>
<td>0.786</td>
<td>0.770</td>
<td>0.466</td>
<td>0.796</td>
<td>0.794</td>
</tr>
<tr>
<td></td>
<td>(0.118)</td>
<td>(0.101)</td>
<td>(0.007)</td>
<td>(0.137)</td>
<td>(0.069)</td>
<td>(0.110)</td>
</tr>
<tr>
<td>ARI</td>
<td>0.747</td>
<td>0.710</td>
<td>0.362</td>
<td>0.710</td>
<td>0.722</td>
<td>0.744</td>
</tr>
<tr>
<td></td>
<td>(0.144)</td>
<td>(0.148)</td>
<td>(0.108)</td>
<td>(0.148)</td>
<td>(0.091)</td>
<td>(0.131)</td>
</tr>
<tr>
<td>breast</td>
<td>NMI</td>
<td>0.661</td>
<td>0.614</td>
<td>0.441</td>
<td>0.657</td>
<td>0.652</td>
</tr>
<tr>
<td></td>
<td>(0.109)</td>
<td>(0.090)</td>
<td>(0.007)</td>
<td>(0.063)</td>
<td>(0.035)</td>
<td>(0.079)</td>
</tr>
<tr>
<td>ARI</td>
<td>0.388</td>
<td>0.441</td>
<td>0.145</td>
<td>0.344</td>
<td>0.340</td>
<td>0.348</td>
</tr>
<tr>
<td></td>
<td>(0.144)</td>
<td>(0.035)</td>
<td>(0.083)</td>
<td>(0.083)</td>
<td>(0.102)</td>
<td>(0.150)</td>
</tr>
<tr>
<td>wdbc</td>
<td>NMI</td>
<td>0.625</td>
<td>0.584</td>
<td>0.468</td>
<td>0.648</td>
<td>0.652</td>
</tr>
<tr>
<td></td>
<td>(0.076)</td>
<td>(0.081)</td>
<td>(0.030)</td>
<td>(0.077)</td>
<td>(0.038)</td>
<td>(0.116)</td>
</tr>
<tr>
<td>ARI</td>
<td>0.722</td>
<td>0.660</td>
<td>0.485</td>
<td>0.651</td>
<td>0.679</td>
<td>0.631</td>
</tr>
<tr>
<td></td>
<td>(0.061)</td>
<td>(0.061)</td>
<td>(0.015)</td>
<td>(0.058)</td>
<td>(0.023)</td>
<td>(0.052)</td>
</tr>
<tr>
<td>CMU faces</td>
<td>NMI</td>
<td>0.913</td>
<td>0.774</td>
<td>0.778</td>
<td>0.883</td>
<td>0.910</td>
</tr>
<tr>
<td></td>
<td>(0.017)</td>
<td>(0.008)</td>
<td>(0.019)</td>
<td>(0.019)</td>
<td>(0.013)</td>
<td>(0.022)</td>
</tr>
<tr>
<td>ARI</td>
<td>0.745</td>
<td>0.456</td>
<td>0.494</td>
<td>0.663</td>
<td>0.769</td>
<td>0.437</td>
</tr>
<tr>
<td></td>
<td>(0.027)</td>
<td>(0.025)</td>
<td>(0.049)</td>
<td>(0.077)</td>
<td>(0.048)</td>
<td>(0.005)</td>
</tr>
<tr>
<td>orlraws10p</td>
<td>NMI</td>
<td>0.852</td>
<td>0.740</td>
<td>0.864</td>
<td>0.852</td>
<td>0.890</td>
</tr>
<tr>
<td></td>
<td>(0.025)</td>
<td>(0.065)</td>
<td>(0.036)</td>
<td>(0.045)</td>
<td>(0.015)</td>
<td>(0.032)</td>
</tr>
<tr>
<td>ARI</td>
<td>0.582</td>
<td>0.408</td>
<td>0.595</td>
<td>0.482</td>
<td>0.686</td>
<td>0.326</td>
</tr>
<tr>
<td></td>
<td>(0.119)</td>
<td>(0.203)</td>
<td>(0.150)</td>
<td>(0.215)</td>
<td>(0.048)</td>
<td>(0.168)</td>
</tr>
<tr>
<td>pixraw10p</td>
<td>NMI</td>
<td>0.919</td>
<td>0.753</td>
<td>0.913</td>
<td>0.821</td>
<td>0.914</td>
</tr>
<tr>
<td></td>
<td>(0.036)</td>
<td>(0.083)</td>
<td>(0.043)</td>
<td>(0.053)</td>
<td>(0.045)</td>
<td>(0.002)</td>
</tr>
<tr>
<td>ARI</td>
<td>0.786</td>
<td>0.331</td>
<td>0.747</td>
<td>0.586</td>
<td>0.813</td>
<td>0.539</td>
</tr>
<tr>
<td></td>
<td>(0.123)</td>
<td>(0.145)</td>
<td>(0.152)</td>
<td>(0.193)</td>
<td>(0.163)</td>
<td>(0.052)</td>
</tr>
<tr>
<td>Yale</td>
<td>NMI</td>
<td>0.659</td>
<td>0.661</td>
<td>0.663</td>
<td>0.538</td>
<td>0.524</td>
</tr>
<tr>
<td></td>
<td>(0.031)</td>
<td>(0.064)</td>
<td>(0.028)</td>
<td>(0.080)</td>
<td>(0.029)</td>
<td>(0.051)</td>
</tr>
<tr>
<td>ARI</td>
<td>0.195</td>
<td>0.176</td>
<td>0.160</td>
<td>0.006</td>
<td>0.022</td>
<td>0.065</td>
</tr>
<tr>
<td></td>
<td>(0.074)</td>
<td>(0.095)</td>
<td>(0.059)</td>
<td>(0.005)</td>
<td>(0.108)</td>
<td>(0.070)</td>
</tr>
<tr>
<td>USPS</td>
<td>NMI</td>
<td>0.267</td>
<td>0.476</td>
<td>0.619</td>
<td>0.487</td>
<td>0.491</td>
</tr>
<tr>
<td></td>
<td>(0.030)</td>
<td>(0.004)</td>
<td>(0.001)</td>
<td>(0.014)</td>
<td>(0.008)</td>
<td>(0.016)</td>
</tr>
<tr>
<td>ARI</td>
<td>0.073</td>
<td>0.348</td>
<td>0.535</td>
<td>0.348</td>
<td>0.369</td>
<td>0.328</td>
</tr>
<tr>
<td></td>
<td>(0.035)</td>
<td>(0.005)</td>
<td>(0.002)</td>
<td>(0.025)</td>
<td>(0.012)</td>
<td>(0.017)</td>
</tr>
</tbody>
</table>

Table 4.5: Rules Discovered by DReaM-FS for the Wine Dataset

<table>
<thead>
<tr>
<th>Cluster</th>
<th>Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.30 &lt; x_7^* &lt; 1.30, 0.47 &lt; x_{11} &lt; 0.85, 1.25 &lt; x_{12} &lt; 2.15</td>
</tr>
<tr>
<td>2</td>
<td>2.31 &lt; x_7^* &lt; 5.11, 759 &lt; x_{13} &lt; 1688</td>
</tr>
<tr>
<td>3</td>
<td>11.01 &lt; x_1 &lt; 12.73, 1.20 &lt; x_{10} &lt; 3.87</td>
</tr>
</tbody>
</table>

The performance of these two methods are comparable, and are better than the rest of the methods. This demonstrates that feature selection helps to improve the performance in these clustering tasks. Kmeans, GMM and DReaM uses all features to do clustering, which perform worse. Although the decision tree is able to achieve feature selection, the training labels for the decision tree is given by GMM that uses all features; the performance of GMM+DT is also not as good.

One of the advantages of DReaM-FS is that it generates a list of interpretable rules. I describe the rules generated for the wine data set in Table 4.5. Although the data set contains 13 features, the list of rules for each cluster contains either 2 or 3 features. The rules in the table clearly define which feature characterizes one cluster and how one cluster is distinguished from others.

To better illustrate the rules generated, I plot some of the rules discovered for the wine data set in Figure 4.7. As shown in Figure 4.7a and 4.7b, most samples that belong to one cluster are located inside the corresponding hyper-rectangle, and most samples that do not belong to this cluster are located outside this hyper-rectangle. I also want to highlight that as plotted in Figure 4.7a, I do not plot boundaries corresponding to
feature 11 for cluster 2, indicating that this is not selected for cluster 2. As shown in Equation (4.45), I select the features based on whether the rule can better distinguish one cluster from the rest than random. We can not use feature 11 to distinguish cluster 3 from cluster 2. Therefore, this feature is not selected to define cluster 2.

We can observe in Table 4.4 that DReaM-FS does not perform well on the USPS handwritten digit data set. I believe that this is due to the characteristic of this clustering task. As demonstrated in Figure 4.7, the model selects features such as feature 7 that distinguish one cluster from the rest, without observing any other features. However, in hand-written digit recognition task, such features might not exist, because the gray-scale value of one certain pixel by itself usually does not imply which digit it is and one needs to observe the rest of the image to identify the digit.

4.2.4.2 Applying DReaM-FS on COPD Subtyping Problem

I develop DReaM-FS for solving a real-world COPDGene disease subtyping problem. We want to generate rules based on 27 features related to clinical information, lung function and computed tomography (CT) chest imaging. These 27 features are the rule-generating features $X$ for the DReaM-FS. We want to discover clusters that are clinically relevant and related to disease progression. I measure disease progression using the change of adjusted lung density (denoted as “adj density”) and change of 15th percentile point of the lung density histogram (denoted as “Perc15”) during the a 5-year period. These are the cluster-preserving features $Y$ for DReaM-FS. “Adj density” and
Table 4.6: Rules Discovered by DReaM-FS for COPD Subtyping Problem

<table>
<thead>
<tr>
<th>Cluster</th>
<th>Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$-0.50 &lt; \text{Tot Emph} &lt; 0.60$</td>
</tr>
<tr>
<td></td>
<td>$0.00 &lt; \log \text{Tot Emph} &lt; 0.42$</td>
</tr>
<tr>
<td></td>
<td>$0.03 &lt; \log \text{RML} &lt; 0.51$</td>
</tr>
<tr>
<td></td>
<td>$0.00 &lt; \log \text{LUL} &lt; 0.49$</td>
</tr>
<tr>
<td></td>
<td>$-0.46 &lt; \text{DistDiff} &lt; 0.55$</td>
</tr>
<tr>
<td>2</td>
<td>$11.85 &lt; \text{Tot Emph} &lt; 60.90$</td>
</tr>
<tr>
<td></td>
<td>$2.66 &lt; \log \text{RUL} &lt; 4.40$</td>
</tr>
<tr>
<td></td>
<td>$2.37 &lt; \log \text{LLL} &lt; 4.15$</td>
</tr>
<tr>
<td></td>
<td>$0.23 &lt; \text{FEV1/FVC} &lt; 0.49$</td>
</tr>
<tr>
<td>3</td>
<td>$1.46 &lt; \log \text{Tot Emph} &lt; 4.36$</td>
</tr>
<tr>
<td></td>
<td>$0.71 &lt; \log \text{RUL} &lt; 1.45$</td>
</tr>
<tr>
<td></td>
<td>$1.05 &lt; \log \text{LUL} &lt; 1.77$</td>
</tr>
<tr>
<td>4</td>
<td>$1.55 &lt; \log \text{UL Ratio} &lt; 2.93$</td>
</tr>
<tr>
<td></td>
<td>$0.48 &lt; \log \text{RUL Residuals} &lt; 1.24$</td>
</tr>
<tr>
<td></td>
<td>$-2.08 &lt; \log \text{RLL Residuals} &lt; -0.52$</td>
</tr>
<tr>
<td>5</td>
<td>$0.40 &lt; \log \text{Tot Emph} &lt; 1.63$</td>
</tr>
<tr>
<td></td>
<td>$0.29 &lt; \log \text{RUL} &lt; 0.77$</td>
</tr>
<tr>
<td></td>
<td>$0.23 &lt; \log \text{RLL} &lt; 0.71$</td>
</tr>
<tr>
<td></td>
<td>$0.34 &lt; \log \text{LLL} &lt; 0.87$</td>
</tr>
<tr>
<td>6</td>
<td>$4.17 &lt; \log \text{Tot Emph} &lt; 12.08$</td>
</tr>
<tr>
<td></td>
<td>$1.47 &lt; \log \text{RUL} &lt; 2.70$</td>
</tr>
<tr>
<td></td>
<td>$1.44 &lt; \log \text{RLL} &lt; 2.45$</td>
</tr>
<tr>
<td></td>
<td>$1.56 &lt; \log \text{LLL} &lt; 2.49$</td>
</tr>
</tbody>
</table>

"perc15" are two features that describe the severity of emphysema for a subject. The change of these values describes the 5-year progression of emphysema for the subject.

I remove the subjects with missing values and separate the remaining 3,073 subjects into training and test sets of equal sizes. I apply DReaM-FS on the training set to separate the subjects into 6 clusters. The rules discovered by DReaM-FS is described in Table 4.6. I choose the number of clusters to be 6 because as shown in Table 1.1, the widely used GOLD Categories for the subjects have 6 categories and we want to fairly compare our results with GOLD Categories.

In Table 4.6, we can observe that cluster 1, 2, 3, 5 and 6 are separated according to the severity of emphysema in the lung, where “Tot Emph” measures the percent of emphysema in the lung, “log Tot Emph” is the logarithm of “Tot Emph” and “log RUL”, “log RML”, “log RLL”, “log LUL” and “log LLL” represent the logarithm of the percent of emphysema in different sections (lobes) in the lung, respectively.

For the cluster 2, which has the largest “Tot Emph” value, we also observe that the
feature “FEV1 % predicted”, “FEV1/FVC” and “FEF2575” are also selected. These values are spirometry measures describing the lung function of the subjects. Since these values are relatively low for cluster 2, we can conclude that cluster 2 represents the subjects with heavy emphysema and they are sick with reduced lung function.

We can also observe that cluster 4 is characterized by how emphysema is distributed in the lung, where “log UL Ratio” is the logarithm of the ratio of the amount of emphysema in the upper lobe to that in the lower lobe. “Dissdiff” is the absolute difference between the amount of emphysema in the upper lung lobes and the emphysema in the lower lung lobes. The “residuals” features represent the residuals after regressing the lobar emphysema values on total lung emphysema. It represents the “lobe-specific” emphysema that is not in keeping with the amount of emphysema in the rest of the lung.

The rule associated with cluster 4 indicates that it is an upper-lobe predominant cluster, characterized by positive “log UL Ratio” and “log RUL Residuals”, as well as negative “log RML Residuals”, “log RLL Residuals” and “log LLL Residuals” values.

In the example above, I demonstrate that with the rules generated by DReaM-FS, it is easier for us to understand each of the clusters discovered. It would be more difficult to understand the clustering results, if each cluster is described using mean and co-variance.

We want the clustering results to be associated with 5-year progression of emphysema. Therefore, I apply the rules in Table 4.6 to subjects in the test set. I summarize the mean and standard deviation of the 5-year progression variables in different clusters in Table 4.7. From Table 4.7, we can observe that cluster 4, is one of the clusters with the most negative mean values in “change perc15” and the second most negative mean values in “change adj density”. This might imply that the upper-lobe predominant subjects are associated with the most 5-year progression of emphysema, which is worthy of further investigation. Note that this result is consistent with the recent COPD research described in [50].

I would like to compare the results given by DReaM-FS with existing methods mentioned in Chapter 4.2.4.1. Therefore, I also train these methods with the COPDGene training set. Since KMeans, GMM, GMM+DT and LFS do not separate features into rule-generating and cluster-preserving features, I train these models using only rule-generating features, such that these methods can properly predict clusters in the test set. I train the DReaM model with both rule-generating and cluster-preserving features.
I predict which cluster a subject belongs to using the trained models. I also apply the GOLD categories described in Table 1.1 to the subjects in the test set for comparison purposes.

For the COPD subtype discovery application, we do not have the ground-truth labels. Therefore, I cannot compute the NMI or ARI values. I adopt an alternative way by measuring whether subjects in different clusters differ in terms of the following 4 values: “Change adj density”, “Change perc15”, “COPD score” and “Emph score”. As mentioned before, the first two variables are used to measure the 5-year progression of emphysema. The last two variables are genetic risk scores that measure the accumulation of genetic risk to different aspects of COPD [51, 52]. In the COPD subtype discovery application, we want each cluster to differ in terms of 5-year emphysema progression, such that we can identify groups of subjects who has more emphysema progression over time; we also want the genetic risk score to differ such that we may highlight biologic differences between clusters.

I measure how the 4 variables differ across different clusters by conducting a Kruskal-Wallis one-way analysis of variance [46], which is a non-parametric method for testing whether samples originate from the same distribution. The p-values for each of these

### Table 4.7: Cluster-Preserving Features for Different Clusters

<table>
<thead>
<tr>
<th>Cluster</th>
<th>Change adj density</th>
<th>Change Perc15</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-3.98 (12.5)</td>
<td>-8.48 (22.7)</td>
</tr>
<tr>
<td>2</td>
<td>-0.38 (10.6)</td>
<td>0.28 (10.7)</td>
</tr>
<tr>
<td>3</td>
<td>1.61 (11.3)</td>
<td>2.58 (13.4)</td>
</tr>
<tr>
<td>4</td>
<td>-6.30 (12.7)</td>
<td>-7.83 (15.0)</td>
</tr>
<tr>
<td>5</td>
<td>-2.16 (10.0)</td>
<td>-0.69 (15.7)</td>
</tr>
<tr>
<td>6</td>
<td>4.71 (12.4)</td>
<td>6.88 (14.0)</td>
</tr>
</tbody>
</table>

### Table 4.8: p-values in Kruskal-Wallis Test

<table>
<thead>
<tr>
<th></th>
<th>Change adj density</th>
<th>Change Perc15</th>
<th>COPD Score</th>
<th>Emph Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>DReaM-FS</td>
<td>7.54e-23</td>
<td>5.08e-30</td>
<td>2.23e-12</td>
<td>2.50e-06</td>
</tr>
<tr>
<td>GOLD</td>
<td>1.57e-08</td>
<td>4.34e-08</td>
<td>2.17e-10</td>
<td>1.85e-03</td>
</tr>
<tr>
<td>GMM</td>
<td>1.84e-22</td>
<td>1.83e-20</td>
<td>9.39e-12</td>
<td>1.43e-04</td>
</tr>
<tr>
<td>KMeans</td>
<td>5.99e-04</td>
<td>2.19e-03</td>
<td>8.13e-04</td>
<td>4.35e-01</td>
</tr>
<tr>
<td>DReaM</td>
<td>3.54e-19</td>
<td>2.56e-11</td>
<td>8.20e-12</td>
<td>1.49e-05</td>
</tr>
<tr>
<td>LFS</td>
<td>5.57e-19</td>
<td>2.64e-22</td>
<td>3.77e-12</td>
<td>6.41e-05</td>
</tr>
<tr>
<td>GMM+DT</td>
<td>5.19e-07</td>
<td>4.28e-07</td>
<td>4.87e-02</td>
<td>8.01e-01</td>
</tr>
</tbody>
</table>
variables are summarized in Table 4.8, where a smaller p-value implies a more statistically significant result. DReaM-FS gives one of the smallest p-values in the tests with all these 4 variables. In particular, it gives a smaller p-value compared to GOLD. This shows that compared with the standard GOLD criteria, the rules generated by DReaM-FS can better separate subjects into groups according to the 5-year disease progression and the genetic risk scores.

4.2.4.3 Sensitivity Test for DReaM-FS

In DReaM-FS, $\zeta$ and $a$ are two important parameters. In this section, I discuss how these two parameters affect the clustering results, by varying these two parameters. I test the model via 5-fold cross validation on the wine data set.

I first keep $a = 100$ and vary $\zeta$. I plot the NMI and number of features selected in Figure 4.9a. As shown in Equation (4.42), $\zeta$ defines how much error the model allows. Therefore, it controls the quality of the features selected. As shown in Figure 4.9a, when $\zeta \to 1$ (e.g. $1 - \zeta = 1e - 4$), I allow almost no error. A small number of features are selected in this case and the model does not perform well in terms of NMI. However, if we choose a small $\zeta$ (e.g. $1 - \zeta = 0.2$), all features including noisy and irrelevant features are selected. This also results a worse performance in terms of NMI. Therefore, I choose $\zeta = 0.01$ for all experiments in Chapter 4.2.4.1 and 4.2.4.2.
Then I keep $\zeta = 0.01$ and vary $a$. I plot the NMI and number of features selected in Figure 4.9b. As shown in Figure 4.6, where $a$ defines the “smoothness” of the soft-thresholding function $g$. Note that as shown in Equation (4.67), I cluster test samples with hard-thresholding. Therefore, in order to get good clustering performance, we require the soft-thresholding function $g$ to be reasonably close to the hard-thresholding function $f$. As shown in Figure 4.6, we need to choose $a$ to be reasonably large. However, if $a$ is too large, the function $g$ will be almost indifferentiable, which harms the optimization process. I choose $a = 100$ for all experiments in Chapter 4.2.4.1 and 4.2.4.2.
Chapter 5

Interpretable Clustering with Similarity Matrices (ICSM)

In Chapter 3 and 4, I focus on discussing clustering methods by analyzing feature matrices, which describe each sample with a set of features. However, in some applications, a similarity matrix that describes the pairwise similarity between each pair of samples. Similarity-matrix-based clustering methods, including Spectral clustering [5–7] and kernel k-means [8] generate the clustering solution by analyzing the similarity matrix. One limitation of existing similarity-matrix-based clustering methods is that these methods usually simply generate clustering results without providing parameters such as mean and covariance to describe each cluster. Therefore, it is usually more difficult to interpret the clustering result.

In this chapter, I introduce the a model for Interpretable Clustering with Similarity Matrices (ICSM). ICSM assumes that both a feature matrix and a similarity matrix are simultaneously provided as the input. ICSM first needs to generate interpretable clustering rules using the feature matrix. To facilitate this, I make use of the rule generating and feature selection mechanism, similar to DReaM-FS as described in Chapter 4.2, but convert the directed graphical model into a factorized model. ICSM represents the rules using lower and upper bounds associated with a subset of selected features in the feature matrix, such that it determines which cluster a sample belongs to, by observing whether rules for each cluster are satisfied or not through soft thresholding. At the same time, ICSM forces the clustering results to be consistent to the observed similarity matrix.
5.1 The ICSM model

5.1.1 A Factor Graph Model

In this section, I introduce the ICSM model. The purpose of the algorithm is to separate the samples into $K$ clusters, represented by an $N \times K$ binary-cluster-indicator matrix $Z$. We want the model to generate interpretable rules based on a subset of features in $X$, while the clustering result is consistent with the similarity matrix $W$.

To achieve this purpose, I introduce latent variables $T = \{\{t_{kd}\}_{k=1}^K\}_{d=1}^D$, $R = \{r_n\}_{n=1}^N$, $S = \{\{s_{kd}\}_{k=1}^K\}_{d=1}^D$, and construct a probabilistic model. I introduce the detail about these variables in the following sections and summarize the descriptions for these variable in Table 5.1. With these variables, I define the joint probability function as

$$p(S, T, R, Z, X, W) = \frac{1}{C} f_1(T) \prod_{n=1}^N f_2(x_n, r_n, T) \prod_{n=1}^N f_3(z_n, r_n, S) f_4(Z, W).$$ (5.1)

where $C$ is the normalization constant that makes sure that Equation (5.1) represents a valid probabilistic distribution. In this equation, I assume that the probability function can be factorized into factors, where $f_1$, $f_2$, $f_3$, $f_4$ are non-negative functions. The factorization is represented in Figure 5.1. I discuss how each factor is defined in the following sections.

5.1.2 Defining Rules with the Feature Matrix

I define each cluster using a set of interpretable rules, with a subset of features in the feature matrix $X$. Similar to DReaM-FS, I assign a lower and an upper bound associated with each feature $d$ for each cluster $k$, denoted as $t_{kd} = \{t^-_{kd}, t^+_{kd}\}$, where $t^-_{kd}, t^+_{kd} \in \mathbb{R}$. I introduce a prior distribution for these variables as the factor $f_1(T)$ in Equation (5.1), such that

$$\log f_1(T) = \frac{1}{2} \alpha \sum_{k=1}^K \sum_{d=1}^D t^-_{kd}^2 - \frac{1}{2} \alpha \sum_{k=1}^K \sum_{d=1}^D t^+_{kd}^2 - \frac{1}{2} \beta \sum_{k=1}^K \sum_{d=1}^D (t^+_{kd} - t^-_{kd})^2$$ (5.2)

In this equation, the first two terms penalize for large $t^-_{kd}$ and $t^+_{kd}$, and the last term penalizes for the square difference between $t^-_{kd}$ and $t^+_{kd}$. $\alpha$ and $\beta$ are the parameters
**Table 5.1: Important Notations**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N$</td>
<td>A positive integer that represents the number of samples</td>
</tr>
<tr>
<td>$K$</td>
<td>A positive integer that represents the number of clusters</td>
</tr>
<tr>
<td>$D$</td>
<td>A positive integer that represents the number of features in the feature matrix</td>
</tr>
<tr>
<td>$X = {x_n}_{n=1}^N$</td>
<td>The feature matrix. Each row in the matrix is a $D$ dimensional vector that represents the feature for the $n$-th sample.</td>
</tr>
<tr>
<td>$W = {{w_{ij}}<em>{i=1}^N}</em>{j=1}^N$</td>
<td>The similarity matrix. Each element in the matrix is a real number that represents the pairwise similarity measure between two samples.</td>
</tr>
<tr>
<td>$t_{kd} = {t_{-kd}^-, t_{+kd}^+}$</td>
<td>A 2-element vector that represents the rule defined for cluster $k$ with feature $d$. Each element in this vector represents the left and right boundaries for the rule respectively.</td>
</tr>
<tr>
<td>$r_{nkd}$</td>
<td>A binary indicator that represents whether the rule $t_{kd}$ is satisfied by a sample $x_n$, i.e., whether $t_{-kd}^- &lt; x_{nd} &lt; t_{+kd}^+$ is true, or not.</td>
</tr>
<tr>
<td>$s_{kd}$</td>
<td>A binary indicator represents whether feature $d$ is used to define cluster $k$ or not</td>
</tr>
<tr>
<td>$z_n$</td>
<td>A $K$-dimensional vector. Each element in the vector is a binary indicator that represents whether the sample $n$ belongs to cluster $k$ or not</td>
</tr>
</tbody>
</table>

that control the tradeoffs between these penalties. This factor introduces regularization to the model to prevent overfitting.

Note that although I assign the boundaries for all features, some features might be irrelevant in defining a certain cluster. I provide a feature selection mechanism to select a subset of features to define a cluster, with the details introduced in Chapter 5.1.4.

### 5.1.3 Determining Whether a Rule is Satisfied via Soft Thresholding

Given a rule $t_{kd}$, we need a mechanism in determining whether a rule is satisfied, such that $t_{-kd}^- < x_{nd} < t_{+kd}^+$ is satisfied. Similar to DReaM-FS, I define a differentiable soft-thresholding function $g(x_{nd}, t_{kd}) \in (0, 1)$ to approximate the step function, such that

$$g(x_{nd}, t_{kd}) = \frac{1}{\left(1 + \exp\{-a(x_{nd} - t_{-kd}^-)\}\right) \left(1 + \exp\{-a(t_{+kd}^+ - x_{nd})\}\right)}$$  \hspace{1cm} (5.3)
Figure 5.1 The factor graph of the ICSM. The square represents the factors, the normal circles represents the latent variables and the shaded circles represents the observed variables. A factor and a variable are connected, only if this factor is a function of this variable.

where $a$ is the parameter that controls the “steepness” of the function. I plot this function with different $a$ in Figure 4.6. Note that when $a \to \infty$, this function approaches the step function.

To facilitate clustering, I convert $g(x_{nd}, t_{kd}) \in (0, 1)$ into binary indicator $r_{nkd} \in \{0, 1\}$. To achieve this, I rewrite Equation (4.48) in DReaM-FS, by defining the factor $f_2(x_n, r_n, T)$ in Equation (5.1) as

$$
\log f_2(x_n, r_n, T) = \sum_{k=1}^{K} \sum_{d=1}^{D} r_{nkd} \log g(x_{nd}, t_{kd}) + \frac{1}{K-1} (1 - r_{nkd}) \log (1 - g(x_{nd}, t_{kd})),
$$

(5.4)

Similar to Equation (4.48), this factor forces $g(x_{nd}, t_{kd})$ and $r_{nkd}$ to be consistent, because

$$
f_2(x_n, r_n, T) \approx 0, \text{ if } \begin{cases} r_{nkd} = 1 \text{ and } g(x_{nd}, t_{kd}) \approx 0, & \text{ or } \begin{cases} r_{nkd} = 0 \text{ and } g(x_{nd}, t_{kd}) \approx 1 \end{cases} \end{cases}
$$

(5.5)
for any \( k \in \{1, \ldots, K\}, d \in \{1, \ldots, D\} \); i.e., it is almost impossible that \( r_{nkd} \) and \( g(x_{nd}, t_{kd}) \) are inconsistent. Note that in Equation (5.4), I introduce a factor \( 1/(K - 1) \) for the second term, because if each cluster is of equal size, then, there are \((K - 1)\) as many samples with \( r_{nkd} = 0 \) as samples with \( r_{nkd} = 1 \). This factor balances the contribution of samples with \( r_{nkd} = 0 \) and \( r_{nkd} = 1 \) to the likelihood function of the overall model.

5.1.4 Clustering and Feature Selection with Rules

Now we want to determine which cluster a sample \( n \) belongs to, by observing whether the rules are satisfied by sample \( n \), represented using binary indicators \( r_n = \{\{r_{nkd}\}_{k=1}^{K}\}_{d=1}^{D} \). In particular, we want the clustering results depend on a subset of selected features only. To achieve this I introduce latent binary indicators \( S = \{\{s_{kd}\}_{k=1}^{K}\}_{d=1}^{D} \) to represent whether feature \( d \) is selected to define cluster \( k \) or not. I make use of clustering and feature selection of DReaM-FS, by defining \( f_3(z_n, r_n, S) \) in Equation (5.1) as

\[
\log f_3(z_n, r_n, S) = \sum_{k=1}^{K} \sum_{d=1}^{D} (1 - s_{kd}) \left( r_{nkd} \log \frac{1}{K} + (1 - r_{nkd}) \log \frac{K - 1}{K} \right) + \sum_{k=1}^{K} \sum_{d=1}^{D} s_{kd} z_{nk} \left( r_{nkd} \log \zeta + (1 - r_{nkd}) \log (1 - \zeta) \right) + \sum_{k=1}^{K} \sum_{d=1}^{D} s_{kd} (1 - z_{nk}) \left( (1 - r_{nkd}) \log \zeta + r_{nkd} \log (1 - \zeta) \right) \tag{5.6}
\]

where \( \zeta \) is a real parameter with \( 0 < \zeta < 1 \). With this equation, \( \zeta \) can be considered as defining the conditional probability

\[
\zeta = p(r_{nkd} = 1 | s_{kd} = 1 \text{ and } z_{nk} = 1) = p(r_{nkd} = 0 | s_{kd} = 1 \text{ and } z_{nk} = 0), \tag{5.7}
\]

i.e., \( \zeta \) is the probability that rule satisfaction indicator \( r_{nkd} \) is consistent with cluster indicator \( z_{nk} \), assuming that feature \( d \) is selected for cluster \( k \) (i.e., \( s_{kd} = 1 \)). In the model, we want to enforce \( r_{nkd} \) to be consistent with \( z_{nk} \), as long as \( s_{kd} = 1 \). Therefore, I let \( \zeta \approx 1 \). I do not enforce \( \zeta \) to be exactly 1, because I do not assume the rules are perfect and I allow for errors. Unless otherwise stated, I choose \( \zeta = 0.99 \) in the experiments. A more detail discussion about the choice of this parameter \( \zeta \) is provided.
in Chapter 5.3.3. The feature selection and clustering mechanism already introduced in Chapter 4.2.1.2.

5.1.5 Clustering with Similarity Matrices

In the previous subsection, I introduce how I generate the cluster indicator based on the rule. Now I introduce how I make sure the cluster indicator to be consistent to the observed similarity matrix, by introducing the factor $f_4(Z, W_{ij})$ in Equation (5.1) as

$$\log f_4(Z, W) = -tr\left\{Z^T(D - W)Z\right\} - \lambda \sum_{k=1}^{K} \left(\sum_{n=1}^{N} z_{nk} - \frac{N}{K}\right)^2$$

(5.8)

where $\lambda$ is a positive parameter, and $D$ is an $N \times N$ diagonal degree matrix with $D_{ii} = \sum_{j=1}^{N} W_{ij}$. Note that because $Z$ is the binary cluster-indicator matrix, the first term in this equation can be rewritten as

$$-tr\left\{Z^T(D - W)Z\right\} = -\frac{1}{2} \sum_{i,j=1}^{N} W_{ij} \sum_{k=1}^{K} (z_{ik} - z_{jk})^2 = -\sum_{i,j} W_{ij}$$

(5.9)

Therefore, maximizing this term is equivalent to minimizing the sum of pairwise similarity measure for samples from different clusters. Note that if $W_{ij}$ is non-negative, then maximizing Equation (5.9) with respect to $Z$ will give a trivial solution that $z_i = z_j$, for all $i, j \in \{1, \ldots, N\}$ (i.e. all samples are assigned to the same cluster), with the maximum value 0.

This trivial solution is an extreme imbalanced clustering solution. To avoid such solution, I introduce the second term in Equation (5.8) to penalize the imbalanced clustering solution. Note that the objective function of spectral clustering is given as

$$\min_{H \in \mathbb{R}^{N \times K}} tr\left\{H^T(D - W)H\right\}, \text{ s.t. } H^T H = I.$$  

(5.10)

where $I$ is an $K \times K$ identity matrix. Maximizing Equation (5.8) with respect to $Z$ can be considered as minimizing the following objective function

$$\min_Z tr\left\{Z^T(D - W)Z\right\}, \text{ s.t. } Z^T Z \approx \frac{N}{K} I, \quad Z \text{ is a binary cluster-indicator matrix}$$

(5.11)
Therefore, maximizing Equation (5.8) with respect to $Z$ is equivalent to optimizing the same objective function as spectral clustering, but I relax the constraint that $Z$ is a real-value variable and assume that $Z$ is a binary cluster-indicator matrix.

### 5.2 Training ICSM with Expectation Maximization

I have presented ICFS model above, as summarized in Equation (5.1) and in Figure 5.1. Now, I introduce how I train the model, by learning the Maximum a Posteriori Probability (MAP) estimator for $T$ through Expectation Maximization (EM) [32].

**Expectation Step**

In the expectation step, I first take the expected value of the log joint probability as a function of $T$, with respect to the posterior distribution $p(R, S, Z|X, W, \hat{T})$, where $\hat{T}$ represents the current estimate of $T$. This expected value can be written as

$$h(T) = \mathbb{E}[\log p(S, T, R, Z, X, W)] = \mathbb{E} \left[ \log f_1(T) \right] + \sum_{n=1}^{N} \mathbb{E} \left[ \log f_2(x_n, r_n, T) \right] + \text{const}$$

where $\text{const}$ represents all terms that are not functions of $T$.

**Maximization Step**

Then, in the maximization step, I update the estimate of $T$ by maximizing $h(T)$, i.e.,

$$\hat{T} = \arg \max_{T} h(T).$$

Since $t_{kd}$ does not depend on other terms $t_{ij}$ for all $i \neq k$ and $j \neq d$, I can compute the optimal $t_{kd}$ with each $k$ and $d$ separately, such that

$$t_{kd} = \arg \max_{t_{kd}} \frac{-1}{2\alpha} t_{kd}^2 - \frac{1}{2\alpha} t_{kd}^2 - \frac{1}{2} \beta (t_{kd} - t_{kd})^2$$

$$+ \sum_{n=1}^{N} \mathbb{E}[r_{nk}] \log g(x_n, t_{kd}) + \frac{1}{K-1} \sum_{n=1}^{N} (1 - \mathbb{E}[r_{nk}]) \log (1 - g(x_n, t_{kd}))$$

This optimization problem can be solved using the conjugate gradient method[44].
Approximating the Posterior Distribution

In the expectation step, we need to derive the posterior distribution $p(R, S, Z|X, W, \hat{T})$. However, this distribution is computationally intractable. I use a variational distribution $q(R, S, Z)$ to approximate it such that

$$q(R, S, Z) \approx p(R, S, Z|X, W, \hat{T}).$$  \hspace{1cm} (5.15)

I make $q(R, S, Z)$ tractable by applying mean-field approximation such that it is factorized as

$$q(R, S, Z) = \prod_{n=1}^{N} \prod_{k=1}^{K} \prod_{d=1}^{D} q(r_{nkd}) \prod_{k=1}^{K} \prod_{d=1}^{D} q(s_{kd}) \prod_{n=1}^{N} q(z_{n}) \hspace{1cm} (5.16)$$

To make sure the variational distribution $q$ is a good approximation for the posterior distribution, I update $q$ such that the KL divergence between the variational distribution and the posterior distribution, denoted as $KL(q||p)$, is minimized. It is straightforward to derive the update equations based on Appendix A, which are summarized as follows.

The update equation for $q(r_{nkd})$ is given by

$$q(r_{nkd}) = \text{Categorical}(v_{nkd})$$ \hspace{1cm} (5.17)

where $v_{nkd}$ is defined as

$$v_{nkd} = \frac{b_{nkd}}{1 + b_{nkd}}$$ \hspace{1cm} (5.18)

and $b_{nkd}$ is defined as

$$b_{nkd} = \exp \left\{ E_q[s_{kd}](2E_q[z_{nk}] - 1)(\log \zeta - \log(1 - \zeta) - (1 - E_q[s_{kd}]) \log (K - 1) \\
+ \log g(x_{n}, \hat{t}_{kd}) - \frac{1}{K - 1} \log(1 - g(x_{n}, \hat{t}_{kd})) \right\}$$ \hspace{1cm} (5.19)

The update equation for $q(s_{kd})$ is given by

$$q(s_{kd}) \sim \text{Bernoulli}(\rho_{kd})$$ \hspace{1cm} (5.20)

where $\rho_{kd}$ is defined as

$$\rho_{kd} = \frac{c_{kd}}{1 + c_{kd}}$$ \hspace{1cm} (5.21)
and $c_{kd}$ is defined as

\[
c_{kd} = \exp \left\{ \sum_{n=1}^{N} \left( \mathbb{E}_{q}[z_{nk}]\mathbb{E}_{q}[r_{nkd}] + (1 - \mathbb{E}_{q}[z_{nk}])(1 - \mathbb{E}_{q}[r_{nkd}]) \right) \log \zeta \right.
\]
\[
+ \sum_{n=1}^{N} \left( \mathbb{E}_{q}[z_{nk}](1 - \mathbb{E}_{q}[r_{nkd}]) + (1 - \mathbb{E}_{q}[z_{nk}])\mathbb{E}_{q}[r_{nkd}] \right) \log(1 - \zeta) \] (5.22)
\]
\[
- \sum_{n=1}^{N} \mathbb{E}_{q}[r_{nkd}] \log \frac{1}{K} - \sum_{n=1}^{N} (1 - \mathbb{E}_{q}[r_{nkd}]) \log \frac{K - 1}{K} \right\}
\]

The update equation for $q(z_n)$ is given by

\[
q(z_n) \sim \text{Categorical}(\pi_n) \] (5.23)

where $\pi_n$ is a $K$-dimensional vector, with each element defined as

\[
\pi_{nk} \propto \exp \left\{ - \sum_{d=1}^{D} \mathbb{E}_{q}[s_{kd}](2\mathbb{E}_{q}[r_{nkd}] - 1) \left( \log \zeta - \log(1 - \zeta) \right) - \sum_{i \neq n} W_{in} \left( 1 - \mathbb{E}[z_{ik}] \right) - \lambda \sum_{i \neq n} \mathbb{E}[z_{ik}] \right\} \] (5.24)

and $\pi_{nk}$ is normalized such that $\sum_{k=1}^{K} \pi_{nk} = 1$.

In the update equations, $\mathbb{E}_{q}$ represents that the expected value is taken with respect to the variational distribution $q$. Since all the distributions $q$ are common distributions in exponential family, it is straightforward to express these expected values using parameters of the corresponding distributions.

I summarize the algorithm in Algorithm 5.

5.3 Experiments with ICSM

5.3.1 Testing ICSM with Benchmark Data

To demonstrate that ICSM is able to analyze the similarity matrix and give interpretable results, I first test the model on benchmark data sets. These benchmark datasets includes 4 UCI data sets [45]: iris, wine, breast, wdbc; 4 human faces data sets: CMU faces, orlraws10p, pixraw10p, Yale; and 1 hand-written digit data set: USPS. The details about these datasets are provided in Appendix B.
Algorithm 5 Expectation Maximization for ICSM

Randomly initialize the linear parameters $\hat{T}$ and the factorized variational distributions $q$.

repeat
  for $n \leftarrow 1$ to $N$ do
    for $k \leftarrow 1$ to $K$ do
      for $d \leftarrow 1$ to $D$ do
        update $q(r_{nkd})$ according to Equation (5.17).
      end for
    end for
  end for
  for $k \leftarrow 1$ to $K$ do
    for $d \leftarrow 1$ to $D$ do
      update $q(s_{kd})$ according to Equation (5.20).
    end for
  end for
  for $n \leftarrow 1$ to $N$ do
    update $q(z_n)$ according to Equation (5.23).
  end for
  for $k \leftarrow 1$ to $K$ do
    for $d \leftarrow 1$ to $D$ do
      Update $\hat{t}_{kd}$ based on the Equation (5.14) using gradient descent.
    end for
  end for
until Converges

For these data sets, I first generate similarity matrix via Gaussian kernel, i.e., each element in the similarity matrix $W$ is computed as

$$W_{ij} = \exp\left(-\frac{||x_i - x_j||^2}{\sigma^2}\right), \quad i, j \in \{1, \ldots, N\}$$

(5.25)

where $\sigma$ is the bandwidth parameter. I choose $\sigma$ to be the median of the pairwise Euclidean distances. I use both the feature matrix $X$ and the similarity matrix $W$ as the input of the ICSM. I choose the parameters of ICSM as $a = 100$, $\zeta = 0.99$ and $\lambda = .5$. A more detailed discussion about the choice of these parameters are given in the supplementary materials.

I compare the ICSM with existing methods. I choose the Gaussian Mixture Model (GMM) [9] and Ratio Cut (RC) spectral clustering [6] as baselines. I use the eigenfunction as described in [53] to generalize the RC clustering results to out-of-sample data. Since ICSM is a local feature selection method, i.e., I allow the features selected to differ across different clusters, I include the local feature selection version of [35], denoted as
LFS. LFS selects features that reveal a clustering structure but cannot generate interpretable rules. I also include a simple rule-generating clustering method I call RC+DT, where I use the RC clustering results as labels to train a decision tree.

Since ICSM generates rules, we want to know how accurately these rules will perform in a predictive task. Therefore, I conduct a 5-fold cross-validation and compare the predictive cluster labels in the validation sets with the ground-truth labels. I measure the performance using Normalized Mutual Information (NMI) and Adjusted Rand Index (ARI), where NMI is the normalized version of mutual information and ARI is the corrected-for-chance version of the Rand index. The definitions of NMI and ARI are given in Appendix C. In Table 5.3, I report the mean value and standard deviation of the 5-fold cross-validation NMI and ARI results on each clustering task. The values in bold are the highest mean NMI or ARI value for each task.

We can observe from Table 5.3 that ICSM and LFS are the two best performers for iris, wine, breast and wdbc datasets. These are simpler data sets in lower dimensions, and the ground-truth clusters for these datasets are more or less linearly separable. ICSM is able to learn the rules that separate most samples correctly. However, for the image data sets, the separation of ground-truth clusters might be non-linear in the original feature space. The RC spectral clustering method performs better on these data sets. The major drawback of spectral clustering is that it is not interpretable; i.e., it is usually difficult to understand what characterizes each cluster and how one cluster differs from others. We might want to interpret the spectral clustering results via post processing, for example, by feeding the clustering results to train a decision tree. However, as shown in the table, the performance of RC+DT is not very good. ICSM outperforms RC+DT method in almost all data sets (except for the USPS data set, which I will discuss later).

One of the advantages of ICSM is that it generates a list of interpretable rules. I describe the rules generated for the wine data set in Table 5.2. Although the data set contains 13 features, the list of rules for each cluster contains either 2 or 3 features. The rules in the table clearly define which feature characterizes one cluster and how one cluster is distinguished from others.

To better illustrate the rules generated, I plot some of the rules discovered for the wine data set in Figure 5.2. As shown in Figure 5.2a and 5.2b, most samples that belong to one cluster are located inside the corresponding hyper-rectangle, and most samples
Table 5.2: Rules Discovered by ICSM for the Wine Dataset

<table>
<thead>
<tr>
<th>Cluster</th>
<th>Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.26 &lt; x_7^2 &lt; 1.58</td>
</tr>
<tr>
<td>2</td>
<td>2.62 &lt; x_7^2 &lt; 4.01</td>
</tr>
<tr>
<td>3</td>
<td>11.35 &lt; x_7^2 &lt; 12.52</td>
</tr>
</tbody>
</table>

Table 5.3: Performance of ICSM on Benchmark Data

<table>
<thead>
<tr>
<th>Dataset</th>
<th>ICSM</th>
<th>GMM</th>
<th>LFS</th>
<th>RC</th>
<th>RC+DT</th>
</tr>
</thead>
<tbody>
<tr>
<td>iris</td>
<td>NMI</td>
<td>.865(1.134)</td>
<td>.810(.083)</td>
<td>.798(.115)</td>
<td>.747(.057)</td>
</tr>
<tr>
<td>wine</td>
<td>NMI</td>
<td>.837(1.156)</td>
<td>.807(.138)</td>
<td>.831(.079)</td>
<td>.686(.170)</td>
</tr>
<tr>
<td>breast</td>
<td>NMI</td>
<td>.652(.118)</td>
<td>.638(.066)</td>
<td>.665(.099)</td>
<td>.605(.095)</td>
</tr>
<tr>
<td>wdbc</td>
<td>NMI</td>
<td>.629(.098)</td>
<td>.584(.081)</td>
<td>.567(.038)</td>
<td>.248(.097)</td>
</tr>
<tr>
<td>CMU faces</td>
<td>NMI</td>
<td>.924(.022)</td>
<td>.769(.083)</td>
<td>.930(.017)</td>
<td>.918(.024)</td>
</tr>
<tr>
<td>orlraws10p</td>
<td>NMI</td>
<td>.852(.025)</td>
<td>.810(.049)</td>
<td>.859(.022)</td>
<td>.856(.027)</td>
</tr>
<tr>
<td>pixraw10p</td>
<td>NMI</td>
<td>.929(.036)</td>
<td>.746(.015)</td>
<td>.938(.026)</td>
<td>.942(.043)</td>
</tr>
<tr>
<td>Yale</td>
<td>NMI</td>
<td>.689(.063)</td>
<td>.598(.050)</td>
<td>.623(.081)</td>
<td>.731(.018)</td>
</tr>
<tr>
<td>USPS</td>
<td>NMI</td>
<td>.391(.004)</td>
<td>.477(.007)</td>
<td>.496(.012)</td>
<td>.586(.014)</td>
</tr>
</tbody>
</table>

that do not belong to this cluster are located outside this hyper-rectangle. I also want to highlight that as plotted in Figure 5.2b, I do not plot boundaries corresponding to feature 11 for cluster 2, indicating that this feature is not selected for cluster 3. Note that ICSM selects features based on whether the rule can better distinguish one cluster from the rest than random. We can not use feature 11 to distinguish cluster 3 from cluster 2. Therefore, this feature is not selected.

We can observe in Table 5.3 that ICSM does not perform well on the USPS handwritten digit data set. I believe that this is due to the characteristic of this clustering task. As demonstrated in Figure 5.2b, the model selects features such as feature 7 that distinguish one cluster from the rest, without observing any other features. However, in hand-written digit recognition task, such features might not exist, because the gray-scale value of one certain pixel by itself usually does not imply which digit it is and one needs to observe the rest of the image to identify the digit.
5.3.2 Applying ICSM on COPD Subtyping Problem

In this subsection, I test ICSM on the real-world COPD subtyping data set. In the data set, we are provided 39 phenotypical features and 630,860 Single Nucleotide Polymorphisms (SNPs) from 5,435 heavy smokers with and without COPD. I try to separate these subjects into clusters making use of SNPs data, such that subjects in different clusters exhibit different phenotypes.

Since most of the SNPs are not relevant to the COPD disease, I first screen the SNPs by conducting Krustal-Wallis test on the following two spirometry measures: 1) FEV1 (forced expiratory volume in 1 second) percent of predicted normal (based on age, sex, height, and race) and 2) FEV1 to forced vital capacity ratio (FEV1/FVC). I use the SNPs with p-values less on 0.01 on both spirometry measures. 3,336 SNPs are selected in this process.

Now, I compute the Genetic Relationship Matrix (GRM) \[54\] between each pair of subjects, with the selected SNPs as follows:

\[
W_{ij} = \frac{1}{L} \sum_{l=1}^{L} \frac{(s_{il} - 2p_l)(s_{jl} - 2p_l)}{2p_l(1 - p_l)}, \quad i, j \in \{1, \ldots, N\} \text{ and } i \neq j \quad (5.26)
\]
Table 5.4: Rules discovered by ICSM for COPD Subtyping Problem

<table>
<thead>
<tr>
<th>Rules for Cluster1:</th>
</tr>
</thead>
<tbody>
<tr>
<td>14.81 &lt; pctEmphSlicer &lt; 58.36</td>
</tr>
<tr>
<td>-1001.33 &lt; Slicer15petInTotal &lt; -951.33</td>
</tr>
<tr>
<td>-927.63 &lt; SlicerIntensityMeanIn &lt; -876.43</td>
</tr>
<tr>
<td>4.47 &lt; FRCslicer &lt; 8.00</td>
</tr>
<tr>
<td>45.79 &lt; pctGasTrapSlicer &lt; 88.81</td>
</tr>
<tr>
<td>7.45 &lt; ExpBelow950Slicer &lt; 50.78</td>
</tr>
<tr>
<td>-1005.30 &lt; Slicer15petExTotal &lt; -926.16</td>
</tr>
<tr>
<td>-919.67 &lt; SlicerIntensityMeanEx &lt; -808.77</td>
</tr>
<tr>
<td>0.93 &lt; SlicerExpInspMeanAttenratio &lt; 1.01</td>
</tr>
<tr>
<td>1.31 &lt; UpperThirdLowerThirdSlicer &lt; 2.07</td>
</tr>
<tr>
<td>129.34 &lt; FRCppraceadjusted &lt; 221.86</td>
</tr>
<tr>
<td>9.30 &lt; FEV1pputah &lt; 45.58</td>
</tr>
<tr>
<td>0.16 &lt; FEV1FVCutah &lt; 0.48</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rules for Cluster2:</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.08 &lt; pctEmph Slicer &lt; 1.17</td>
</tr>
<tr>
<td>-911.21 &lt; Slicer 15petIn Total &lt; -816.65</td>
</tr>
<tr>
<td>-0.51 &lt; pctGasTrap Slicer &lt; 10.23</td>
</tr>
<tr>
<td>-0.13 &lt; Exp Below950 Slicer &lt; 0.55</td>
</tr>
<tr>
<td>-839.44 &lt; Slicer 15petEx Total &lt; -668.34</td>
</tr>
<tr>
<td>-703.38 &lt; Slicer IntensityMean Ex &lt; -537.15</td>
</tr>
<tr>
<td>0.66 &lt; Slicer ExpInspMeanAtten ratio &lt; 0.83</td>
</tr>
<tr>
<td>0.12 &lt; UpperThird LowerThird Slicer &lt; 0.68</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rules for Cluster3:</th>
</tr>
</thead>
<tbody>
<tr>
<td>25.37 &lt; pctGasTrap Slicer &lt; 47.02</td>
</tr>
<tr>
<td>2.31 &lt; Exp Below950 Slicer &lt; 7.55</td>
</tr>
<tr>
<td>-928.15 &lt; Slicer 15petEx Total &lt; -885.50</td>
</tr>
<tr>
<td>-810.56 &lt; Slicer IntensityMean Ex &lt; -759.28</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rules for Cluster4:</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.84 &lt; pctEmph Slicer &lt; 6.46</td>
</tr>
<tr>
<td>9.62 &lt; pctGasTrap Slicer &lt; 28.31</td>
</tr>
<tr>
<td>0.25 &lt; Exp Below950 Slicer &lt; 2.40</td>
</tr>
<tr>
<td>-887.12 &lt; Slicer 15petEx Total &lt; -838.09</td>
</tr>
<tr>
<td>-770.07 &lt; Slicer IntensityMean Ex &lt; -702.60</td>
</tr>
<tr>
<td>0.83 &lt; Slicer ExpInspMeanAtten ratio &lt; 0.90</td>
</tr>
<tr>
<td>0.26 &lt; UpperThird LowerThird Slicer &lt; 0.91</td>
</tr>
</tbody>
</table>

where $s_{nl}$ represents the $l$-th SNP for subject $n$, $p_l$ is the frequency of the minor allele for the $l$-th SNP, and $L$ is the number of SNPs, i.e., $L = 3,336$. 
Table 5.5: p-values in Kruskal-Wallis Test

<table>
<thead>
<tr>
<th></th>
<th>ICSM</th>
<th>GMM</th>
<th>LFS</th>
<th>RC</th>
<th>RC+DT</th>
</tr>
</thead>
<tbody>
<tr>
<td>copdScore</td>
<td>7.89e-15</td>
<td>3.90e-15</td>
<td>2.88e-15</td>
<td>2.46e-01</td>
<td>4.40e-01</td>
</tr>
<tr>
<td>lungfxScore</td>
<td>6.58e-04</td>
<td>2.29e-02</td>
<td>8.33e-03</td>
<td>1.08e-14</td>
<td>4.51e-01</td>
</tr>
<tr>
<td>emphScore</td>
<td>8.47e-16</td>
<td>1.23e-11</td>
<td>5.18e-12</td>
<td>1.04e-01</td>
<td>1.38e-01</td>
</tr>
<tr>
<td>airScore</td>
<td>6.38e-01</td>
<td>2.79e-01</td>
<td>7.90e-01</td>
<td>3.43e-26</td>
<td>9.68e-01</td>
</tr>
</tbody>
</table>

I separate the data set into training and test sets of equal size. I use the phenotypical features as $X$ and the GRM as $W$ to train ICSM. I summarize the rules discovered by ICSM in Table 3.4. To better understand the clustering results, I plot some selected features in Figure 3.5. As shown in the table, the clusters are separated based on $pct\text{GasTrap Slicer}$, $Exp\ Below950\ Slicer$, $Slicer\ 15\ pct\text{Ex Total}$ and $Slicer\ IntensityMean\ Ex$, where all clusters are defined using these features. These are highly correlated features that measure the severity of emphysema (destruction of lung tissue). $pct\text{Emph Slicer}$ and $Slicer\ Exp\ Insp\ Mean\ Atten\ ratio$ also measure the severity of emphysema. These two features are used to define all clusters except for cluster 3. As shown in Figure 5.3b, it is more difficult to define cluster 3 with these two features.

Cluster 1 is also defined with $FRC\ Slicer$, $FRCpp\ race\ adjusted$, $FEV1pp\ utah$ and $FEV1\ FVC\ utah$, which are spirometry features that measure the lung function of the subjects. Cluster 1 is characterized with high $FRC$ and low $FEV1$, as plotted in Figure 5.3c. This implies that subjects in cluster 1 are the sickest subjects with severe disease.

I also compare ICSM with existing methods mentioned in Chapter 5.3.1. I train GMM and LFS using phenotypical features only. I train RC with the GRM. I train RC+DT by training a decision tree to predict clustering indicator generated by RC, using phenotypical features. I measure the performance of these methods in the test set. In this application, we do not have ground-truth labels. Therefore, I adopt an alternative way, by measuring whether subjects in different clusters differ in terms of the following 4 values: $copdScore$, $lungfxScore$, $emphScore$, $airScore$. These scores are genetic risk scores that measure the accumulation of genetic risk to different aspects of COPD [51, 52].

I measure how the 4 variables differ across different clusters by conducting a Kruskal-Wallis one-way analysis of variance [46], which is a non-parametric method for testing whether samples originate from the same distribution. The p-values for each of these
variables are summarized in Table 5.5, where a smaller p-value implies a more statistically significant result. The values in bold highlight the smallest p-values across different methods.

In Table 5.5, we can observe that methods based on phenotypical features (ICSM, GMM, LFS) gives more statistically significant results on copdScore and emphScore. However, RC that are based on GRM give more significant results on lungfxScore and airScore. In particular, we can observe that I cannot duplicate the clustering with RC using phenotypical data, because RC+DT gives insignificant results. This might imply that the phenotypical features and the GRM might contain different dominant clustering structure. Compared with GMM and LFS, ICSM gives more statistically significant results in terms of lungfxScore and airScore, because it makes use of the information in GRM to adjust the rules defined with phenotypical features. Another advantage of ICSM over other existing methods is that it gives interpretable results as shown in Table 3.4.

5.3.3 Sensitivity Test for ICSM

In ICSM, a, ζ and λ are important parameters. In this section, I discuss how these parameters affect the clustering results. I test the model via 5-fold cross validation on the wdbc data set.

I first keep a = 100 and λ = .5, vary ζ. I plot the NMI and number of features selected in Figure 5.4a. ζ defines how much error the model allows. Therefore, it controls the quality of the features selected. As shown in Figure 5.4a, when ζ → 1 (e.g. 1−ζ = 1e−6), I allow almost no error. A small number of features are selected in this case and the
model perform less well in terms of NMI. However, if I choose a small $\zeta$ (e.g. $1 - \zeta = 0.2$), almost all features including noisy and irrelevant features are selected. This also results a worse performance in terms of NMI. I choose $\zeta = 0.99$ in the experiments.

Next I keep $\zeta = 0.99$, $\lambda = 0.5$ and vary $a$. I plot the NMI in Figure 5.4b. $a$ defines the “smoothness” of the soft-thresholding function $g$. Note that eventually I cluster test samples using the rules through hard-thresholding. Therefore, in order to get good clustering performance, we require the soft-thresholding function $g$ to be reasonably close to the hard-thresholding step function. Therefore, we need to choose $a$ to be reasonably large. However, if $a$ is too large, the function $g$ will be almost indifferentiable, which harms the optimization process. I choose $a = 100$ for all experiments in Chapter 5.3.1 and 5.3.2.

Now I keep $a = 100$, $\zeta = 0.99$ and vary $a$. I plot the NMI in Figure 5.4c. I also plot a quantity that measures how balanced the clustering results is, which is defined as

$$v = \max_k N_k - \min_k N_k, \quad (5.27)$$

where $N_k$ is the number of samples in the $k$-th cluster in the training set. This quantity has 0 value if the clusters are perfectly balanced; while it equals to $N$ if the clustering result is trivial, such that all samples are in the same cluster. The wdbc training set has 452 samples and the ground-truth label is imbalanced, with $v = 112$. We can observe in the plot that if $\lambda = 0$, ICSM gives trivial solution, such that all samples are in the same cluster. If I choose a large $\lambda$, the clustering result is almost perfectly balanced. Since the ground-truth labels are imbalanced, large $\lambda$ gives worse NMI. In the experiments in Chapter 5.3.1 and 5.3.2, I choose $\lambda = 0.5$. 
Chapter 6

An Interpretable Crowdclustering Model with Partition Labels (ICMPL)

In this chapter, I focus on improving interpretability for crowdclustering problem, i.e., to generate a consensus clustering solution by analyzing clustering labels provided by multiple experts. Unlike existing crowdclustering models, ICMLP directly makes use of partition labels rather than pairwise similarity labels. This allows us to learn the mapping between the latent clusters and the expert labels, and therefore improves the interpretability of crowdclustering algorithm, by revealing the relationships between clustering solutions from different experts. At the same time, ICMLP simultaneously determines the number of clusters, based on a “rich get richer” fashion; and partition samples into clusters with discriminative hyper-planes.

6.1 The ICMPL Model

In this section I introduce the ICMPL model. I assume that the dataset contains $N$ samples with $D$ features. I let $\phi_n \in \mathbb{R}^{D+1}$ be a $(D + 1)$-dimensional vector associated with each sample $n \in \{1 \ldots N\}$. I let the first $D$ dimensions in this vector be the observed features for the $n$-th sample, and the $(D + 1)$-th dimension to be a constant 1.
I train a linear model for $\phi_n$ and the corresponding weight parameter for the $(D + 1)$th dimension plays a role of the bias term. I denote features for all samples as $\Phi$.

I assume that labels are provided by $M$ experts. I let $y_n^{(m)} \in \{1 \ldots J_m\}$ with $m \in \{1 \ldots M\}$ be the labels given by the $m$-th expert for the $n$-th sample, where $J_m$ represents the number of clusters the $m$-th expert chooses to partition the dataset. I let $Y$ represent labels from all experts.

### 6.1.1 Modified Multinomial Logistic Regression

Now I introduce how to generate the latent cluster indicators, such that the model simultaneously determines the number of clusters and partitions samples into clusters.

The number of clusters is usually a predefined parameter for clustering algorithms, including several crowdclustering methods [11–14]. This number might not be easy to determine for crowdclustering, especially when experts partition data into different number of clusters. One possible way to automatically determine this number is to apply Dirichlet Process (DP) [55, 56] as prior for the cluster indicators in a generative model. However, a generative model requires strong assumptions about the distribution of the observed samples. These assumptions are usually inaccurate in practice. Therefore, I decide to develop a discriminative model.

It is not straightforward to incorporate DP in a discriminative model. Therefore, I develop a novel approach, based on a modified multinomial logistic regression model, to automatically learn the number of clusters. This approach is inspired from the “rich get richer” fashion adopted by DP, such that big clusters that already have many members are more likely to be assigned with more new members. To achieve this, I define $z_n \in \{1 \ldots K\}$ to be the cluster indicator for the $n$-th sample, where $K$ is a predefined integer parameter that represents the maximum possible number of clusters. I let $K$ have a large value such that $K = 50$ in the experiment, i.e., samples can potentially be partitioned into up to 50 clusters. Because of the “rich get richer” property of ICMPL, only a few clusters will remain non-empty after we train the model.

I let $z_n$ follow a Categorical distribution such that

$$z_n | \mathbf{W}, \Phi \sim \text{Categorical}(\pi_n). \quad (6.1)$$
Figure 6.1 The problem setup. ICMPL analyzes both features and partition cluster labels from multiple experts. ICMPL simultaneously determines the number of clusters, partitions samples into clusters, and learns the mapping between the learned clusters and the expert labels.

In this equation, $\pi_n$ is a $K$ dimensional non-negative vector, such that $\sum_{k=1}^{K} \pi_{nk} = 1$. $\pi_{nk}$ gives the probability that the $n$-th sample belongs to cluster $k$, which is defined as

$$\pi_{nk} = \frac{\exp(w_k^T \phi_n + \lambda w_k^T w_k)}{\sum_{i=1}^{K} \exp(w_i^T \phi_n + \lambda w_i^T w_i)}$$

(6.2)

where $w_k \in \mathbb{R}^{D+1}$ with $k \in \{1 \ldots K\}$ is a $(D + 1)$-dimensional vector, each element of which represents the weight for each feature in $\phi_n$ and $\lambda$ is a predefined non-negative parameter. In equation (6.1), I use $W$ to denote all the weight vectors $\{w_k\}_{k=1}^{K}$.

I assign a Gaussian prior for each $w_k$ such that

$$w_k \sim \mathcal{N}(0, \sigma^2 I),$$

(6.3)

where $\sigma^2$ is the variance parameter.

Note that if we let $\lambda = 0$, then the model becomes the regular logistic regression. I modified logistic regression by introducing an additional $\lambda w_k^T w_k$ term.
6.1.2 Determining the Number of Clusters

I let $\lambda > 0$ to enforce the “rich-get-richer” property. When $\lambda > 0$, Equation (6.2) indicates that given the same prediction performance of the linear model (determined by $w^T \phi_n$), a sample is more likely to be assigned to a cluster with larger $w^T_k w_k$. Note that $w^T_k w_k$ is not a function of the features $\phi_n$; and serves as an additional non-negative bias term.

Now if I apply the Expectation Maximization (EM) [32] to learn the maximum a posteriori probability (MAP) estimator for $W$, then the derived maximization step that updates $W$ is given as

$$
\hat{W} = \arg \max_W \sum_{k=1}^K \left( \frac{1}{2\sigma^2} - \lambda \sum_{n=1}^N E[1(z_n = k)] \right) w_k^T w_k + \sum_{k=1}^K \sum_{n=1}^N E[1(z_n = k)] w_k^T \phi_n - \sum_{n=1}^N \log \sum_{i=1}^K \exp \left( w_i^T \phi_n + \lambda w_i^T w_i \right),
$$

(6.4)

where $E$ represents an expected value is taken with respect to the posterior distribution of $z_n$. $1(z_n = k)$ is an indicator function that returns 1, only if $z_n = k$; and returns 0, otherwise. Note that $\sum_{n=1}^N E[1(z_n = k)]$ represents the expected number of samples assigned to cluster $k$. By observing the first term in Equation (6.4), I conclude that for the cluster $k$ that contains more members, the model penalizes less with respect to $w_k^T w_k$. Therefore, if a cluster $k$ has more members, it tends to have a larger $w_k^T w_k$ value.

From Equation (6.4), we can conclude that big clusters with more members tend to have larger $w_k^T w_k$. As shown in Equation (6.2), samples are more likely to be assigned to clusters with larger $w_k^T w_k$. Therefore, in the EM optimization process, large clusters that already have many members are more likely to be assigned with more new members. This exhibits the “rich get richer” property. This property allows us to initialize the model with a large number of clusters (50 in our experiments). Similar to the updates in variational inference of Dirichlet process [56], only a few clusters remain non-empty when the optimization converges. The number of these non-empty clusters is the number of clusters automatically determined by the model.

$\lambda$ is a parameter that controls the trade off between prediction accuracy and cluster size. Larger $\lambda$ makes Equation (6.2) depends more on the cluster size (represented by
\( \mathbf{w}_k^T \mathbf{w}_k \), but less on prediction accuracy (represented by \( \mathbf{w}_k^T \mathbf{w}_k \)). In the experiments, I choose \( \lambda = 1/(2N\sigma^2) \) to ensure that the coefficients for \( \mathbf{w}_k^T \mathbf{w}_k \) in Equation (6.4) are non-positive and introduce \( l_2 \) regularization for the optimization.

### 6.1.3 Generating Expert Labels

I demonstrated how I generate the latent cluster indicators \( \mathbf{Z} \). Now, I introduce how I model the expert labels \( \mathbf{Y} \), given the latent cluster indicators \( \mathbf{Z} \).

As illustrated in the table shown in Figure 6.1, I model a mapping between the clusters and expert labels by assigning the cluster to the expert labels. I let \( t_k^{(m)} \in \{1 \ldots J_m\} \) indicate which label the cluster \( k \) is assigned to for expert \( m \), such that \( t_k^{(m)} = j \) implies that cluster \( k \) is assigned to the \( j \)-th label given by expert \( m \). I assign a uniform prior for \( t_k^{(m)} \), such that

\[
    t_k^{(m)} \sim \text{Categorical} \left( \left[ \frac{1}{J_m}, \ldots, \frac{1}{J_m} \right] \right),
\]

(6.5)

Note that for each expert \( m \), each cluster \( k \) is assigned to exactly one label \( j \), but this label \( j \) might be associated with multiple clusters, as illustrated in the table of Figure 6.1.

After assigning the cluster to the labels, I want to make sure that expert labels can be accurately predicted with the observed cluster indicator \( z_n \). For example, if cluster \( k \) is assigned to label \( j \) for expert \( m \) and sample \( n \) belongs to cluster \( k \), such that \( t_k^{(m)} = j \) and \( z_n = k \), then it should be very likely that expert \( m \) gives the label \( j \) for the \( n \)-th sample, i.e., the probability that \( y_n^{(m)} = j \) is high. Therefore, if we define a non-negative \( J_m \)-dimensional vector \( \eta_l^{(mj)} \) with each element \( \sum_{l=1}^{J_m} \eta_l^{(mj)} = 1 \), to represent the conditional probability

\[
    \eta_l^{(mj)} \overset{\text{def}}{=} p \left( y_n^{(m)} = l \mid z_n = k \text{ and } t_k^{(m)} = j \right),
\]

(6.6)

then it must be true that

\[
    \eta_j^{(mj)} \gg \eta_l^{(mj)}, \text{ for all } l \neq j.
\]

(6.7)
Note that Equation (6.6) is equivalent to letting $y_n^{(m)}$ follow a mixture of categorical distribution such that

$$y_n^{(m)} | \mathcal{Z}, \mathbf{T} \sim \prod_{k=1}^{K} \prod_{j=1}^{J_m} \left\{ \text{Categorical} \left( \eta^{(mj)} \right) \right\} 1(t_k^{(m)} = j) 1(z_n = k).$$

(6.8)

I enforce Condition (6.7) by assigning a Dirichlet distribution prior for $\eta^{(mj)}$, such that

$$\eta^{(mj)} \sim \text{Dirichlet} \left( \Psi^{(m)} \right),$$

(6.9)

where $\Psi^{(m)} = \{ \psi_i^{(m)} \}_{i=1}^{J_m}$ is a $J_m$-elemental vector, each of whose elements defined as

$$\psi_i^{(m)} = \begin{cases} \alpha, & \text{if } l = j \\ \beta, & \text{if } l \neq j. \end{cases}$$

(6.10)

$\alpha$ and $\beta$ are concentration parameters for the Dirichlet distribution. In order to make Condition (6.7) be satisfied, I chose $\alpha \gg \beta$.

In the experiments, I choose $\alpha = 40(J^{(m)} - 1)$ and $\beta = 10$. With these chosen parameters, the expected value for each element of $\eta^{(mj)}$ is given by $E[\eta_j^{(mj)}] = 0.8$ and $E[\eta_l^{(mj)}] = 0.2/(J_m - 1)$ for all $l \neq j$, which satisfies $E[\eta_j^{(mj)}] \gg E[\eta_l^{(mj)}]$. Choosing these parameters is equivalent to assuming that given the cluster indicator $z_n$, the model is able to predict the label $j$ given by expert $m$ with an average accuracy of 80% by default. I want to emphasize that this is the expected value of the prior distribution; and the posterior distribution for $\eta$ are learned through training.

6.1.4 The Overall Model of ICMPL

I have described the proposed discriminative probability model above. The joint distribution conditioned on the observed features $\Phi$ is given by

$$p(Y, W, Z, T, \eta | \Phi) = \prod_{k=1}^{K} p(w_k | \sigma^2) \prod_{m=1}^{M} \prod_{j=1}^{J_m} p(\eta^{(mj)} | \alpha, \beta) \prod_{m=1}^{M} \prod_{k=1}^{K} p(t_k^{(m)} | \beta) \prod_{n=1}^{N} p(z_n | W, \phi_n) \prod_{m=1}^{M} \prod_{n=1}^{N} p(y_n^{(m)} | z_n, \eta, T).$$

(6.11)

ICMPL is summarized using a graphical model in Figure 6.2.
Figure 6.2 The graphical model for interpretable crowdclustering.

6.2 Training ICMLP via Variational Inference

In Chapter 6.1, I presented our model. In this section, I introduce how to train the model. In this model, I learn the maximum a posteriori probability (MAP) estimator for $\mathbf{W}$ through Expectation Maximization (EM) [32]. In the expectation step, I first calculate the expected value of Equation (6.11) with respect to the posterior distribution $p(Z, T, \eta|\Phi, Y, \widehat{W})$, where $\widehat{W}$ is the current estimate of $\mathbf{W}$. I denote this expected value as $f(\mathbf{W}) = \mathbb{E}_{p(Y, W, Z, T, \eta|\Phi)}$. Then, in the Maximization step, I update the estimate of $\mathbf{W}$ to maximize this quantity, i.e.,

$$\widehat{W} = \arg\max_{\mathbf{W}} f(\mathbf{W}). \quad (6.12)$$

I have already derived the objective function of the Maximization step in Equation (6.4). Note that as shown in the graphical model in Figure 6.2, $\mathbf{W}$ is conditionally independent with $\eta$ and $T$ given $Z$. Therefore, Equation (6.4) is a function of $\mathbb{E}[z_n]$ but does not involve the random variables $\eta^{(mj)}$ or $t_k^{(m)}$. I obtain the optimal $\widehat{W}$ using conjugate gradient method.

I have derived the maximization step. However, we have a problem in the Expectation step, because the posterior distribution $p(Z, T, \eta|\Phi, Y, \widehat{W})$ is computationally intractable. Therefore, I use a variational distribution $q(Z, T, \eta)$ to approximate it such that

$$q(Z, T, \eta) \approx p(Z, T, \eta|\Phi, Y, \widehat{W}). \quad (6.13)$$
To ensure $q(Z, T, \eta)$ is tractable, I apply mean-field approximation such that

$$q(Z, T, \eta) = \prod_{n=1}^{N} q(z_n) \prod_{m=1}^{M} \prod_{j=1}^{J_m} q(\eta^{(mj)}) \prod_{m=1}^{M} \prod_{k=1}^{K} q(t_k^{(m)}).$$  \hfill (6.14)

In the inference, I derive an optimal variational distribution that minimizes the KL divergence between the variational distribution and the posterior distribution, denoted as $KL(q||p)$. As described in Appendix A, minimizing $KL(q||p)$ is equivalent to maximizing the lower-bound $L$, defined as

$$L = \sum_{m=1}^{M} \sum_{k=1}^{K} \mathbb{E}_q \left[ \log p(t_k^{(m)}) \right] + \sum_{n=1}^{N} \mathbb{E}_q \left[ \log p(z_n|\hat{W}, \phi_n) \right]$$

$$+ \sum_{m=1}^{M} \sum_{j=1}^{J_m} \mathbb{E}_q \left[ \log p(\eta^{(mj)}|\alpha, \beta) \right] + \sum_{m=1}^{M} \sum_{n=1}^{N} \mathbb{E}_q \left[ p(y_n^{(m)}|z_n, \eta, T) \right]$$

$$+ \sum_{n=1}^{N} H(q(z_n)) + \sum_{m=1}^{M} \sum_{k=1}^{K} H(q(t_k^{(m)})) + \sum_{m=1}^{M} \sum_{j=1}^{J} H(q(\eta^{(mj)})),$$  \hfill (6.15)

where $\mathbb{E}_q$ represents that the expected value is taken with respect to the variational distribution $q$ and $H(\cdot)$ represents the entropy of the variational distribution.

As described in Appendix A, it is straightforward to derive the updates for the variational distribution by applying variational calculus. I omit the details of derivation, and list the update equations as follows:

$$q(z_n) \sim \text{Categorical}(\rho_n)$$  \hfill (6.16)

where $\rho_n$ is a $k$-dimensional vector, each element of which is given by

$$\rho_{nk} \propto \exp \left\{ \sum_{m=1}^{M} \sum_{j=1}^{J_m} \mathbb{E}_q [1(t_k^{(m)} = j)] \sum_{l=1}^{J_k} \mathbb{I}(y_n^{(m)} = l) \mathbb{E}_q [\log \eta_l^{(mj)}] \right\}$$

$$+ \hat{w}_k^T \phi_n + \lambda \hat{w}_k^T \hat{w}_k - \log \sum_{i=1}^{K} \exp \left( \hat{w}_i^T \phi_n + \lambda \hat{w}_i^T \hat{w}_i \right).$$  \hfill (6.17)

$\rho_n$ is normalized such that $\sum_{k=1}^{K} \rho_{nk} = 1$.

$$q(t_k^{(m)}) \sim \text{Categorical}(\zeta^{(mk)}),$$  \hfill (6.18)
where $\zeta^{(mk)}$ is a $J_m$ dimensional vector, each element of which is given by

$$\zeta_j^{(mk)} \propto \exp \left\{ \sum_{n=1}^{N} \mathbb{E}_q[1(z_n = k)] \sum_{l=1}^{J_m} 1(y_n^{(m)} = l) \mathbb{E}_q[\log \eta_l^{(mj)}] \right\}, \quad (6.19)$$

$\zeta^{(mk)}$ is normalized such that $\sum_{j=1}^{J_m} \zeta_j^{(mk)} = 1$.

$$q(\eta^{(mj)}) \sim \text{Dirichlet}(\alpha_{\eta^{(mj)}}), \quad (6.20)$$

where $\alpha_{\eta^{(mj)}}$ is a $J_m$ dimensional vector defined as

$$\alpha_{\eta^{(mj)}} = \begin{cases} \alpha + \sum_{n=1}^{N} \sum_{k=1}^{K} \sum_{j=1}^{J_k} \mathbb{E}_q[1(z_n = k)] \mathbb{E}_q[1(t_k^{(m)} = j)] 1(y_n^{(m)} = l), & \text{if } l = j \\ \beta + \sum_{n=1}^{N} \sum_{k=1}^{K} \sum_{j=1}^{J_k} \mathbb{E}_q[1(z_n = k)] \mathbb{E}_q[1(t_k^{(m)} = j)] 1(y_n^{(m)} = l), & \text{if } l \neq j. \end{cases} \quad (6.21)$$

The expected values involved are given as follows:

$$\mathbb{E}_q[1(z_n = k)] = \rho_{nk}, \quad (6.22)$$

$$\mathbb{E}_q[1(t_k^{(m)} = j)] = \zeta_j^{(mk)}, \quad (6.23)$$

$$\mathbb{E}_q[\log \eta_l^{(mj)}] = \psi \left( \alpha_{\eta^{(mj)}} \right) - \psi \left( \sum_{i=1}^{J_m} \alpha_{\eta_i^{(mj)}} \right), \quad (6.24)$$

where $\psi$ is the digamma function, i.e. the logarithmic derivative of the gamma function.

I summarize the training process in Algorithm 6.

### 6.3 Experiments with ICMPL

In this section, I first present the experimental results on benchmark data. I demonstrate that ICMPL is able to learn the number of clusters and reveal the clustering structure by applying the method on benchmark data. Then I further illustrate the usefulness of the method with a real-world application.
Algorithm 6 Variational Expectation Maximization

repeat
  repeat
    for $n \leftarrow 1$ to $N$ do
      update $q(z_n)$ according to Equation (6.16).
    end for
    for $m \leftarrow 1$ to $M$ do
      for $k \leftarrow 1$ to $K$ do
        update $q(t_k^{(m)})$ according to Equation (6.18).
      end for
    end for
    for $m \leftarrow 1$ to $M$ do
      for $j \leftarrow 1$ to $J_m$ do
        update $q(\eta^{(mj)})$ according to Equation (6.20).
      end for
    end for
  until $L$ defined in Equation (6.15) converges
end repeat

Update $\hat{W}$ based on the Equation (6.4) using conjugate gradient method.

until $\hat{W}$ converges

6.3.1 Testing ICMPL with Benchmark Data

I test the ICMPL with 5 UCI datasets\cite{UCI}: iris, seeds, breast, glass and steel. I also test on 3 face recognition datasets: Yale, warpAR10P and warpPIE10P. The details about these datasets are provided in Appendix B.

For benchmark data, I only have access to the ground-truth cluster labels, but multi-expert labels are not available. Therefore, I generate labels for 10 synthetic experts, based on the ground-truth labels. For each expert, I first randomly partition the ground-truth cluster labels into 3 sets. For samples whose labels are in the each of the 3 sets, I assume the expert gives positive labels, negative labels and missing labels, respectively. This simulates the situation that each expert is interested in one particular binary classification task related to the ground-truth clusters. The expert might not be sure what label should be given for samples from certain ground-truth clusters, and decides to give missing labels. I randomly flip 10\% of the labels to simulate the error of expert labels. I generate binary expert labels only, such that the ground-truth number of clusters is not obvious by observing the number of clusters from each expert.

In the experiments, I vary the percentage of the labels observed from each expert from 10\% to 100\%. To achieve this, I randomly pick a subset of labels from each expert independently. I conduct 5-fold cross validation, and measure the performance by comparing
Normalized Mutual Information (NMI) [38] between the learned cluster indicators and ground-truth labels in both training and validation sets. NMI is the normalized version of mutual information such that it has a value between 0 and 1, where a larger value indicates a better performance. The definition of NMI is introduced in Appendix C.

**Competing Methods**

I compare ICMPL with the following methods:

*K-Means-Based Consensus Clustering (KCC) [42]* is a cluster ensemble method that learns a median consensus clustering solution such that the similarity between consensus result and all given clustering solutions is maximized. This method uses partition expert labels only, without accessing the features. It can not directly predict out-of-bag validation samples. Therefore, I train a multinomial logistic regression using the features and learned cluster labels in the training set. Then, I predict the cluster assignment in the validation set using the trained logistic regression model. I denote this method using $KCC+LR$.

*Metric Pairwise Constrained KMeans (MPCKMeans)[57]* is a semi-supervised learning algorithm that combines constrained clustering and metric learning. I generate pairwise must link and cannot link between two samples, if 80% of synthetic experts agree that they should be in the same cluster and in different clusters, respectively. Since we include some high-dimensional data, I apply a scalable version that learns diagonal diagonal covariance matrices, ignoring the covariance between features.

*Semi-crowdsourced Clustering (SemiCrowd)[12]* is a crowd clustering method that first completes the similarity matrix via convex optimization and then learns a distance metric that makes use of observed features. It is not straightforward to predict out-of-bag samples with this model, and the matrix-completion optimization is not scalable. I cannot apply this method to datasets with more than 500 samples.

*Multi-Expert Constrained Clustering (MECC) [14]* is a crowd clustering method that fit a multinomial logistic regression model to generate a clustering result that best predict the observed pairwise similarity labels.

In addition, I apply *k-means [3]* as a baseline, which makes use of observed features only, without using the expert labels. I also include *Dirichlet Process Gaussian Mixture Model (DPGMM) [55]* because it automatically learns the number of clusters.
Experimental Results

I report NMI in training and validation sets in Figure 6.3. I observe from this figure that ICMPL is one of the best performers in terms of NMI in both training and validation set. Note that all other methods, except for DPGMM, are provided with the ground-truth number of clusters as a given parameter. ICMPL is at a disadvantage, because it automatically learns the number of clusters, and thus provided with less ground-truth information.

KCC performed badly when less labels are observed in the training data, because it only makes use of the observed expert labels without accessing the features. ICMPL is able to combine expert labels with observed features, which makes it perform better when less expert labels are given. In the training set of seeds and steel, KCC gives a higher NMI than ICMPL when more labels are observed. However, in the validation sets, KCC+LR does not outperform ICMPL. This suggests that when features are noisy, expert labels might be more trustworthy. ICMPL might be negatively influenced by the noisy features in the training results. However, because of the noisy features, KCC+LR does not generalized better in the validation set.

We can also observed in the figure that the pairwise similarity based methods, including MPCKMeans, semicrowd and MECC, usually perform worse. Note that in the experiments I convert the partition labels into pairwise similarity labels. This results in dense pairwise similarity matrices. These methods might not perform well on dense similarity matrices.

As mentioned previously, ICMPL is able to automatically determine the number of clusters. I summarize the mean and standard deviation of the number of clusters discovered by ICMPL for different tasks in Table 6.1. I also report the results of DPGMM for comparison. It can be concluded from the table that the ICMPL is able to recover the number of clusters pretty accurately in most of the datasets. We can also observed that, ICMPL overestimates the numbers of clusters for steel and warpPIE10P dataset, probably because the features in these datasets are more noisy. Note that as shown in Figure 6.3, ICMPL still performs comparably with other methods on these two tasks in terms of NMI. DPGMM does not make use of the expert-label information, and performs worse. In steel, Yale and warpPIE10P datasets, DPGMM fails probably because each cluster in these datasets does not follow a Gaussian distribution.
Table 6.1: Number of Clusters Discovered by ICMPL

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Ground Truth</th>
<th>DPGMM</th>
<th>10%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>16%</td>
<td>30%</td>
</tr>
<tr>
<td>iris</td>
<td>3</td>
<td>3.4 (0.4)</td>
<td>3.0 (0.0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3.2 (0.4)</td>
<td>3.4 (0.5)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3.2 (0.4)</td>
<td>3.2 (0.4)</td>
</tr>
<tr>
<td>seeds</td>
<td>3</td>
<td>5.8 (0.4)</td>
<td>3.0 (0.0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3.6 (0.5)</td>
<td>3.8 (0.4)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4.4 (0.5)</td>
<td>4.6 (0.5)</td>
</tr>
<tr>
<td>breast</td>
<td>6</td>
<td>10.6 (1.7)</td>
<td>3.8 (0.4)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4.8 (0.4)</td>
<td>4.8 (0.4)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5.4 (0.8)</td>
<td>5.8 (0.4)</td>
</tr>
<tr>
<td>glass</td>
<td>6</td>
<td>16.8 (1.5)</td>
<td>4.6 (0.5)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7.0 (0.6)</td>
<td>6.4 (1.5)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8.4 (0.8)</td>
<td>8 (0.8)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8.4 (0.8)</td>
<td>8.4 (0.8)</td>
</tr>
<tr>
<td>steel</td>
<td>7</td>
<td>18.1 (1.9)</td>
<td>23.2 (2.8)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>28.4 (3.8)</td>
<td>31.0 (1.1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>31.4 (1.5)</td>
<td>31.4 (1.5)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>30.4 (1.4)</td>
<td>30.0 (2.8)</td>
</tr>
<tr>
<td>Yale</td>
<td>15</td>
<td>1.8 (0.4)</td>
<td>16.4 (1.6)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>14.6 (1.5)</td>
<td>13.2 (2.6)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>14.6 (1.9)</td>
<td>16.4 (2.1)</td>
</tr>
<tr>
<td>warpAR10P</td>
<td>10</td>
<td>21.4 (11.1)</td>
<td>11.0 (1.9)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11.8 (2.8)</td>
<td>11.8 (1.7)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>14.8 (1.9)</td>
<td>13.2 (1.2)</td>
</tr>
<tr>
<td>warpPIE10P</td>
<td>10</td>
<td>1.2 (0.4)</td>
<td>20.2 (1.3)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>17.2 (2.9)</td>
<td>14.8 (1.7)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>16.4 (1.6)</td>
<td>16.0 (2.3)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>19.4 (1.7)</td>
<td>19.4 (1.7)</td>
</tr>
</tbody>
</table>

6.3.2 Applying ICMPL on COPD Subtyping Problem

I apply ICMPL to analyze labels provided by multiple experts in the COPD dataset. This dataset contains 2,109 subjects with 39 features. The subjects include heavy...
cigarette smokers with and without COPD. The features are collected based on clinical information, lung function, and measures from computed tomography (CT) chest imaging. There are 63 clustering solutions provided by a cohort of COPD researchers, including pulmonologists, radiologists and data analysts. In each clustering solution, a subset (ranging from 85 to 2,109 samples) of the subjects are partitioned into different number of clusters (ranging from 2 to 10).

I randomly split the dataset into training and validation sets of equal sizes. I first train ICMPL using the training set. ICMPL finds 4 clusters in the dataset. Note that since the number of clusters varies across different expert solutions, it is not easy to manually choose the number of clusters for the consensus result. ICMPL automatically determines the number of clusters, which is useful in this application.

I also train the competing methods to partition samples into 4 clusters. I predict clusters in the validation set using the trained models to check how well the learned clustering solution generalizes for out-of-bag samples. Since I do not have the ground-truth labels, I cannot compute NMI. Rather, I check whether subjects in different clusters differ in terms the following 4 genetic risk scores [51, 52]: copdScore, lungfxScore, emphScore and airScore. These genetic risk scores measure the accumulation of genetic risk to different aspects of COPD and differences in genetic risk between COPD clusters may highlight biologic differences between clusters. Genetic risk score differences were evaluated via Kruskal-Wallis one-way analysis of variance [46], which is a non-parametric method for testing whether samples in different groups originate from the same distribution. Note that the 39 features I used to train the models do not directly contain gene features. I summarize the p-value of Kruskal-Wallis test in Table 6.2.

In the table, I bold all p-values that are less than 0.05, which implies statistical significance. ICMPL is the only method that achieves statistical significance in all 4 genetic risk scores. This suggests that ICMPL outperforms other methods, in terms of discovering clusters that are more correlated with the COPD-relevant genetic information of the subjects.

In this application, we are not only interested in finding a clustering solution. We also want to understand how the expert labels are related, and what the experts agree or disagree with each other.
Table 6.2: p-values in Kruskal-Wallis test

<table>
<thead>
<tr>
<th></th>
<th>copdScore</th>
<th>lungfxScore</th>
<th>emphScore</th>
<th>airScore</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICMPL</td>
<td>3.73E-07</td>
<td>1.52E-02</td>
<td>1.89E-04</td>
<td>1.18E-02</td>
</tr>
<tr>
<td>KCC+LR</td>
<td>1.14E-07</td>
<td>3.61E-01</td>
<td>2.44E-03</td>
<td>2.43E-02</td>
</tr>
<tr>
<td>MPCKMeans</td>
<td>1.96E-05</td>
<td>9.92E-01</td>
<td>6.64E-03</td>
<td>7.81E-01</td>
</tr>
<tr>
<td>MECC</td>
<td>7.45E-05</td>
<td>6.27E-01</td>
<td>5.61E-02</td>
<td>6.01E-02</td>
</tr>
<tr>
<td>kmeans</td>
<td>3.38E-01</td>
<td>3.29E-02</td>
<td>7.07E-01</td>
<td>1.06E-01</td>
</tr>
</tbody>
</table>

Table 6.3: Clustering Results of ICMPL on COPD Subtyping Problem

<table>
<thead>
<tr>
<th>No. samples</th>
<th>FEV1pp_utah</th>
<th>pctEmph</th>
<th>WallAreaPct_seg</th>
<th>Emph_UL_LL_ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>382</td>
<td>93.6 (14.6)</td>
<td>1.8 (1.7)</td>
<td>59.9 (2.5)</td>
</tr>
<tr>
<td>2</td>
<td>137</td>
<td>65.7 (16.9)</td>
<td>1.9 (1.9)</td>
<td>65.4 (2.3)</td>
</tr>
<tr>
<td>3</td>
<td>139</td>
<td>91.0 (13.5)</td>
<td>7.6 (4)</td>
<td>59.1 (2.3)</td>
</tr>
<tr>
<td>4</td>
<td>311</td>
<td>44.9 (18.6)</td>
<td>22.2 (12.1)</td>
<td>62.4 (2.7)</td>
</tr>
</tbody>
</table>

Table 6.4: Mapping between Clusters and Expert Labels

<table>
<thead>
<tr>
<th>Solutions</th>
<th>Mapping</th>
</tr>
</thead>
</table>
| A, B, C, D, E, F, G, H, I | Label 0: Cluster 1, 2 & 3  
Label 1: Cluster 4 |
| J         | Label 0: Cluster 4  
Label 1: Cluster 1, 2 & 3 |
| K         | Label 0: Cluster 2  
Label 1: Cluster 1, 3 & 4 |
| L, M      | Label 0: Cluster 1 & 3  
Label 1: Cluster 2 & 4 |
| N         | Label 0: Cluster 2 & 4  
Label 1: Cluster 1 & 3 |
| O         | Label 0: Cluster 1 & 3  
Label 1: Cluster 2  
Label 3: Cluster 4 |

I first summarize the mean and standard deviation of some important features for each learned cluster in Table 6.3. In this table, we can observe that the Cluster 1 contains subjects that are more resistant to cigarette smoking, which is characterized by a high $FEV_{1pp\_utah}$ value. Cluster 2 corresponds to airway disease predominant group, which is characterized by lower $FEV_{1pp\_utah}$, and higher $WallAreaPct_{seg}$. Cluster 3 corresponds to resistant cigarette smoker with mild emphysema, which is characterized by high $FEV_{1pp\_utah}$ and mild $pctEmph$. Cluster 4 corresponds to the sickest subjects, with lower $FEV_{1pp\_utah}$, higher $pctEmph$ and higher $WallAreaPct_{seg}$.

Now I analyze the relationship between the clustering results and expert labels. I pick the expert solutions that are accurately predicted based on learned clustering results,
such that the prediction accuracy is above 80% in the training set, where the predicted labels are given by estimating \( \mathbb{E}_q[p(y^{(m)}_n|z_n, T)] \), i.e., the expected value of Equation (6.8).

Then, I analyze \( q(T) \) to observe the mapping between the learned clusters and the observed labels, where I use alphabet letters to represent the experts. The results are summarized using 6 groups, as shown in Table 6.4. We can observe in the table that groups 1 contains 9 solutions. These solutions agree to separate the cluster 4 from the rest, i.e., they separate the sickest subjects from the healthier subjects. Group 2 agrees with group 1, but with positive and negative labels flipped. Group 3 separates the airway disease predominant group from the rest. Both groups 4 and 5 separate clusters 1 and 3 from clusters 2 and 4, with labels flipped. They separate the more resistant cigarette smokers from the sicker subjects. Expert O in fact partition the subjects into 5 groups, but label 2 and label 4 are not matched by any learned clusters. Labels 0, 1, and 3 in this solution corresponds to resistant cigarette smokers, airway predominant subjects and the sickest subjects, respectively.

As shown in this table, ICMPL helps us better understand the expert labels. Existing crowdclustering methods analyze the pairwise similarity labels and it would be more difficult to reveal such relationship between expert solutions.
Chapter 7

Conclusions

This dissertation focuses on improving interpretability of clustering algorithms by targeting the following three aspects: 1) Clustering with interpretable rules. 2) Interpretable clustering with similarity matrices. 3) Interpretable crowdclustering with partition labels. Most of these algorithms developed are motivated by the COPD subtyping problem.

In order to achieve clustering with interpretable rules, I first introduce the Discriminative Rectangle Mixture (DReaM) model that automatically generates interpretable rectangular rules in Chapter 3. DReaM is applied to COPD subtyping problem and allows us to update the existing COPD clinical guidelines. One limitation of the DReaM model is that it always uses all features and does not incorporate feature selection. In order to overcome this limitation, I propose a generative and discriminative interpretable clustering model with feature selection in Chapter 4.1 and 4.2, respectively.

Then, I further extend the interpretable clustering model to analyze similarity matrices in Chapter 5. The model generates a set of interpretable rules for each cluster, using a subset of selected features in the feature matrix; and at the same time, it forces the clustering results to be consistent with the observed similarity matrix. This model is applied to the COPD dataset and separates subjects into clusters by generating interpretable rules based on phenotypical data, and forcing the clustering results to be consistent with the Genetic Relationship Matrix (GRM) at the same time.

In Chapter 6, I focus on improving the interpretability of crowdclustering model. In order to improve interpretability, I propose to analyze partition labels, rather than pairwise
similarity labels. This allows us to learn the mapping between the latent clusters and the expert labels, and therefore improves the interpretability of crowdclustering algorithm, by revealing the relationships between clustering solutions from different experts. At the same time, ICMLP simultaneously determines the number of clusters, based on a “rich get richer” fashion; and partition samples into clusters with discriminative hyper-planes.
Appendix A

Variational Inference

I train several methods introduced in this dissertation using variational inference. In these models, we want to derive the posterior distribution \( p(V|U) \), where \( V \) represents all the latent variables and \( U \) represents all the observed variables. However, it may be intractable to compute this posterior distribution in some models. Therefore, I use a variational distribution \( q(V) \) to approximate it, such that

\[
q(V) \approx p(U|V) \tag{A.1}
\]

To make sure \( q(V) \) is tractable, I partition the elements of \( V \) into disjoint groups, denoted by \( V_i \) with \( i = 1 \ldots M \) assume \( q(V) \) can be factorizes with respect to these group, such that

\[
q(V) = \prod_{i=1}^{M} q_i(V_i) \tag{A.2}
\]

To find the best variational distribution \( q(V) \) is a good approximation for \( p(U|V) \), we minimize the Kullback-Leibler (KL) divergence between \( KL(q||p) \), defined as

\[
KL(q||p) = \int q(V) \log \left\{ \frac{q(V)}{p(V|U)} \right\} dV \tag{A.3}
\]

By Bayes’ theorem we have

\[
p(V|U) = \frac{p(U,V)}{p(U)} \tag{A.4}
\]
Substitute this equation into Equation (A.3), we have

\[ \text{KL}(q||p) = \int q(V) \log p(U) dV - \int q(V) \log \left( \frac{p(U, V)}{q(V)} \right) dV \]

\[ = \log p(U) - \mathcal{L}(U) \tag{A.5} \]

where I define \( \mathcal{L}(U) \) as

\[ \mathcal{L}(U) = \int q(V) \log \left( \frac{p(U, V)}{q(V)} \right) dV \tag{A.6} \]

Since \( U \) represent the observed variables, \( \log p(U) \) is a constant. Therefore, minimizing \( \text{KL}(q||p) \) is equivalent to maximizing \( \mathcal{L}(U) \).

Now we make use of the assumption that \( q(V) \) factorizes, as shown in Equation (A.2), I rewrite \( \mathcal{L}(q) \) as

\[ \mathcal{L}(q) = \int \prod_{i=1}^{M} q_i(V_i) \left\{ \log p(U, V) - \sum_{i=1}^{M} \log q_i \right\} dZ \tag{A.7} \]

I optimize \( \mathcal{L}(q) \) with respect to each distribution \( q_j \), for each \( j \in \{1, \ldots, M\} \), we rewrite \( \mathcal{L}(q) \) as a function of \( q_j \), such that

\[ \mathcal{L}(q_j) = \int q_j(V_j) \left\{ \prod_{i \neq j} q_i(V_i) \log p(U, V) dV_i \right\} dV_j - \int q_j(V_j) \log q_j(V_j) dV_j + \text{const} \]

\[ = \int q_j(V_j) \mathbb{E}_{i \neq j} \left[ \log p(U, V) \right] - \int q_j(V_j) \log q_j(V_j) dV_j + \text{const} \tag{A.8} \]

where \( \text{const} \) represents constants that are not a function of \( q_j \). \( \mathbb{E}_{i \neq j} \) represents that the expected value is taken with respect to all other variational distribution \( q_i(V_i) \) for all \( i \neq j \), i.e.

\[ \mathbb{E}_{i \neq j} \left[ \log p(U, V) \right] = \int \log p(U, V) \prod_{i \neq j} q_i(V_i) dV_i \tag{A.9} \]

Note that Equation (A.8) represents the negative KL divergence between \( q_j(V_j) \) and \( \mathbb{E}_{i \neq j} \left[ \log p(U, V) \right] \). Maximizing \( \mathcal{L}(q) \) is equivalent to minimizing this KL divergence, with the optimal \( q^*_j(V_j) \) given by

\[ \log q^*_j(V_j) = \mathbb{E}_{i \neq j} \left[ \log p(U, V) \right] + \text{const} \tag{A.10} \]
where $\text{const}$ represents the normalization constant that make sure $q_j^*(V_j)$ is a valid distribution.

With Equation (A.10), we are able to optimize $\mathcal{L}(q)$ by first initializing each $q_j(V_j)$, for all $j \in \{1, \ldots, M\}$ properly. Then we iteratively update each $q_j(V_j)$ with Equation (A.10) until convergence. Because $\mathcal{L}(q)$ is convex with respect to each of the factors $q_j(V_j)$, convergence is guaranteed.
Appendix B

Benchmark Datasets

This dissertation introduces 5 clustering models. These models are tested using benchmark datasets. The benchmark datasets that are used in the dissertation include 8 UCI data sets [45]: iris, wine, breast, seeds, wdbc, page, glass and steel; 2 gene expression data sets: lung[48] and carcinomas[49]; the following 6 human face image datasets: orlraws10p, pixraw10p, warpAR10p, warpPIE10p, Yale and CMU faces [45]; and 2 hand-written digit datasets: USPS and semeion. We summarize the number of samples, number of features and number of clusters in Table B.1.

<table>
<thead>
<tr>
<th>Data Set</th>
<th>#samples</th>
<th>#features</th>
<th>#clusters</th>
</tr>
</thead>
<tbody>
<tr>
<td>iris</td>
<td>150</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>wine</td>
<td>178</td>
<td>13</td>
<td>3</td>
</tr>
<tr>
<td>breast</td>
<td>106</td>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>seeds</td>
<td>210</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>wdbc</td>
<td>569</td>
<td>30</td>
<td>2</td>
</tr>
<tr>
<td>page</td>
<td>5,473</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>glass</td>
<td>214</td>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>steel</td>
<td>1,941</td>
<td>27</td>
<td>7</td>
</tr>
<tr>
<td>lung</td>
<td>203</td>
<td>3,312</td>
<td>5</td>
</tr>
<tr>
<td>carcinomas</td>
<td>174</td>
<td>9,182</td>
<td>11</td>
</tr>
<tr>
<td>orlraws10P</td>
<td>100</td>
<td>10,304</td>
<td>10</td>
</tr>
<tr>
<td>pixraw10P</td>
<td>100</td>
<td>10,000</td>
<td>10</td>
</tr>
<tr>
<td>warpAR10P</td>
<td>130</td>
<td>2,400</td>
<td>10</td>
</tr>
<tr>
<td>warpPIE10P</td>
<td>210</td>
<td>2,420</td>
<td>10</td>
</tr>
<tr>
<td>Yale</td>
<td>165</td>
<td>1,024</td>
<td>15</td>
</tr>
<tr>
<td>CMU faces</td>
<td>642</td>
<td>960</td>
<td>20</td>
</tr>
<tr>
<td>USPS</td>
<td>9,298</td>
<td>256</td>
<td>10</td>
</tr>
<tr>
<td>semeion</td>
<td>1,593</td>
<td>256</td>
<td>10</td>
</tr>
</tbody>
</table>
Appendix C

Evaluation of Clustering

This dissertation introduce clustering methods. I evaluate the quality of the clustering results generated by these methods, by comparing the clustering results with ground-truth labels. I evaluate the quality of the clustering results using Normalized Mutual Information (NMI) and Adjusted Rand Index (ARI). The definitions of these two quantity are provided in the following sections.

C.1 Normalized Mutual Information (NMI)

The Normalized Mutual Information (NMI) [38] between two discrete random variables X and Y can be defined as:

\[
NMI = \frac{I(X;Y)}{\sqrt{H(X)H(Y)}}
\]  

(C.1)

where \(I(X;Y)\) is the mutual information of X and Y, defined as

\[
I(X;Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) \log \left( \frac{p(x,y)}{p(x)p(y)} \right).
\]  

(C.2)

\(H(X)\) and \(H(Y)\) are entropy of X and Y, respectively. The definition of entropy is given as

\[
H(X) = -\sum_{x \in X} p(x) \log p(x)
\]  

(C.3)

NMI has a maximum value of 1, representing X and Y perfectly agree with each other, and a minimum value of 0, representing X and Y are irrelevant.
C.2 Adjusted Rand Index (ARI)

The adjusted Rand index is the corrected-for-chance version of the Rand index \[58\]. Given a set \( N \) elements, and two clusterings of these elements, namely \( X = \{X_1, X_2, \ldots, X_r\} \) and \( Y = \{Y_1, Y_2, \ldots, Y_s\} \), the overlap between \( X \) and \( Y \) can be summarized in a contingency table \( n_{ij} \), where each entry \( n_{ij} \) denotes the number of objects in common between \( X_i \) and \( Y_j \), i.e.,

\[
    n_{ij} = |X_i \cap Y_j|
\]

(A.4)

A contingency table can be illustrated as follows:

<table>
<thead>
<tr>
<th></th>
<th>( Y_1 )</th>
<th>( Y_2 )</th>
<th>( \ldots )</th>
<th>( Y_s )</th>
<th>Sums</th>
</tr>
</thead>
<tbody>
<tr>
<td>( X_1 )</td>
<td>( n_{11} )</td>
<td>( n_{12} )</td>
<td>( \ldots )</td>
<td>( n_{1s} )</td>
<td>( a_1 )</td>
</tr>
<tr>
<td>( X_2 )</td>
<td>( n_{21} )</td>
<td>( n_{22} )</td>
<td>( \ldots )</td>
<td>( n_{2s} )</td>
<td>( a_2 )</td>
</tr>
<tr>
<td>( \vdots )</td>
<td>( \vdots )</td>
<td>( \vdots )</td>
<td>( \ldots )</td>
<td>( \vdots )</td>
<td>( \vdots )</td>
</tr>
<tr>
<td>( X_r )</td>
<td>( n_{r1} )</td>
<td>( n_{r2} )</td>
<td>( \ldots )</td>
<td>( n_{rs} )</td>
<td>( a_r )</td>
</tr>
<tr>
<td>Sums</td>
<td>( b_1 )</td>
<td>( b_2 )</td>
<td>( \ldots )</td>
<td>( b_s )</td>
<td>( )</td>
</tr>
</tbody>
</table>

Then the Adjusted Rand Index is defined as

\[
ARI = \frac{\sum_{ij} \left( \begin{array}{c} n_{ij} \\ 2 \end{array} \right) - \sum_i \left( \begin{array}{c} a_i \\ 2 \end{array} \right) \sum_j \left( \begin{array}{c} b_j \\ 2 \end{array} \right) / \left( \begin{array}{c} n \\ 2 \end{array} \right)}{\frac{1}{2} \left\{ \sum_i \left( \begin{array}{c} a_i \\ 2 \end{array} \right) + \sum_j \left( \begin{array}{c} b_j \\ 2 \end{array} \right) \right\} - \sum_i \left( \begin{array}{c} a_i \\ 2 \end{array} \right) \sum_j \left( \begin{array}{c} b_j \\ 2 \end{array} \right) / \left( \begin{array}{c} n \\ 2 \end{array} \right)}
\]

(C.5)

where \( n_{ij}, a_i, b_j \) are values from the contingency table.

ARI has a maximum value of 1, representing \( X \) and \( Y \) perfectly agree with each other; but might yield a negative value, indicating the agreement between \( X \) and \( Y \) is less than random.
Bibliography


